D4.4 Trust, Privacy and Security related tools

Document Identification			
Status	Final	**Due Date**	31/08/2022
Version	1.2	**Submission Date**	06/09/2022

Related WP	WP4	**Document Reference**	D4.4
Related Deliverable(s)	D4.1	**Dissemination Level (*)**	PU
Lead Participant	INTRA	**Lead Author**	Olga E. Segou, PhD, Panos Matzakos, Athanasios Papadakis
Contributors	FILL, HOLO, i2CAT, ICCS, ENG	**Reviewers**	Orfeas Voutyras (ICCS) Estela Carmona Cejudo (I2CAT)

Keywords:
Security, Privacy, Trust, Threat Analysis, Cloud-Edge Infrastructures

(*) Dissemination level: **PU**: Public, fully open

Document information

List of Contributors	
Name	Partner
Olga E. Segou, PhD	INTRA
Panos Matzakos	INTRA
Athanasios Papadakis	INTRA
Gabrielle Giammatteo	ENG
Francesco Iadanza	ENG
Verena Stanzl	FILL
Orfeas Voutyras	ICCS
Alexandros Psychas	ICCS
Antonis Litke	ICCS
Nour Fendri	HOLO
Carina Pamminger	HOLO

Document History			
Version	Date	Change editors	Changes
0.1	17/05/2022	Olga E. Segou, PhD	Document structure created/Table of Contents by INTRA
0.2	15/07/2022	Olga E. Segou, PhD	Integrated contributions by ENG (4.2, 5.3)
0.3	22/07/2022	Olga E. Segou, PhD	Annexes added by INTRA
0.4	25/07/2022	Olga E. Segou, PhD	Integrated contributions by INTRA (4.3.2, 5.1)
0.5	01/08/2022	Olga E. Segou, PhD	Integrated contributions by HOLO (5.3.4, 6.4)
0.6	02//08/2022	Olga E. Segou, PhD	Threat analysis updated by INTRA (3.3)
0.7	03/08/2022	Olga E. Segou, PhD	Introduction, Conclusions finalised by INTRA
0.8	04/08/2002	Olga E. Segou, PhD	References, figures, tables, acronyms finalised by INTRA
0.9	25/08/2022	Estela Carmona Cejudo	1st internal review by i2CAT
0.10	29/08/2022	Orfeas Voutyras	2nd internal review and final contributions by ICCS (4.3.3, 5.2)
1.0	31/08/2022	Panos Matzakos	Reviewer comments addressed (INTRA, all)
1.1	02/09/2022	Carmen San Román	Quality Assurance review
1.2	06/09/2022	Lara López Muñiz	Final version to be submitted

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Olga E. Segou, PhD (INTRA)	31/08/2022
Quality manager	Carmen San Roman Alonso (ATOS)	02/09/2022
Project Coordinator	Lara Lopez Muñiz (ATOS)	06/09/2022

Table of contents

List of tables

List of figures

List of acronyms

Abbreviation / acronym	Description
ACL	Access Control List
CA	Certificate Authority
CD	Continuous Deployment
CI	Continuous Integration
DB	Database
DoS/DDoS	Denial of Service/Distributed Denial of Service
Dx.y	Deliverable number “y” belonging to WP “x”
ENISA	European Union Agency for Cybersecurity
IaaS	Infrastructure as a Service
ICMP	Internet Control Message Protocol
ICT	Information and Communication Technologies
K8s	Kubernetes
LDAP	Lightweight Directory Access Protocol
MITM	Man-In-The-Middle attack
ML	Machine Learning
OWASP	Open Web Application Security Project
QoE	Quality of Experience
QoS	Quality of Service
RBAC	Role Based Access Control
SSH	Secure Shell Protocol
SSL	Secure Sockets Layer
SYN	Synchronization
T&R	Trust and Reputation
TARA	Threat Assessment and Remediation Analysis
TLS	Transport Layer Security
VM	Virtual Machine
Tx.y	Task number “y” belonging to WP “x”
WP	Work Package
XSS	Cross-Site Scripting

Executive summary

The dynamic and decentralised nature of Cloud-Edge deployments requires a comprehensive approach to assess cyber threats, prioritise them according to their criticality and propose remediation measures to minimise cyber risks. A methodology is herein presented to catalogue the most important threats to security, privacy and trust and assess their potential impact to the Pledger platform. Pledger considers that assessing the potential impacts to integrity, confidentiality and availability is not sufficient to address cyber risks in the Cloud-Edge environment. Often, the degradation in a service or application Quality of Service/Quality of Experience (QoS/QoE) metrics can have devastating effects, especially when it comes to critical, latency-sensitive services.

A set of countermeasures is therefore compiled, to address the risks uncovered by the preliminary threat analysis. Countermeasures are prioritised according to their overall utility and cost. Results of the threat analysis and of actual metrics selected by the Pledger Intrusion Detection system are herein presented. The results show that any public-facing service needs to be secured as multiple daily intrusions and alerts are recorded.

The use of hardening measures and security, privacy and trust assets are considered a necessity to secure the Cloud-Edge continuum and ensure uninterrupted access to critical services. Within the document, the specifications of several security assets such as the decentralised Intrusion Detection system, and the Pledger Honeypot are presented, along with various hardening measures for the Kubernetes cluster, Big Data, Continuous Integration/Continuous Delivery and Blockchain subsystems, based on the results of the threat and countermeasure analysis.

Results from the use of Deep Learning methods to detect cyber threats on the infrastructure are further presented. Experiments showed that Machine Learning methods for anomaly detection were not yielding adequate results, therefore further experimentation with Neural Networks was required to successfully detect a variety of threats. The result of the Pledger Trust and Reputation subsystem is the implementation of a model that allows unknown entities to enter into safer interactions establishing a root of trust based on Trust and Reputation (T&R) QoS metrics extrapolated by the system.

Pledger's approach to cybersecurity allows the deployment of a number of different assets and provides best practices for the application of threat countermeasures. The approach allows each member of the Pledger value chain to access security, privacy, and trust assets, and enjoy the benefits of a secure cloud-edge environment.

1 Introduction

1.1 Purpose of the document

This document focuses on the security, privacy, and trust mechanisms applied in and by the Pledger platform. Pledger adopts a methodological approach, starting with a preliminary threat analysis on the main assets and infrastructures, along with a prioritisation of threats based on a variety of criticality factors such as the potential impact of a threat to confidentiality, integrity, availability, Quality of Service, among others, as well as the overall probability of its occurrence.

Following the threat analysis, Pledger selected a variety of countermeasures to detect, mitigate, prevent or limit the negative effects of cyber threats and prioritises them according to their utility and cost. A selection of countermeasures was then planned (and some of them already applied) in order to develop a secure, hardened Pledger platform. The specifications for selected countermeasures as well as the security assets deployed for task 4.1 (T4.1) is also herein provided.

Algorithmic information is also provided on the Deep Learning and Trust and Reputation subsystems that provide additional intelligence other than the traditional defences applied on the Pledger architecture.

To sum up, the main implementation updates provided in this document compared to its initial version (deliverable D4.1 [1]) cover the following:

- Updates on the architecture of privacy, security and trust components, namely with respect to the infrastructure hardening, as well as the deployed security assets (section 4).
- The description of security asset demonstrations to showcase the functionality and value of the individual Pledger security assets (section 5).

1.2 Approach to securing the Pledger platform

Pledger dedicates effort in T4.1 "Trust, Security, and Privacy on combined edge/cloud deployments" to ensure that privacy, security and trust mechanisms are in place, tailored to the specificities of the cloud/edge environment. T4.1 considers four main security pillars:

Pillar 1: Secure Architecture: The selection of tools and configurations to harden the Pledger infrastructure is based on an extensive threat analysis and countermeasure selection process.

Pillar 2: Flexible mechanism selection: This refers to the utilisation of tools that provide the users with the ability to define security and privacy characteristics and securing the data shared between edge and cloud, both at rest and in-transit.

Pillar 3: Risk assessment and Trusted nodes: This refers to a framework to identify trusted nodes, based on their compliance to specific rules and requirements provided by the user.

Pillar 4: Anomaly detection, classification and visualisation: This pillar refers to the use of data analytics to identify anomalies, as well as the data aggregation, indexing and visualisation mechanisms for complex network and threat data.

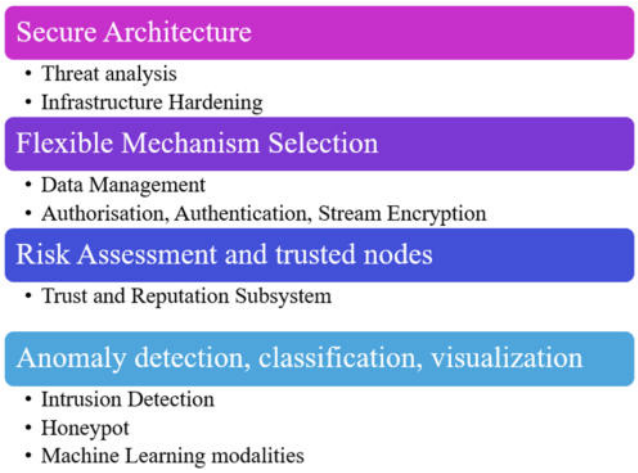


Figure 1: Pledger security pillars and key developments.

The T4.1 methodology starts with the definition of key security assets and features, providing the draft architecture for the Security subsystem in D2.3 [2]. Following this assessment, a threat analysis methodology is created in order to assess the major threats to security, privacy and trust, along with a set of countermeasures to alleviate their negative effects (section 3.1). The threat assessment allows Pledger to prioritise specific security mechanisms to be applied, covering all four pillars.

1.2.1 Security benefits across the Pledger value chain

Demonstrations of the key assets developed in T4.1 were performed (section 5), in order to showcase the utility of the deployed solutions. These assets protect the infrastructure from a variety of cyber threats and provide actionable intelligence to the infrastructure operators, as well as their individual clients that build and deploy their applications to the edge. For example, each client/tenant has access to any instances of the Intrusion Detection system they deploy, while the infrastructure operator has a bird's eye view of the threats across their entire client base. The development and deployment infrastructure for client applications was secured to prevent attacks that allow remote execution and code injection, while vulnerability scanning was applied to ensure certified applications maintained certain security standards. The trust and reputation subsystem allows for trusted interactions among any two parties. Figure 2 illustrates the main benefits, per stakeholder type, across the Pledger value chain, for the main T4.1 components that were deployed and tested on the project infrastructure.

Clients/Tenants

- Access to own security services (virtualized firewall, Intrusion Detection and Prevention etc.)
- Access to threat information from external sources (IaaS providers, threat databases etc.)
- Access to Deep Learning
- Access to runtime vulnerability scans
- Access to Trust & Reputation metrics
- Access of client applications on authorised data streams
- Secure blockchain & Smart Contract negotiation

Application Developers

- Access to secure infrastructure to develop their applications
- Access to secure infrastructure to deliver their applications
- Access to runtime vulnerability scans
- Secure blockchain & Smart Contract negotiation
- Access to Trust & Reputation metrics

IaaS/PaaS/SaaS providers

- Bird's eye view of threats in their infrastructure, across clients.
- Access to registry of certified, signed applications
- Access to Trust & Reputation metrics
- Access to runtime vulnerability scans
- Secure blockchain & Smart Contract negotiation
- Access to Deep Learning

Figure 2: Benefits across the Pledger main actors.

1.3 Relation to other project work

This document provides an update to the D4.1 [1] preliminary version of "Trust, Privacy and Security related tools" [1]. It updates the specifications of the components of the Pledger Security subsystem and illustrates the use of the security components. This document also references Deliverable D2.3 "Pledger Overall Architecture" [2], as the analysis herein is performed based on the Pledger conceptual architecture. An overview of the Pledger architecture is provided in the following Section in order to provide quick reference to the project components.

1.4 Structure of the document

This document is organised as follows:

Section 1 "Introduction" (present chapter) serves as an introduction to the document and delineates its scope.

Section 2 "Pledger Overview" provides an overview of the Pledger core platform and the project Use Cases as a handy reference of technical components.

Section 3 "Threat modelling" provides the preliminary threat analysis conducted for the Pledger Core and Use Cases, including the prioritisation of suggested countermeasures.

Section 4 "Architecture of privacy, security and trust components" provides the description of the T4.1 components and infrastructure hardening measures applied in Pledger.

Section 5 "Security Asset Demonstrations" provides the descriptions of the security demos created within the project, illustrating the real-world value of the Pledger security assets.

Section 6 "Conclusions" provides the final conclusions of the document.

Annex A provides additional information on the threat modelling methodology.

Annex B contains a history of changes incorporated in the current document, compared to its preliminary version D4.1 [1].

2 Pledger overview

This section provides an overview of the Pledger Core architecture, as defined in D2.3 [2], in order to provide the necessary context for this document.

2.1 Pledger core

This deliverable introduces the main functionalities and functional components of Pledger developed within Task "T4.1 – Trust, Security and Privacy on combined edge/cloud deployments". It also provides useful information related to the development of the corresponding Pledger **Security subsystem** and **Big Data Platform subsystem** which are two of the Pledger core subsystems (Figure 3), as identified in Deliverable "D2.3 Pledger Overall Architecture" [2] that acts as the roadmap for all development and integration tasks of the project.

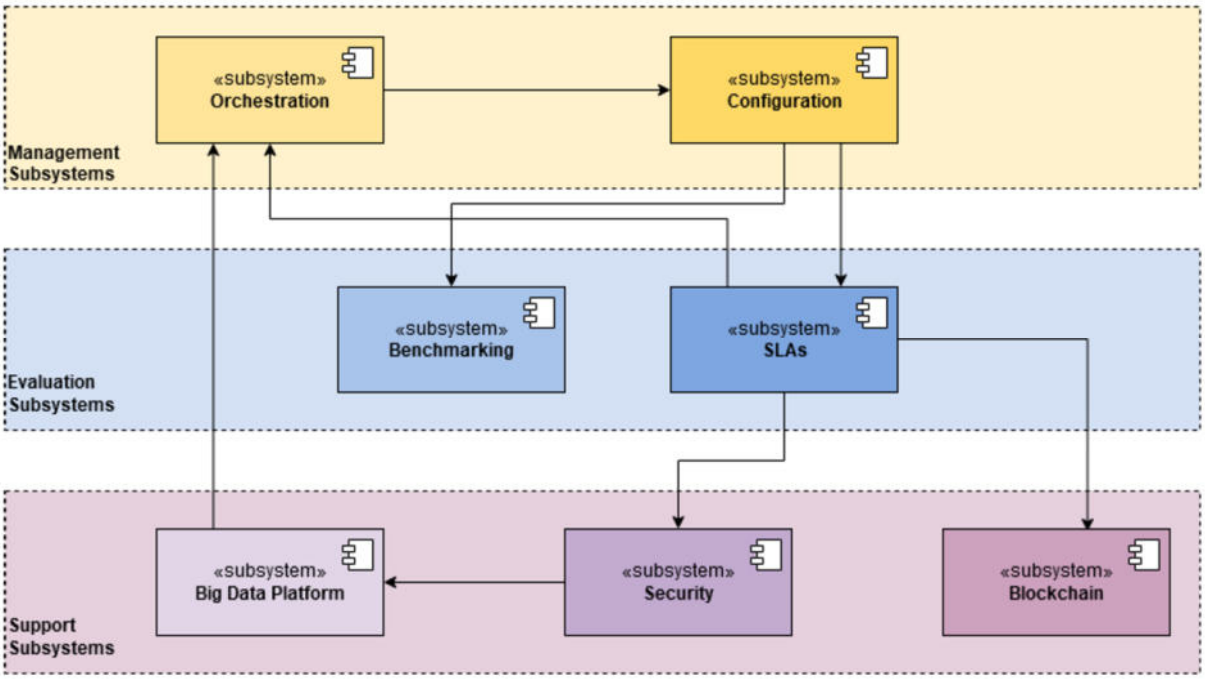


Figure 3: The Pledger Core Subsystems.

As also illustrated in D2.3 [2], a multi-layer approach is required to address security challenges at the edge. The Security subsystem includes components that enhance the security of the system as a whole. Other security features and mechanisms are included in components of other subsystems as part of their corresponding functionalities, but this subsystem is dedicated to security aspects of Pledger. Figure 4 provides the security subsystem architecture, improved upon the results of the threat analysis performed within Pledger. Table 1 describes the functionality of the individual components of the security subsystem. Section 4 updates the architecture for security components and provides their implementation details, as of August 2022.

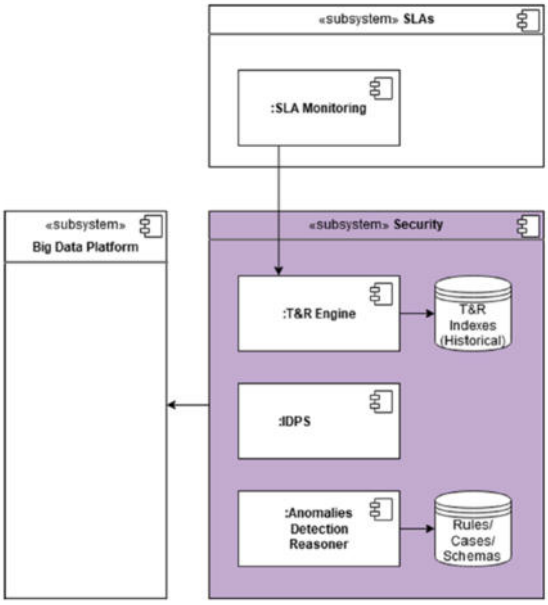


Figure 4: Security Subsystem draft architecture.

Table 1: Security subsystem Components.

ID	Component	Functionality
FC.7.1	**IDPS**	IDPS stands for Intrusion Detection and Prevention System. The component is used as network threat detection engine with real time intrusion detection capabilities. Such a component, when placed in the Edge to Cloud gateway, may inspect network traffic using different rules and detect possible threats.
FC.7.2	**Anomalies Detection Reasoner**	The Anomalies Detection Reasoner is responsible for the identification of rare items, events or observations which raise suspicions by differing significantly from much of the data. Typically, the anomalous items will translate to problems such as fraud or a structural defect. Pledger follows a Rules Based Reasoning (RBR) approach.
FC.7.3	**Rules/ Cases/ Schemas**	This database (DB) will include the identifiers of anomalies. Depending on the reasoning technique that will be selected, it may include Cases, Rules or other schemas.
FC.7.4	**T&R Engine**	This component assigns Trust & Reputation indices to the Infrastructure as a Service (IaaS) providers, based on their performance and reliability. That way, the ranking of the IaaS providers based on their trustworthiness is enabled.
FC.7.5	**T&R Indices**	In this DB the T&R stores the calculated Trust & Reputation indices, thus making possible the historical trustworthiness evaluation of actors.

The functionalities of the above components (Table 1) are presented in more detail in Section 4. The same section also includes some Security practices and solutions implemented in Pledger that do not take the form of a separate component and, as such, are not presented in Table 1. Similarly, Section 4 includes mitigation actions focusing on e.g., security design decisions for specific components that appear in other subsystems. As such, from an architectural point of view, it should be kept in mind that the Security subsystem does not encapsulate all the security decisions and activities of the project.

3 Threat modelling

3.1 Overview of threat modelling methodology

Pledger makes adaptations to the **Threat Assessment and Remediation Analysis** (TARA) [3] methodology proposed by MITRE[1]. TARA (Figure 5) proposes a stepwise approach where threats are catalogued and rated according to their overall criticality during the Cyber Threat Susceptibility Assessment. The Cyber Risk Remediation Analysis entails the discovery and prioritisation of effective risk remediation countermeasures. The methodology was adapted to consider the specificities of a collaborative project and in order to provide tailor-made scoring systems. An overview of the methodology and scoring methods is provided in Annex A.

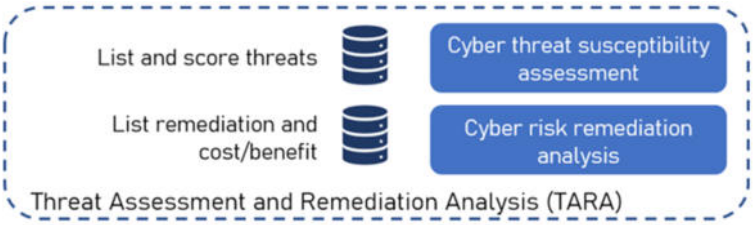


Figure 5: Threat Assessment and Remediation Analysis (TARA) overview.

The first step in the threat modelling process adopted by the project is to analyse threats to the main components and infrastructure elements. A list was compiled, looking into a variety of resources, such as reports, documentation, known threat databases and the Pledger honeypot. The threat severity was then scored based on the probability and criticality of the threat. This was performed in dedicated workshops within the consortium and collecting scores across all partners involved in the Core and Use Cases systems. Following the collection of the severity scores, a list of countermeasures was compiled, correlating each possible countermeasure with a threat. The countermeasures considered were then ranked based on their overall utility and the estimated cost/effort to obtain and operate them. A detailed analysis of the methodology and the scoring factors is included in Annex A. Figure 6 provides a quick overview of the threat modelling process adopted by Pledger.

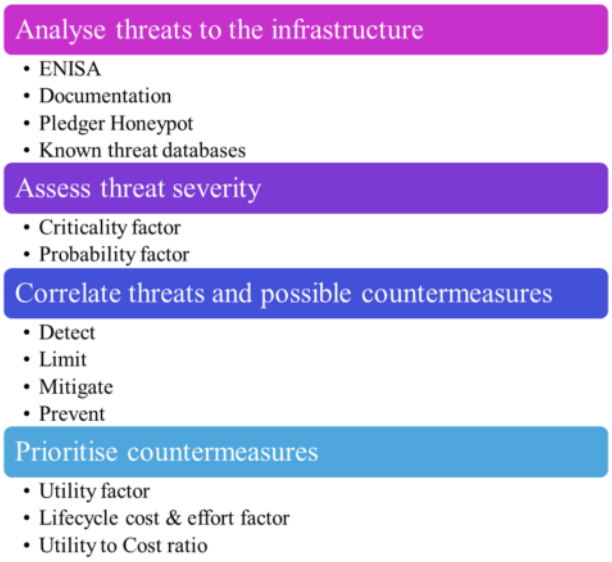


Figure 6: Overview of the Pledger threat modelling methodology.

[1] MITRE Corporation: https://www.mitre.org/ (Accessed August 2022).

3.2 Threat descriptions per Pledger information asset

In order to assess what the most important threats against the Pledger assets are, multiple sources of information were utilized (Figure 7). The consortium investigated well-known threat and vulnerability databases such as MITRE ATT&CK, CVE etc. Reports, scientific publications and press regarding attack modalities were also consulted. The relevant documentation for many open-source components as well as related standards were utilized. Furthermore, data analysis from the Pledger Honeypot was utilized, providing insight on the real-world attacks and suspicious traffic that raised alerts on our infrastructure. More information on the Honeypot architecture is provided in Subsection 0.

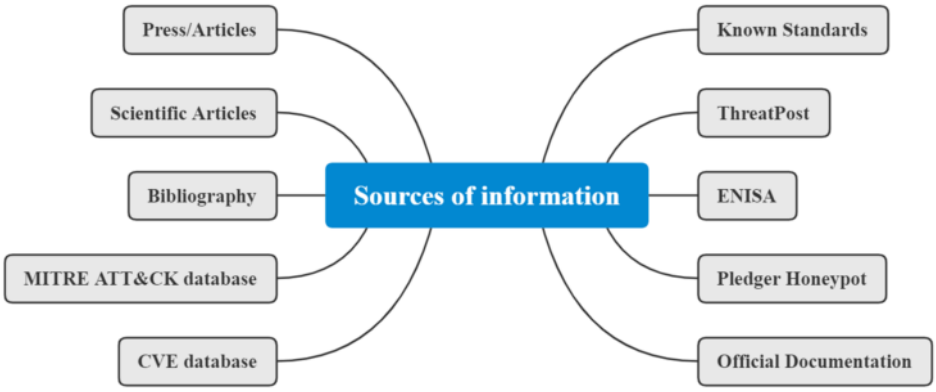


Figure 7: Sources of threat information.

The following subsections detail the most relevant threats per information asset, as well as some selected mitigation measures. This analytic threat analysis is also available in the preliminary version D4.1 [1]. In the current document, this information is updated in the corresponding threat tables with the mitigation measures per threat that have been implemented as of August 2022. More details on the implementation methods (countermeasures) that were used for each threat are provided in Table 14. to deal with each one of the threats In many cases, there are common TTPs that can affect more than one information asset. The aggregate table with unique rows and the prioritization of the TTPs based on their scoring follows in subsection 3.3.

3.2.1 Configuration and decision support system

In Pledger, Configuration and Decision Support System are two functional components deployed as a unique service, so they share some common threats. The former provides configuration data to the DSS in order to execute commands through the Orchestrator. The main threats of this service are represented by the configuration data stored and the possibility for an attacker to leverage on vulnerabilities of multiple kinds to get unauthorized access to the infrastructure to execute malicious code, as well as the possibility to access potential sensitive data shared among the running services.

Table 2: Common threats for the Configuration & DSS services.

TTP ID	CDSS-001
Title	**Cloud Infrastructure Discovery**
Source(s)	https://attack.mitre.org/techniques/T1580/ https://attack.mitre.org/mitigations/M1018/
Description	An attacker could exploit the Configuration service, get information about the infrastructure and use it to host malicious code in case he/she has a compromised user's access key. This could be a severe security threat as it exposes information about the infrastructure topology and availability of resources, in particular all the resource properties like specific hardware that could help the attacker to leverage on known CVEs to get unauthorized access.

Mitigations	▸ Configuration service should limit permission to access infrastructure information using the least privilege approach, ensuring that only those with specific authorisation can get access, and also tracking suspicious activities. For example, setting in place a Role Based Access Control (RBAC) and tracking the failed user login attempts to identify suspicious behaviour.
Implemented mitigations	▸ Implemented RBAC and audit logs in Configuration and DSS services.
TTP ID	**CDSS-002**
Title	**Cloud Service Dashboard**
Source(s)	https://attack.mitre.org/techniques/T1538/ https://attack.mitre.org/mitigations/M1018/
Description	An attacker could exploit the Configuration service and get information about the services that are running, leverage on known CVEs and jeopardise their execution in case he/she gets a compromised user's access keys
Mitigations	▸ Configuration service should limit permission to access infrastructure information using the least privilege approach, ensuring that only those with specific authorisation can get access, and also tracking suspicious activities. For example, setting in place a Role Based Access Control and tracking the failed user login attempts.
Implemented mitigations	▸ Implemented login failed audit logs when accessing configuration data through API or UI in Configuration and DSS services
TTP ID	**CDSS-003**
Title	**Account Manipulation**
Source(s)	https://attack.mitre.org/techniques/T1098/ https://attack.mitre.org/mitigations/M1026/ https://attack.mitre.org/techniques/T1059/
Description	An attacker could manipulate accounts to get access to victim systems if it manages to get compromised admin's access keys. This is a potential severe threat as administrator accounts are usually capable of modifying other accounts privileges and subvert security policies.
Mitigations	▸ Limit the administrator accounts that can create/manage user accounts, for example avoiding any admin-like accounts for day-to-day operations [-https://attack.mitre.org/techniques/T1098/] and restricting their capabilities only to configuration operations minimize the security impact. ▸ Use an Authentication/Authorization/Accounting system that stores the history of failed login attempts and of the most critical activities so that unauthorized use and abuse can be easily detected and reported.
Implemented mitigations	▸ Implemented clear separation of roles in Configuration and DSS services.
TTP ID	**CDSS-004**
Title	**Valid Accounts/Stolen credentials**
Source(s)	https://attack.mitre.org/techniques/T1078/
Description	An attacker could obtain and abuse credentials from existing account and use it to get increased privileges to specific systems
Mitigations	▸ Enforce secure password policies
Implemented mitigations	▸ Implemented hashed password management in Configuration and DSS services.

TTP ID	CDSS-005
Title	**Network Sniffing**
Source(s)	https://attack.mitre.org/techniques/T1040/ https://attack.mitre.org/mitigations/M1041/ https://attack.mitre.org/mitigations/M1037/
Description	An attacker could sniff network traffic and capture sensitive information shared among the application services
Mitigations	▸ Enforce HTTPS or TLS (Transport Layer Security) on all services communication or put a reverse proxy using HTTPS/TLS in front of existing HTTP services ▸ Filter network traffic to allow only the necessary ingress/egress and perform protocol-based filtering also in virtualized environments
Implemented mitigations	▸ Implemented Pod to Pod security policies to limit traffic for Configuration and DSS services.
TTP ID	**CDSS-006**
Title	**Create or Modify System Process**
Source(s)	https://attack.mitre.org/techniques/T1543/ https://attack.mitre.org/mitigations/M1022/ https://attack.mitre.org/mitigations/M1033/
Description	An attacker could create or modify the processes that are executed by the Orchestrator as directed by the DSS using vulnerabilities of background processes that are run along with the main service. This is a severe security breach as it potentially allows access to service data and the execution of malicious code.
Mitigations	▸ Restrict File and Directory Permissions, to limit access to specific paths of the operating system containing sensitive data ▸ Limit Software installation, to avoid the execution of unrequired software along with the main service
Implemented mitigations	▸ Added whitelist of image registry and applied immutable image guidelines for Configuration and DSS services.
TTP ID	**CDSS-007**
Title	**Modify System Image**
Source(s)	https://attack.mitre.org/techniques/T1601 https://attack.mitre.org/mitigations/M1045/
Description	An attacker could change the operating system of a container image executed by the Orchestrator as directed by the DSS to weaken its defences or add new malicious features. This is a severe threat that could affect both opensource images and those developed within the project
Mitigations	▸ Code Signing, to enforce binary integrity through code signature and prevent execution of untrusted software
Implemented mitigations	▸ Added CVE image scanning to limit access to malicious images.

3.2.2 Orchestration

The Orchestration subsystem is in charge of selecting the best nodes of a cluster (best effort) to run an application based on the SaaS provider's preferences or the profile of the application (e.g., app requires GPU to run) or the recommendations added by DSS at runtime. The Orchestrator also receives input from a Monitoring Engine to collect different metrics of the infrastructure in a time series database. With these functionalities we can settle the base for a QoS control system. The same subsystem could be part of the Trust & Reputation management system of the project, as described in section 4.3.3.

Table 3: Major threats for the Orchestration subsystem.

TTP ID	ORCH-001
Title	**Container and Resource Discovery**
Source(s)	https://attack.mitre.org/techniques/T1613/
Description	Adversaries may attempt to discover containers and other resources that are available within a container's environment. Other resources may include images, deployments, pods, nodes, and other information such as the status of a cluster. These resources can be viewed within web applications such as the Kubernetes dashboard or can be queried via the Docker and Kubernetes APIs. In Docker, logs may leak information about the environment, such as the environment's configuration, which services are available, and what cloud provider the victim may be utilizing. The discovery of these resources may inform an adversary's next steps in the environment, such as how to perform lateral movement and which methods to utilize for execution.
Mitigations	▸ Limit Access to Resource Over Network: Limit communications with the container service to local Unix sockets or remote access via Secure Shell Protocol (SSH). Require secure port access to communicate with the APIs over TLS by disabling unauthenticated access to the Docker API and Kubernetes API Server. ▸ Network Segmentation: Deny direct remote access to internal systems through the use of network proxies, gateways, and firewalls. User Account Management: Enforce the principle of least privilege by limiting dashboard visibility to only the required users.
TTP ID	**ORCH-002**
Title	**Container image vulnerabilities**
Source(s)	https://www.wwt.com/article/common-container-security-threats https://www.veritis.com/blog/container-security-an-overview-of-risks-and-security-best-practices/
Description	The use of container images with vulnerabilities can cause a wide range of security issues from allowing malware to be installed, to creating kernel panics and access to the host kernel. There are a number of reports of rogue containers infected with cryptocurrency mining malware embedded in community docker images. Docker hub has removed in the past images with backdoors to run cryptocurrency mining malware that may still be in use as of today. Container image risks include configuration defects, embedded malware, embedded clear text secrets, and use of untrusted images. Application images are among key vulnerable areas for security risks in a container environment. These images can be outdated ones, insecure versions of the software, applications carrying bugs, those containing hidden malware and those loosely configured. Moreover, these images often carry along authentication keys or certificates.
Mitigations	▸ Reading vulnerabilities across all layers of the image, from base layer to application frameworks and custom software. This visibility should ideally include flexible reporting and monitoring in line with organizational goals. ▸ Usage of container technology-specific vulnerability management tools and processes. ▸ At the policy level, there should be quality gates that check image vulnerability at every stage of the image lifecycle and allow only those meeting the set standards.
Implemented mitigations	▸ Partially implemented with a CVE scanner over publicly available images used as a baseline for all Pledger components plus the secure access to the image repository for the Pledger components

TTP ID	ORCH-003
Title	**Container risks**
Source(s)	https://www.veritis.com/blog/container-security-an-overview-of-risks-and-security-best-practices/
Description	Besides the entities such as apps and images they hold, containers themselves are often considered vulnerable to security risks. The most common security issue with containers occurs when container runtimes that manage containers themselves contain vulnerabilities. Common risks are vulnerabilities within the runtime software, unbounded network access from containers, insecure container runtime configurations, App vulnerabilities, and Rogue containers. A vulnerable runtime can expose all containers it runs, and also the host OS to potential security risks. Misconfigured runtimes may enjoy unrestricted access to all devices that can cause potential threats to all containers running on the host.
Mitigations	▸ Monitoring container runtimes and remediating them in case of any issues. ▸ Paying special attention to all the configurable options associated with the runtimes.
TTP ID	**ORCH-004**
Title	**Registry risks**
Source(s)	https://www.veritis.com/blog/container-security-an-overview-of-risks-and-security-best-practices/
Description	A container registry is a repository, or collection of repositories, used to store container images for Kubernetes, DevOps, and container-based application development. Registry Risks include insecure connections to registries, stale images in registries, insufficient authentication, and authorization restrictions.
Mitigations	▸ Configuring the development tools, orchestrators and container runtimes only to registries over encrypted channels. ▸ Pulling images only from trusted sources. When accessing from a poorly configured registry, the access should happen through encrypted and authenticated connections. ▸ The registry should undergo continuous monitoring to ensure all stale images that offer scope to vulnerabilities are cleared. This can be done by automating, based on time triggers and labels related images, the procedure of identifying registries with unsafe and vulnerable images that are no longer useful. The other way is to use operational procedures to access images using immutable names that identify discrete versions of useful images.
Implemented mitigations	▸ Implemented secure access to the image repository for all the Pledger components
TTP ID	**ORCH-005**
Title	**Orchestrator risks**
Source(s)	https://www.veritis.com/blog/container-security-an-overview-of-risks-and-security-best-practices/
Description	Orchestrator risks include unbounded administrative access, unauthorized access, poorly separated inter-container network traffic, mixing of workload sensitivity levels and orchestrator node trust, etc.
Mitigations	▸ Administrative interface that involves 'single orchestrator managing multiple apps' needs special attention. ▸ Use a least privilege access model allowing users to only perform specific actions on specific hosts, containers and images that their work demands. ▸ Tight access control to cluster-wide administrative accounts through effective authentication methods such as multifactor authentication beyond just password authentication.

	▸ Different orchestrators configured to separate network traffic categorized into discrete virtual networks. Public-facing web applications should be isolated from highly-sensitive workloads. Workloads should be allowed to run in their assigned security levels.
Implemented mitigations	▸ Pledger orchestration components running behind a VPN
TTP ID	**ORCH-006**
Title	**Network related threats**
Source(s)	https://www.wwt.com/article/common-container-security-threats
Description	With Kubernetes pod concepts, each self-contained pod/container can talk to all other pods/containers over the network overlay. It is possible a compromised container/pod can spread the malware across multiple containers/pods on multiple container hosts. Usually, this type of network traffic is east-west traffic and may not be detected without enhanced security mechanisms or network policies due to the dynamic scaling capabilities of the Kubernetes or docker container orchestration. Containers/pods exposed to the Internet are under the same type of threats that affect the non-containerized workloads such as the denial of services (DOS), SQL injection, cross-site scripting (XSS).
Mitigations	▸ Given the flexibility that containers enjoy through dynamic IPs over the network, there is a need to identify network anomalies in such an environment and apply relevant filtering tools to address any possible vulnerabilities.
Implemented mitigations	▸ Pledger orchestration components running behind a VPN
TTP ID	**ORCH-007**
Title	**Access control exploits**
Source(s)	https://www.wwt.com/article/common-container-security-threats
Description	Overlooking basic authentication and authorization to orchestrators like Kubernetes and Docker can offer attackers full access to container deployment. Securing the container orchestrator console/dashboard and APIs is the first step of securing the container infrastructure. Another example of a security vulnerability with unsecured API is the use of Docker API to create new rogue containers on container hosts with the community images and malware for cryptocurrency mining.
Mitigations	▸ Authorisation and authentication mechanisms, Identity and Access Management (IAM) mechanisms
Implemented mitigations	▸ Configured the APIs used to manage the different orchestrators (Kubernetes, Docker etc.) to use HTTPS and API token authentication

3.2.3 Big data management

Streamhandler is a highly scalable, distributed event-driven streaming platform based on Apache Kafka [4]. It organizes the data in partitioned topics and makes possible the simultaneous processing of data that are entering the platform from multiple clients. Additionally, Streamhandler allows to publish and subscribe to streams of records (topics) similar to the functionality provided by a message queue. The streams of records are stored in a fault-tolerant durable way and consumers can process them as they occur.

In the context of the Pledger project, Streamhandler is used as an internal middle layer of the Pledger Core System enabling the different systems of the platform to be integrated asynchronously. With this way, producer and consumers of different applications are really decoupled and scaled independently at their speed and requirements.

Table 4: Common threats for the Big Data infrastructure.

TTP ID	BDM-001
Title	**Man-In-The-Middle (MITM) attack**
Source(s)	MITRE: https://attack.mitre.org/techniques/T1557/
Description	An attacker could sniff network traffic and capture sensitive information routed to the Streamhandler if the data sent to the cluster sent in PLAINTEXT.
Mitigations	▸ Encryption of data using Secure Sockets Layer (SSL) **[Implemented]** ▸ SSL Authentication **[Implemented]**
TTP ID	**BDM-002**
Title	**Root SSH brute force attack**
Source(s)	MITRE: https://attack.mitre.org/techniques/T1110/
Description	In root ssh brute-force attacks, attackers often launch repeatedly and systematically the root ssh command in order to access the machine remotely and attempt password combinations until they find the password that works.
Mitigations	▸ Disable root ssh: Streamhandler platform has been implemented to Virtual Machines (VMs) where root ssh is disabled. **[Implemented]** ▸ Firewall setup: Firewall rules have been implemented in order to ssh connections after a number of unsuccessful trials **[Implemented]**
TTP ID	**BDM-002**
Title	**Denial of Service**
Source(s)	Pledger Honeypot, European Union Agency for Cybersecurity (ENISA) Big Data Threat Landscape Report 2016
Description	In this kind of attacks, the attacker exhausts the network or computing/memory resources of a server. As a result, the system crashes and denies access to real users.
Mitigations	▸ Firewall setup: Firewall rules have been implemented in order to allow the connection of IPs only from trusted nodes. **[Implemented]**
TTP ID	**BDM-003**
Title	**Unsecure application API**
Source(s)	ENISA Big Data Threat Landscape Report 2016 Scientific book publication on Security in Data Intensive Computing systems [5]. Specialised Databases: https://cve.mitre.org and https://www.cvedetails.com OWASP Top Ten https://owasp.org/www-project-top-ten/
Description	Various sources claim that Big Data is often built with little security. New software components are usually provided with service-level authorization, but few utilities are available to protect core features and application interfaces (APIs). Since Big Data applications are built on web services models, APIs may be vulnerable to well-known attacks, such as the Open Web Application Security Project (OWASP) Top Ten list, with few facilities for countering common web threats.
Mitigations	▸ Apply app authorisation features **[Partially implemented]**
TTP ID	**BDM-004**
Title	**Eavesdropping**
Source(s)	ENISA Big Data Threat Landscape Report 2016
Description	A common issue that affects any Information and Communication Technologies (ICT) infrastructure is when offenders can intercept communications between nodes by targeting the communication links. Various sources claim that inter-node communication with new Big Data tools is often unsecured, that it is not difficult to hijack a user session or gain unauthorized access. There is evidence of flaws in

	communication protocols. Big Data software distributions (for example Hadoop, Cassandra, MongoDB, Couchbase) rarely have the protocols that ensure data confidentiality and integrity between communicating applications (e.g., TLS and SSL) enabled by default or configured properly (e.g., changing default passwords).
Mitigations	▸ In-transit and on-disk encryption must be applied. **[Implemented]** ▸ Utilising secure protocols (SSH/TLS/HTTPS etc) **[Implemented]** ▸ Enforcing strong password policies **[Implemented]**

3.2.4 Continuous Integration/Continuous Delivery (CI/CD)

The Pledger CI/CD platform is implemented as a collection of open-source software components, which collectively aim to create an automated build system capable of integrating changes performed by developers working on individual tools of the Pledger project. In addition to that, the CI/CD platform acts as a testing and system integration environment, which can accommodate development instances and effectively integrate, test and release all the software services of the project.

Table 5: CI/CD Infrastructure threats.

TTP ID	**CICD-001**
Title	**Root SSH brute force attack**
Source(s)	MITRE: https://attack.mitre.org/techniques/T1110/
Description	In root ssh brute-force attacks, attackers often launch repeatedly and systematically the root ssh command in order to access the machine remotely and attempt password combinations until they find the password that works.
Mitigations	▸ Disable root ssh: Streamhandler platform has been implemented to VMs that root ssh is disabled. **[Implemented]** ▸ Firewall setup: Firewall rules have been implemented in order to ssh connections after a number of unsuccessful trials. **[Implemented]**
TTP ID	**CICD-002**
Title	**Denial of Service**
Source(s)	MITRE, Pledger Honeypot
Description	Denial of Service is a type of attack that aims to exhaust resources on the target system until it crashes and denies access to real users. Multiple modalities exist, exploiting network or application level mechanisms, with varied behaviours such as reflection, flooding, amplification attacks, etc. Within the Pledger Honeypot, there were also recorded Denial-of-Service attacks exploiting the NTP protocol (monlist mechanism[2]) to deploy an amplification-type attack, as well as attacks based on failed handshakes and malformed packets.
Mitigations	▸ Firewall setup: Firewall rules have been implemented in order to allow the connection of IPs only from trusted nodes. **[Implemented]** ▸ Blacklist policies: Refusing entry of traffic from IP addresses that have been blacklisted. **[Implemented]**

[2] Monlist command of the NTP protocol returns a list of hosts that recently connected to the NTP server. In an NTP amplification attack, the attacker sends repeatedly the request for this list while spoofing the requester's IP to the one of the victim server.

TTP ID	CICD-003
Title	**Remote Code Execution**
Source(s)	Shadow Containers attack: https://blog.aquasec.com/host-rebinding-and-shadow-containers-at-blackhat-2017 Execution: https://attack.mitre.org/tactics/TA0002/
Description	Execution consists of techniques that result in adversary-controlled code running on a local or remote system. Techniques that run malicious code are often paired with techniques from all other tactics to achieve broader goals, like exploring a network or stealing data. For example, the Shadow containers attack aims to execute malicious code by a three-stage process to inject code and elevate the attacker's privileges by exploiting the Same Origin Policy to build malicious code.
Mitigations	▸ Code Signing ▸ Establishing a private Code Repository **[Implemented]**

3.2.5 Blockchain

Hyperledger is an open-source collaborative effort created to advance cross-industry blockchain technologies. It is a global collaboration, hosted by The Linux Foundation, including leaders in finance, banking, IoT, supply chains, and Technology.

Table 6: Blockchain threats.

TTP ID	BLC-001
Title	**(Distributed) Denial of Service attacks**
Source(s)	https://attack.mitre.org/techniques/T1499/ https://www.apriorit.com/dev-blog/578-blockchain-attack-vectors https://geth.ethereum.org/docs/whisper/whisper-overview
Description	A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt the normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic. When attacking a blockchain network using DDoS, hackers intend to bring down a server by consuming all its processing resources with numerous requests. DDoS attackers aim to disconnect a network's mining pools, e-wallets, crypto exchanges, and other financial services. A blockchain can also be hacked with DDoS at its application layer using DDoS botnets.
Mitigations	▸ In order to prevent a DDoS attack in the Whisper protocol, proof-of-work (PoW) algorithm is used. The messages sent by the nodes in the private blockchain (created by Geth), will be processed (and forwarded further) only if their PoW exceeds a certain threshold, otherwise they will be dropped. **[Implemented]**
TTP ID	**BLC-002**
Title	**Spying on the HTTP Endpoint**
Source(s)	MITRE https://www.apriorit.com/dev-blog/578-blockchain-attack-vectors
Description	All the traffic generated between nodes is being sent through HTTP endpoints. The data being transmitted is not encrypted and thus anyone can connect to the IP and can start listening in to the private information.
Mitigations	▸ This can be solved by setting up an HTTPS reverse proxy that communicates securely with the HTTP endpoint and handles sending the data to the other nodes that are also using an HTTPs reverse-proxy. That way, the data will be encrypted and there would be no danger of someone spying on the sensitive data being transmitted.

TTP ID	BLC-001
Title	**False Friends/ Eclipsing Attack**
Source(s)	https://www.apriorit.com/dev-blog/578-blockchain-attack-vectors https://arxiv.org/pdf/1908.10141.pdf
Description	In an eclipse attack, the attacker eclipses the node from the network. The attacker makes sure that the node will not communicate with the blockchain network. The node will believe in a completely different truth than the rest of the network after the node is compromised by the attack. The attack requires a hacker to control a large number of IP addresses or to have a distributed botnet. Then the attacker overwrites the addresses in the "tried" table of the victim node and waits until the victim node is restarted. After restarting, all outgoing connections of the victim node will be redirected to the IP addresses controlled by the attacker. This makes the victim unable to obtain transactions they're interested in. Researchers from Boston University initiated an eclipse attack on the Ethereum network and managed to do it using just one or two machines. **False Friends:** A type of an eclipse attack on the Geth blockchain. It could be done with limited resources (only 2 IP addresses with /24 subnets). This used to work on Geth v1.8.0 but it has been patched on v 1.9.0.
Mitigations	▸ Use a higher version of Geth. The one used is v1.9.20. **[Implemented]**
TTP ID	**BLC-001**
Title	**Sybil Attack**
Source(s)	https://www.apriorit.com/dev-blog/578-blockchain-attack-vectors
Description	A Sybil attack is arranged by assigning several identifiers to the same node. Blockchain networks have no trusted nodes, and every request is sent to a number of nodes. During a Sybil attack, a hacker takes control of multiple nodes in the network. Then the victim is surrounded by fake nodes that close up all their transactions. Finally, the victim becomes open to double-spending attacks.
Mitigations	▸ Increasing the cost of creating a new identity, requiring some type of trust for joining the network, or determining user power based on reputation. **[Implemented]**

3.2.6 Benchmarking

Pledger's Benchmarking module automatically assesses the performance of the infrastructures controlled by Pledger, by executing benchmarking tests on the infrastructure's nodes.

In order to assess the performance of an infrastructure, the Benchmarking components need to access the infrastructure. For this purpose, the Benchmarking Suite manages configuration data that contains sensitive information, in particular the credentials that will be used by the tool to access the infrastructure. This data can be provided by the user in the GUI or retrieved from the Pledger's Configuration service, it is stored in the internal MongoDB database and finally it is used by the Executor to establish a connection with the target infrastructure.

Clearly, infrastructure configuration and credentials data are very sensitive because an attacker in possession of this data could easily gain access to the infrastructure and exploit its resources. This risk is so concrete and has so big impact that it is listed as #3 in the OWASP's "Top 10 Web Application Security Risks" as "Sensitive Data Exposure". In addition, "Exposure of Sensitive Information to an Unauthorized Actor" is one of the "Top 25 Most Dangerous Software Weaknesses" in the list of common software weaknesses produced by CVE.

Furthermore, the performance of the infrastructures is evaluated by executing benchmarking tests on the target infrastructures and collecting metrics of the executions. The Benchmarking Library component keeps the list of legitimate benchmarking tests that can be executed. In future releases, also benchmarking tests provided by the users will be available for the execution on the infrastructures. The automatic execution of code on the target infrastructures brings several security risks because if malicious code is provided in the Benchmarking Library or malicious code is injected in a running benchmarking test, it will be executed by the Benchmarking components on the infrastructures without the need for the attacker to have credentials (and the privileges) to do that.

Table 7: Benchmarking system threats.

TTP ID	**BNCH-001**
Title	**Sensitive data exposure**
Source(s)	This risk is so concrete and has so big impact that it is listed as #3 in the OWASP's "Top 10 Web Application Security Risks" as "Sensitive Data Exposure". In addition, "Exposure of Sensitive Information to an Unauthorized Actor" is one of the "Top 25 Most Dangerous Software Weaknesses" in the list of common software weakness produced by CVE. This is particularly relevant in the case of multiple tenants/slices.
Description	Attackers can use multiple techniques to get access to sensitive data: ▸ Sniff data in-transit, like sniffing the network traffic (MITRE Technique #T1040). ▸ Steal data at-rest from databases or files (MITRE Technique #T1005) looking for poorly secured stored credentials (MITRE Technique #T1552).
Mitigations	In order to mitigate the risk of exposing sensitive data, several countermeasures are suggested by MITRE and OWASP. The following ones have been applied in the Benchmarking components: ▸ Sensitive data is never included in application API and GUI responses (CWE-201). ▸ Sensitive data is always encrypted (MITRE Mitigation #M1041) both in-transit using TLS for all communications and at-rest encrypting data before storing it in the database with a strong encryption algorithm (i.e., AES) and keeping keys outside the application and the database. A further planned improvement is to use a secrets vault system like Hashicorp's Vault. ▸ Requests to application's API and GUI are authenticated and authorized using signed JWT tokens issued by Keycloak. System, application and third-party software is regularly updated (MITRE Mitigation #M1051).
Implemented mitigations	▸ Implemented data encryption using HTTPS and JWT tokens via KeyClock in the Benchmarking suite
TTP ID	**BNCH-002**
Title	**Remote code execution**
Source(s)	MITRE recognizes different techniques that exploit the execution of malicious code in order to gain higher privileges on a system: #T1204 "User Execution", #T1055 "Process Injection" and T1068 "Exploitation for Privilege Escalation". Proper network segmentation is also important among the different slices.
Description	However, security of virtual machines and/or containers is not total and other techniques are known that could still compromise the target infrastructure by running malicious container images (i.e., MITRE #T1204 "Malicious Image") or break the container isolation to gain access to the host (i.e., MITRE #T1611 "Escape to Host").

Mitigations	In Pledger's Benchmarking subsystem, a first and important mitigation of this risk comes from the usage of virtualization and containerization technologies for the execution of benchmarking tests (MITRE Mitigation M1048 "Application Isolation and Sandboxing"). These technologies ensure a high level of isolation of the execution environment from the physical target infrastructure. A further set of countermeasures will be considered to minimize the security risks generated by the execution of code on target infrastructures: ▸ Use read-only containers and minimal images for the benchmarking tests to prevent the execution of non-allowed commands (MITRE Mitigation M1038 "Execution Prevention"). ▸ Use code signing algorithms and vulnerability scanning software (e.g., Clair, Falco) to ensure the integrity of the code available in the Benchmarking Library. When running in Kubernetes infrastructures, improve isolation of containers by adjusting the security policies, e.g. execute containers as unprivileged, disable mounting of host paths, enforce selinux, require read-only root filesystem.
Implemented mitigations	▸ Added CIS benchmarking to trace security vulnerabilities and to refer to official mitigation guidelines.

3.2.7 QoS, SLAs, and T&R assessment and management systems

As one of the Evaluation Subsystems of Pledger, the QoS Assessment entity will be the tool responsible for reinforcing the Quality of Service (QoS) agreed between the provider of the infrastructure (IaaS/PaaS) and the consumer or service developer (SaaS). This component assures performance and availability for software components captured as KPI or quantifiable metrics measurements within an SLA agreement. The QoS aspects are extracted from the SLA agreement and used as performance guarantees for the infrastructure used by the applications. From the architecture point of view, the SLA subsystem is the subsystem responsible for creating, managing, and evaluating the SLAs associated to the applications running on the Pledger Cloud and Edge environment. This subsystem relies on the information gathered by external monitoring tools, which are used to continuously evaluate the SLAs, and to notify other components about violations of the QoS agreed.

As presented in Section 4.3.3, this subsystem is closely related to the Trust & Reputation management system that Pledger could implement, in parallel with the usage of Blockchain technologies. Interestingly enough, although such systems are used to mitigate malicious behaviour in networks where independent actors interact with each other (thus mitigating specific security threats), the systems themselves are prone to other security attacks, such as the ones presented in Figure 8 [6] (individual malicious actors, malicious collectives, malicious actors with camouflage, sibyl attack, slanderers, man in the middle attack, etc.):

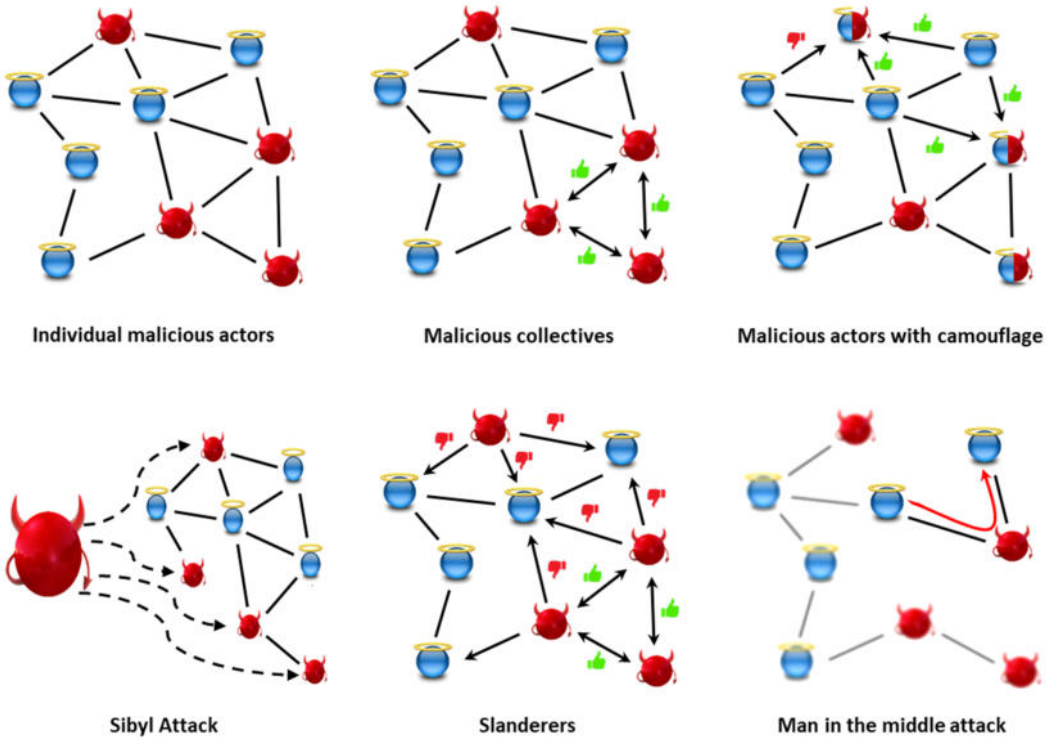


Figure 8: Common attacks in T&R systems.

The following table includes some of the major threats for such evaluation systems, under which most of the direct or indirect attacks can take place.

Table 8: Major threats for QoS, SLAs, and T&R assessment and management systems.

TTP ID	QSTR-001
Title	**Oscillating malicious service provision**
Source(s)	http://iot-cosmos.eu/sites/default/files/iot-cosmos/public/content-files/deliverables/
Description	A malicious actor that demonstrates oscillating behaviour is trying to trick the collective "mind" of the network by causing conflicts between the observations of individual nodes when providing services or recommendations. The oscillation can occur between different services or within the same service. In the first case, an actor can act maliciously for one service while being benevolent for another one. In the second and harder to identify case, a node generally provides good service, but not always. This way, when nodes try to collectively decide the reputation of the node, they might disagree with each other.
Mitigations	▸ Decoupling the Trust & Reputation Indices for different (types of) services. [Implemented] ▸ Providing a weight factor for the computation of the Trust & Reputation Indices, based on the criticality of the service being provided. [Implemented] ▸ Using both a Trust and a Reputation Index. [Implementedd] ▸ Usage of several time-windows to extract long-term and short-term T&R indices and fading factors. [Implemented]

TTP ID	QSTR-002
Title	**Malicious a posteriori evaluation**
Source(s)	http://iot-cosmos.eu/sites/default/files/iot-cosmos/public/content-files/deliverables/
Description	This behaviour is encountered when the nodes of a system have social attributes and T&R models are used. An actor may provide inaccurate satisfaction ratings/ evaluation for consumed services, in order to e.g. maliciously reduce the reputation of another actor (case of slanderers). It should be mentioned that these malicious nodes do not necessarily demonstrate malicious service provision.
Mitigations	▸ Include a mechanism to evaluate the evaluators (by e.g. identifying great differences between evaluations of different actors for the same service). **[Implemented]**
TTP ID	**QSTR-002**
Title	**Malicious collectives**
Source(s)	http://iot-cosmos.eu/sites/default/files/iot-cosmos/public/content-files/deliverables/
Description	At this point a malicious node can also be a part of a smaller network of malicious nodes which collude in order to increase their reputation (or decrease that of others) in the entire network. They do that firstly by signing trivial SLAs with each other and secondly by providing good feedback for each other and bad for everyone else. It should be mentioned that these malicious nodes do not necessarily demonstrate malicious service provision. The Sibyl attack and the 51% security risk fall under this category.
Mitigations	▸ Identifying trusted nodes (evaluators) and giving greater weight to their evaluation. ▸ Using both a Trust and a Reputation Index. **[Implemented]** ▸ Identifying the proper Dependability calculation formula to make the system resilient to 51%+ attacks. **[Partially implemented]**
TTP ID	**QSTR-002**
Title	**Direct Attacks to the SLA subsystem and T&R components**
Source(s)	https://attack.mitre.org/techniques/T1543/
Description	Attacks may be focused directly on the SLA Evaluator, the SLA Monitoring, and/or the T&R engine components, so as to change the evaluation results extracted by these components in favour of specific individuals.
Mitigations	▸ Common development and integration best practices to reduce the attack surface of these components. **[Implemented]** ▸ Regular cross-check of the appropriate results/models of the said components. **[Implemented]**
TTP ID	**QSTR-002**
Title	**Direct Attacks to the IaaS evaluation and the T&R Indices DBs**
Source(s)	https://www.scirp.org/journal/paperinformation.aspx?paperid=106429
Description	Attackers may directly target the final outcome of these components (evaluation metrics) and change the stored data in their favour (e.g., database tampering).
Mitigations	▸ Using tampering detection techniques and tools. ▸ Using Blockchain for the storage of (some of the) results, to acquire tamper-proof results. **[Partially implemented]**

3.2.8 UC1 subsystem

To overcome the computation limitations of Head Mounted Devices (HMDs), remote rendering is implemented to utilize the high processing ability of servers. In UC1, we use the remote rendering technology to bypass the limitations of the HoloLens2 (client) by allowing the laptop (server) to do the heavy work while streaming it back to the device. But, for this peer-to-peer connection to be established, the client and server need to first agree on a set of rules through signalling. This process comes with many threats that are detailed in the table below along with their mitigations.

Table 9: UC1 threats.

TTP ID	**UC1-001**
Title	**DDoS Attacks**
Source(s)	https://www.apriorit.com/dev-blog/578-blockchain-attack-vectors https://geth.ethereum.org/docs/whisper/whisper-overview
Description	A malicious attempt to disrupt the normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic.
Mitigations	▸ To prevent a DDoS attack in the Whisper protocol, proof-of-work (PoW) algorithm is used. **[Implemented]**
TTP ID	**UC1-002**
Title	**Spying on the HTTP Endpoint**
Source(s)	https://www.securitymetrics.com/blog/are-http-websites-insecure
Description	All the traffic generated between nodes is being sent through HTTP endpoints. The data being transmitted is not encrypted and thus anyone can listen (and pull) the sensitive data being exchanged.
Mitigations	▸ Setup an HTTPS reverse proxy so that communication only happens with that proxy which is secured through HTTPS.
TTP ID	**UC1-003**
Title	**False Friends/ Eclipsing Attack**
Source(s)	https://www.apriorit.com/dev-blog/578-blockchain-attack-vectors https://arxiv.org/pdf/1908.10141.pdf
Description	In an eclipse attack, the attacker eclipses the node from the network. The attacker makes sure that the node will not communicate with the blockchain network. The node will believe in a completely different truth than the rest of the network after the node is compromised by the attack. Researchers from Berlin University initiated an eclipse attack on the Ethereum network and managed to do it using just one or two machines. A type of an eclipse attack on the Geth blockchain. It could be done with limited resources (only 2 IP addresses with /24 subnets). This used to work on Geth v1.8.0 but it has been patched on v 1.9.0.
Mitigations	• Using a higher version of Geth. (v1.9.20 for example) **[Implemented]**
TTP ID	**UC1-004**
Title	**Sybil Attack**
Source(s)	https://www.apriorit.com/dev-blog/578-blockchain-attack-vectors

Description	A Sybil attack is arranged by assigning several identifiers to the same node. Blockchain networks have no trusted nodes, and every request is sent to a number of nodes.
Mitigations	▸ Increasing the cost of creating a new identity, requiring some type of trust for joining the network, or determining user power based on reputation. **[Implemented]**

3.2.9 UC2 subsystem

The main threats related to UC2 are three-fold:

- Radio communication related threats, the possibility of attackers performing jamming attacks and disrupting the use case's functionalities should be analysed.
- Infrastructure related threats, related to the possibility of attackers gaining unauthorised access to all infrastructure tiers (cloud, edge and radio), and executing malicious code.
- Application and data related threats related to the possibility of attackers gaining unauthorised access to sensitive data either at the infrastructure or users' on-board units.

All identified threats and mitigation actions are detailed below, following the MITRE knowledge base and the European Union Agency for Cybersecurity's good practices for security of smart cars [7].

Table 10: UC2 threats.

TTP ID	**UC2-001**
Title	**Radio communication related threats and mitigation actions** **Jamming attacks**
Source(s)	Jamming or Denial of Service, Technique T1464 - Mobile | MITRE ATT&CK®
Description	An attacker could jam radio signals to prevent wireless communication between users and infrastructure.
Mitigations	▸ This type of attack cannot be easily mitigated with preventive controls.
TTP ID	**UC2-002**
Title	**Infrastructure related threats and mitigation actions** **Infrastructure discovery**
Source(s)	Cloud Infrastructure Discovery, Technique T1580 - Enterprise | MITRE ATT&CK®
Description	An attacker may attempt to discover compute and network resources in the infrastructure, gaining knowledge about the infrastructure topology and availability of resources. This knowledge can help the attacker gain access to administrative privileges.
Mitigations	▸ Limit permissions to discover cloud infrastructure in accordance with least privilege. **[Implemented]** ▸ Limit the number of users with administrative privileges. **[Implemented]**
TTP ID	**UC2-003**
Title	**Kubernetes administration command**
Source(s)	Container Administration Command, Technique T1609 - Enterprise | MITRE ATT&CK®
Description	An attacker with access to the infrastructure may gain remote execution in containers within the Kubernetes infrastructure cluster.
Mitigations	▸ Use read-only containers where possible. ▸ Require secure port access to communicate with the APIs over TLS by disabling unauthenticated access to the Kubernetes API Server. **[Implemented]**

TTP ID	UC2-004
Title	**V2X DoS**
Source(s)	ENISA good practices for security of Smart Cars — ENISA (europa.eu), ISO - ISO/SAE FDIS 21434 - Road vehicles — Cybersecurity engineering
Description	An attacker may be able to cause V2X DSRC or MEC to crash or stop remotely by flooding the area with invalid data and causing resource exhaustion. Severity on this threat will depend on the target, being the MEC server the worst-case scenario.
Mitigations	▸ Implement data validation and shutdown communications channel to V2X ECU if flooding is detected. ▸ Potentially enter in a special safety mode (e.g.: based fallback strategy) if central ECU is malfunctioning or unavailable. ▸ Have a distributed architecture of MEC servers.
TTP ID	**UC2-005**
Title	**Application and data related threats and mitigation actions** **Network sniffing**
Source(s)	Network Sniffing, Technique T1040 - Enterprise | MITRE ATT&CK®
Description	An attacker could sniff data traffic and capture sensitive information delivered by application services at all infrastructure tiers, including application-related user data logs and authentication material such as user credentials. In addition, an attacker may also be able to create fake data logs and trigger unwanted application events.
Mitigations	▸ Use multi-factor authentication where possible. **[Implemented]** ▸ Ensure that all traffic is encrypted appropriately. ▸ Use best practices for authentication protocols, e.g. private key authentication rather than user+password authentication. **[Implemented]**
TTP ID	**UC2-006**
Title	**V2X Spoofing**
Source(s)	ENISA good practices for security of Smart Cars — ENISA (europa.eu), ISO - ISO/SAE FDIS 21434 - Road vehicles — Cybersecurity engineering
Description	An attacker could impersonate a malicious V2V connected vehicle to deliver malicious V2X updates, such as hazards on the lane when none are present or trigger critical events.
Mitigations	▸ Use a Public Key Infrastructure. **[Implemented]** ▸ Use message authentication codes. ▸ Implement data validation.
TTP ID	**UC2-007**
Title	**Environmental manipulation**
Source(s)	ENISA good practices for security of Smart Cars — ENISA (europa.eu), ISO - ISO/SAE FDIS 21434 - Road vehicles — Cybersecurity engineering
Description	An attacker might be able to manipulate environmental data to reproduce false Barcelona Trams during the video processing phase, triggering fake alarms over the area to vulnerable users.
Mitigations	▸ Combine video-processing solutions with trusted V2X data. ▸ Work with additional sensors in case a V2X solution is not available in the Tram.

TTP ID	UC2-008
Title	**K8s Application Vulnerability**
Source(s)	Protecting Kubernetes Against MITRE ATT&CK: Initial Access
Description	Containerized applications that are public-facing can allow initial attacker access, and if these containers have vulnerabilities, they leave organizations susceptible to exploits. For example, a remote code execution (RCE) vulnerability could be exploited and since service account credentials are mounted to containers by default in Kubernetes, these credentials could then be used to make requests to the Kubernetes API server. Besides, by default all pods in a namespace can communicate with all other pods in the same one. A compromised pod may allow an attacker to reach other applications, access the kubelet read-only API server, access sensitive data stored on cloud provider instance metadata servers, cause a denial of service attack, or other malicious activities.
Mitigations	▸ Network policies limit the cluster communication. A deny-all policy should be combined with egress & ingress rules specifying which container communicates with which, on which port and with which protocol. ▸ Additionally, service account permissions should be restricted using Kubernetes RBAC according to the principle of least privilege. **[Implemented]**
TTP ID	**UC2-009**
Title	**K8s Node Resource Starvation – DoS**
Source(s)	K8s Resource Quotas (K8s official doc.) XI Commandments of K8s security (Publication)
Description	As pods by default consume the resources that they need on the fly (CPU, network and disk I/O), if a pod mistakenly or deliberately uses far more resources than it needs, this could lead to node's starvation, affecting the performance of the node and the running pods. This would also prevent the scheduler from being able to instantiate pods on this node, potentially affecting the offloading process (contemplated in the Use Case 2), as the node would not have enough resources to accept new pods.
Mitigations	▸ A K8s resource quota, defined by a ResourceQuota object, provides constraints that limit aggregate resource consumption per namespace. It can limit the quantity of objects that can be created in a namespace by type, as well as the total amount of compute resources that may be consumed by resources in that namespace. **[Implemented]**

3.2.10 UC3 subsystem

The UC3 subsystem consists of a microservice architecture based on Docker containers. The central interface is a message broker (i.e., an instance of RabbitMQ) that receives the data from the machine and all analytics applications consume the corresponding data from it. Database systems, e.g., MongoDB[3] and InfluxDB are also used. The message broker represents a weak point if it were to be attacked. The Advanced Message Queuing Protocol (AMQP) and MongoDB[4] - two central technologies in this UC - are used for the threat analysis.

[3] MongoDB: https://www.mongodb.com/ (Accessed August 2022).
[4] InfluxDB: https://www.influxdata.com/ (Accessed August 2022).

3.2.10.1 RabbitMQ – AMQP

RabbitMQ[5] is an open-source message broker software that implemented the Advanced Message Queuing Protocol (AMQP) and other protocols. AMQP is an open standard for passing business messages between applications or organizations. Vulnerabilities of AMQP mainly involve the broker component and affect processes, like access control, message and identity validation and message queue management. Possible attacks are described in the following table. MongoDB is a document-based, distributed NoSQL database. In UC3 it is used to store documents with analytics results. The following table collects the possible attacks on Rabbit MQ as well as MongoDB.

Table 11: UC3 threats.

TTP ID	**UC3-001**
Title	**Man-In-The-Middle Attack – Manipulation of transmitted data**
Source(s)	https://attack.mitre.org/techniques/T1557/
Description	An attacker could position himself/herself between the data producer and the message broker as well as message broker and consumer to manipulate the transmitted data. This would lead to falsified data, which could lead to incorrect results in the evaluation. In the worst case, this could be used to manipulate a deployed machine learning algorithm, which would issue a false alarm and thus cause an unwanted machine shutdown.
Mitigations	▸ Ensure that all wired and/or wireless traffic is encrypted appropriately. Use best practices for authentication protocols, such as Kerberos, and ensure web traffic that may contain credentials is protected by SSL/TLS. **[Implemented]** ▸ Limit access to network infrastructure and resources that can be used to reshape traffic or otherwise produce MiTM conditions. ▸ Network segmentation can be used to isolate infrastructure components that do not require broad network access. This may mitigate, or at least alleviate, the scope of MiTM activity. ▸ Network intrusion detection and prevention systems that can identify traffic patterns indicative of MiTM activity can be used to mitigate activity at the network level.
TTP ID	**UC3-002**
Title	**Denial of Service Attack**
Source(s)	CVE-2021-22116: http://cve.mitre.org/cgi-bin/cvename.cgi?name=2021-22116 Security vulnerabilities of NoSQL and SQL Databases scientific publication [8]
Description	An attacker could send malicious AMQP messages to the target RabbitMQ instance having the AMQP 1.0 plug-in enabled. In doing so, he/she can exploit an improper input validation in the AMQP 1.0 client connection endpoint. This would lead to unavailability of the service and no data can be processed and analysed.
Mitigations	▸ Network intrusion detection and prevention systems that can identify traffic patterns indicative of DoS activity can be used to mitigate activity at the network level.
TTP ID	**UC3-003**
Title	**Valid Accounts**
Source(s)	https://attack.mitre.org/techniques/T1078/ Security vulnerabilities of NoSQL and SQL Databases scientific publication [8]

[5] RabbitMQ: https://www.rabbitmq.com/ (Accessed August 2022).

Description	An attacker could obtain credentials of existing accounts and use them to get increased rights.
Mitigations	▸ Ensure that applications do not store sensitive data or credentials insecurely. (e.g., plaintext credentials in code, published credentials in repositories, or credentials in public cloud storage). [Implemented] ▸ Applications and appliances that utilize default username and password should be changed immediately after the installation, and before deployment to a production environment. When possible, applications that use SSH keys should be updated periodically and properly secured. ▸ Audit domain and local accounts as well as their permission levels routinely to look for situations that could allow an adversary to gain wide access by obtaining credentials of a privileged account.
TTP ID	**UC3-004**
Title	**Cross Site Scripting (XSS) Attacks**
Source(s)	https://blog.shiftleft.io/mitigating-nosql-injection-attacks-part-2-42a2d8890f70
Description	An attacker could steal sensitive information from the database by inserting arbitrary JavaScript code to bypass security authentication. This leads to loss of integrity.
Mitigations	▸ Sanitize and validate user input: Allow and expect matches what the user provided ▸ Use prepared statements to secure the data ▸ Avoid using JavaScript functions to parse user input: eval(), setTimeout(), setInterval(), or Function() to parse user-provided input **[Implemented]**
TTP ID	**UC3-005**
Title	**Query injection attacks**
Source(s)	https://blog.shiftleft.io/mitigating-nosql-injection-attacks-part-2-42a2d8890f70
Description	MongoDB permits unserialized JSON and JavaScript expressions in several alternative query parameters. As JavaScript is a fully featured language, an attacker could manipulate data and run arbitrary code. This would lead to falsified data and the user might take wrong decisions based on them.
Mitigations	▸ Sanitize and validate user input: Allow and expect matches what the user provided ▸ Use prepared statements to secure the data ▸ Avoid using JavaScript functions to parse user input: eval(), setTimeout(), setInterval(), or Function() to parse user-provided input **[Implemented]**

3.2.11 Other underlying cloud infrastructure

Regarding the underlying cloud infrastructure that hosts the Pledger components, additional data were gathered by placing honeypot instances on the INTRA cloud infrastructure. The honeypot consists of an Intrusion Detection System integrated with monitoring and logging tools (more information follows on Section 0) that documented threats to a public-facing service. Figure 9 illustrates some basic statistics for a 7-day period. Almost 150K logs were acquired and analysed, more than 1000 potential threats were detected, raising 66 warnings and 11 critical alerts.

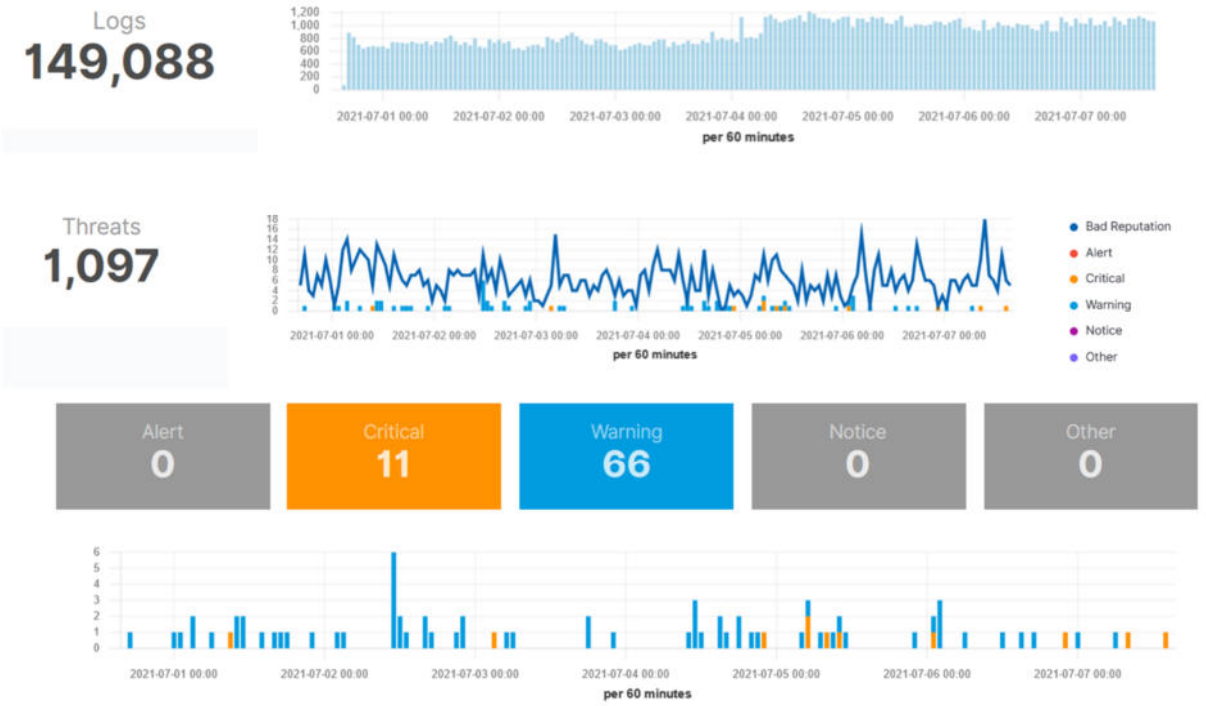


Figure 9: Basic honeypot statistics for a 7-day period.

Table 12: Honeypot alert logs for a 7-day period.

Signature	Signature ID	# Alerts
SURICATA TCPv4 invalid checksum	2200074	113
SURICATA TLS invalid handshake message	2230003	22
SURICATA TLS invalid record/traffic	2230010	22
ET DOS Possible NTP DDoS Inbound Frequent Un-Authed MON_LIST Requests IMPL 0x03	2017919	16
SURICATA STREAM Packet with invalid ack	2210045	16
SURICATA STREAM SHUTDOWN RST invalid ack	2210046	16
ET DROP DShield Block Listed Source group 1	2402000	7
SURICATA STREAM 4-way handshake SYNACK with wrong SYN	2210012	4
SURICATA STREAM Packet with invalid timestamp	2210044	3
SURICATA STREAM 3way handshake SYNACK in wrong direction	2210003	2
ET CINS Active Threat Intelligence Poor Reputation IP group 63	2403362	1
ET CINS Active Threat Intelligence Poor Reputation IP group 78	2403377	1
ET CINS Active Threat Intelligence Poor Reputation IP group 94	2403393	1
ET DOS Possible NTP DDoS Inbound Frequent Un-Authed MON_LIST Requests IMPL 0x02	2017918	1
SURICATA TCP option invalid length	2200036	1

Table 12 includes the analysis of the threats captured from the Pledger honeypot, as well as other known threats against cloud infrastructure in bibliography. Table 13 collects the most significant threat signatures recorded by the IDPS up to August 2022.

Table 13: Threats against the underlying Pledger infrastructure.

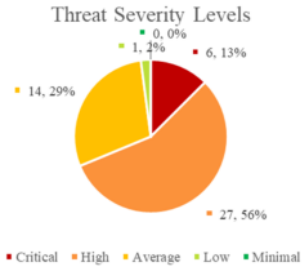
TTP ID	INFRA-001
Title	**(Distributed) Denial of Service**
Source(s)	MITRE, Pledger Honeypot
Description	Denial of Service is a type of attack that aims to exhaust resources or exploit protocol vulnerabilities to crash or slow down a victim server so that it denies access to real users. Within the Pledger Honeypot, there were also recorded Denial-of-Service attacks exploiting the NTP synchronisation protocol (in particular, the monlist mechanism) to deploy an amplification-type attack. Furthermore, multiple network-based DoS attacks through malformed packets were detected.
Mitigations	▸ Firewall setup: Firewall rules have been implemented to allow the connection of IPs only from trusted nodes. **[Implemented]** ▸ Blacklist policies: Refusing entry of traffic from IP addresses that have been blacklisted. **[Implemented]**
TTP ID	**INFRA-002**
Title	**Traffic from low-reputation IPs**
Source(s)	Pledger Honeypot
Description	DShield is a community-based collaborative firewall log correlation system. It receives logs from volunteers worldwide and uses them to analyse attack trends. It is used as the data collection engine behind the SANS Internet Storm Center (ISC). Low-reputation IPs can be identified and proactively blocked.
Mitigations	▸ Firewall setup: Firewall rules have been implemented in order to allow the connection of IPs only from trusted nodes. **[Implemented]** ▸ Blacklist policies: Refusing entry of traffic from low-reputation IP addresses that have been blacklisted. **[Implemented]**
TTP ID	**INFRA-003**
Title	**RST Injection/DoS**
Source(s)	Pledger Honeypot CAPEC MITRE: https://capec.mitre.org/data/definitions/596.html
Description	An adversary injects one or more TCP RST packets to a target after the target has made a HTTP GET request. The goal of this attack is to have the target and/or destination web server terminate the TCP connection.
Mitigations	▸ Proper configuration of routers **[Implemented]** ▸ Inclusion of Intrusion Detection and Prevention system **[Implemented]**

3.3 Threat & countermeasure prioritisation

According to the previous analysis, the following prioritization was completed by scoring the selected threats and sanitizing the inputs to provide a list of unique entries. Overall, a total of 48 unique threats were assessed, 6 of which are considered critical, while 27 are considered highly critical (Figure 10).

Severity Level		
Critical	**6**	**12.50%**
High	**27**	**56.25%**
Average	**14**	**29.17%**
Low	**1**	**2.08%**
Minimal	**0**	**0.00%**
SUM	**48**	**100.00%**

(a)



(b)

Figure 10: Threat Severity basic statistics.

The following table provides: (a) the final results of the TTP prioritization, and (b) the selected countermeasures and mitigations, after ranking them according to utility and cost, and marking the ones applied at the time of the project's first and second releases. The full countermeasures table, which assigns utility score per threat is provided in Annex A.

Table 14: (a) Threat prioritisation, (b) Countermeasure prioritisation.

TTP ID	Source	TTP	Threat Severity Level	Max TTP Score	Orchestration	Configuration &Benchmarking	SLA	Decision Support System	Big Data	Blockchain	CI/CD	UC1 Subsystem	UC2 Subsystem	UC3 Subsystem	Other infrastructure
TH001	MITRE, OWASP, CVE	Sensitive Data Exposure	**Critical**	**4.2**	3.9	4.2			4						4
TH002	MITRE	Account Manipulation	**Critical**	**4.1**				4.1							
TH003	MITRE	Process Injection	**Critical**	**4**		4									
TH004	MITRE	User Execution	**Critical**	**4**		4									
TH005	ThreatPost, Honeypot	(Distributed) Denial of Service Attack	**Critical**	**4**					4	2.9	3.4	2.9		3	3.5
TH006	MITRE, OWASP	Remote Code Execution	**Critical**	**4**		4					4				
TH007	ENISA	Insecure application API	**High**	**3.9**					3.9						
TH008	MITRE	Modify System Image	**High**	**3.8**							3.8				
TH009	MITRE	Kubernetes administration command	**High**	**3.8**									3.8		
TH010	Articles/Bibliography	Orchestrator risks	**High**	**3.8**	3.8										
TH011	Articles/Bibliography	Network related threats	**High**	**3.7**	3.7										
TH012	MITRE	Create or Modify System Process	**High**	**3.6**							3.6				
TH013	MITRE	Exploitation for Privilege Escalation	**High**	**3.6**		3.6									
TH014	MITRE	Escape to Host	**High**	**3.6**		3.6									
TH015	MITRE	Root SSH brute force attack	**High**	**3.6**					3.6		3.5				
TH016	Articles/Bibliography	Container risks	**High**	**3.6**	3.6										
TH017	Articles/Bibliography	Malicious collectives	**High**	**3.6**			3.6								
TH018	MITRE	Cloud Service Dashboard	**High**	**3.5**				3.5							
TH019	MITRE	Unsecured Credentials	**High**	**3.5**		3.5									
TH020	Pledger Honeypot	RST Injection	**High**	**3.5**											3.5
TH021	MITRE	Valid Accounts	**High**	**3.4**				3.4						2.3	
TH022	Scientific Publication	Sybil Attack	**High**	**3.4**						3.4		3.4			
TH023	Articles/Bibliography	Container image vulnerabilities	**High**	**3.4**	3.4										
TH024	Articles/Bibliography	Threats to SLA subsystem/T&R components	**High**	**3.4**			3.4								

TTP ID	Source	TTP	Threat Severity Level	Max TTP Score	Orchestration	Configuration &Benchmarking	SLA	Decision Support System	Big Data	Blockchain	CI/CD	UC1 Subsystem	UC2 Subsystem	UC3 Subsystem	Other infrastructure
TH025	Pledger Honeypot	Traffic from Bad Reputation IPs	**High**	**3.4**											3.4
TH026	MITRE, Pledger Honeypot	Cloud Infrastructure Discovery	**High**	**3.3**				3.3							
TH027	ATT&CK	Manipulation of transmitted data	**High**	**3.3**										3.3	
TH028	Articles/Bibliography	Threats to IaaS evaluation/T&R Indexes DBs	**High**	**3.3**			3.3								
TH029	Openshift, MITRE	K8s Application Vulnerability	**High**	**3.2**									3.2		
TH030	Scientific Publication	False Friends / Eclipsing Attack	**High**	**3.1**						3.1		3.1			
TH031	ENISA, ISO 21434	V2X DoS	**High**	**3.1**									3.1		
TH032	MITRE, Honeypot, ENISA	Network Sniffing/Eavesdropping	**High**	**3**		3		3	3						
TH033	MITRE	Man-in-the-middle attack	**High**	**3**					3						
TH034	MITRE	Data from Local System	**Average**	**2.9**		2.9									
TH035	MITRE, Pledger Honeypot	Infrastructure discovery	**Average**	**2.9**									2.9		
TH036	Articles/Bibliography	Access control exploits	**Average**	**2.9**	2.9										
TH037	Scientific Publication	Query injection attacks	**Average**	**2.8**										2.8	
TH038	K8s official doc	K8s Node Resource Starvation/DoS	**Average**	**2.8**									2.8		
TH039	Articles/Bibliography	Container and Resource Discovery	**Average**	**2.8**	2.8										
TH040	Articles/Bibliography	Registry risks	**Average**	**2.8**	2.8										
TH041	Scientific Publication	XSS attacks	**Average**	**2.6**										2.6	
TH042	MITRE	Jamming attacks	**Average**	**2.6**									2.6		
TH043	ENISA, ISO 21434	Environmental Manipulation	**Average**	**2.6**									2.6		
TH044	Articles/Bibliography	Oscillating malicious service provision	**Average**	**2.5**			2.5								
TH045	Articles/Bibliography	Malicious a posteriori evaluation	**Average**	**2.5**			2.5								
TH046	ENISA, ISO 21434	V2X Spoofing	**Average**	**2.4**									2.4		
TH047	ThreatPost	Decryption Key Leaked	**Average**	**2.3**								2.3			
TH048	ThreatPost	Spying on the Endpoint	**Low**	**1.9**								1.9			

Countermeasure ID (R1)	Countermeasure ID (R2)	Countermeasure	Utility/Cost	Cost	Utility
C01	C01	Privileged Account Management	19.1	1.05	20
C02	C02	Intrusion Detection and Prevention System	16.67	2.1	35
C03	C03	Encrypt sensitive information in transit (SSL/TLS)	14.96	1.15	17.2
C04	C04	Firewall setup/Blacklist policies	14	2	28
C05	C05	Enforce secure password policies	11.67	1.2	14
C06	C06	Application Isolation and Sandboxing	9.68	1.9	18.4
C07	C07	Access Control Lists/ Authorisation	8.28	2.9	24
C08	C08	Least privilege access model	7.86	1.4	11
C09	C09	Avoid JavaScript functions to parse user input	7.83	1.15	9
C10	C10	Adjust container security policies	7.33	2.4	17.6
C11	C11	Execution Prevention	7.17	2.65	19
C12	C12	Multifactor authentication	6.61	2.3	15.2
C13	C13	Recent version of Geth	6.5	1.6	10.4
C14	C14	Use trusted network	6.11	1.8	11
C15	C15	Code Signing	5.86	2.9	17
C16	C16	User account management	5.83	2.4	14
C17	C17	Pulling images only from trusted sources	5.83	1.2	7
C18	C18	Encrypt Sensitive Information at Rest	5.22	2.3	12
C19	C19	Software Upgrade to higher version	5	0.8	4
C20	C20	Monitoring and remediating container runtimes	4.79	3.3	15.8
C21	C21	Sanitize and validate user input	4.57	1.05	4.8
C22	C22	Update Software	4.36	1.65	7.2
C23	C23	Remove sensitive data from responses	3.6	2	7.2

Countermeasure ID (R1)	Countermeasure ID (R2)	Countermeasure	Utility/Cost	Cost	Utility
C24	C24	HTTPS reverse proxy	3.33	1.8	6
C25	C25	Prepared statements	3.31	1.45	4.8
C26	C26	Network anomalies detection	3.29	3.4	11.2
C27	C27	Limit access to resource over network	3.26	2.15	7
C28	C28	Private key management	3	1	3
C29	C29	Container image vulnerabilities detection	2.84	3.1	8.8
C30	C30	Container image vulnerabilities management tools	2.82	3.4	9.6
C31	C31	Encrypt Sensitive Information	2.77	2.6	7.2
C32	C32	Decoupled T&R Indices	2.61	2.3	6
C33	C33	Discrete virtual networks and isolated applications	2.55	2.75	7
C34	C34	Limit software installation	2.38	2.1	5
C35	C35	Attack surface reduction	2.33	2.7	6.3
C36	C36	Using Blockhain for logs storage	2	2.6	5.2
C37	C37	RF Hardening	1.8	5	9
C38	C38	Configuration only to registries over encrypted chai	1.58	1.9	3
C39	C39	Tampering detection techniques and tools	1.52	2.9	4.4
C40	C40	Network Segmentation	1.25	2.4	3
C41	C41	Assignment of trusted nodes	1.21	2.65	3.2
C42	C42	Weights, time-windows, fading factors for calculati	1.07	2.8	3
C43	C43	Testing of proper Dependability calculation formula	1.07	3	3.2
C44	C44	Evaluators evaluation	0.97	3.1	3
C45	C45	Regular cross-check of the appropriate results/mod	0.96	2.5	2.4

Already deployed

Considered for the future (High to medium priority)

Considered for the future (Low priority)

Not deployed

4 Architecture of privacy, security and trust components

4.1 High level architecture

In addition to the planned configurations and the development of the security subsystem in T4.1, Pledger selected a subset of infrastructure hardening countermeasures, and added a layer of security defences ranging from traditional perimeter defences to security scans, code analysis etc. A third layer deals with the creation of data gathering, indexing and visualisation tools, that allows access to the generated threat information. Figure 11 provides an overview of the work performed to secure the Pledger Core and Use Cases.

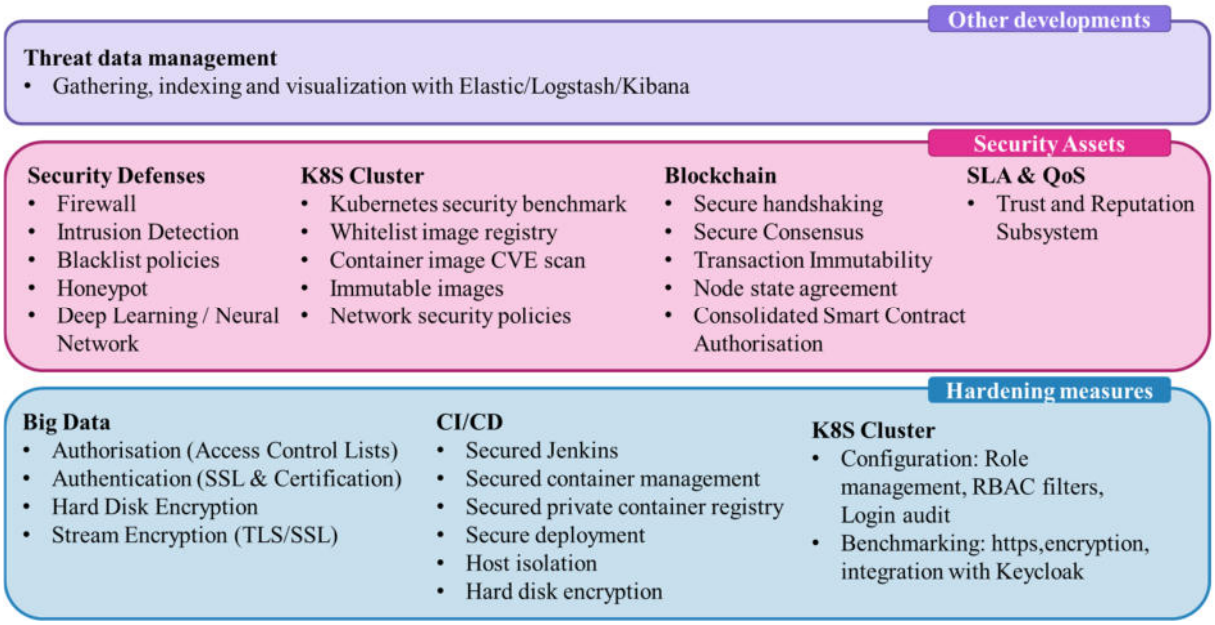


Figure 11: Pledger Security, Privacy and Trust tools and measures.

4.2 Infrastructure hardening

The current section provides an overview of the hardening measures that were integrated in the different infrastructure blocks of Pledger. Compared to the preliminary version D4.1 [1], integration updates of the second project period, as well as more technical details of the followed approaches are provided here for:

- The K8s cluster: CVE scanning, CIS benchmarking and data aggregator model for the security assessment of the underlying infrastructures (section 4.2.3).
- Blockchain security over the Whisper protocol: describing the integrated mechanisms to achieve data integrity, anonymity, DDoS and MitM attacks prevention (section 4.2.4).

4.2.1 Streamhandler big data platform

In a standard Kafka setup[6], any user or application can write any messages to any topic, as well as read data from any topics. As Streamhandler platform will be used from different teams and different software components it is critical to protect confidential information and implement security to the cluster. StreamHandler applies three layers of security:

- **Encryption of data (in-flight) using Secure Sockets Layer/Transport Layer Security protocols (SSL/TLS)**: The data are encrypted between Brokers and Clients by utilizing SSL(TLS). This is achieved by providing each Broker and Client of the cluster with their own encryption key.
- **Authentication using SSL**: Only clients from trusted IPs can be authenticated to the cluster. Brokers and clients are configured with a truststore to distinguish allowed certificates and verify their identity. These certificates are issued to the clients and signed by the certificate authority (CA) of the cluster.
- **Authorization using Access Control Lists (ACLs)**: Once the clients are authenticated, Kafka brokers can run against access control lists (ACL) to determine whether a particular client can be authorized to read to some topic.

4.2.2 CI/CD

A set of security features has been considered and incorporated in the CI/CD infrastructure to protect both the CI/CD infrastructure and services and the deployed project's artifacts. The following paragraphs described in detail these provided security assets.

4.2.2.1 Encryption - Secure communication over Hypertext Transfer Protocol Secure (HTTPS)/Transport Layer Security (TLS)

Access to the offered services is secured with HTTPS to protect the connections of the users to the deployed applications. Specifically, the continuous integration (Jenkins), the management and the artifacts repository (JFrog Container Registry) have been secured with HTTPS, allowing a secure experience from a client's side. Data sent using HTTPS is secured via Transport Layer Security protocol (TLS), which provides three key layers of protection:

- Encryption - encrypting the exchanged data to keep it secure from eavesdroppers. That means that while users are using an HTTPS secured service, nobody can "listen" to their conversations, track their activities across multiple pages, or steal their information.
- Data integrity - data cannot be modified or corrupted during transfer, intentionally or otherwise, without being detected.
- Authentication - proves that users communicate and send data to the intended service. It protects against man-in-the-middle (MitM) attacks and builds user trust.

The OpenSSL library that provides an open-source implementation of the TLS protocol has been used to generate the private keys, certificate signing requests, SSL certificates (self-signed or Certificate Authority (CA)-signed) and for certificate format conversion.

Additionally, the connection to the servers that will be used for the deployment of the project's artifacts, including the testing, staging and production environments, has been also secured with HTTPS. Access to the Docker daemon socket is protected by enabling TLS, allowing Docker to be reachable through the network in a safe manner. From the server side, connections from clients authenticated by a certificate (self-signed or signed by a CA) are allowed to the servers in which the project's artifacts will be deployed. From the client side, clients can only connect to these servers with a certificate, that can again be self-signed or signed by a CA.

[6] Details on the platform and the data integration capabilities are provided within Pledger deliverable D5.2.

4.2.2.2 Hard disk encryption at rest of the CI/CD infrastructure servers

All hard disks mounted to the servers (virtual machines) that are used for the deployment of the Pledger CI/CD solution are encrypted. Disk encryption at rest ensures that files are always stored on disk in an encrypted form. The files only become available to the operating system and applications in readable form while the system is running and unlocked by a trusted user. An unauthorized person looking at the disk contents directly, will only find garbled random-looking data instead of the actual files, securing the actual data stored in cases that the hard disk or server is lost, stolen or discarded after its end-of-life.

4.2.2.3 User authentication for the CI/CD services

The services offered in the Pledger CI/CD solution, such as the source code repository (Gitlab), the continuous integration (Jenkins) and the artifacts repository (Artifactory) are secured using user authentication. The integration of an existing Lightweight Directory Access Protocol (LDAP) server or Active Directory implementation is possible.

4.2.2.4 Firewall protection of the CI/CD and Development & Testing Environment infrastructure

The security of the infrastructure used by the CI/CD platform and Development & Testing Environment is ensured by proper firewalling, controlling the allowed connections to the servers used for the deployment of services and project's artifacts. Specifically, one of the most popular and flexible open-source Linux firewall software is used for this purpose, iptables. A set of iptables rules has been defined and configured to the different servers to block some of the common network attacks (Syncronization (SYN) flood attacks, smurf attacks, land attacks, attacks by malfunctioning Internet Control Message Protocol (ICMP) packets and other forms of Denial of Service (DoS attacks)). The default policy is to drop incoming, outgoing or forwarded packets from any source to any destination, unless there is a specific rule set to allow a particular communication. Examples of firewall configurations include:

- Blocking null packets of scanning bots that look for security holes to exploit.
- Prevent syn-flood attack that can take up the servers resources.
- Drop remote packets that are claiming to be from a loopback address.
- Prevent ping flooding - up to 6 pings per second from a single source, with log limiting. Also prevents us from ICMP REPLY flooding some victim when replying to ICMP ECHO from a spoofed source.
- Drop invalid packets.
- Do not log packets that are going to ports used by SMB (Samba / Windows Sharing).
- Do not log late replies from nameservers.
- Explicitly reject AUTH traffic so that it fails fast.
- Prevent DOS by filling log files.
- Allow incoming and outgoing traffic on new, established and related connections only from trusted sources.

4.2.2.5 SSH key-based authentication to the infrastructure

SSH access to administer and manage the CI/CD platform servers has been configured to use only key-based authentication. This is a more secure alternative in comparison to the most commonly used password authentication. Although passwords are sent to the server in a secure manner, they are sometimes not complex or long enough to be resistant to repeated, persistent attackers. Modern processing power combined with automated scripts make brute forcing a password-protected account very possible. Thus, SSH public key authentication in which we generate and store a pair of cryptographic keys and then configure the servers to recognize and accept the generated keys is a more secure approach.

4.2.3 K8S cluster

Within "Cloud Service Discovery" threat category, MITRE identifies all the possible threats that an infrastructure may suffer. An attacker might exploit the infrastructure to get unauthorized access to both infrastructure and running services or execute malicious code. Such threat cannot be easily mitigated and is heavily dependent on the infrastructure used in the project.

About the Pledger infrastructure provided by ENG, it is based on Kubernetes hosted on virtual machines provided by OpenStack. OpenStack and Kubernetes have dedicated working groups and provide main guidelines about security activities to harden their infrastructure; in the project, we did not go over an exhaustive check of the threats, as potentially very long, but we selected those we considered more critical based on following list. For OpenStack [9], the security activities adopted are based on the official security guidelines:

1. All VMs are accessible only to administrator through private keys stored outside the Pledger project's boundaries.
2. Services published by Kubernetes on OpenStack are controlled by SecurityGroups that filter the ingress/egress traffic to only those allowed.
3. Services published by Kubernetes are accessible from remote infrastructure only if connected using a VPN.
4. VPN accounts are managed on a separate Virtual Machine, unreachable by Pledger project.
5. A different VPN account is provided for each infrastructure, have limited validity and can be revoked anytime with a single line of code.

For Kubernetes, the planned security activities are based on the official security guidelines identified by the CNCF Certified Kubernetes Security Specialist in order to secure the whole container supply chain: the hosting infrastructure, the container images built for the project, the opensource binaries from public repositories, the restriction to work on trusted registries, the audit to track the activities performed.

By the **end of first period**, we identified the following activities to secure Pledger and ENG infrastructure at different levels:

1. The use of a CIS benchmark to identify the major threats of the Kubernetes infrastructure; this is specific for Kubernetes and helps to keep track of the major vulnerabilities and also suggest possible countermeasures.
2. The reduction of the image surface of attack, by using best practice to limit the surface of attack of running containers to only the strictly necessary code and best practice to produce immutable containers that cannot be used to launch/host malicious code. This affects the image packaging process and aims at reducing the possibility for an attacker to install malicious code on a given image.
3. The scanning of images for known vulnerabilities based on CVEs database. This is feature that should allow to track the vulnerabilities of opensource container images and help to choose those with less critical ones.
4. The whitelist image registries, sign and validate images to filter out images that are not considered safe, while allowing those that are subject to periodic security. This should enforce the usage of trusted registries.
5. The configuration of secure Pod-to-Pod communication and allow traffic only within Pods that have explicit requirements to communicate with each other, using PodSecurityPolicies [10]. This should limit traffic also within Kubernetes infrastructure, with an additional level of filtering that limits the damages in case of security breaches.
6. The configuration of Audit Log to track Kubernetes main activities. It is meant to facilitate further analysis in case of security breach.

In the **second project** period, we decided to **focus** on the scanning of **CVE** vulnerabilities and on **CIS** benchmarking to address the **development** stage and the **runtime** stage with the goal to reduce the surface of attack of the running containers.

With **CVE** scanning, we integrated Kubei[7], an existing opensource tool and provided a UI to developers to better track the opensource Docker images potentially used in their application to help them choose those with equivalent functionalities and less security flows.

Then, with **CIS** benchmarking, we integrated kube-bench[8] and docker-bench-security[9] opensource tools through an agent to be run on the different infrastructures to collect CIS benchmarks and a service to aggregate them and visualize along with the different level of security implemented.
The high-level schema of this **CVE/CIS data aggregator** is shown in Figure 12. A demo has been published on Pledger YouTube channel[10] and is described with more details in section 5.3.1.

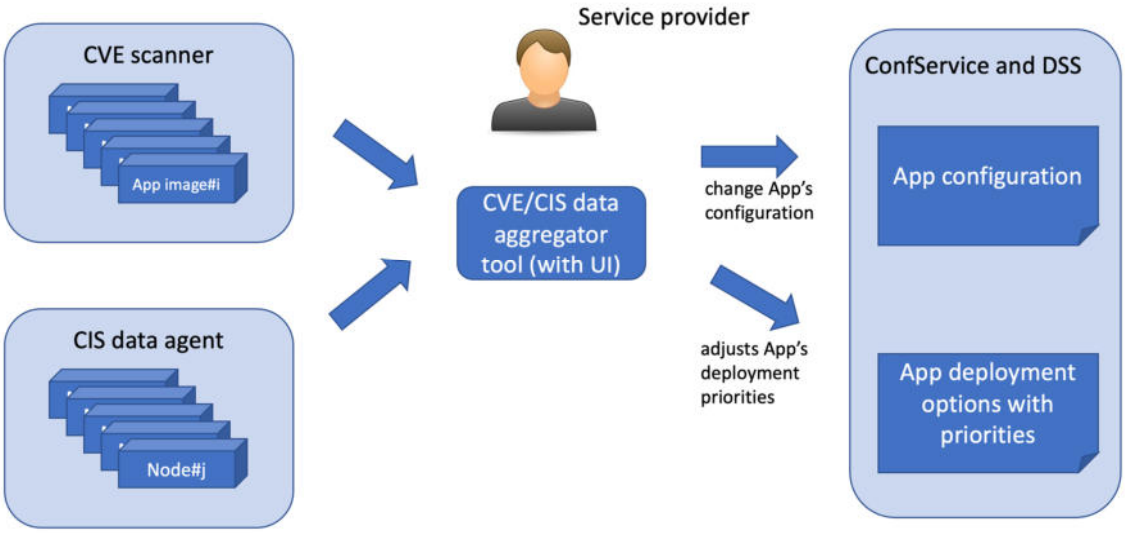


Figure 12: CVE/CIS data aggregator schema

In the end, the **goal** of the CVE/CIS data aggregator is to provide a simple interface where to check the security assessment of the infrastructures configured in Pledger using common set of best practices and their severities used as a **reference**.

As a **result**, the CVE/CIS data aggregator, along with the T&R, should help the service providers to be more aware of the infrastructures capabilities they might want to use from security and trust perspective when deploying DApps, so that they can adjust the priorities used by the DSS when taking a decision on the infrastructure where to offload computation.
To complement the work about infrastructure hardening, additional activities have been taken on ENG infrastructure because it hosts Pledger control plane. Security policies and audit rules have been applied to restrict network traffic only among authorized Pods and to track critical activities such as access to Kubernetes secrets. Pod Security Policies[11] have been applied to limit traffic within the DSS containers (namely, "confservice" and "confservice-mysql" Pods), as well as Audit Logs[12] to improve the DSS security at infrastructure level.
On the other hand, we preferred to postpone the whitelist of the internal Docker registry, as well as the usage of immutable images to allow a degree of freedom up to the very end of the project to simplify the configuration (and debugging) for the final UC validation stage.

[7] https://github.com/openclarity/kubeclarity (Accessed August 2022).
[8] https://github.com/aquasecurity/kube-bench (Accessed August 2022).
[9] https://github.com/docker/docker-bench-security (Accessed August 2022).
[10] https://www.youtube.com/watch?v=OtD3_Qg-xGU (Accessed August 2022).
[11] https://kubernetes.io/docs/concepts/security/pod-security-policy/ (Accessed August 2022).
[12] https://kubernetes.io/docs/tasks/debug/debug-cluster/audit/ (Accessed August 2022).

So, at the time of writing this deliverable (M33), it is still possible to reference an external Docker registry image and to access Pods using bash shell, for easy operations. Nevertheless, these (very easy) actions will be applied by M36.

The Docker registry whitelist will require the configuration of an Admission Controller[13] whereas immutable image will be produced following well-known security best practices[14], including multistage-builds[15] and the removal of shell binaries.

4.2.4 Blockchain security through the Whisper protocol

Whisper is a pure identity-based messaging system and protocol with a simple low-level API. The protocol allows peer-to-peer (P2P) communication between the nodes of the Whisper network using the underlying ÐΞVp2p[16] Wire Protocol (i.e., RLPx Protocol). In addition, Whisper provides two types of encryptions, symmetric and asymmetric. Symmetric encryption creates a key that is used for both encrypting and decrypting messages. While Asymmetric encryption creates a public and private key pair that are used to decrypt each other. It is important to note that nodes could generate multiple symmetric and asymmetric keys. Additionally, multiple nodes could have the same symmetric key. However, asymmetric keys are unique to each node.

The Whisper protocol employs neighbour forwarding and topic filtering techniques to exchange messages within the network. This means that each node that receives an encrypted message (symmetric or asymmetric) primarily reads the message's topic. If the topics match its interests it attempts to decrypt the message, otherwise, the node forwards it to its neighbours. When attempting to decrypt messages, nodes use all their available keys. Regardless of decryption success, nodes forward the message to their neighbouring nodes.

4.2.4.1 Data integrity

When exchanging messages, nodes insert their data into Whisper messages and encrypt using symmetric or asymmetric keys. While the message is circulating the network, the content can only be accessed by authorised nodes (i.e., node with adequate decryption keys). Otherwise, nodes without proper decryption key act as routing nodes whose purpose is to forward the message to neighbouring nodes. Therefore, these nodes have no access to the contents of the message. Consequently, they are unable to change the data within. This in turn, guarantees data integrity within the chain.

4.2.4.2 Anonymity

Users sending messages with the Whisper protocol are hiding behind public keys that they use to identify within the network. In addition, with the neighbour forwarding mechanism, nodes without the adequate decryption key don't acquire information about the initial sender and eventual receiver. Therefore, this method of message exchange guarantees the anonymity of sender and receiver.

4.2.4.3 DDoS attacks prevention

When creating Whisper messages, nodes are expected to perform a certain amount of work to reach an agreed upon PoW Target. The higher the value of the target, the longer it takes for the message to be created. Nodes in the network rate the other peers on how well they respect the PoW Target and assign a trust value on them respectively.

In DDoS attacks, malicious nodes rapidly send messages to flood the network. However, this causes each created message to have unsatisfactory PoW. Consequently, the nodes receiving these messages reduce the trust value for these malicious nodes until they are completely dropped. This method

[13] https://sysdig.com/blog/kubernetes-admission-controllers/ (Accessed August 2022).
[14] https://sysdig.com/blog/dockerfile-best-practices/ (Accessed August 2022).
[15] https://docs.docker.com/develop/develop-images/multistage-build/ (Accessed August 2022).
[16] Pronounced devp2p.

eventually ensures that malicious nodes get dropped by the rest of the network, leading to the prevention of DDoS attacks.

4.2.4.4 MitM attacks prevention

During signaling, a third-party user, known as a hacker, usually get access to information exchanges between the peers that allows him to spy on the P2P direct channel communication. This type of attacks is called a Man in the Middle (MitM) attack.

The Whisper solution prevents these kinds of attacks even in the case of 100% - 2 case (i.e., all the nodes except the ones occupied by the communicating peers are jeopardized). Signaling messages are encrypted using the recipient's public key. This means that this message can only be decrypted by the recipient. Therefore, all the other malicious nodes that try to get access to the contents will fail and forward the message to the neighbors until it eventually safely reaches the recipient as shown by Figure 13.

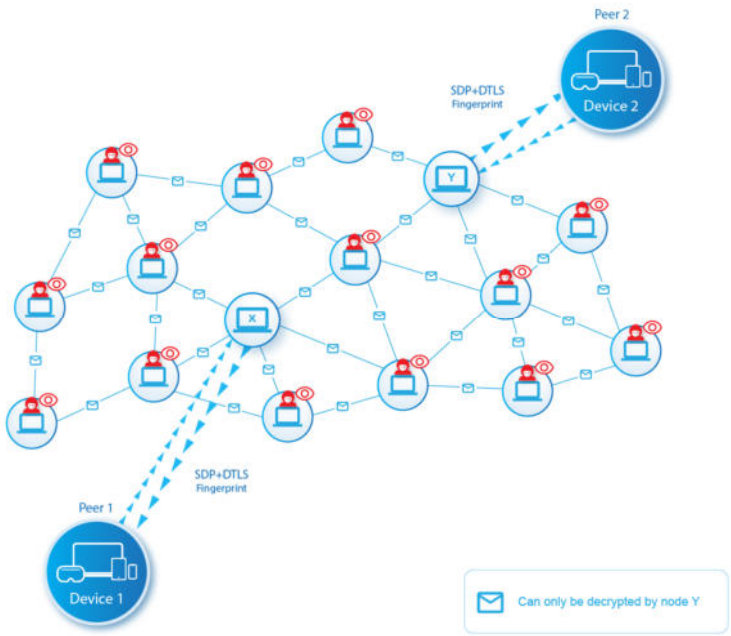


Figure 13: Solving MitM attack using Whisper.

4.3 Deployed security Assets

The current section provides an overview of the security assets that were deployed for Pledger. Compared to the preliminary D4.1 [1] version, the updates provided here cover:

- New machine learning experiments which were performed to evaluate the use of machine learning for intrusion detection, specifically comparing anomaly detection and neural networks (section 4.3.2).
- Implementation updates of the T&R management system (section 4.3.3) which cover:
  - The finalization of Pledger T&R indices definition.
  - The finalization of the computation scheme that is presented in detail, together with the corresponding computation formulas.

- The introduction of the T&R rules-based anomalies detection system for the very first time, covering a brand-new aspect of the Pledger platform. In this context, indicative implemented rules are presented.
- Integration of all the above with the Pledger platform, as can be presented through a functioning demo (section 5.2).

4.3.1 Distributed IDPS cybersecurity solution

4.3.1.1 Overview

The amount of malicious traffic keeps increasing, and new threats are created every day, getting more and more serious, complex and sophisticated. Companies need efficient and easy to use tools to monitor and prevent their systems from being compromised and that is where computer security tools come to the rescue. Most companies use a **Firewall** in order to deny or permit traffic based on pre-set rules. Even in the scenario that a firewall is well designed and configured, there are threats that can pass through it, because filtering decisions are based only on network packet header data. Analysing packet payload is often essential for detecting packets with malicious content. This is where **Intrusion Detection Systems (IDS)** come to help. An IDS is a monitoring system, either a device or a software application, that detects suspicious activities and generates alerts when they are detected. Based upon these alerts, a security analyst or incident responder can investigate the issue and take the appropriate actions to remediate the threat.

There are two different techniques an IDS uses to detect malicious traffic/activity: **statistical anomaly-based IDS** and **signature-based IDS**. Statistical anomaly-based IDS monitor system activity and classifies it based on heuristics and rules and attempts to detect any type of misuse that falls out of normal system operation. The signature-based IDS monitor the network and compare the packets against pre-set signatures, which are signatures of known attacks. When the signature-based IDS detect packets or traffic that match some of the signatures in the database, it will create an alarm about that malicious traffic and report to administrator. The issue with signature-based IDS, is that it cannot detect malicious traffic when database is not up to date with newest signature threats. On the other hand, statistical anomaly-based IDS will be able to detect this. Statistical anomaly-based detection, however, often suffers from false positive or false negative detections and should be assessed as to their performance.

In the context of Pledger project, a distributed cybersecurity network streaming platform has been implemented aiming to offer real time intrusion detection (IDS) and network security monitoring (NSM). The **business objectives** that drive this solution are to:

- Increase security, privacy and trust from the Edge to the Cloud
- Detect network threats and security anomalies and provide prompt alerts
- Provide sophisticated data analytics of the network traffic through a user-friendly UI

4.3.1.2 Architecture

The architecture of the cybersecurity solution can be found in the following figure (Figure 14):

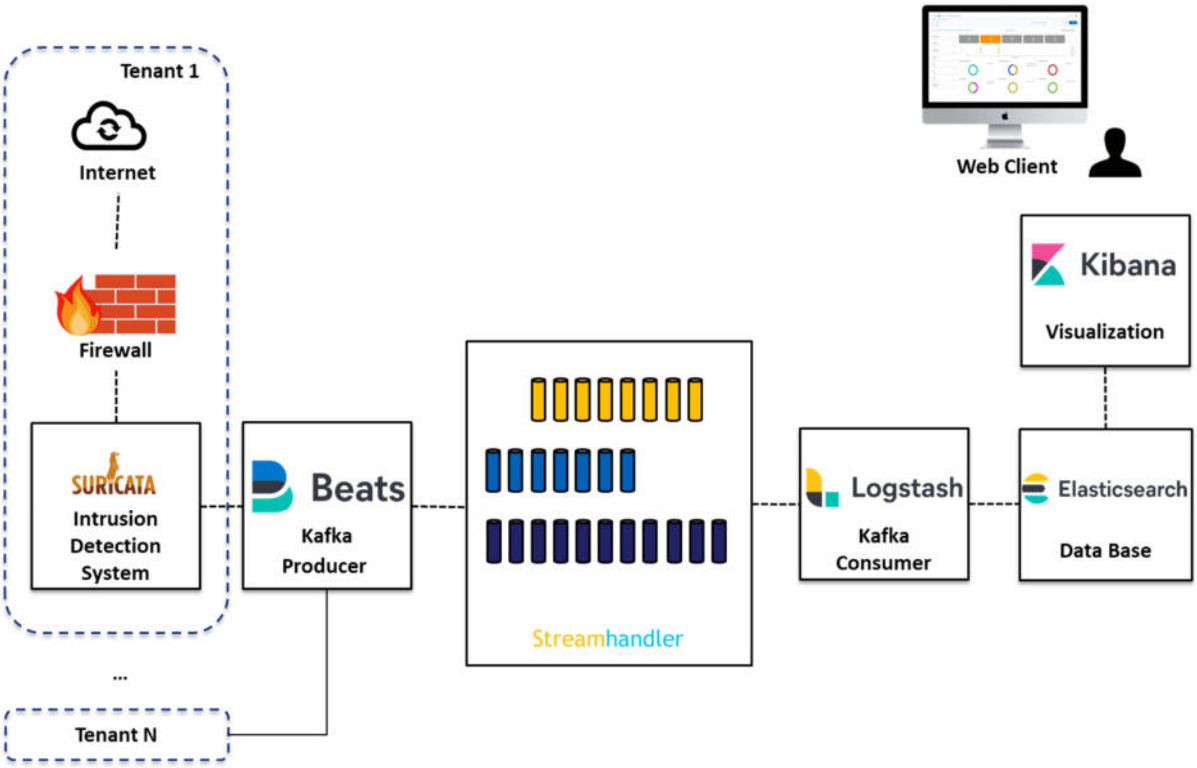


Figure 14: Distributed Cybersecurity platform architecture.

Our solution consists of the following software components:

- **Suricata**, which is a very popular open source, mature, fast, multi-threaded and robust network threat detection engine. The Suricata engine is capable of real time intrusion detection (IDS), inline intrusion prevention (IPS), network security monitoring (NSM) and offline log (in pcap format) processing. It inspects the network traffic using a powerful signature language to match on known threats, policy violations and malicious behavior. Moreover, it has a powerful Lua scripting support for detection of complex threats. With standard input and output formats like YAML and JSON integration with tools like existing SIEMs, Splunk, Logstash/Elasticsearch, Kibana and other frameworks, become effortless. Suricata constitutes the heart of the Cybersecurity solution as it acts as an IDS and NSM application.
- **Filebeat**, which is part of the ELK stack, belongs to Beats collection of lightweight data shippers for forwarding and centralizing log data in a distributed environment. Installed as an agent on your servers, Filebeat monitors the log files or locations that you specify, collects log events, and forwards them either to Elasticsearch or Logstash for indexing. In current solution, Filebeat acts as a Kafka Producer that collects JSON data generated by Suricata and forwards them to StreamHandler (Message Broker)
- **StreamHandler**, as already mentioned in 0, which is a data streaming and analytics platform based on Apache Kafka offering:
  - Real-time monitoring and event-processing
  - Distributed messaging system
  - High fault-tolerance
  - Elasticity - High scalability

- **Logstash**, which is part of the ELK stack, is used to dynamically ingest, transform and ship data using data processing pipelines (input-filter-output). It can be used to ingest data of all shapes, sizes or sources, parse and transform data on the fly and route data to a great variety of outputs.
- In current solution, Logstash acts as a Kafka Consumer that is responsible for:
  - Collecting data from StreamHandler (Message Broker).
  - Decoding, parsing, formatting and enriching of data.
  - Forwarding data to Elasticsearch
- **Elasticsearch**, which is part of the ELK Stack, is an open-source, distributed, RESTful search and analytics engine that centrally stores your data for lightning-fast search, fine tuned relevancy, and powerful analytics that scale with ease. In current solution, Elasticsearch acts as a time series Data Base, where all network traffic from all distributed servers is centrally persisted.
- **Kibana**, which is part of the ELK Stack, is used for data visualization using custom visualization panels & dashboards. It has built-in support for authentication and authorization and a great variety of tools for interacting with Elasticsearch.

This architecture is specifically designed to support a decentralized approach in the deployment of cybersecurity appliances on the edge (virtual Intrusion Detection, virtual Honeypot, virtual Deep Packet Inspection etc.). It then allows the project to aggregate all the network logs and threat information into a SIEM-like interface based on the ELK Stack. The use of the Kafka-based Streamhandler platform ensures that high throughput can be achieved, thus facilitating the data ingestion process from multiple instances across the Cloud-Edge continuum. This solution is designed to be able to aggregate data from multiple compute tenants/slices, as dedicated security services run in each slice can stream their results through Kafka providing an operational picture across the infrastructure that allows threats to be efficiently addressed by the Pledger provider.

4.3.1.3 IDS Modes

Pledger examines and implements two different IDS approaches, that both add significant value to gain clear insight about potential network threats and therefore help security analysts to apply efficient mitigation actions. Both approaches are compliant with the decentralised architecture, although the deployment and configuration of the IDS differs in order to provide two distinct functionalities.

Threat Detection mode

In the Threat Detection mode, as it can be seen in the following figure, the IDS component analyses only the traffic that is accepted by the applied Firewall rules. In this way, administrators can detect any severe security vulnerabilities that have to be addressed. The IDS acts as part of the perimeter defenses and identifies threats within the cloud infrastructure.

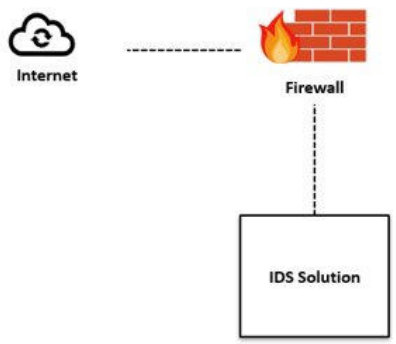


Figure 15: Threat Detection Mode – IDS.

Honeypot Mode

In computer security terms, a cyber honeypot is a sacrificial computer system that is intended to attract cyberattacks. It mimics a target for hackers, and uses their intrusion attempts to gain information about cybercriminals and the way they are operating or to distract them from other targets.

In the Honeypot mode, IDS follows a different deployment and configuration in order to log the threats and is attached directly to the network interface(s), before any firewall rules are applied, as it can be seen in the following figure. In this way, IDS can be configured properly to capture all the external network traffic, including network threats and attacks, so that they can be studied in order to improve the applied security policies.

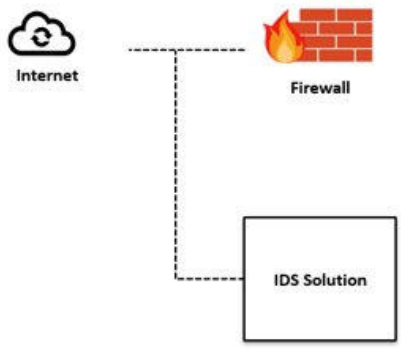


Figure 16: Honeypot Mode – IDS.

4.3.2 Machine learning experiments for intrusion detection

Additional experimentation took place on INTRA infrastructure to assess the viability of Machine Learning methods for cybersecurity [11], particularly comparing anomaly detection and neural networks. Previous work [12] has shown that anomaly detection can provide adequate detection in cases of 0-day attacks. However, the detection rate was very dependent on the traffic characteristics as it tended to detect more "covert" attempts dealing in low volume and velocity of the offending traffic. Methods based on analysing network logs using Natural Language Processing methods (such as Latent Dirichlet Allocation [13]) also showcased another important flaw, which is the existence of a high false positive rate especially for attacks that featured voluminous offending flows. This is considered a major shortcoming preventing this type of algorithm to be adopted in real operational conditions. Should an operator consider the information actionable, they would be unaware of the false positive status and deny legitimate user requests, thus creating an unintentional "Denial of Service" effect. In such cases, additional metrics (like information entropy) should be sought out to improve the detection capabilities, although this increases the processing time and thus provides a longer attack window to the attacker.

INTRA thus performed additional experiments to assess the viability of alternative Deep Learning methods. The dataset we worked on is the intrusion detection evaluation dataset (CIC-IDS2017) [14]. The dataset was created by Canadian Institute for Cybersecurity (CIC) and University of New Brunswick (UNB). The CIC-IDS2017 consists of labelled network flows, including payloads in pcap format, the corresponding profiles and the labelled flows and csv files for machine and deep learning purposes. The datasets included different types of cyber security attacks, among them being various types of Distributed Denial of Service and port scanning attacks. Three experiments were performed with DDoS, port scans and combinations/traffic mixes of both. Infiltration attacks and web scan attacks are greatly imbalanced in their datasets (i.e., labelled data were not enough) and for that reason the experiments results were not very promising in the current datasets.

The three experiments were comprised of Machine Learning (ML) models able to detect DDoS attacks, PortScan attacks and their combination. Figure 17 illustrates the methodology adopted to create AI pipelines consisting of three modules, namely data analysis, data engineering and AI modelling.

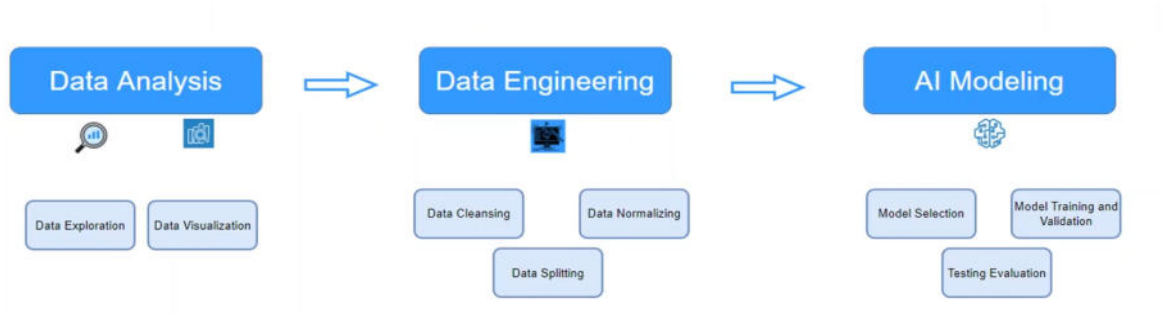


Figure 17: Methodology for the creation of AI pipelines.

The data analysis module focuses on Data Exploration and the Data Visualization. The Data Engineering module focuses on data pre-processing such as Data Cleansing, Data Normalisation and Data Splitting. Finally, the AI modelling module is comprised of the Model Selection, Model Training, Validation modules and performs the model's evaluation. In the INTRA experiments, the following pipeline was followed. First, features correlation was measured by constructing a matrix and following the Pearson correlation coefficient [15]. This technique was selected because it achieves two goals. The first one is to **decrease the dataset's features** to work with fewer features which decreases the time of each model's training phase, and second, so as **not to include features that are correlated** which can result to redundant information in the dataset and additional in increased fault accuracies. After removing the redundant features, the next step was to encode our labels, and split our main dataset to train, validate and test. The resulting datasets were normalised with our training test mean value and variance. The model created is an artificial neural network and more specifically a Multilayer Perceptron (MLP) [16] which has only one hidden layer comprised of 128 neurons.

4.3.2.1 DDoS Attacks Experiment

The initial dataset consisted of 78 features which, after the computation of the correlation matrix, were decreased to 30 as shown in Figure 18. The benign samples are almost 98k and the DDoS samples are 128k. Both classes do have sufficient samples in the dataset, and we can be optimistic that the class imbalance will not pose a problem to our experiment.

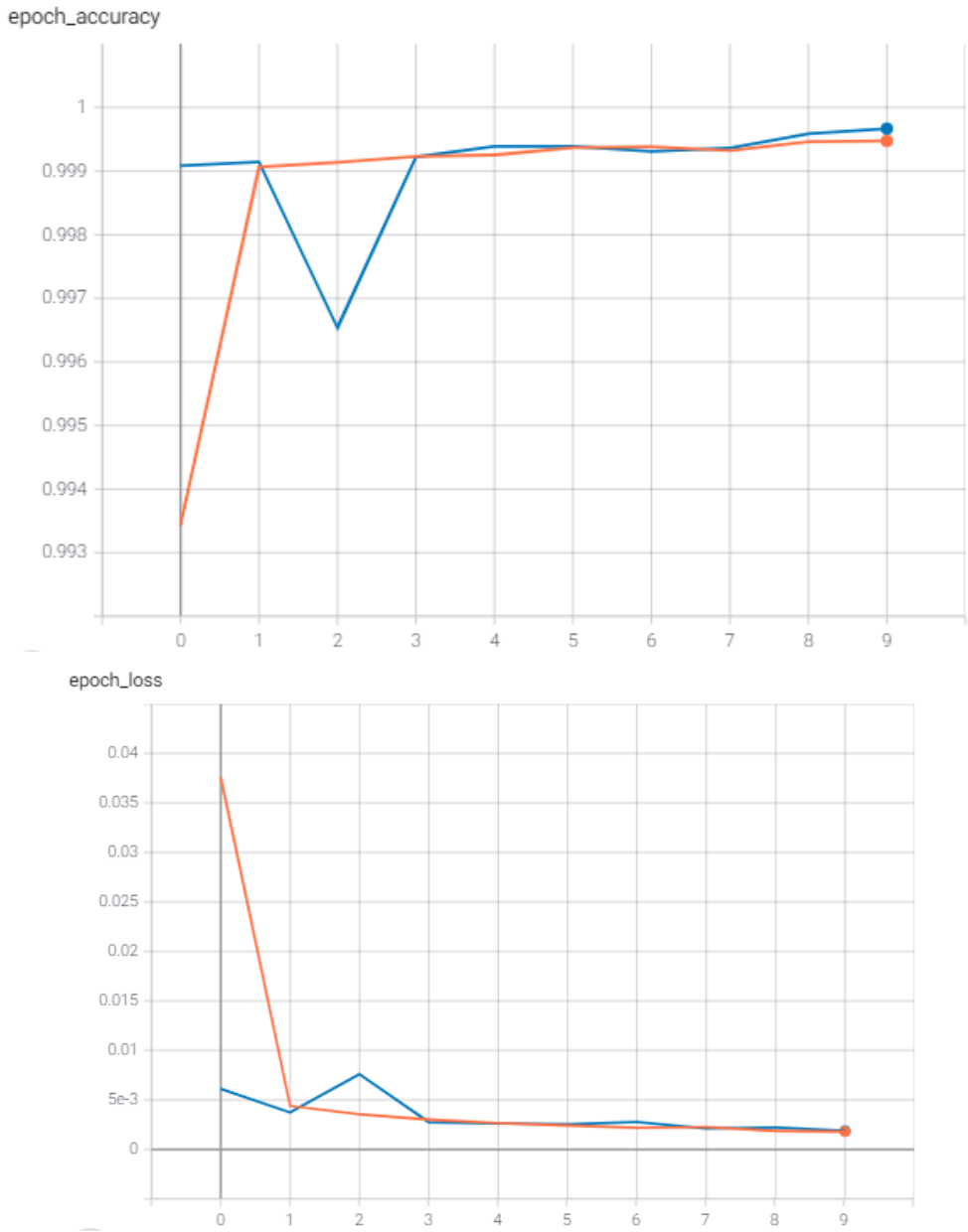
```
_destination_port            _bwd_packets/s
_flow_duration               _min_packet_length
_total_fwd_packets           fin_flag_count
total_length_of_fwd_packets  _rst_flag_count
_fwd_packet_length_min       _psh_flag_count
bwd_packet_length_max        _ack_flag_count
_bwd_packet_length_min       _urg_flag_count
flow_bytes/s                 _down/up_ratio
_flow_packets/s              init_win_bytes_forward
_flow_iat_mean               _init_win_bytes_backward
_flow_iat_min                _min_seg_size_forward
_fwd_iat_min                 active_mean
bwd_iat_total                _active_std
_bwd_iat_mean                _idle_std
fwd_psh_flags                _label
```

Figure 18: Selected features (after reduction).

After the split, the test dataset resulted in 20% of the initial dataset and the training and validation dataset consisted of the remaining 80% (with a further 80-20% split between them). The labels were encoded with the label encoder scaler and the problem resulted in a binary classification problem. Furthermore,

all the resulted datasets were normalized according to the train dataset's mean value and variance with the sklearn's StandardScaler [17].

Figure 19 illustrates the training and validation accuracy and loss during the model's training phase. Their graphical visualization shows great accuracy, a representation which is the same in the evaluation phase too. As depicted in the classification report below the model achieved its task and could detect DDoS traffic mixes with sufficient accuracy.

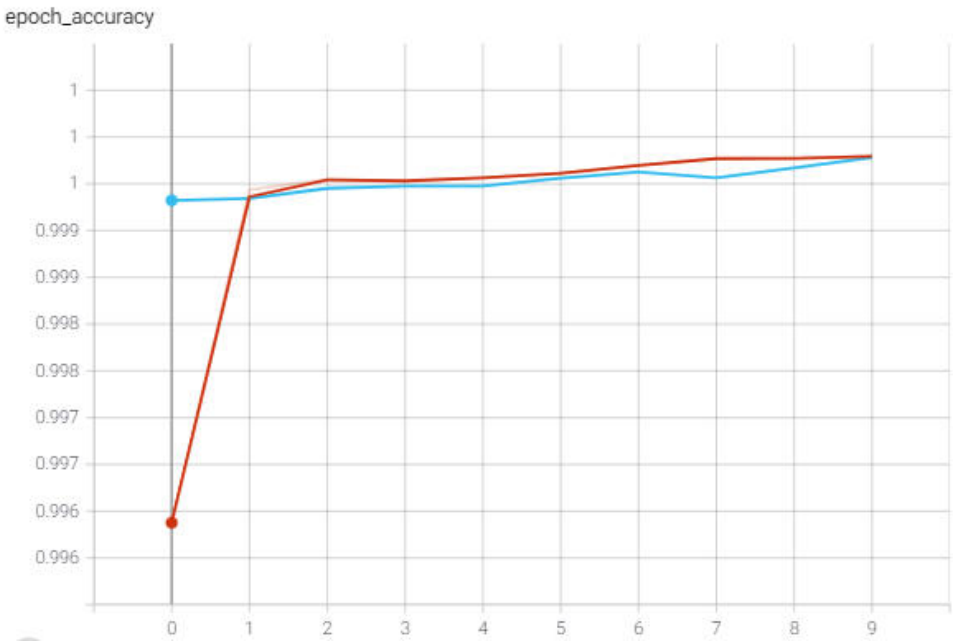


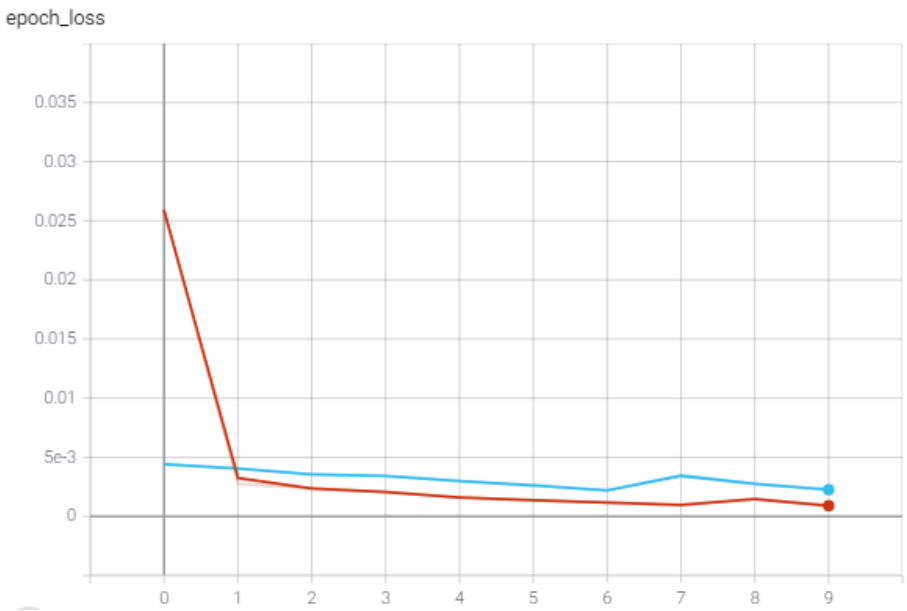
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     19566
           1       1.00      1.00      1.00     25577

    accuracy                           1.00     45143
   macro avg       1.00      1.00      1.00     45143
weighted avg       1.00      1.00      1.00     45143
```

Figure 19: Results for the DDoS detection experiment.

4.3.2.2 PortScan Attacks Experiment

In the next experiment, the PortScan attacks were analysed and an additional artificial neural network was created. The experiment's flow is the same as in the DDoS attacks experiment and followed the same pre-processing for the dataset. Again the problem is a binary classification problem and the dataset consists of 128k benign class samples and of 159k PortScan class samples. Once again, the class imbalance problem will not play a part in our experiment due to the good ratio of the classes in the dataset. In the images below we can observe the accuracy and the loss in the model's training phase. We can find that the model is capable to fully learn its training dataset and to reach high performances. Finally in the classification report we produced we can observe once more that the test dataset was able to be fully predicted and that our test instances can be perfectly separatable from our model.



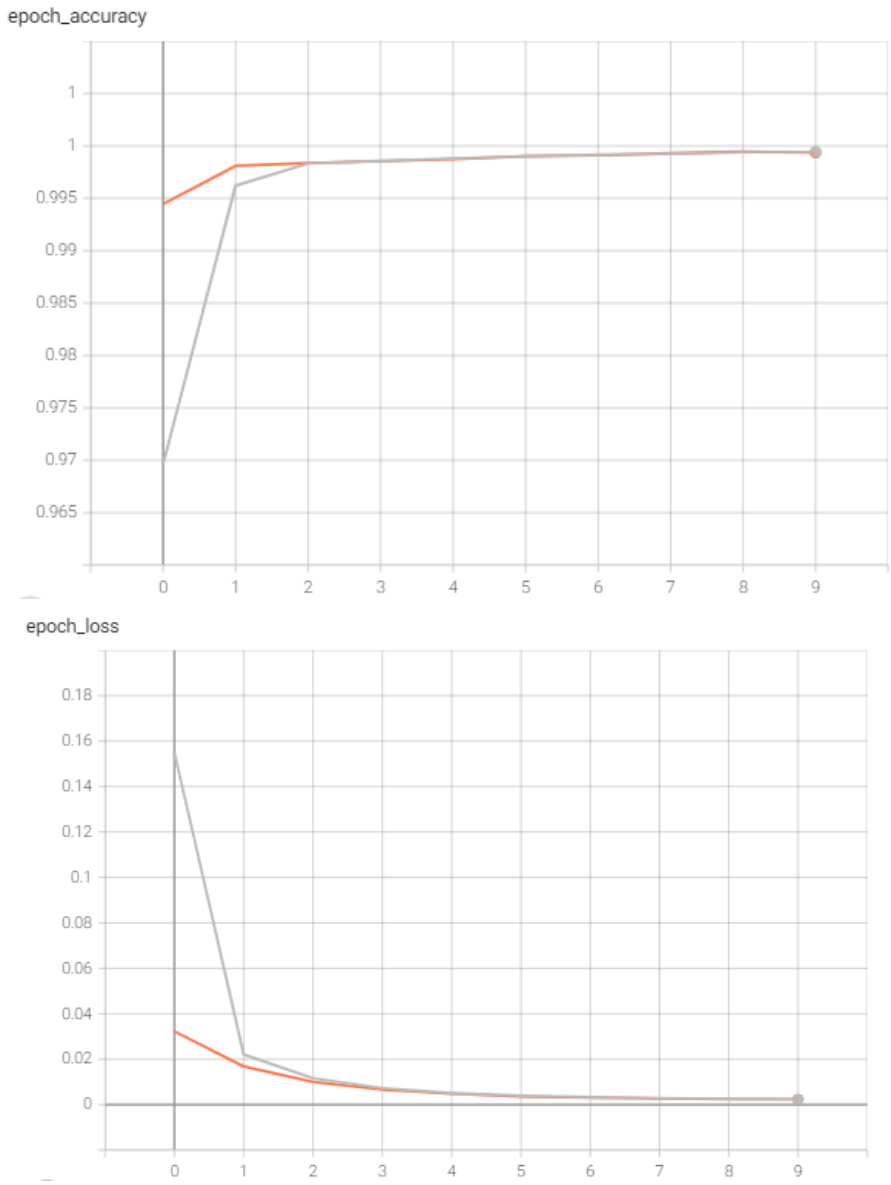


	precision	recall	f1-score	support
0	1.00	1.00	1.00	25384
1	1.00	1.00	1.00	31836
accuracy			1.00	57220
macro avg	1.00	1.00	1.00	57220
weighted avg	1.00	1.00	1.00	57220

Figure 20: Results of the PortScan attacks.

4.3.2.3 DDoS and PortScan Attacks Combined Experiment

In the last experiment performed, the two previous datasets were combined to create a unified model able to detect both attacks (without having to use the previous models). The same AI pipeline logic was used to develop our final experiment. Having more than two classes means that the problem is no longer a binary classification problem, but a multi-class classification with three classes. Thus, the label encoder used in the previous experiments is now converted to a label binarizer encoder. The datasets are again normalized by using the training's dataset mean value and variance in order to be ready to be fed to the artificial neural network. From the training's phase accuracy and loss graphs it is evident that the model is able to achieve high performance. Finally, from the classification report produced the model can be evaluated, showing that is able to generalize its gained knowledge to the test dataset too.



	precision	recall	f1-score	support
0	1.00	1.00	1.00	44912
1	1.00	1.00	1.00	25563
2	1.00	1.00	1.00	31887
accuracy			1.00	102362
macro avg	1.00	1.00	1.00	102362
weighted avg	1.00	1.00	1.00	102362

Figure 21: Results from the combined DDoS and PortScan attack experiments.

Overall, the Deep Learning model was able to be learn from the training datasets and produce overall very good results. Preliminary results of the infiltration and web attacks produced models weaker in performance models not reliable enough to be reused. Real world attacks with the additional same features would provide a good scenario to test the generalization of the deployed models, in new unseen data.

4.3.3 Trust and reputation

Trust and Reputation (T&R) are two indices that can be defined and monitored to guarantee a minimum level of security between entities of a distributed system that want to execute transactions or interactions between them. Many methods and mechanisms can be used to manage and model trust and reputation in environments like P2P networks, Wireless Sensor Networks (WSN), Vehicular Ad-hoc Networks (VANETs), Collaborative Intrusion Detection Networks (CIDN), Cloud Computing Systems, etc.

T&R management is a very useful and powerful tool in environments where a lack of previous knowledge about the system can lead participants to undesired situations, specifically in virtual communities where users do not know each other at all or, at least, do not know everyone. In such scenarios, through a T&R management mechanism, a peer is provided the option to identify the most trustworthy (as identified by the system) participant to have an interaction with, preventing thus the selection of a fraudulent or malicious one.

4.3.3.1 The Pledger T&R management system

T&R management systems follow five steps, as described by Marti and Garcia-Molina [18] (Figure 22):

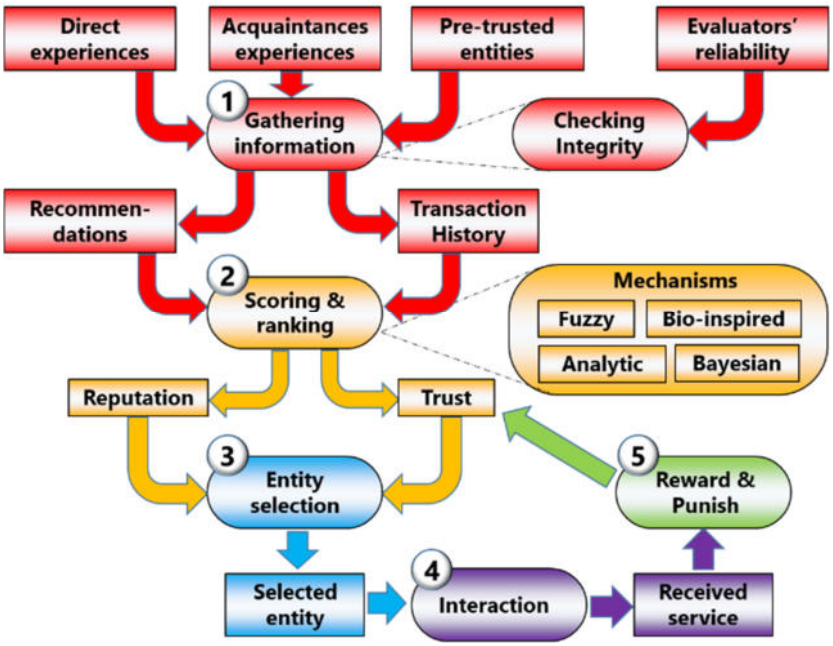


Figure 22 General steps followed in T&R models.

1. **Gathering information** about the behaviour of a certain participant.
2. **Scoring & ranking** for every peer in the network by calculating T&R indices.
3. **Selecting the most trustworthy or reputable entity** in the community providing a certain service.
4. **Effectively having an interaction** (and/or executing a transaction) with it, assessing posteriori the satisfaction with the received service.
5. **Rewarding or Punishing** according to the satisfaction obtained, adjusting consequently the global trust (or reputation) deposited in the selected service provider.

The overall **Pledger T&R management system** is of the integration between several components and subsystems of the Pledger Core System.

The first step in Figure 22 (gathering information) is realised through the SLA subsystem, as this subsystem is responsible for monitoring part of the behaviour of several actors related to their compliance with specific contracts. Information about the trustworthiness of actors (human-users) could

be extracted from other UC-specific mechanisms, but as this would lead to UC-dependent mechanisms, this scenario is not considered at a later stage of the project.

The second and third step in Figure 22 is the one on which T4.1 focuses and is presented extensively in section 4.3.3.2. It is related to the definition of Trust & Reputation Indices and the mechanisms for the calculation of the said indices. The "entity selection" step per se refers to the selection of a specific service provider from a service consumer, after the latter takes under consideration not only the results of the T&R scoring/ranking, but also those of the Orchestration subsystem, and more specifically the "Recommender" component, as described in D3.2.

In the fourth step, the actual service from the selected provider is consumer. The interactions (and transactions) of the several peers take place through the several subsystem of Pledger (such as the Pledger Blockchain and Smart Contracts implementation).

Finally, step five is related to the action of updating the T&R indices and scoring/ranking based on the new interactions that have taken place, thus establishing a feedback loop were benevolent actions are rewarded.

4.3.3.2 The Pledger T&R model

4.3.3.2.1 Entity roles, T&R properties, and T&R indices

In general, the trust relationship involves two entities/things; a trustee and trustor. The trustor needs to have confidence in the trustees in term of benevolence and honesty. Since trust is characterized by uncertainty and involves risk, it is difficult to ensure that the trustor will be satisfied by the trustee's behaviour. Under the context of Pledger, using the nomenclature already presented in D2.3 [2], a trustee is an IaaS provider (actor offering edge/ cloud infrastructure as a service) and a trustor is a SaaS provider (actor offering Software-as-a-Service, an Application provider) choosing to trust the trustee and select them for their services.

Different models manage concepts such as Trust or Reputation in different ways. Although there are some generic data structures for the domain of T&R provided, for example by the Open Reputation Management Systems (ORMS)[17] of OASIS, **there are no standards** for concepts like Trust and Reputation. However, Trust properties have been derived from social sciences [6]:

- **Asymmetry:** Trust can flow asymmetrically between entities. Where an entity-A trusts entity-B, it does not imply that also entity-B trusts entity-A.
- **Subjectivity:** Trust is subject to evaluation by one entity with another. Assume that the common opinion of the community of interest about an entity-B is that entity-B is well-behaved. However, for entity-A it may still be possible to have quite the opposite opinion about entity-B.
- **Partial transitivity:** Trust can be transitive or non-transitive. For instance, assume an entity-A trusts entity-B, entity-B trusts entity-C, it is not compulsory for entity-A to trust entity-C.
- **Context sensitivity:** If an entity-A forms a trust index about an entity-B, the judgment also considers the context based on which entity-A has established that judgment.
- **Dynamicity:** The behaviour of the several entities/actors and their corresponding T&R indices can change (dramatically) as times passes.

Taking under consideration these properties, in Pledger, the following definitions are considered:

- **Trust (T):** The expectation that an interaction will be satisfactory based on *our personal experience*. The belief of entity-A that entity-B is going to deliver the correct service, based on previous interactions of entity-A with entity-B. The Trust Index of entity-B provided by entity-A is a property which states how many times entity-B has successfully shared its services with entity-A. Trust is "subjective", because it is estimated from perspective of the individual trustor (entity-A in this case).

[17] https://www.oasis-open.org/committees/orms/charter.php

- **Reputation (R):** The belief that an interaction will be satisfactory based on the experience of *our social circle*. The belief of entity-A that entity-B is going to deliver the correct service based on previous interactions of other actors. The Reputation Index can be calculated from the Trust that other entities (apart from entity-A) have on entity-B. In other words, this metric determines the belief of others on an entity and is useful especially when entity-A does not have enough data to extract a Trust Index for entity-B (e.g., because there are no interactions between the two entities yet).
- **Popularity (P):** A counter which monitors how many times entity-B has received or may receive a request (how many "hits" it has). The Popularity Index is an accumulative and comparative indicator and is used to determine the stability of Reputation and Trust.

Generally, an entity can be evaluated only by information gathered from other entities. Its trustworthiness can be calculated by each and every other entity of the community (a subjective estimation) or by the whole system (a more, but not totally, objective estimation).

Reputation connects closely to the concept of Trust, but there is a clear difference, which can be illustrated by the following two scenarios:

- Entity-A trusts entity-B because entity-B has a good Reputation. This reflects that Reputation can be used to build Trust.
- Entity-A trusts entity-B despite the bad Reputation of Entity-B. This reflects that even if entity-A knows the Reputation of entity-B, entity-A has its own private knowledge (e.g. direct experience with entity-B) which is considered to be more important.

4.3.3.2.2 The T&R computation scheme

Trust computation is the processes that encompass techniques for extracting trustworthiness information regarding interacting entities. Trust computation schemes categorized into trust composition, trust propagation, trust aggregation, trust update, and trust formation [19].

Trust composition indicates the components (trust-properties) that are considered in trust computation processes. Trust components fall either under social trust (honesty, connectivity, unselfishness, intimacy) or QoS trust. Considering the scope of Pledger, the selected approach is that of identifying a QoS trust, indicating the belief that an entity can provide services in the quality as agreed or requested by another peer.

Trust propagation relates to the style of exchanging trustworthiness information among entities. In the literature, trust propagation is categorized in one of two schemes, either centralized or distributed. Considering that Pledger is focusing on the composition of QoS trust and that it uses a centralized mechanism for the related information gathering (i.e. SLA monitoring), Pledger follows a centralized trust propagation schema.

Trust aggregation entails the aggregation of trust information (observations), through either self-experience or peers recommendations. In the literature, trust aggregation techniques include weighted sum, inference approaches, and regression analysis. The weighted sum approach has been deemed as the most appropriate one, for various reasons, one of them being that the levels of the Pledger SLA violations notifications can be used for the extraction of the importance of a violation (a negatively evaluated interactions). These levels range from mild to catastrophic.

Trust update is the process of involving new observations and recommendations into trust credit. Trust update follows either event-driven approach or time-driven approach. Building on top of the periodic updates of the Pledger SLA monitoring mechanisms, Pledger follows a time-driven approach, although it can also be used in an event-driven context. From a computation perspective, the Pledger T&R system uses both big and small time-windows to quickly detect malicious or unsatisfactory behaviour and avoid the fast redemption of blacklisted actors, and incorporates a fading factor to give a higher weight to recent interactions.

Trust formation represent the way of computing the **overall trust**. In the literature, trust formation is found in either multi-trust form or single-trust form. Pledger follows the multi-trust approach,

decoupling trust from popularity and reputation, and identifying different T&R indexes between a trustor and a trustee for a different service.

Based on the above, for the actual calculation of the T&R indices, the following parameters have to be considered:

- **Satisfaction (s):** This value is essentially a subjective QoS indicator. The Satisfaction is the a posteriori assession of the interactions. A malicious or faulty actor will quickly lose any trust. If the behaviour of the actor changes, the system will allow it to build trust again.
- **Weight (w):** This is a value indicating how crucial the interaction is for the well-being of the actor. It is used in order to prevent a malicious actor from exploiting previously built trust. Due to this value, the Trust can drop quickly in case of a catastrophic interaction.
- **Fading factor (f):** When new interactions take place, the importance of older ones should decrease. The fading factor addresses this issue and forces peers to stay consistent with their previous behaviour. Old interactions have lower fading factor values, so an actor cannot misbehave relying on their good history. The fading factor makes the Social Reintegration of ex-malicious nodes possible, meaning that if they become benevolent, it is possible for them to get a second chance and form new ties with the network. Of course, multiple incidents of misbehaviour can get an actor permanently black-listed. This fading factor can be set by the system administrator, so that the actors take under consideration the last N interactions with any other actor.

When an actor wants to calculate the **Trust Index** of another one, it investigates the appropriate Log Files and calculates the trust value as the weighted average of the log entries using:

$$\mu_{c_i,p_j,s_k,t} = \frac{\sum_{x=1}^{N}(s_x \cdot w_x \cdot f_x)}{W} = \frac{\sum_{x=1}^{N}(s_x \cdot w_x \cdot f_x)}{\sum_{x=1}^{N}(w_x \cdot f_x)}$$

where mean value (μ) is a measure of the overall observed behaviour of the actor and indicates the expected satisfaction value of the next interaction between a consumer c_i and provider p_j for a service (type) s_k at time *t*, and where W is the normalization co-efficient which ensures that the trust value will be between [0,1].

To know how confident we can be about the value of μ i.e. how much the satisfaction from the service may actually deviate from μ, the **standard deviation (σ)** of the behaviour is also calculated. To reduce the computational overhead, the calculation of the later occurs simultaneously with the calculation of the mean value following the formula:

$$\sigma_{c_i,p_j,s_k,t} = \sqrt{\frac{\sum_{x=1}^{N}({s_x}^2 \cdot w_x \cdot f_x) \cdot W - (\sum_{x=1}^{N}(s_x \cdot w_x \cdot f_x))^2}{W}}$$

Finally, a strict definition of Trust is:

$$T_{c_i,p_j,s_k,t} = \mu_{c_i,p_j,s_k,t} - \sigma_{c_i,p_j,s_k,t}$$

To sum up, μ shows the satisfaction that entity-A should expect from entity-B, while σ shows how predictable the behaviour of entity-B is. That way the service providers that are not consistent and have an ever changing and oscillating behaviour will have lower Trust indices even if their μ value is higher.

Similar approach is being followed to calculate the **Reputation Index**, although this metric needs to extract more interactions logs.

4.3.3.4 Trust attacks and Anomalies Detection

Robust T&R systems are essential for sustaining the functionality of applications. However, several types of attacks that can compromise the T&R systems themselves. Trust attacks can be categorized into a biased recommendations, inconsistent behaviour, and identity attacks. Such attacks and measures against them are discussed in sections 5.2 as well as in the entirety of section 4.3.3.

Pledger's decisions to decouple Trust from Reputation and monitor the distribution of the several evaluations (subjectivity), calculate different T&R indexes for each service and provide different weights for different interactions (context sensitivity), and track the historical evolution of the T&R indices (dynamicity) makes it possible formulate and implement a **T&R rules-based anomalies detection system**. Detection of the said anomalies can trigger notifications that provide the context of the perceived anomaly (which could be a potential attack on the T&R system per se or on one of the entities involved in it). Indicatively, the following rules trigger anomalies detection warnings:

- **Consistent Trust vs Reputation misalignment:** The trust indexes T_{c_i} of a consumer for other providers is consistently (way) bellow or above the corresponding Reputation Indexes of the providers. This can be due to either slandering/white-washing attacks, or due to discrimination.
- **Consistent Trust alignment:** The trust indexes T_{c_i} of different consumers for other providers is consistently (almost) the same for all related providers. In case of high standard deviation between these providers and others, this may be an indication of malicious collectives.
- **Popularity vs Reputation misalignment:** The Popularity of a service provider is increasing in an accelerating manner, even though their reputation had been decreasing (a time-delay is expected). In a more formalized manner, this rule is set as:

$$\frac{\Delta^2 P_{p_j,s_k,t}}{\Delta t^2} > 0 \text{ while } R_{p_j,s_k,t'} < 0$$

5 Security asset demonstrations

5.1 Virtualised intrusion detection

5.1.1 Motivation

The IDPS demo utilizes two instances of the Intrusion Detection system, one of which has been set up in "sacrificial" infrastructure acting as a honeypot and deployed outside of the project's perimeter defenses. The protection of the Big Data Management subsystem was selected, as it is a central component featuring many integration points with the rest of the components of the Pledger infrastructure, although instances can be deployed in any Point-of-Presence. A second instance was deployed within the perimeter defenses, illustrating the actual impact of the security configurations applied to the firewall.

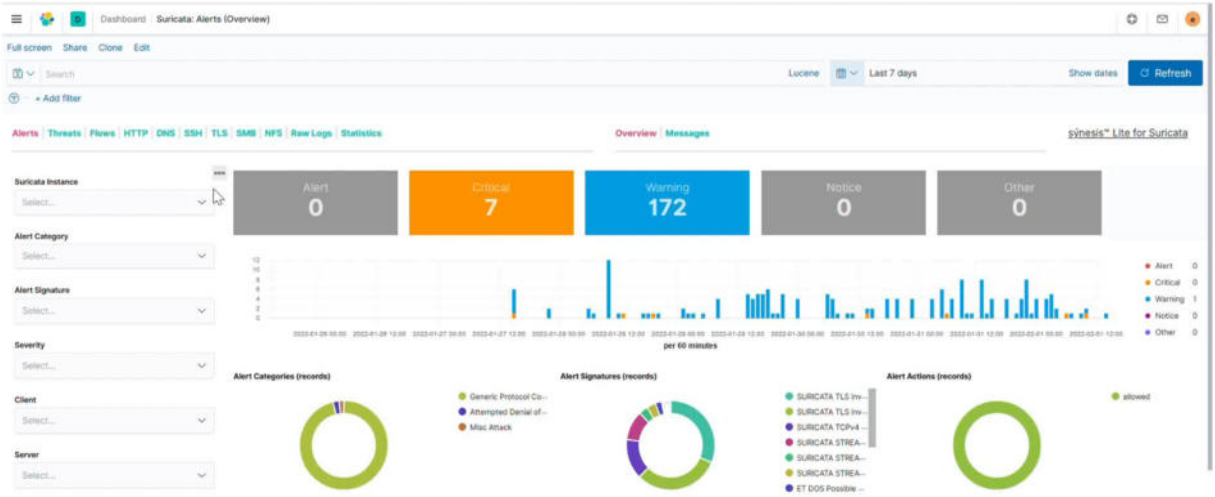
The benefit of this deployment is two-fold: a) each client or tenant of the infrastructure/service provider gets unfettered access to the virtualized security assets that are deployed for their benefit, and b) the infrastructure provider can aggregate threat information from across the different instances and thus have a bird's eye view of threats on their infrastructure. The provider may also select specific instances dedicated to a single client and review the threat information. Security operators can seamlessly switch between these views and provide remediation to the clients.

The business potential of providing such services can be invaluable and can help clients reduce their OPEX/CAPEX expenditures by purchasing managed security services offered by the provider. In fact, the managed security services market is projected to grow "from USD 22.8 billion in 2021 to USD 43.7 billion by 2026, at a Compound Annual Growth Rate (CAGR) of 13.9% during the forecast period", according to MarketsAndMarkets [20]. The rapid CAGR is also corroborated by other sources [21]. The existence of managed services can be invaluable as small and medium enterprises [22] find the costs in maintaining dedicated security teams to be prohibitive, and are often the most attractive targets of cyber attacks [23]. On the operator's side, the virtualized nature makes remote operation easier, further driving down the costs.

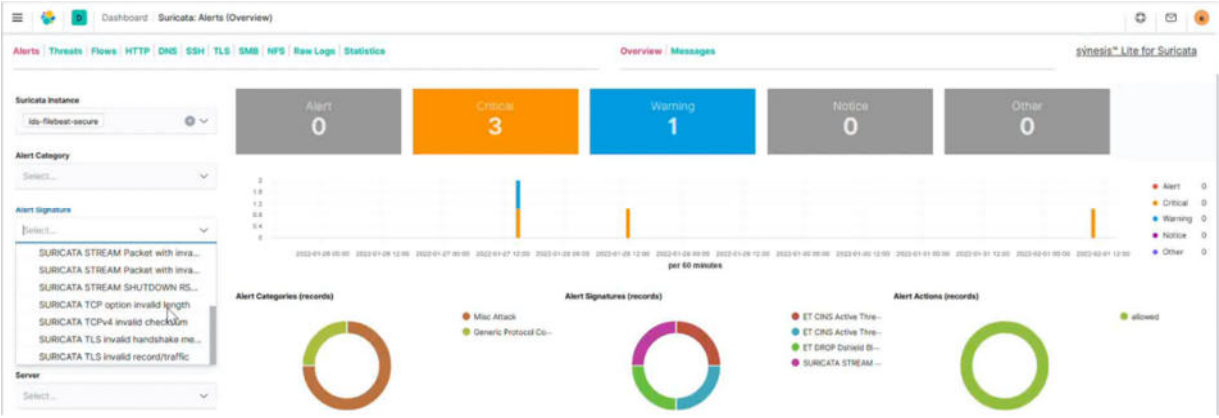
5.1.2 Results

The ELK stack deployed in the INTRA infrastructure is responsible for the aggregation, indexing and visualization of the data gathered from the running IDPS instances. In our case, we are running a "secure" instance, meaning it is contained within our perimeter defenses and configured to be on detection mode, whereas the second "insecure" instance is the sacrificial "honeypot".

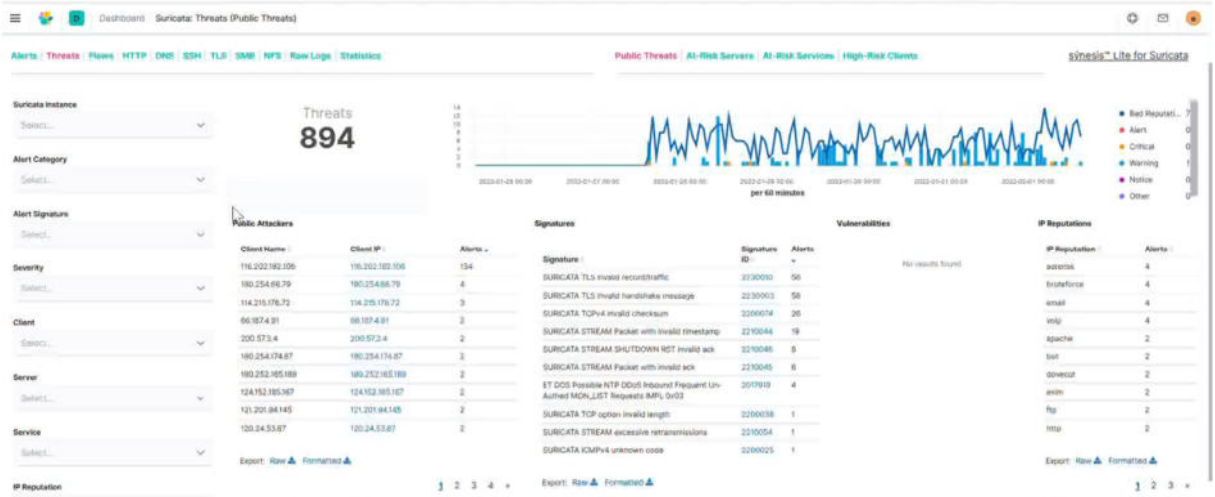
Figure 23 illustrates the "bird's eye view" of the infrastructure operator, that may access aggregated threat statistics or statistics from each individual instance. A variety of filters are available in order to review threats per instance, alert signature, severity, etc. The alerts screen provides a view the alerts created in response to the documented threats. These are classified in terms of their severity. In this case, during a seven-day period, 172 warnings and 7 critical alerts were issued. Filtering per instance yields one warning and only 3 critical alerts in the secure instance. Upon closer inspection, these alerts are caused by low-reputation IPs that are blacklisted in public databases like DShield.



(a)



(b)



(c)

Figure 23: (a) Alerts screen, (b) filtering per instance and alert signature (c) the threats screen allows review of the specific threat statistics and signatures.

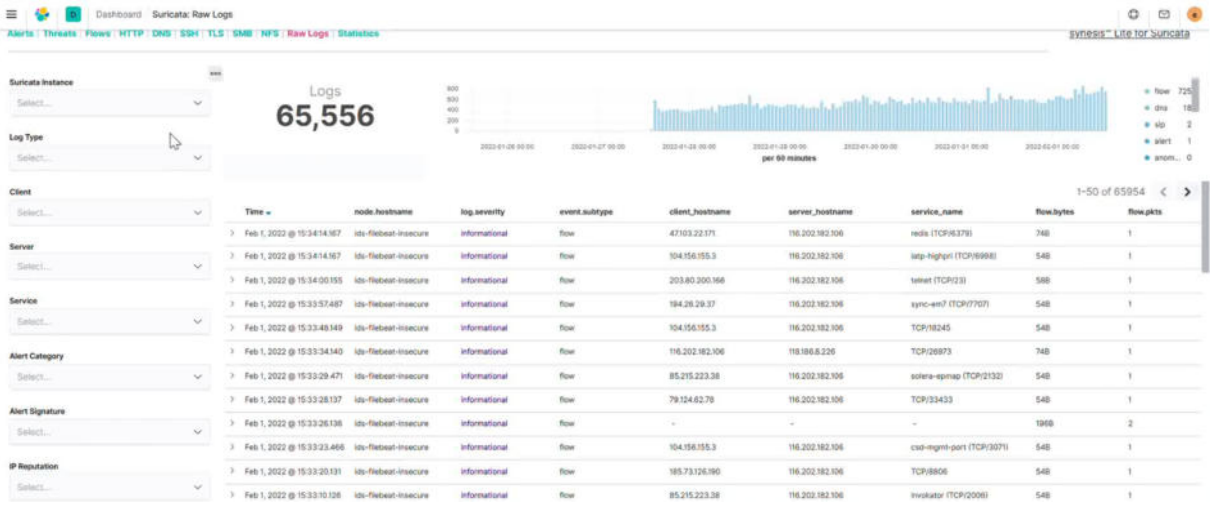


Figure 24: Logs screen.

As illustrated in Figure 24, the user may also see statistics related to the logs kept by Suricata. In a seven-day period, more than 65K logs were collected. The frequency of log captures can be configured. Statistics are also available per client and server (in the Flows screen, Figure 25), by protocol (Figure 26), or per geographical location of origin (Figure 27).

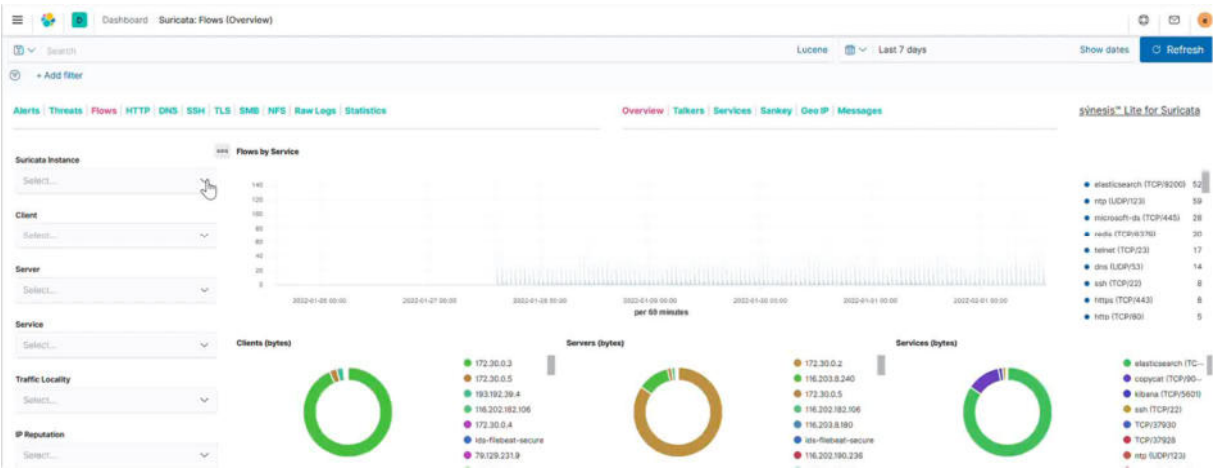


Figure 25: Flows Screen.

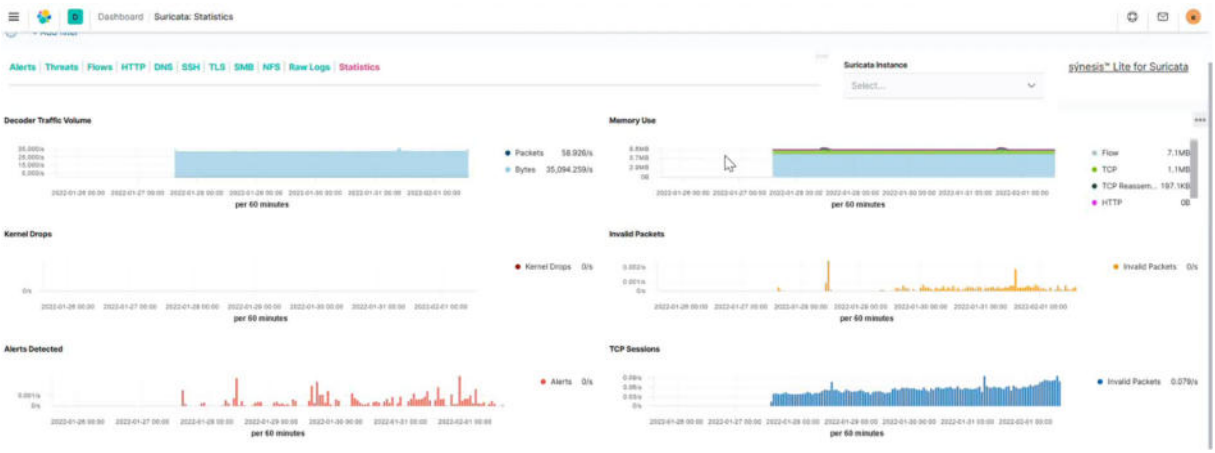


Figure 26: Statistics per protocol.

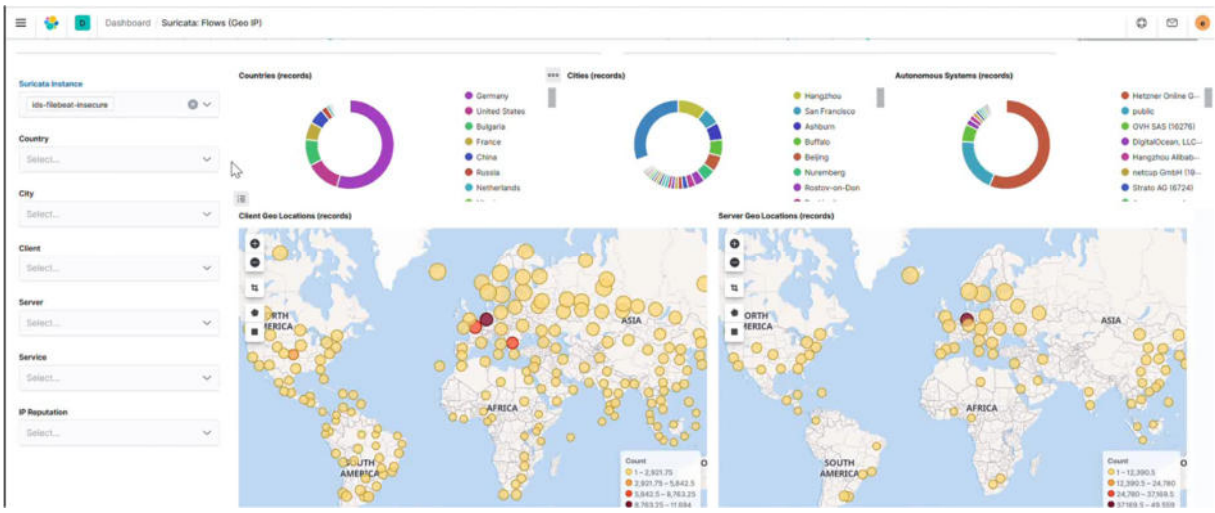


Figure 27: Geographical origin of offending flows.

This high level view of threat information can be further developed as the infrastructure provider can push anonymized threat data in a dedicated stream which all clients can access, thus providing them with near real-time threat intelligence. This can work against the attackers' use of automation and drastically reduce attack windows in case of large-scale attacks.

The full demo is available on the project's YouTube page. A dedicated blog post is also available in the project website.

5.2 Trust and reputation subsystem

5.2.1 Motivation

The Trust & Reputation (T&R) demo focuses, on the one hand, on the actual implementation of the Pledger T&R computation scheme and its integration with the rest of the Pledger components, and on the other hand, exposes the results of the said computations, thus showing in a graphical manner the T&R outputs to be taken under consideration by potential service consumers. More specifically, two scenarios are taken into consideration in this demonstration.

In the first scenario, a potential service consumer (SaaS provider) wants to select an appropriate service provider. Similarly to the Recommender presented in work package 3 (WP3), in this scenario we aspire to provide extra information to the potential consumer, so as to facilitate this selection. As such, a

ranking list of the said providers is extracted, presenting the corresponding Reputations indices (Figure 29), which can be used as an extra source of information for the consumer to make their final selection.

In the second scenario, a service consumer has already established a relation with a service provider and can track their other "performance". To that end, Pledger provides a set of gauges which present the T&R indices of the selected provider, including not only their reputation, but also the Trust on them, as well as an indication of their Popularity (Figure 30).

5.2.2 Results

The Trust and Reputation systems' implementation is based on Node-RED flows. Furthermore, MongoDB is used for data storage and retrieving purposes. Node-RED is an open source flow based tool, built on Node.js which enables work with ready-to-use predefined nodes or the creation of custom nodes written in javascript. MongoDB is a popular NoSQL, document-based database that organizes data into collections. The database model consists of six collections which are described as follows:

- **SLA_Contracts:** The SLA contracts which contain all the defined details as for example are the id, the name, the provider details, the client details etc.
- **SLA_Violations:** Every SLA violation that has occurred, including the importance of each violation.
- **Trusts_Indexes:** The Trust indexes between every client and provider.
- **Reputation_Indexes:** The Reputation indexes of every provider.
- **Status:** The live violation status of each active contract.
- **History:** The status of every active contract for every past moment the status was recorded.

The Trust and Reputation engine has two Kafka listeners that wait for new SLA contract notifications (start or stop) or new SLA violations (Figure 28) to be sent in the relative Kafka topics. In the first occasion, the new contract is parsed and based on the contract status field the contract is either added to or deleted from.

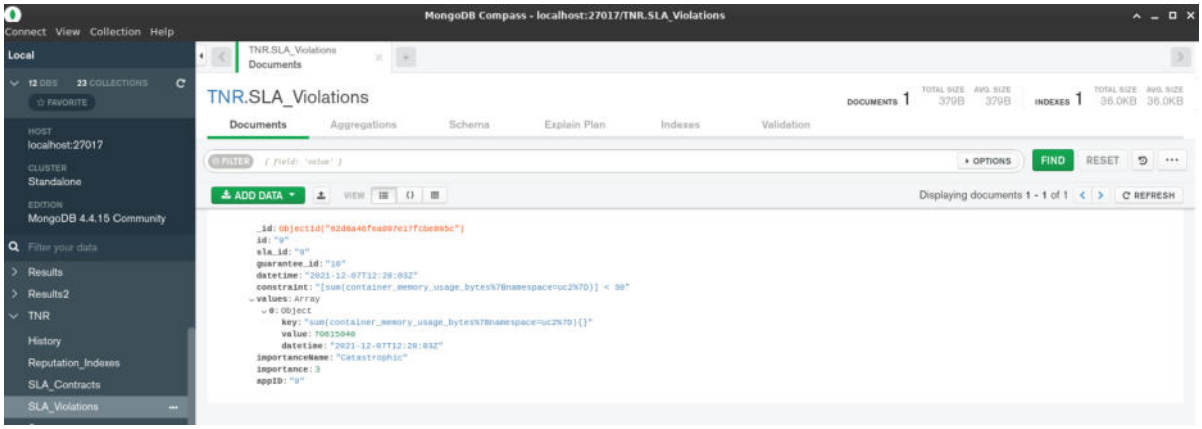


Figure 28: SLA violations collection.

Upon the creation of a new SLA contract, a query just inserts the new contract in the Contracts collection. Next, the Status collection is updated by inserting a new document that contains the contract id and the ids of the two sides i.e. the client id and the provider id.

The status field is initially set to 1, which means that the contract is not violated. Consequently, a new trust index between the client and provider of the new contract is created in the Trust_Indexes collection. The trust field is set to "fresh", since there are no data yet available to calculate the trust index. Lastly, the Reputation_Indexes collection is updated and a new reputation index for the provider is created and set to 0 (in case this is the first contract referred to the specific provider).

After the first contract is added, an infinite loop starts that periodically checks the status of all the active contracts i.e. the contracts that are saved in the Status collection at that time. Every time the Status

collection empties, the loop is reset and is ready to start when a new contract appears. Based on the Status collection, the History collection is updated. That means that, for the specific moment the history recording occurs, the status of every tuple of contracts, providers, and clients for every active contract is stored in the History collection.

Having a mechanism that checks and stores the status of active contracts periodically, Pledger is ready to calculate whatever is needed. For this purpose, every active contract is retrieved and, combining the previous Trust index and the current status, the new Trust index is calculated.

Firstly, the status field of every contract is reset to 1. That means that for the next iteration we assume that the contract is not violated, unless a new violation notification appears. After that, all the stored trust indexes are updated in the Trust_Indexes collection. Lastly, an index for the Reputation_Indexes collection, based on the providers ids is created, which will be necessary for a later aggregation step that will produce the reputation indexes.

In this aggregation pipeline, the trust indexes are grouped by the provider id field. For each group the average of all the indexes is calculated which results in the reputation index of every provider.

The aforementioned process is repeated as long as there are active SLA contracts.

In the case an SLA violation arrives, the violation is saved and the status of the contract for which the violation happened is changed so that the previously described procedure can continue. The new violation is inserted in the SLA_Violations collection and then the status field of the SLA contract that corresponds to the violation is set to minus the value of the importance of the violation. This means that a more important violation will reduce the provider indexes more than a less important one.

As far as the graphical part is concerned, the popular Node-RED dashboard module is used. After a query for the best Reputation indexes is executed, the results are presented through a chart (Figure 29). Additionally, queries for the trust indexes and reputation indexes for specific providers are executed and presented through gauges (Figure 30).

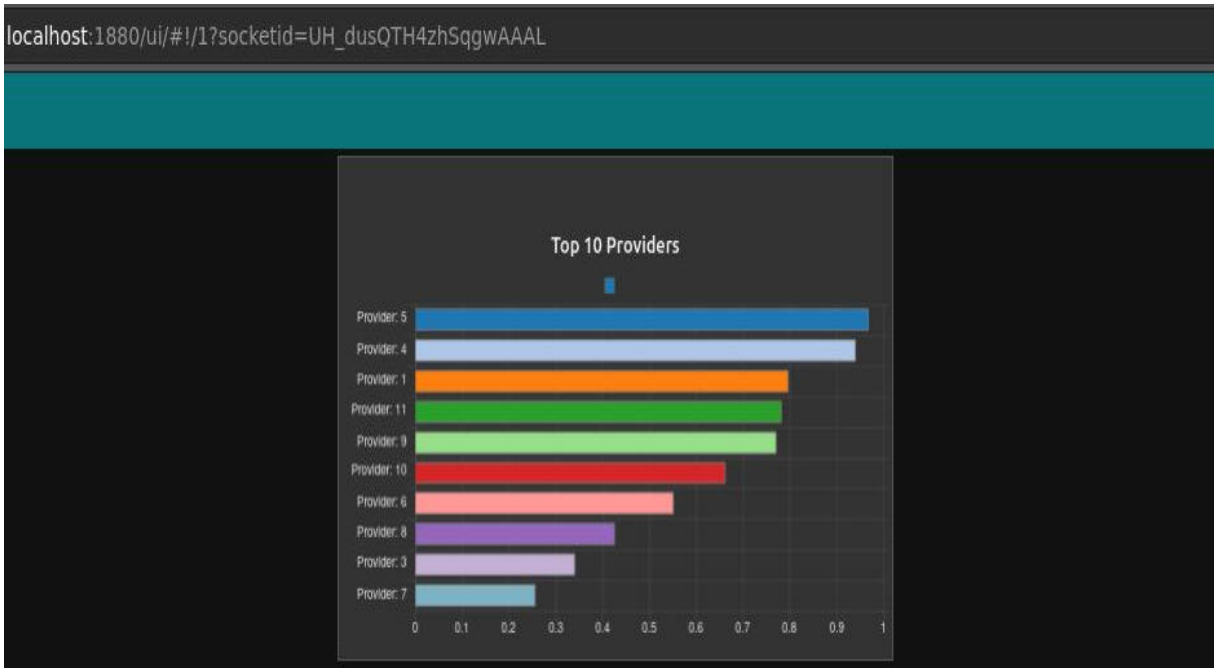


Figure 29: Reputation ranking of service providers.

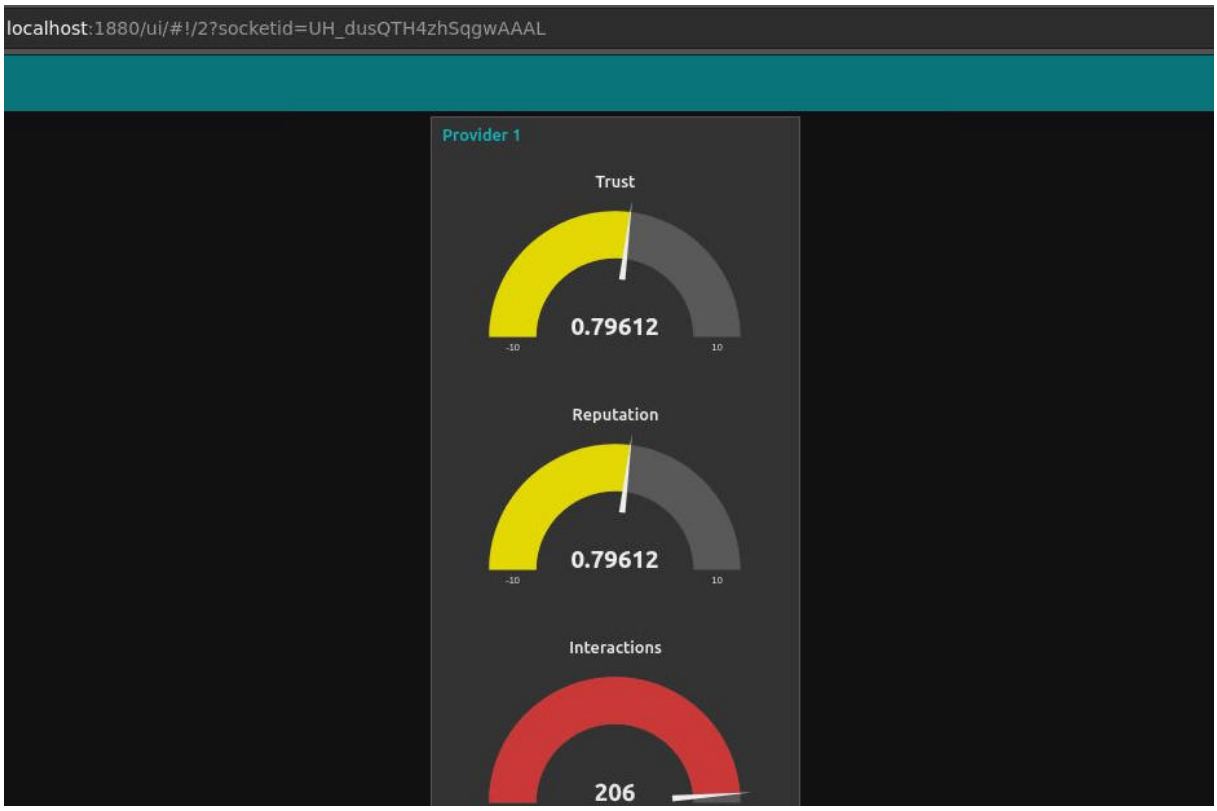


Figure 30: T&R indices of a specific provider.

5.3 CVE/CIS data aggregator

5.3.1 Motivation

The idea behind the CVE/CIS data aggregator is to allow service providers to have a better understanding of the security features of the infrastructures used in Pledger according to an official consensus-driven security guideline produced by **CIS**[18], a nonprofit global IT community that promotes the identification of security best-practices as well as possible corrective actions to mitigate threats.
As a result, service providers can access a dedicated UI on the CVE/CIS data aggregator and check the security rating of each infrastructure, so to better configure DSS priorities when choosing where to offload computation. At the same time, reports about corrective actions are available to infrastructure providers to possibly fix the threat found according to the CIS security best practices.

5.3.2 Results

An example of the CVE/CIS data aggregator UI is shown in Figure 31, with a convenient summary of the number, status and severity of the tests done on each infrastructure.

[18] https://www.cisecurity.org/ (Accessed August 2022)

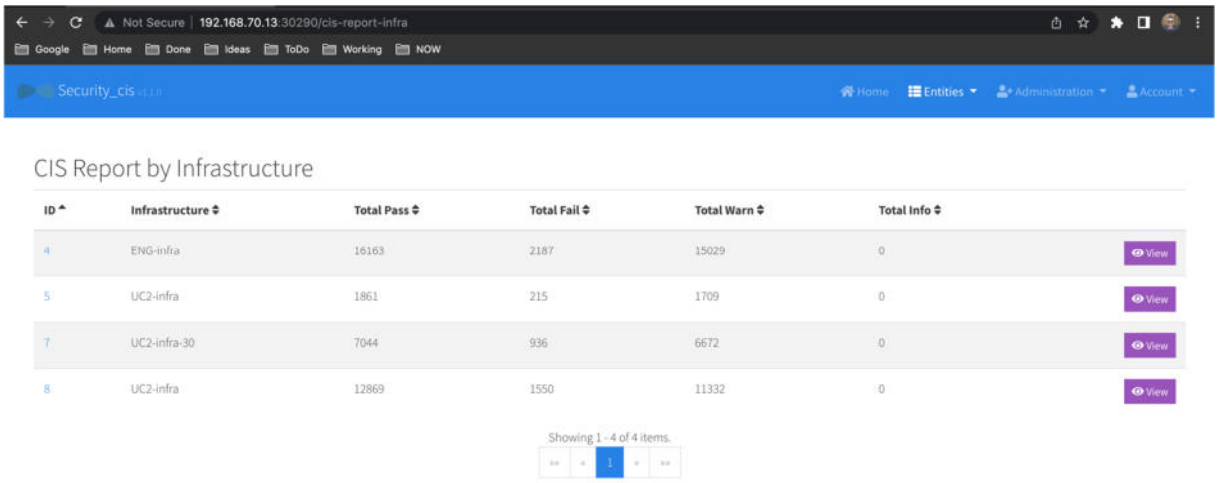


Figure 31: CVE/CIS data aggregator UI with synthetic reports.

Figure 32 shows an example of the different checks done according to the CIS security best practices and the infrastructure component involved, with the example reported about the key Kubernetes services.

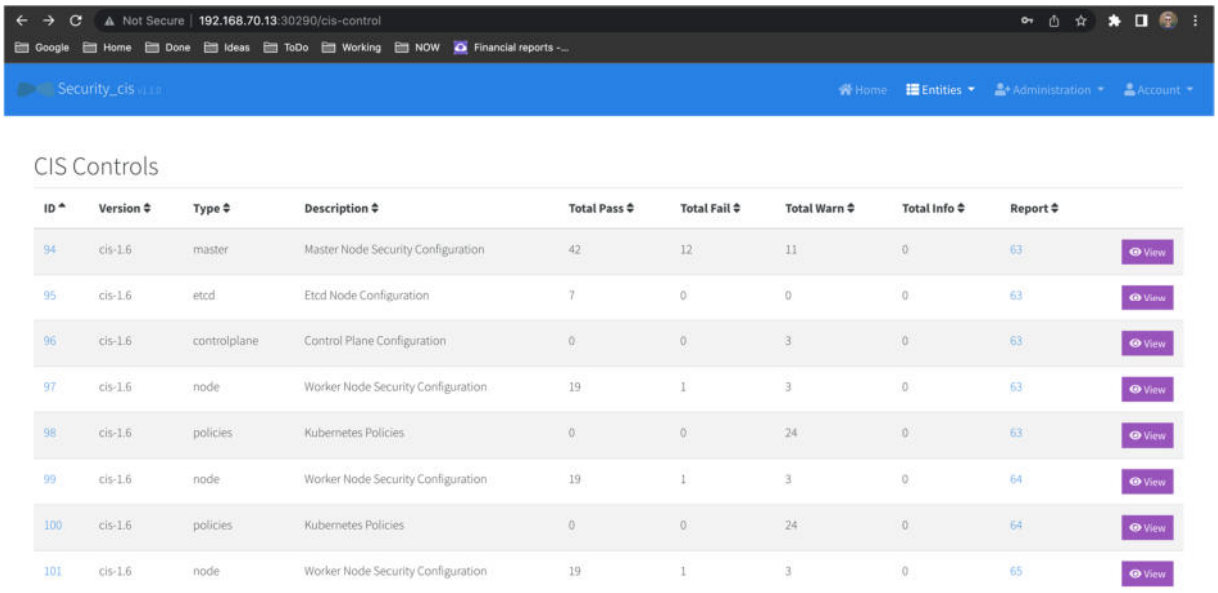


Figure 32: CVE/CIS data aggregator checks example.

Finally, Figure 33 shows the suggested corrective actions to mitigate the identified threats. While some are easy enough to apply so that could be even included in an automatic process, others require manual intervention as more complex to configure.

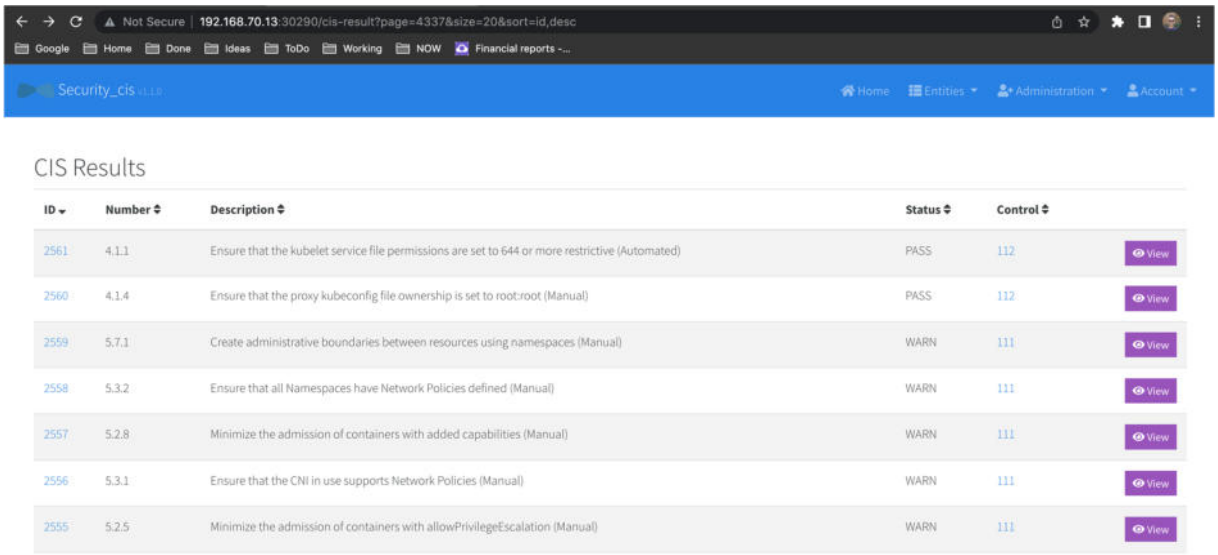


Figure 33: CVE/CIS data aggregator suggested corrective actions.

5.4 Whisper integration with the Pledger blockchain

5.4.1 Motivation

The Whisper protocol permits nodes to exchange messages anonymously. This means that a message transported using the Whisper protocol does not give an indication of the sender, the receiver, and the contents of the message.

This is beneficial for all cases looking to exchange low bandwidth messages in complete security and darkness. For example, in UC1, the Whisper protocol is used to encrypt and forward the messages generated by two peers attempting WebRTC signalling. It is crucial to secure the contents of these messages as access to this information jeopardises the integrity and security of the following P2P direct communication channel.

5.4.2 Results

The blockchain hosting the nodes with the Whisper protocol will be separated from the Pledger DLT but will be considered as a part of the Pledger Core platform. Therefore, users are allowed to connect to the blockchain nodes and utilise its functionalities by sending JSON-RPC requests.

6 Conclusions

During the period leading to August 2022, infrastructure hardening measures were taken to secure the Kubernetes cluster, the blockchain/DLT of the project, the CI/CD and big data infrastructures, and other related parts of the Pledger Architecture. Specific security assets were also deployed, in order to secure the Pledger Core and Use Cases. Pledger applied a threat analysis methodology based on the review of bibliographical resources as well as real threats encountered and analysed on the Pledger Honeypot. This allowed the consortium to apply carefully selected countermeasures to harden its infrastructure as well as dedicated cybersecurity tools to address a wide range of cyberthreats.

The top-6, critical threats against Pledger (in order of severity) were found to be:

- Sensitive Data Exposure
- Account Manipulation
- Process Injection
- User Execution
- (Distributed) Denial of Service Attack
- Remote Code Execution

The proper deployment and configuration of perimeter defences such as Firewalls and Intrusion Detection and Prevention Systems is key to address a wide range of threats. Furthermore, live threats that were encountered on the Pledger Honeypot, such as various types of Denial-of-Service attacks, malformed packets, failed handshakes, untrusted IPs and suspicious TOR exit traffic etc. were addressed and blocked by iptables-based firewalls.

The use of secure communication protocols was paramount to secure the Pledger architecture. Use of SSH or TLS is not always enabled by default and had to be configured on our infrastructure. Looking into the documentation of open-source components thus becomes necessary, in order to properly configure them and minimise security, privacy and trust risks.

The use of Trust and Reputation mechanisms was also considered, in order to secure interactions among untrusted, unknown entities. The Big Data, CI/CD, Blockchain and Kubernetes cluster for the project were sufficiently hardened against the uncovered threats. Privacy mechanisms were applied to the data streams, to ensure authorised access. Particular attention was paid to enable a secure environment for the development and delivery of trusted, certified applications. The project registry was secured against known threats that allow code injections, and runtime monitoring was applied to running application instances.

The analysis showed the potential to improve machine learning methods, though certain trade-offs need to be taken into account in terms of the information utilised, processing time required, deployment configurations etc. In order to have a production-level system, additional work is required. The existence of appropriate open datasets to train and further experiment with Machine Learning methods is a significant enabler in that respect.

References

[1] The Pledger Consortium, "D4.1 preliminary verson of "Trust, Privacy and Security related tools"," 2021.

[2] The Pledger Consortium, "D2.3 Pledger Overall Architecture," 2021.

[3] Jackson E. Wynn, The MITRE Corporation, "Threat Assessment and Remediation Analysis (TARA)," 2014.

[4] Apache Foundation, "Apache Kafka: a distributed streaming platform," [Online]. Available: https://kafka.apache.org/.

[5] E. Fernandez, "Security in Data Intensive Computing Systems," in *Handbook of Data Intensive computing*, Springer, 2011.

[6] O. Khalid, S. Khan, S. Madani, K. Hayat, M. Khan, N. Min-Allah, J. Kolodziej, L. Wang, S. Zeadally and D. Chen, "Comparative study of trust and reputation systems for wireless sensor networks," *Secure Communication Networks,* vol. 6, pp. 669-688, 2013.

[7] ENISA, "ENISA good practices for security of Smart Cars," 2019.

[8] H. Shahriar and H. Haddad, "Security Vulnerabilities of NoSQL and SQL Databases for MOOC Applications," *International Journal of Digital Society,* vol. 8, no. 1, 2017.

[9] "OpenStack security guide," [Online]. Available: https://docs.openstack.org/security-guide/. [Accessed 29 08 2022].

[10] "Pod Security Policies," [Online]. Available: https://kubernetes.io/docs/concepts/policy/pod-security-policy/.

[11] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *4th International Conference on Information Systems Security and Privacy (ICISSP)*, Portugal, 2018.

[12] C. Mathas, O. E. Segou, G. Xylouris, D. Christinakis, M. Kourtis and C. Vassilakis, "Evaluation of Apache Spot's machine learning capabilities in an SDN/NFC enabled environment," in *Proceedings of CyberTIM Workshop, ARES Conference*, Hamburg, Germany, 2018.

[13] D. Blei, A. Ng and M. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning,* vol. 3, pp. 993-1022, 2003.

[14] Canadian Institute for Cybersecurity, "Intrusion Detection Evaluation Dataset," [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html . [Accessed August 2022].

[15] J. L. Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *JSTOR The American Statistician,* vol. 42, no. 1, pp. 59-66, Feb 1988.

[16] S. Haykin, Neural Networks: A Comprehensive Foundation (2 ed.)., Prentice Hall, 1988.

[17] "scikit-learn Standard Scaler," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[18] S. Marti and H. Garcia-Molina, "Taxonomy of trust: categorizing P2P reputation systems", Computer Networks 2006," *Computer Networks,* 2006.

[19] R. C. J. T. J. Guo, "A survey of trust computation models for service management in internet of things systems," *Computer Communication,* vol. 97, pp. 1-14, 2017.

[20] MarketsAndMarkets, "Managed Security Services (MSS) Market by Security Type (Managed IAM, Managed SIEM, Managed Firewall, and MDR), Service Type (Fully MSS and Co-managed), Organization Size, Vertical (BFSI, Government, and Healthcare), and Region - Global Forecast to 2026".

[21] Allied Market Research, "Managed Security Services Market [..] Global Opportunity Analysis and Industry Forecast, 2021-2030".

[22] ENISA, "Cybersecurity for SMEs," [Online]. Available: https://www.enisa.europa.eu/topics/national-cyber-security-strategies/sme_cybersecurity.

[23] E. Segal, "Small businesses are more frequent targets for cyber attacks than large companies: New report".*Forbes.*

Annex A: Threat modeling methodology

The first step in the Pledger methodology is to compile a list of Tactics, Techniques and Procedures (TTPs) and prioritise them according to a severity score (Figure 34).

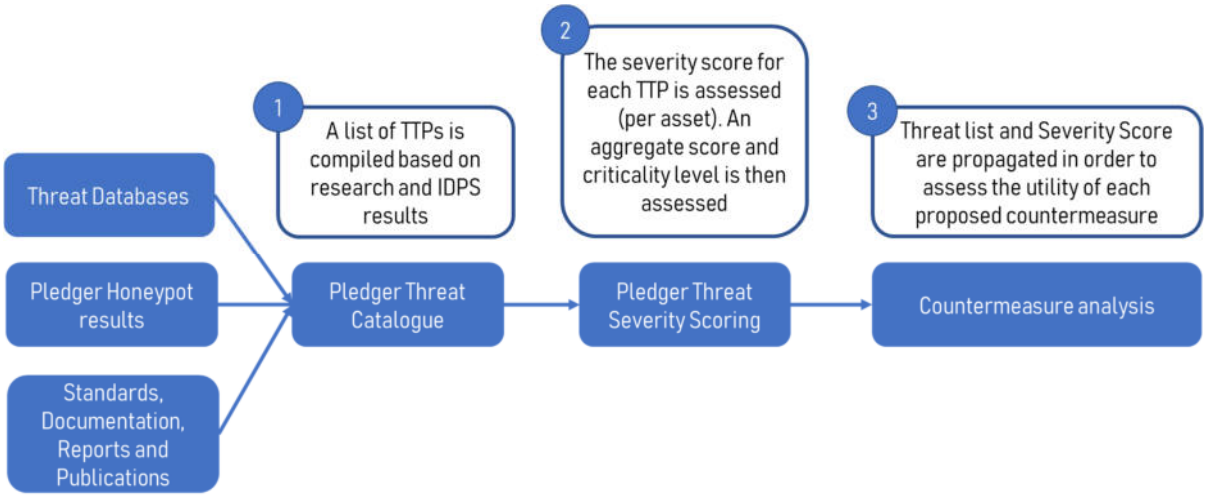


Figure 34: Overview of the Threat Assessment stage.

The next step would be to create a correlation matrix, associating the threats with the Pledger subsystems, as described in D2.3 [2]. A severity score is assessed for each pair of subsystem and threat, assessing the severity of the TTP. The scoring factors are selected from two broad categories:

- **Impact factors** that define how severe the effects of the TTP might be on the Pledger subsystems: examples are impacts to availability, latency etc.
- **Probability factors** that define how probable is that the risk of the TTP will materialize: examples are the known frequency of attacks, the required skill level of the attacker etc.

Table 15 provides an overview of the scoring factors selected for Pledger. For each TTP/information asset pair, an average or weighted score can be calculated. An aggregate score is then calculated and translated to a criticality level. Although MITRE provides a scoring system, it also suggests that it should be adapted to suit the needs of each analysis. Pledger developed its scoring based on the impact of a TTP and propagates the threat severity score to the selection process of the countermeasures.

The following figure shows how the aggregated score is translated to a criticality level. An important consideration in threat analysis is the level of information required. More detailed analysis can be performed on the basis of the specific digital assets (software, hardware, storage and network), for example by specific software versions and device models. This provides a more actionable list of countermeasures and can provide a higher level of protection in a realistic production environment. On the other hand, high-level analysis benefits in that it can be reused more readily.

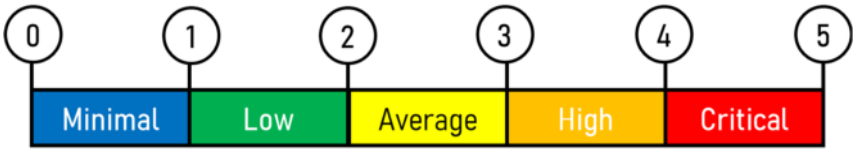


Figure 35: Threat Severity level scale.

The second stage of the analysis focuses on the identification and prioritisation of measures to counter TTPs against Cloud-Edge deployments.

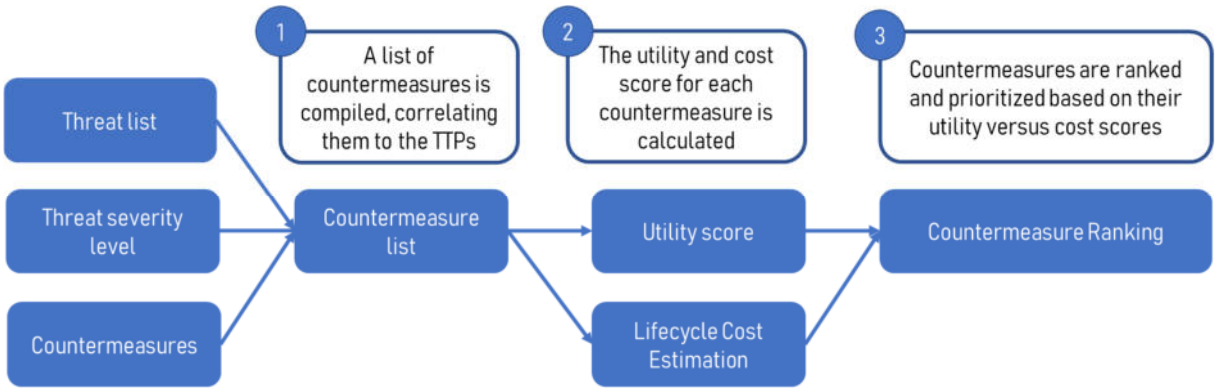


Figure 36: Countermeasure scoring method.

The prioritization of countermeasures is performed by assessing their **utility** in addressing a specific cyber threat as well as the overall cost for their implementation and operation. The utility score is selected according to the following factors:

- **The rate of success:** When applicable, countermeasures that have a higher rate of success are scored higher than countermeasures with a lower rate of success (i.e., detection rate, false negative rate, etc.)
- **The effects of the countermeasures:** Preventative measures are scored higher, followed by mitigation measures, and measures that limit the effects of an attack. Detection measures are scored lower.
- **The criticality of the threat:** The utility of a countermeasure that addresses a critical threat is scored higher than the utility of a countermeasure that addresses a minimal threat.

Table 16 provides the utility score system and a few tentative examples. The maximum utility score (u_{max}=5) is attributed to a countermeasure that has a *"HIGH"* chance to *"PREVENT"* a *"CRITICAL"* threat. The lowest possible score (u_{min}=0.1) is attributed to a countermeasure that has a *"LOW"* chance to *"DETECT"* a *"MINIMAL"* threat.

After the Utility Score has been selected, MITRE proposes the use of Life Cycle Cost estimators (LCC) to assess the cost to acquire and operate a specific countermeasure. The costs are broken down to the acquisition and operation costs. For the purpose of the project, we consider the cost in terms of human effort and cloud resources, translated in a scale of [1, 5] (Table 17). In a real-life deployment, a more comprehensive analysis can be performed to propose an accurate and realistic benchmark. Finally, the ratio of Utility Score versus Cost, provides the final score for each countermeasure. Countermeasures can then be prioritized on the basis of the highest utility to cost ratio.

Table 15: Pledger Threat Scoring: A weighted score is calculated across these factors.

		Minimal	Low	Average	High	Critical
		1	2	3	4	5
Impact	**Severity:** How severe are the impacts? Is the attack's effect localised on one or can it affect many subsystems?	Minimal impact, localised to the asset	Average impact, localised to the asset	Average impact, can affect the performance of other assets as well	High impact, critically affects one or more assets performance	High impact, critically affects one or multiple components of the Pledger architecture
Impact	**Loss of confidentiality:** Is there potential for loss of confidentiality, i.e., data leak, user accounts and passwords, orpayment information, among others?	Minimal impact to data confidentiality	Low impact to data confidentiality, data leak is unlikely	Some non-critical data may be leaked	A large volume of non-critical data or a small volume of critical data may be leaked	A large volume of critical data may be leaked
Impact	**Impact to availability:** Is there loss of availability?	a few seconds	a few minutes	a few hours	a day	more than a day
Impact	**Impact to latency\QoS:** Does the attack have the potential to affect latency-critical applications? Does it degrade a service, so it is unusable?	minor impact to latency/QoS, performance degradation is barely noticeable	low impact to latency/QoS, performance degradation is noticeable, but usability is not threatened	average impact to latency/QoS, usability is starting to be hindered	high impact to latency/QoS, usability is very hindered	critical impact, the service is too degraded to be useful

		Minimal	Low	Average	High	Critical
		1	2	3	4	5
Impact	**Impact to integrity:** What is the impact to integrity, i.e., integrity of data, integrity of user accounts, integrity of configurations, integrity of infrastructure?	minor subcomponent compromised	low chance of a subsystem being compromised	average chance that a subsystem is compromised	High chance one subsystem is compromised	High chance that multiple subsystems are breached or otherwise compromised
Impact	**Attack recovery:** How long would it take to recover from such an attack? How many recovery actions are needed? i.e., bringing services back online, restoring back-up data or user accounts)	minutes	a few hours	a day	three days	more than three days
Probability	**Complexity of attack:** Is the attack easily deployed? Does it require highly advanced vectors and payloads?	attack is very complex, with multiple advanced attack vectors and payloads	attack requires and advanced vector/payload	advanced tools may be necessary	tools and processes to launch the attack are widely available	It is fairly common and there are multiple existing tools, leveraging automation
Probability	**Frequency:** Are there recent attacks in a threat database? Known cybersecurity events? Is it a frequent occurrence?	Almost no recent examples were found	Sparse evidence of this attack was found	Confirmed recent cases	Frequent recent cases	Multiple recent examples of this attack exist, attack is widespread

		Minimal	Low	Average	High	Critical
		1	2	3	4	5
Probability	**Detectability:** How detectable is the attack?	Can easily be detected even without advanced monitoring	Can be detected once its impact becomes visible	Requires constant intrusion detection	Requires advanced monitoring and anomaly detection/SIEM	Can easily go undetected, even with advanced monitoring
Probability	**Requires skilled attacker:** Does the attack require highly skilled attackers? Are there easy to deploy automated tools to deploy the attack at scale?	requires highly organised and skilled attackers (e.g., Advanced Persistent Threat)	requires highly skilled attacker(s) or insiders	requires attacker(s) with technical know-how to develop & deploy the attack	moderately easy to deploy using existing tools	easy to deploy, does not require highly skilled attacker

Table 16: Utility Score selection.

		THREAT LEVEL														
		MINIMAL [0.1,1]			LOW [0.2,2]			AVERAGE [0.3,3]			HIGH [0.4,4]			CRITICAL [0.5,5]		
		1			2			3			4			5		
UTILITY SCORE	Prevent	1	0.8	0.6	2	1.6	1.2	3	2.4	1.8	4	3.2	2.4	5	4	3
	Mitigate	0.8	0.6	0.4	1.6	1.2	0.8	2.4	1.8	1.2	3.2	2.4	1.6	4	3	2
	Limit	0.6	0.4	0.2	1.2	0.8	0.4	1.8	1.2	0.6	2.4	1.6	0.8	3	2	1
	Detect	0.4	0.2	0.1	0.8	0.4	0.2	1.2	0.6	0.3	1.6	0.8	0.4	2	1	0.5
		High	Medium	Low	High	Medium	Low	High	Medium	Low	High	Medium	Low	High	Medium	Low
		95%>	50-95%	<50%	95%>	50-95%	<50%	95%>	50-95%	<50%	95%>	50-95%	<50%	95%>	50-95%	<50%
		Rate of success			Rate of success			Rate of success			Rate of success			Rate of success		

LEGEND

Highest Utility Score: good rate of success to mitigate the threat

Lowest Utility Score: poor rate of success to detect

EXAMPLES:
This countermeasure has a HIGH chance to MITIGATE a CRITICAL threat:
Utility Score = 4
This countermeasure has a LOW chance to DETECT a HIGH criticality threat:
Utility Score = 0.4
This countermeasure has a MEDIUM chance to PREVENT an AVERAGE criticality threat:
Utility Score = 2.4

Table 17: Life Cycle Cost estimation.

		Very Low	Low	Medium	High	Very High
	LEGEND	**1**	**2**	**3**	**4**	**5**
Acquisition	Cost to research	<1PM	<2PM	<4PM	<6PM	>6PMs
	Cost to develop	simple development, requires security configuration of existing component, low effort for one partner	simple development effort for one partner	average development effort for one partner	average development effort from multiple partners	Development is complex and requires contribution from multiple partners
	Cost to deploy	simple configuration can be applied in existing Pledger infrastructure	simple deployment in a single VM or container	average development effort for one partner	4-6 VMs or containers	deployment in multiple VMs, additional infrastructure costs
	Cost to integrate	easy to integrate to Pledger	single integration point to Pledger components, over standardised interfaces	single integration point to Pledger components over non-standardised interfaces	several integration points exist with Pledger components, through standard interfaces	multiple integration points with Pledger components, standard interfaces are not supported
Utilisation	Cost to operate	minimal cloud resources	single integration point to Pledger components, over standardised interfaces	average development effort for one partner	multiple VMs/containers incur medium infrastructure costs, but are not resource heavy	multiple integration points with Pledger components, standard interfaces are not supported

		Very Low	Low	Medium	High	Very High
	LEGEND	**1**	**2**	**3**	**4**	**5**
	Cost to train	requires minimal training	1PM	>2PM	>3PM	requires training and specific skills to operate
	Cost to maintain	minimal cost to maintain	some reconfiguration is necessary	medium reconfiguration and maintenance are required	frequent maintenance and reconfiguration is needed	requires constant upgrades, monitoring and maintenance in order to be utilised by the project
	Cost to dispose	simple to dispose	some reconfiguration of Pledger components might be required	requires disposing of multiple components and revert configurations	close integration with another Pledger component makes it difficult to dispose	close integration with multiple Pledger components makes it difficult to dispose

Table 18 Countermeasures per threat list.

Countermeasure ID (at R1)	Countermeasure ID (at R2)	Countermeasure	Utility/Cost Score	Cost	Total Utility Score	Sensitive Data Exposure	Account Manipulation	Process Injection	(Distributed) Denial of Service Attack	User Execution	Remote Code Execution	Insecure Application API	Modify System Image	Kubernetes administration command	Orchestrator risks	Network related threats	Create or Modify System Process	Exploitation for Privilege Escalation	Escape to Host	Root SSH brute force attack	Container risks	Malicious collectives	Cloud Service Dashboard	Unsecured Credentials	RST Injection	Valid Accounts	Sybil Attack	Container image vulnerabilities	Threats to SLA subsystem/T&R components	Traffic from Bad Reputation IPs	Cloud Infrastructure Discovery	Manipulation of transmitted data	Threats to IaaS evaluation/T&R indices DBs	K8s Application Vulnerability	False Friends / Eclipsing Attack	V2X DoS	Network Sniffing	Man-in-the-middle attack	Data from Local System	Infrastructure discovery	Access control exploits	Query injection attacks	K8s Node Resource Starvation/DoS	Container and Resource Discovery	Registry risks	XSS attacks	Jamming attacks	Environmental Manipulation	Oscillating malicious service provision	Malicious a posteriori evaluation	V2X Spoofing	Decryption Key Leaked	Spying on the Endpoint
						TH001	TH002	TH003	TH004	TH005	TH006	TH007	TH008	TH009	TH010	TH011	TH012	TH013	TH014	TH015	TH016	TH017	TH018	TH019	TH020	TH021	TH022	TH023	TH024	TH025	TH026	TH027	TH028	TH029	TH030	TH031	TH032	TH033	TH034	TH035	TH036	TH037	TH038	TH039	TH040	TH041	TH042	TH043	TH044	TH045	TH046	TH047	TH048
C01	C01	Privileged Account Management	19.05	1.05	20		4	4					4										4	4																														
C02	C02	Intrusion Detection and Prevention System	16.67	2.1	35			4	4		5	3													4					4	4					4								3										
C03	C03	Encrypt sensitive information in transit (SSL/TLS)	14.96	1.15	17.2				3																						4	4						3.2	3														
C04	C04	Firewall setup/Blacklist policies	14	2	28			4			5														4					4	4					4								3										
C05	C05	Enforce secure password policies	11.67	1.2	14		4													4						3													3															
C06	C06	Application Isolation and Sandboxing	9.68	1.9	18.4			4			4							4	2.4															4																				
C07	C07	Access Control Lists/ Authorisation	8.28	2.9	24		4				5	3					4							4								4																						
C08	C08	Least privilege access model	7.86	1.4	11					4					4																								3															
C09	C09	Avoid JavaScript functions to parse user input	7.83	1.15	9																																		3				3										
C10	C10	Adjust container security policies	7.33	2.4	17.6						4		4				3.2		2.4															4																				
C11	C11	Execution Prevention	7.17	2.65	19					3	4	4						4																4																				
C12	C12	Multifactor authentication	6.61	2.3	15.2		4								4								3.2																		4													
C13	C13	Recent version of Geth	6.5	1.6	10.4				4																		2.4								4																			
C14	C14	Use trusted network	6.11	1.8	11				3														4								4																								
C15	C15	Code Signing	5.86	2.9	17					5	4		4				4																																					
C16	C16	User account management	5.83	2.4	14		4																	4																		3			3										
C17	C17	Pulling images only from trusted sources	5.83	1.2	7									4																																3								
C18	C18	Encrypt Sensitive Information at Rest	5.22	2.3	12	4													3.2							3.2																												1.6
C19	C19	Software Upgrade to higher version	5	0.8	4				4																																													
C20	C20	Monitoring and remediating container runtimes	4.79	3.3	15.8						3.2			4							2.4													3.2									3											
C21	C21	Sanitize and validate user input	4.57	1.05	4.8																																					2.4				2.4								
C22	C22	Update Software	4.36	1.65	7.2													3.2																4																				
C23	C23	Remove sensitive data from responses	3.6	2	7.2	4																															3.2																	

Countermeasure ID (at R1)	Countermeasure ID (at R2)	Countermeasure	Utility/Cost Score	Cost	Total Utility Score	Sensitive Data Exposure	Account Manipulation	Process Injection	(Distributed) Denial of Service Attack	User Execution	Remote Code Execution	Insecure Application API	Modify System Image	Kubernetes administration command	Orchestrator risks	Network related threats	Create or Modify System Process	Exploitation for Privilege Escalation	Escape to Host	Root SSH brute force attack	Container risks	Malicious collectives	Cloud Service Dashboard	Unsecured Credentials	RST Injection	Valid Accounts	Sybil Attack	Container image vulnerabilities	Threats to SLA subsystem/T&R components	Traffic from Bad Reputation IPs	Cloud Infrastructure Discovery	Manipulation of transmitted data	Threats to IaaS evaluation/T&R indices DBs	K8s Application Vulnerability	False Friends / Eclipsing Attack	V2X DoS	Network Sniffing	Man-in-the-middle attack	Data from Local System	Infrastructure discovery	Access control exploits	Query injection attacks	K8s Node Resource Starvation/DoS	Container and Resource Discovery	Registry risks	XSS attacks	Jamming attacks	Environmental Manipulation	Oscillating malicious service provision	Malicious a posteriori evaluation	V2X Spoofing	Decryption Key Leaked	Spying on the Endpoint
						TH001	TH002	TH003	TH004	TH005	TH006	TH007	TH008	TH009	TH010	TH011	TH012	TH013	TH014	TH015	TH016	TH017	TH018	TH019	TH020	TH021	TH022	TH023	TH024	TH025	TH026	TH027	TH028	TH029	TH030	TH031	TH032	TH033	TH034	TH035	TH036	TH037	TH038	TH039	TH040	TH041	TH042	TH043	TH044	TH045	TH046	TH047	TH048
C24	C24	HTTPS reverse proxy	3.33	1.8	6																											4																						2
C25	C25	Prepared statements	3.31	1.45	4.8																																				2.4				2.4								
C26	C26	Network anomalies detection	3.29	3.4	11.2					4	4					3.2																																					
C27	C27	Limit access to resource over network	3.26	2.15	7																		4																					3									
C28	C28	Private key management	3	1	3																																																3	
C29	C29	Container image vulnerabilities detection	2.84	3.1	8.8									4														1.6						3.2																				
C30	C30	Container image vulnerabilities management tools	2.82	3.4	9.6									4														2.4						3.2																				
C31	C31	Encrypt Sensitive Information	2.77	2.6	7.2	4																															3.2																	
C32	C32	Decoupled T&R Indices	2.61	2.3	6																	3																											3				
C33	C33	Discrete virtual networks and isolated applications	2.55	2.75	7										4																								3														
C34	C34	Limit software installation	2.38	2.1	5						5																																										
C35	C35	Attack surface reduction	2.33	2.7	6.3									3.9															2.4																								
C36	C36	Using Blockchain for logs storage	2	2.6	5.2																												4							1.2														
C37	C37	RF Hardening	1.8	5	9																																									3	3			3		
C38	C38	Configuration only to registries over encrypted channels	1.58	1.9	3																																							3									
C39	C39	Tampering detection techniques and tools	1.52	2.9	4.4																												3.2							1.2														
C40	C40	Network Segmentation	1.25	2.4	3																																							3									
C41	C41	Assignment of trusted nodes	1.21	2.65	3.2																	3.2																																
C42	C42	Weights, time-windows, fading factors for calculations	1.07	2.8	3																																											3					
C43	C43	Testing of proper Dependability calculation formula	1.07	3	3.2																	3.2																																
C44	C44	Evaluators evaluation	0.97	3.1	3																																													3				
C45	C45	Regular cross-check of the appropriate results/models	0.96	2.5	2.4																								2.4																									

Annex B: History of changes since D4.1

Affected subsection	Change type	Description	Pages
D4.1 Chapter 3	**Deleted**	Chapter 3 was removed	-
Annex A	**Added**	Important information for the threat scoring added in annex (originally in D4.1/Chapter 3)	76-83
Section 1	**Modified**	Modified introduction, added Task methodology. Also, new material added to reflect security benefits per stakeholder and across the Pledger value chain	9-12
Section 2	**Modified**	Slight modifications to the architecture. Removed subsections 2.2-2.4	13-14
Subsection 3.1	**Added**	Small description of threat modelling added. The rest of the information is annexed.	15
Subsections 4.1, 4.2.X, 4.3.1, 4.3.3	**Modified**	Updated to reflect changes till August 2022, added more technical detail on hardening measures and security assets.	42-63
Subsection 4.3.2, 4.3.2.X	**Added**	New results on the use of Machine Learning for cybersecurity were added	52-58
Section 5	**Added**	Added chapter on demonstrations to reflect the utility of security assets	64-72
Section 6	**Modified**	Updated to reflect status as of August 2022	73