



D4.1 Trust, Privacy and Security related tools

Document Identification			
Status	Final	Due Date	31/07/2021
Version	1.0	Submission Date	30/07/2021

Related WP	WP4	Document Reference	D4.1
Related Deliverable(s)	D2.3	Dissemination Level (*)	PU
Lead Participant	INTRA	Lead Author	Olga E. Segou, PhD Ioannis Sarris Ioannis Sarantidis
Contributors	FILL, HOLO, i2CAT, ICCS, ENG	Reviewers	Gabriele Giammatteo, ENG
			Antonio Castillo Nieto, ATOS

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

This document is issued within the frame and for the purpose of the PLEDGER project. This project has received funding from the European Union's Horizon2020 Framework Programme under Grant Agreement No. 871536. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

[The dissemination of this document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains. This deliverable is subject to final acceptance by the European Commission.

This document and its content are the property of the PLEDGER Consortium. The content of all or parts of this document can be used and distributed provided that the PLEDGER project and the document are properly referenced.

Each PLEDGER Partner may use this document in conformity with the PLEDGER Consortium Grant Agreement provisions.

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

Document Information

List of Contributors	
Name	Partner
Olga E. Segou, PhD	INTRA
Ioannis Sarris	INTRA
Ioannis Sarantidis	INTRA
Gabrielle Giammatteo	ENG
Francesco Iadanza	ENG
Verena Stanzl	FILL
Orfeas Voutyras	ICCS
Alexandros Psychas	ICCS
Antonis Litke	ICCS
Estrela Carmona Cedujo	i2CAT
August Betzler	i2CAT
Bruno Cordero	i2CAT
Adrian Pino	i2CAT
Nour Fendri	HOLO
Carina Pamminger	HOLO

Document History			
Version	Date	Change editors	Changes
0.1	01/05/2021	Olga E. Segou, PhD	Document structure created/Table of Contents
0.2	25/05/2021	Olga E. Segou, PhD	Executive Summary, Chapter 1 provided by INTRA
0.3	07/06/2021	Olga E. Segou, PhD	Merged Section 2 Contributions, ICCS (2.1), HOLO (2.2), I2CAT (2.3), FILL (2.4), Merged Section 3 (provided by INTRA)
0.4	14/06/2021	Olga E. Segou, PhD	Merged Section 4 Contributions by ENG (4.1.1, 4.1.6), INTRA (4.1.3, 4.1.4), HOLO (4.1.5, 4.1.8), I2CAT (4.1.9), FILL (4.1.10)
0.5	21/06/2021	Olga E. Segou, PhD	Merged Section 5 contributions by ENG (5.2.3), INTRA (5.2.1, 5.2.2, 5.3.1), Merged Section 4 contributions by ICCS (4.1.2, 4.1.7),
0.6	28/06/2021	Olga E. Segou, PhD	Merged Section 5 INTRA (5.1), ICCS (5.3.2)
0.7	8/07/2021	Olga E. Segou, PhD	INTRA finalised Subsection 4.2
0.8	15/07/2021	Olga E. Segou, PhD	Chapter 6 added by INTRA, HOLO (5.2.4)
0.9	21/07/2021	Olga E. Segou, PhD	Addressed reviewer's comments (ENG and ATOS)
1.0	28/07/2021	Olga E. Segou, PhD	Final version

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Olga E. Segou, PhD (INTRA)	26/07/2021
Quality manager	Carmen San Roman Alonso (ATOS)	28/07/2021
Project Coordinator	Lara Lopez Muñiz (ATOS)	30/07/2021

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	2 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

Table of Contents

Document Information	2
Table of Contents	3
List of Tables.....	5
List of Figures	6
List of Acronyms.....	7
Executive Summary	8
1 Introduction	9
1.1 Purpose of the document.....	9
1.2 Relation to other project work.....	9
1.3 Structure of the document	10
2 Pledger Overview	11
2.1 Pledger Core.....	11
2.2 Use Case 1: Mixed Reality Applications	13
2.3 Use Case 2: Edge infrastructure for enhancing the safety of Vulnerable Road Users (VRUs) ...	14
2.4 Use Case 3: Data Mining on the Edge for Manufacturing.....	15
3 Pledger security approach.....	17
3.1 Approach to securing the Pledger platform.....	17
3.2 Cyber Threat Modelling approach	18
3.2.1 Methodology	19
4 Threat modelling	29
4.1 Threat descriptions per Pledger information asset.....	29
4.1.1 Configuration and Decision Support System	29
4.1.2 Orchestration	31
4.1.3 Big Data Management.....	34
4.1.4 Continuous Integration/Continuous Delivery (CI/CD)	35
4.1.5 Blockchain.....	37
4.1.6 Benchmarking	38
4.1.7 QoS, SLAs, and T&R assessment and management systems	40
4.1.8 UC1 subsystem.....	42
4.1.9 UC2 subsystem.....	44
4.1.10 UC3 subsystem.....	46
4.1.11 Other underlying cloud infrastructure	48
4.2 Threat & countermeasure prioritisation	51
5 Architecture of privacy, security and trust components	56
5.1 High level architecture	56

Document name:	D4.1 Trust, Privacy and Security related tools				Page:	3 of 73
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status: Final

5.2 Infrastructure Hardening	56
5.2.1 Streamhandler Big Data Platform	56
5.2.2 CI/CD	57
5.2.3 K8S Cluster	58
5.2.4 Blockchain security through the Whisper protocol	59
5.3 Deployed Security Assets	60
5.3.1 Distributed IDPS Cybersecurity Solution	60
5.3.2 Trust and Reputation	63
6 Conclusions and Next Steps	69
6.1 Key results.....	69
6.2 Plan for future activities.....	69
7 References	72

Document name:	D4.1 Trust, Privacy and Security related tools				Page:	4 of 73
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status: Final

List of Tables

Table 1 Security subsystem Components.	12
Table 2 Pledger Threat Scoring: A weighted score is calculated across these factors.	21
Table 3 Utility Score selection.	26
Table 4 Life Cycle Cost estimation.	27
Table 5 Common threats for the Configuration & DSS services.	29
Table 6 Major threats for the Orchestration subsystem.	31
Table 7 Common threats for the Big Data infrastructure.	34
Table 8 CI/CD Infrastructure threats.	36
Table 9 Blockchain threats.	37
Table 10 Benchmarking system threats.	39
Table 11 Major threats for QoS, SLAs, and T&R assessment and management systems.	41
Table 12 UC1 threats.	43
Table 13 UC2 threats.	44
Table 14 UC3 threats.	47
Table 15 Honeypot alert logs for a 7-day period.	49
Table 16 Threats against the underlying Pledger infrastructure.	50
Table 17 (a) Threat prioritisation, (b) Countermeasure prioritisation.	52
Table 18 Reference list of penetration testing tools publicly available.	70

List of Figures

Figure 1: The Pledger Core Subsystems.	11
Figure 2 Security Subsystem draft architecture	12
Figure 3 UC1 component diagram.	13
Figure 4 UC2 component diagram.	15
Figure 5 UC3 component diagram.	16
Figure 6 Security, Privacy and Trust methodological approach.	17
Figure 7: Pledger security pillars and key developments.	18
Figure 8 Threat Assessment and Remediation Analysis (TARA) overview.	19
Figure 9 Overview of the Threat Assessment stage.	19
Figure 10 Threat Severity level scale.	20
Figure 11 Countermeasure scoring method.	24
Figure 12 Sources of threat information.	29
Figure 13 Common attacks in T&R systems.	41
Figure 14 Basic honeypot statistics for a 7-day period.	49
Figure 15 Threat Severity basic statistics.	51
Figure 16 Pledger Security, Privacy and Trust tools and measures.	56
Figure 17 Whisper protocol architecture.	60
Figure 18 Distributed Cybersecurity platform architecture.	61
Figure 19 Threat Detection Mode – IDS.	63
Figure 20 Honeypot Mode – IDS.	63
Figure 21 General steps followed in T&R models.	64
Figure 22 TRMSim-WSN.	68

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	6 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

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
EC	European Commission
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
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
VRU	Vulnerable Road User
WP	Work Package
XSS	Cross-Site Scripting

Executive Summary

Pledger is an innovative project that delivers a new architectural paradigm and a toolset that will pave the way for next generation edge computing infrastructures. The project tackles the modern challenges faced today and coupling the benefits of low latencies on the edge, with the robustness and resilience of cloud infrastructures.

The dynamic and decentralised nature of Cloud-Edge deployments, however, 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 then compiled, to address the risks uncovered by the preliminary threat analysis. Countermeasures are prioritised according to their overall utility and cost. The utility is assessed on the basis of the criticality of the threat they address, the effect of the countermeasures (in terms of prevention, mitigation, limitation, or detection of a threat) and the rate of success. The cost is assessed based on the acquisition and operational costs of a specific countermeasure. A plan to deploy specific countermeasures is provided, listing measures to harden the Cloud-Edge infrastructure and functional components of the Pledger platform, as well as a list of specific security, privacy and trust assets deployed within the project.

Preliminary 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 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 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 Trust and Reputation subsystem, 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.

Document name:	D4.1 Trust, Privacy and Security related tools			Page:	8 of 73
Reference:	D4.1	Dissemination:	PU	Version:	1.0
				Status:	Final

1 Introduction

1.1 Purpose of the document

Pledger is an innovative project that delivers a new architectural paradigm and a toolset that will pave the way for next generation edge computing infrastructures. The project tackles the modern challenges faced today and coupling the benefits of low latencies on the edge, with the robustness and resilience of cloud infrastructures. The project will deliver a set of tools and processes that will enable:

- a) **edge computing providers** to enhance the stability and performance effectiveness of their edge infrastructures, through modelling the overheads and optimal groupings of concurrently running services, runtime analysis and adaptation, thus gaining a competitive advantage,
- b) **edge computing adopters** to understand the computational nature of their applications, investigate abstracted and understandable QoS metrics, facilitate trust and smart contracting over their infrastructures and identify how they can balance their cost and performance to optimise their competitiveness and monitor their SLAs.

By providing this toolset, the project will also allow third parties to act as independent validators of QoS features in IoT applications, enabling new decentralised applications and business models, thus filling a large gap in the emerging edge/IoT computing market landscape.

The complex and decentralised nature of Edge-Cloud infrastructures, along with their dynamic nature introduces cyber risks. When applications and services can be instantiated and turn down in seconds, have critical QoS demands and perform data-intensive operations, it is crucial to ensure that the infrastructure is appropriately hardened, and proper cybersecurity assets are in place to address evolving cyber threats and ensure privacy and service continuity.

This document focuses on the security, privacy and trust mechanisms applied in 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 T4.1 is also herein provided.

1.2 Relation to other project work

This document provides reference to the deliverable D2.3 “Pledger Overall Architecture” [1], as the analysis herein is performed based on the Pledger conceptual architecture. An overview of the Pledger architecture is provided in the following Section.

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	9 of 73				
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status:	Final

1.3 Structure of the document

This document is organised as follows:

Chapter 1 “Introduction” (present chapter) serves as an introduction to the document and delineates its scope.

Chapter 2 “Overview of the Pledger platform” provides an overview of the Pledger core platform and the project Use Cases.

Chapter 3 “Threat analysis methodology” presents the threat analysis methodology and scoring system.

Chapter 4 “Threat analysis” provides the preliminary threat analysis conducted for the Pledger Core and Use Cases, including the prioritisation of suggested countermeasures.

Chapter 5 “Pledger Security Architecture” provides the description of the security components and infrastructure hardening measures applied in Pledger.

Chapter 6 “Conclusions and Next Steps” provides the final conclusions of the document as well as the plans for the coming months.

Document name:	D4.1 Trust, Privacy and Security related tools				Page:	10 of 73	
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status:	Final

2 Pledger Overview

This section provides an overview of the Pledger Core and UC architecture, as defined in D2.3, 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” and provides useful information related to the development of the corresponding Pledger **Security subsystem** and **Big Data Platform subsystem**, as identified in Deliverable “D2.3 Pledger Overall Architecture” [1] which acts as the roadmap for all development and integration tasks of the project.

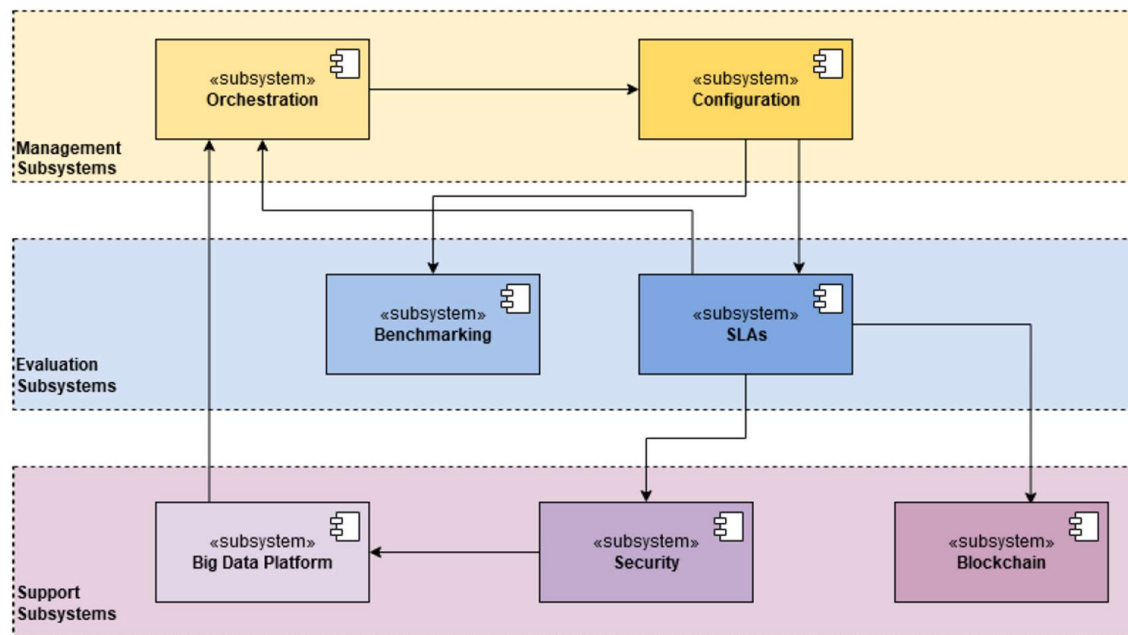


Figure 1: The Pledger Core Subsystems.

As also illustrated in D2.3, 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. Of course, several other security features and mechanisms are included in other components of other subsystems, but this subsystem is specifically focused on the security aspects of Pledger. Figure 2 provides the security subsystem architecture, improved upon based on the results of the threat analysis performed within Pledger. Section 5 updates the architecture for security components and provides their implementation details, as of July 2021.

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	11 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

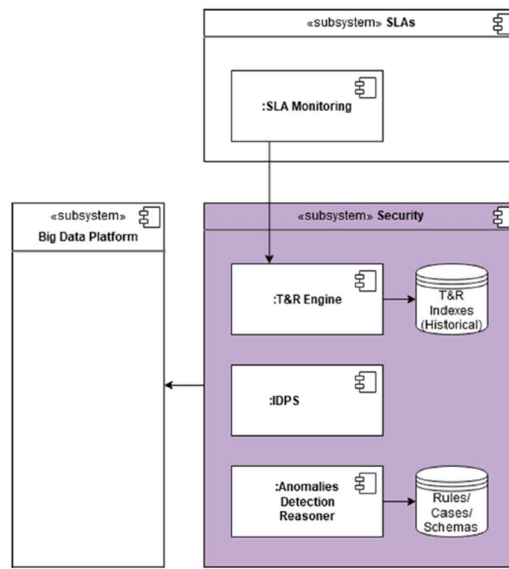


Figure 2 Security Subsystem draft architecture

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. Several reasoning techniques will be considered for the implementation of this component, including Rules Based Reasoning (RBR), Case Based Reasoning (CBR), and ML algorithms.
FC.7.3	Rules/ Cases/ Schemas	This 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 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 are presented in more detailed in Section 5. The same section also includes some Security practices and solutions implemented in Pledger that do not take the form a separate component and, as such, are not presented in the above tables. 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.

2.2 Use Case 1: Mixed Reality Applications

The goal of Use Case 1 (UC1) is to enhance the capabilities of Augmented/Mixed/Virtual Reality (AR/MR/VR) solutions by coupling them with edge computing technologies and the corresponding load allocation and optimisation tools, in order to provide high-level services in industrial environments. Specifically, the industrial environment which will run the pilots of UC1 is the one provided in Use Case 3 [2]. In particular, the UC will focus on three case studies integrating machine data into MR interfaces and optimising them via edge computing technology:

- ▶ Collaborating in Fast Prototyping,
- ▶ Assisting in Manufacturing and Service & Utility procedures,
- ▶ Enhancing Training & Interaction,

The following Component diagram is extracted for this specific Use Case.

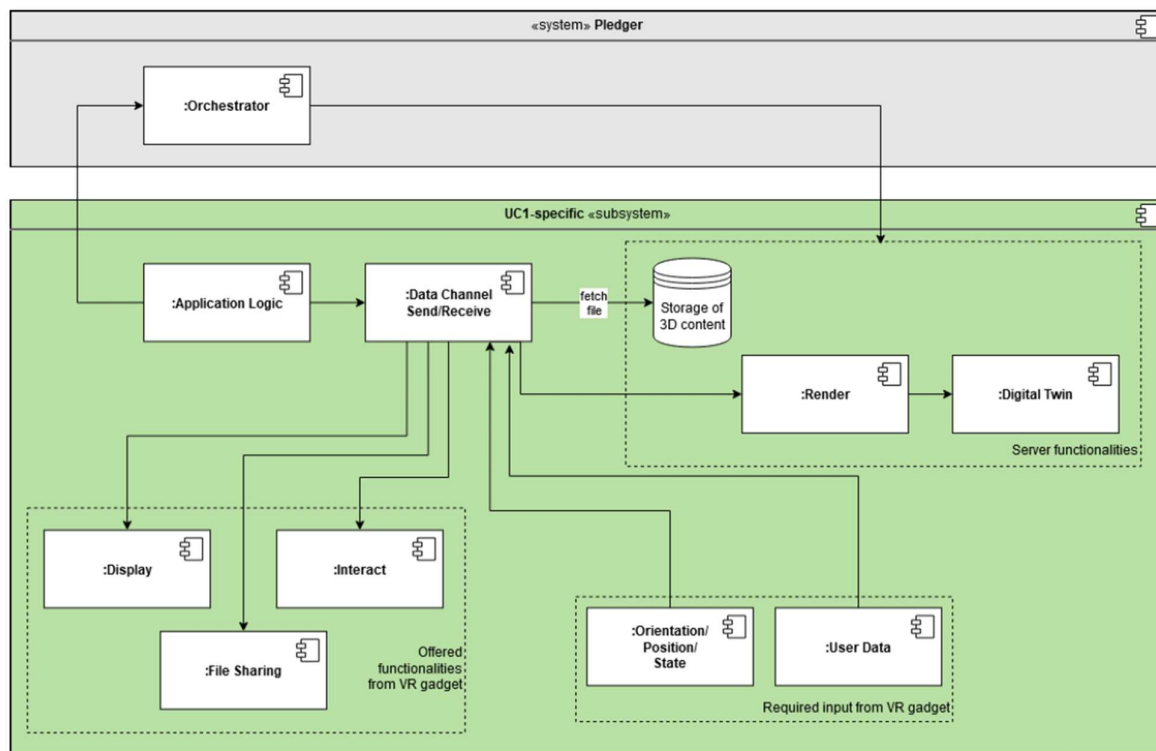


Figure 3 UC1 component diagram.

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	13 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

2.3 Use Case 2: Edge infrastructure for enhancing the safety of Vulnerable Road Users (VRUs)

Over the last years in Europe, the number of traffic accidents resulting in critical injuries or deaths has been decreasing. Yet, the number of accidents involving vulnerable road users (VRUs) in cities has not seen this steady decrease and has even increased in some cities. One reason is that in some cities, the road infrastructure is not designed to support an increasing number of VRUs, such as bicycles or electric scooters. The regulations of the mobility of bikes and scooters are often missing or not correctly applied and as such, these VRUs are often involved in accidents.

In general terms, increasing the security of VRUs is a task that can be accomplished in many ways: Safety can come from having a well-defined regulatory framework. This ensures that VRUs and their interactions with other vehicles and the infrastructure follow a set of rules and behaviour of each participant in the traffic is well-defined. Another way to increase safety is to build dedicated infrastructure elements and by adapting the city layout to respect the mobility of VRUs. This can come in the form of dedicated bicycle lanes or pedestrian areas, a methodology that is widely pushed in cities to generate safe spaces for VRUs. State of the art ICT technologies can also be used to implement a “smart” infrastructure that increases safety: using sensors to gather data about the position and trajectory of VRUs and other traffic participants, algorithms can be used to process the gathered information and coupled with notification systems, VRUs can be alerted if situations of risk are detected. This latter approach is investigated in this UC.

This use case explores the use of the Pledger platform together with a series of infrastructure enhancements to increase the safety for said VRUs in a specific area of the city, where pedestrians exiting the public transport must cross a bicycle lane. By integrating Pledger with Barcelona’s city infrastructure available to the project, we can introduce risk avoidance mechanisms for said pedestrians and cyclists or users of electric scooters: using radio equipment, networking infrastructure, compute nodes and dedicated applications – all controlled by Pledger – risky situations for the VRUs can be detected and measures can be taken to reduce those risks. The city infrastructure offers a multi-tier compute node hierarchy (cloud and edge) and the virtualized risk detection and notification system can be deployed on top of this infrastructure. Underneath these elements that are integrated with the Pledger platform, the radio infrastructure and a detection system based on cameras is responsible for feeding the risk detection and notification service with the locations data of the VRUs. The VRU location detection is powered by a V2X stack that enables communication between onboard units and roadside units over 802.11p. Further, the detection of other elements, such as pedestrians and public transport is handled by an external solution that uses cameras and image recognition and feeds this information to the risk detection and notification system, enabling to alert VRUs in time to avoid possible accidents.

Document name:	D4.1 Trust, Privacy and Security related tools				Page:	14 of 73
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status: Final

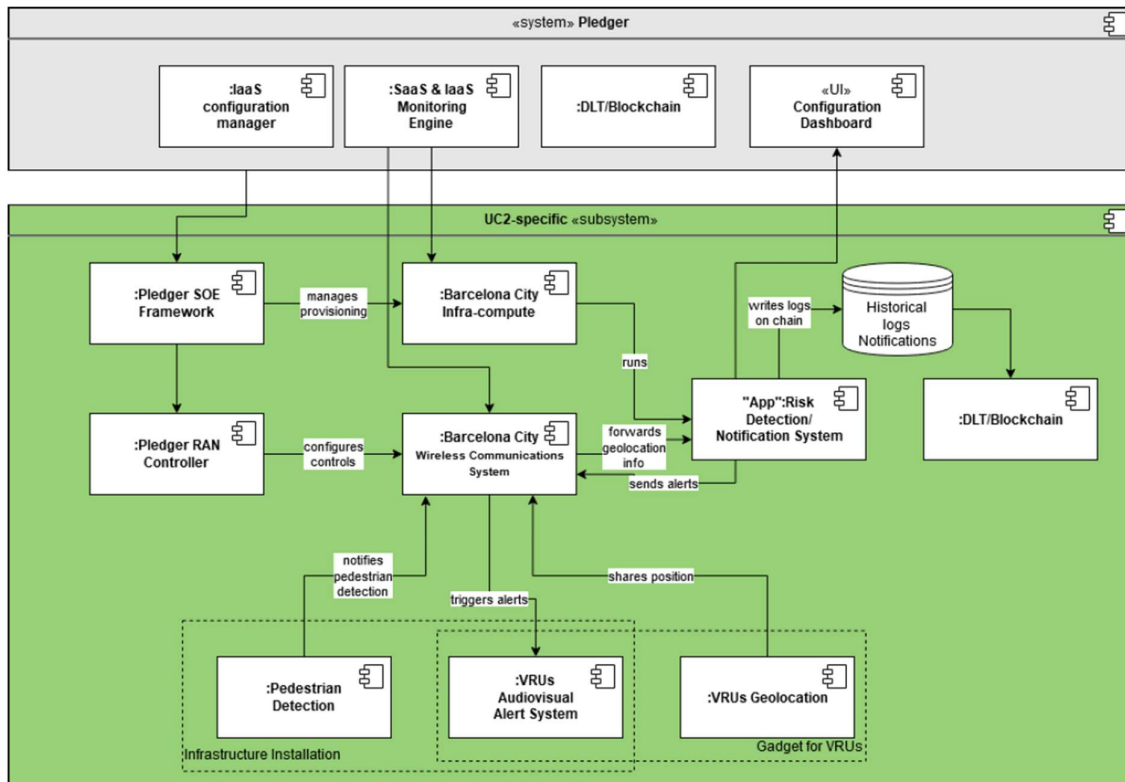


Figure 4 UC2 component diagram.

2.4 Use Case 3: Data Mining on the Edge for Manufacturing

As FILL edge computing devices face strict customer requirements (use low energy, harsh environments, low costs), Cybernetics, Fill’s Industrial IoT-Platform, is forced to outsource computationally intensive operations to a cloud service. Data transfer and transfer of computational power enables the generation of richer datasets and the computation of more complex machine learning models to improve predictions and analytics. These analysis by the data analysts enable the engineering experts to better evaluate the machine mechanics and to improve the engineering process. Furthermore, they provide valuable insights into the machine for the customer. These analyses are developed in Python3.

The data is collected and processed in dockerized applications on the edge device, which is based on a X86 architecture and running on Linux. Basic analysis is also performed there, e.g., evaluation of the cycle-time and quality of a component or the average power of the machine. These analysis scripts are deployed on edge as Docker containers and the derived indicators can be visualized on a dashboard. For more complex analyses and the use of analysis services towards predictive maintenance, data must be transferred to the cloud, where the customer wants to decide which data streams to release for whom. Customers’ machine data transfers knowledge about the underlying processes and a special focus on security is required to ensure a secure data transmission and to prevent unauthorised access by third parties. The UC will be deployed in FILL’s own premises, in the research & development area of the FILL Future Zone, where WiFi is deployed for wireless data transfer within the building.

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	15 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

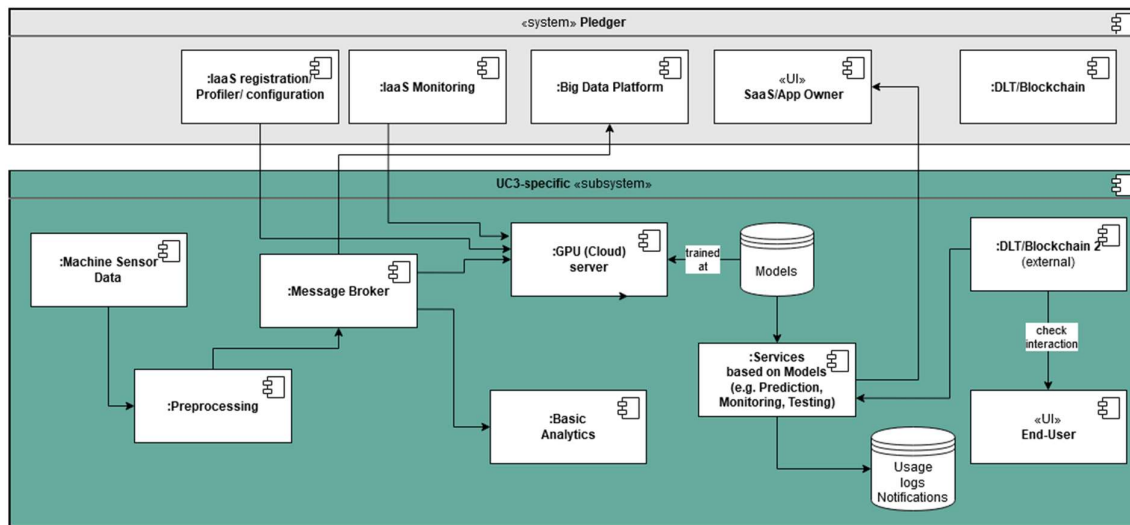


Figure 5 UC3 component diagram.

3 Pledger security approach

3.1 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.

The T4.1 methodology starts with the preliminary definition of key security assets and features, providing the draft architecture for the Security subsystem in D2.3. Following this preliminary 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. The threat assessment allows Pledger to prioritise specific security mechanisms to be applied, covering all four pillars. The following figure illustrates some of the existing developments, as of July 2021.

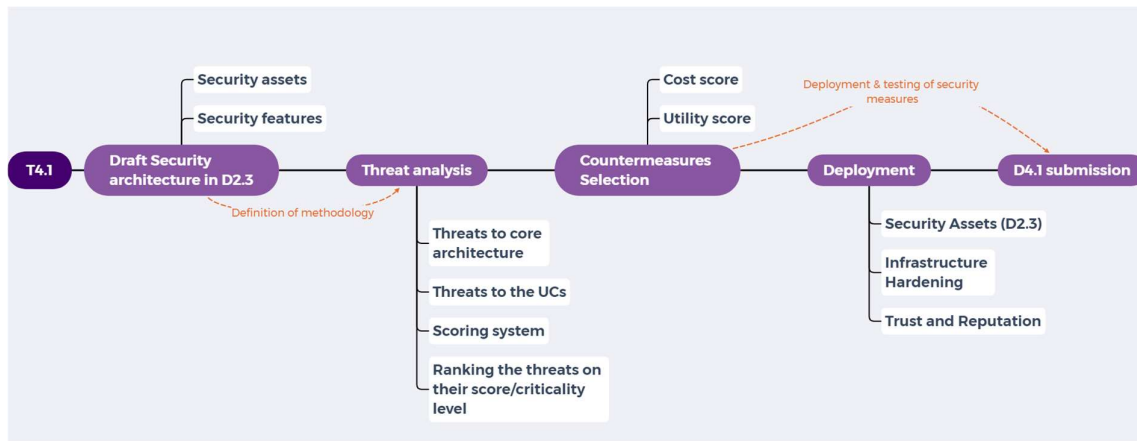


Figure 6 Security, Privacy and Trust methodological approach.

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	17 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

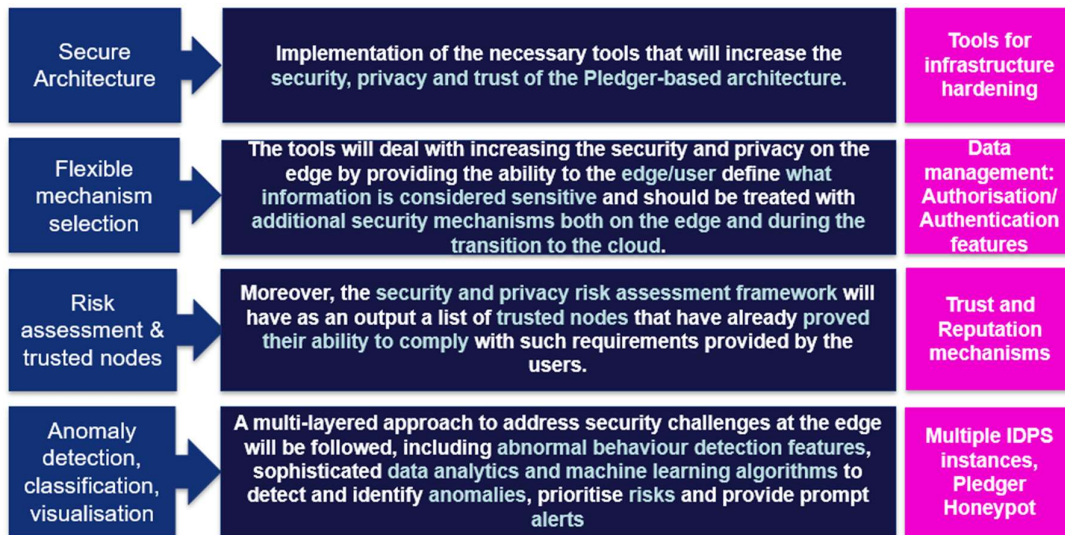


Figure 7: Pledger security pillars and key developments.

3.2 Cyber Threat Modelling approach

Cyber Threat Modelling is the process by which security vulnerabilities can be uncovered so proper defensive and remedial measures can be applied to minimize cyber risks. There is no specific industry standard when it comes to threat modelling. A variety of methodologies have been proposed, each serving a specific purpose. DREAD [3] is a model developed by Microsoft that focused on the factors of **D**amage, **R**eproducibility, **E**xploitability, **A**ffected users, and **D**iscoverability of a cyber threat. Although widely used, it is no longer maintained by Microsoft. STRIDE [4] was also developed by Microsoft, in order to classify threats on the basis on their effect: **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of Service, and **E**levation of Privilege.

Pledger accepts that in the case of Edge-Cloud deployments, it is necessary not only to apply threat modelling, but also extend it in key areas. When it comes to the deployment of services on the mobile edge, multiple stakeholders may be involved, forming complicated value chains. Taking into account the complexity of the integration of multiple software, hardware, network and storage technologies, there needs to be a complete methodology that also provides a way to prioritise threats and remediations. Furthermore, new factors other than the traditional Confidentiality, Integrity and Availability triplet should be accounted for, especially for use cases with strict QoS/QoE requirements. Degradation of service quality can easily become a major problem in mission-critical services.

To that extent, Pledger adopts the **Threat Assessment and Remediation Analysis (TARA)** [5] methodology proposed by MITRE [6]. TARA 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 take into account the specificities of a collaborative project and in order to provide tailor-made scoring systems.

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	18 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

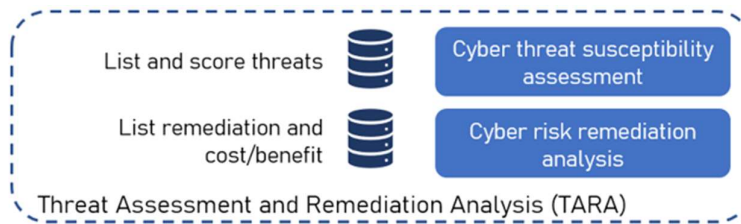


Figure 8 Threat Assessment and Remediation Analysis (TARA) overview.

3.2.1 Methodology

Threat Assessment and Remediation Analysis (TARA) is an engineering methodology used to identify and assess cyber vulnerabilities and select countermeasures effective at mitigating those vulnerabilities. TARA is part of a MITRE portfolio of systems security engineering (SSE) practices that contribute to achievement of mission assurance (MA) for systems during the acquisition process. Unique aspects of the methodology include use of catalog-stored mitigation mappings that preselect plausible countermeasures for a given range of attack vectors and use of countermeasure selection strategies that prescribe the application of countermeasures based on level of risk tolerance. TARA considers the adversary’s “Threats, Techniques and Procedures” (referred as TTPs) as an all-encompassing definition that includes the threats and methodologies for the deployment of a cyber attack.

3.2.1.1 Threat Assessment

The first step in the Pledger methodology is to compile a list of TTPs and prioritise them according to a severity score. During the threat assessment stage, a threat list is compiled and scored (Figure 9).

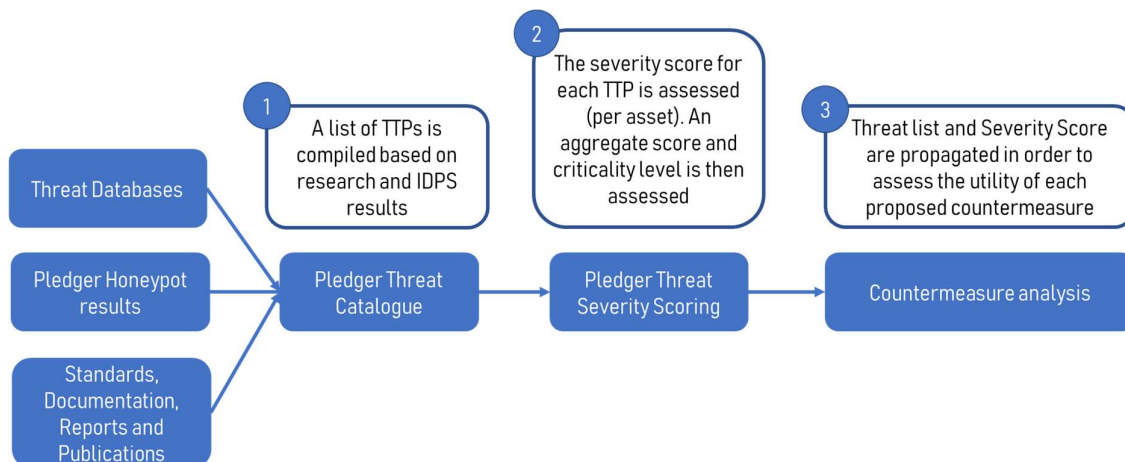


Figure 9 Overview of the Threat Assessment stage.

The next step would be to create a correlation matrix, associating the threats with the Pledger subsystems, as described in D2.3. 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.

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	19 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

► **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 2 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.

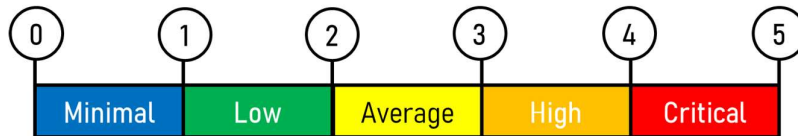


Figure 10 Threat Severity level scale.

Figure 10 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.



Table 2 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, or payment 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

3.2.1.2 Countermeasure catalogue and scoring

The second stage of the analysis focuses on the identification and prioritisation of measures to counter cyber threats (TTPs) against Cloud-Edge deployments. Experts and security practitioners often warn against the “*security theatre*” [7], the state where misplaced security investments create a false sense of security with little to no reduction to overall risk. While risk assessment products exist in the market, they are usually vendor-specific and create tie-in conditions for the clients or provide a very high-level analysis that cannot be easily translated to actionable policies. Especially in the case of Pledger, the lack of risk assessment tools that consider the complexity of the Cloud-Edge environment, creates the need to assess the usefulness of possible countermeasures and prioritise the ones that can minimise the overall risk.

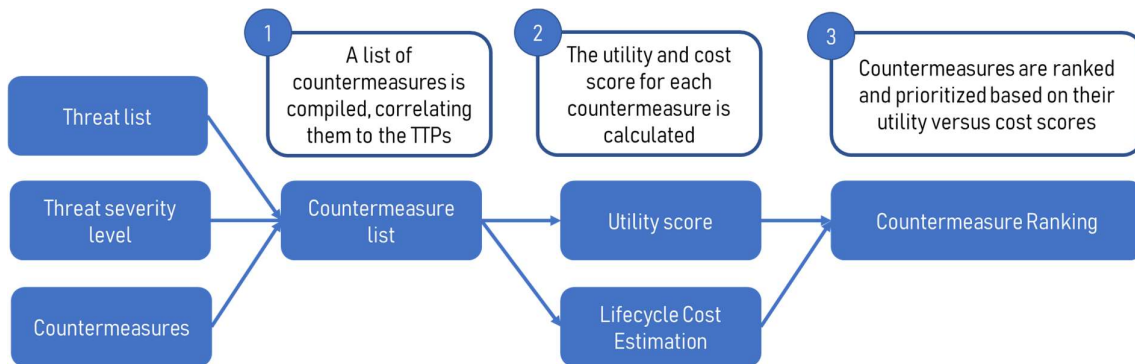


Figure 11 Countermeasure scoring method.

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 (Table 3) 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 3 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 following categories:

- ▶ **Acquisition Costs:**
 - Cost to research
 - Cost to develop
 - Cost to deploy
 - Cost to integrate

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	24 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

► **Operation Costs:**

- Cost to train
- Cost to operate
- Cost to maintain
- Cost to dispose

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 4). 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.

Document name:	D4.1 Trust, Privacy and Security related tools			Page:	25 of 73		
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status:	Final



Table 3 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 4 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

4 Threat modelling

4.1 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. 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 5.3.1.

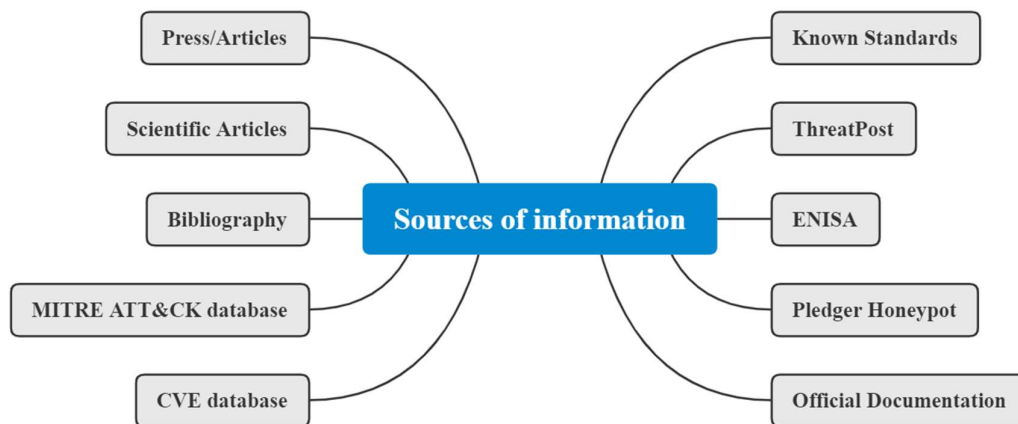


Figure 12 Sources of threat information.

The following subsections detail the most relevant threats per information asset, as well as some selected mitigation measures. 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 0.

4.1.1 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 5 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

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	29 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

	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 knows 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 and tracking the failed user login attempts to identify suspect behaviour.
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 knows 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.
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 thread as administrator accounts are usually capable to modify other accounts privileges and subvert security policies.
Mitigations	<ol style="list-style-type: none"> 1. 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. 2. Use an Authentication/Authorization/Accounting system that stores the history failed login attempts and of the most critical activities so that unauthorized use and abuse can be easily detected and reported.
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
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	<ol style="list-style-type: none"> 1. Enforce HTTPS or TLS on all services communication or put a reverse proxy using HTTPS/TLS in front of existing HTTP services 2. Filter network traffic to allow only the necessary ingress/egress and perform protocol-based filtering also in virtualized environments
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	<ul style="list-style-type: none"> ▶ 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
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

4.1.2 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 5.3.2.

Table 6 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	<ul style="list-style-type: none"> ▶ Limit Access to Resource Over Network: Limit communications with the container service to local Unix sockets or remote access via 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. <p>User Account Management: Enforce the principle of least privilege by limiting dashboard visibility to only the required users.</p>
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	<ul style="list-style-type: none"> ▶ 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. <p>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.</p>
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	<ul style="list-style-type: none"> ▶ 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	<ul style="list-style-type: none"> ▶ 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. <p>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, 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.</p>
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	<ul style="list-style-type: none"> ▶ 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 high-sensitivity workloads. Workloads should be allowed to run in their assigned security levels.
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 container pod can talk to all other pods/containers over the network overlay. It is possible a compromised container/pods 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 (DDOS), 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.
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

4.1.3 Big Data Management

Streamhandler is a highly scalable, distributed event-driven streaming platform based on Apache Kafka [8]. 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 7 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 SSL SSL Authentication
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	<ol style="list-style-type: none"> 1. Disable root ssh: Streamhandler platform has been implemented to VMs where root ssh is disabled. 2. Firewall setup: Firewall rules have been implemented in order to ssh connections after a number of unsuccessful trials
TTP ID	BDM-002
Title	Denial of Service

Source(s)	Pledger Honeypot, 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.
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 [9] 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
TTP ID	BDM-004
Title	Eavesdropping
Source(s)	ENISA Big Data Threat Landscape Report 2016
Description	A common issue that affects any 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. Utilising secure protocols (SSH/TLS/HTTPS etc) Enforcing strong password policies

4.1.4 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 8 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	<ol style="list-style-type: none"> 1. Disable root ssh: Streamhandler platform has been implemented to VMs that root ssh is disabled. 2. Firewall setup: Firewall rules have been implemented in order to ssh connections after a number of unsuccessful trials
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 etc attacks. Within the Pledger Honeypot, there were also recorded Denial-of-Service attacks exploiting the NTP protocol (monlist mechanism) 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. Blacklist policies: Refusing entry of traffic from IP addressed that have been blacklisted.
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

4.1.5 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 9 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.
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 on 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-003
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 on 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	<p>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.</p> <p>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.</p>
Mitigations	Use a higher version of Geth. The one used is v1.9.20.
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.

4.1.6 Benchmarking

Pledger’s Benchmarking assess the performance of the infrastructures controlled by Pledger automatically by executing benchmarking tests on the infrastructure’s nodes. The main components of the subsystem are:

- ▶ Benchmarking GUI: web portal to configure the infrastructure, the tests and the execution schedules.
- ▶ Benchmarking Library: the repository for all the benchmarking tests that can be executed.
- ▶ Benchmarking Scheduler: accordingly, to the user configuration, schedule the execution of benchmarking tests on the target infrastructures.
- ▶ Benchmarking Executor: executes benchmarking tests by connecting to the target infrastructure and issuing commands to start the execution and retrieval of results.
- ▶ Benchmarking Metrics DB: the repository where all metrics collected are store.

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.

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	38 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

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 weakness produced by CVE.

Furthermore, performance of the infrastructures is evaluated 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 10 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	<p>Attackers can use multiple techniques to get access to sensitive data:</p> <ul style="list-style-type: none"> ▶ 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	<p>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:</p> <ul style="list-style-type: none"> ▶ 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. <p>System, application and third-party software is regularly updated (MITRE Mitigation #M1051).</p>
TTP ID	BNCH-002
Title	Remote code execution
Source(s)	MITRE recognizes different techniques that exploits 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	<p>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:</p> <ul style="list-style-type: none"> ▶ Use read-only containers and minimal images for the benchmarking tests to prevent the execution of commands not allowed (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. <p>When running in Kubernetes infrastructures, improve isolation of containers adjusting the security policies like execute containers as unprivileged, disable mounting of host paths, enforce selinux, require read-only root filesystem.</p>

4.1.7 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 5.3.2, 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 13.

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	40 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

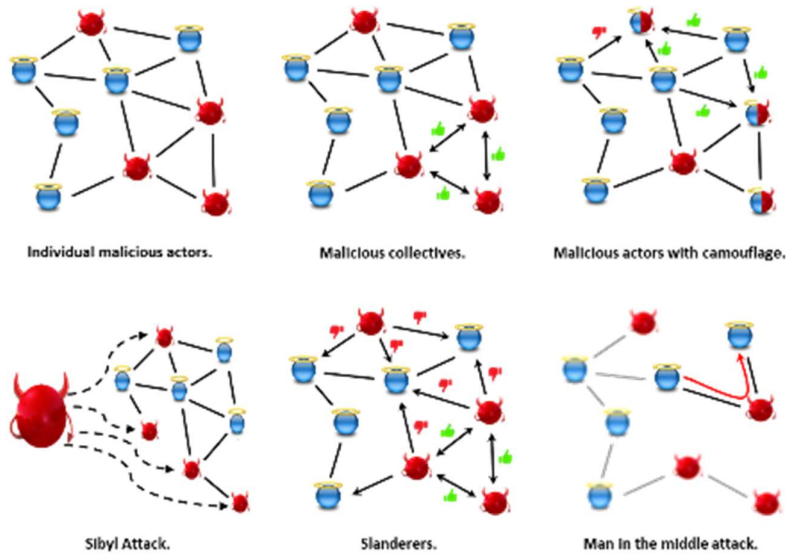


Figure 13 Common attacks in T&R systems.

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

Table 11 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	<ul style="list-style-type: none"> ▶ Decoupling the Trust & Reputation Indices for different (types of) services. ▶ Providing a weight based on the criticality of the service being provided. ▶ Using both a Trust and a Reputation Index. ▶ Usage of several time-windows to extract long-term and short-term T&R indices and fading factors.
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).
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	<ul style="list-style-type: none"> ▶ Identifying trusted nodes (evaluators) and giving greater weight to their evaluation. ▶ Using both a Trust and a Reputation Index. ▶ Identifying the proper Dependability calculation formula to make the system resilient to 51%+ attacks.
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	<ul style="list-style-type: none"> • Common development and integration best practices so as to reduce the attack surface of these components. • Regular cross-check of the appropriate results/models of the said components.
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	<ul style="list-style-type: none"> • Using tampering detection techniques and tools. • Using Blockchain for the storage of (some of the) results, to acquire tamper-proof results.

4.1.8 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.

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	42 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

Table 12 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.
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 Boston 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)
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.

4.1.9 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 [10].

Table 13 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	<ul style="list-style-type: none"> ▶ Limit permissions to discover cloud infrastructure in accordance with least privilege. ▶ Limit the number of users with administrative privileges.
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	<ul style="list-style-type: none"> ▶ 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.
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	<ul style="list-style-type: none"> ▶ 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	<ul style="list-style-type: none"> ▶ Use multi-factor authentication where possible. ▶ Ensure that all traffic is encrypted appropriately. ▶ Use best practices for authentication protocols, e.g. private key authentication rather than user+password authentication.
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	<ul style="list-style-type: none"> ▶ Use a Public Key Infrastructure. ▶ 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	<ul style="list-style-type: none"> ▶ 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	<ul style="list-style-type: none"> ▶ 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 Role-Based Access Control (RBAC) according to the principle of least privilege.
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.

4.1.10 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 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 - two central technologies in this UC - are used for the threat analysis.

4.1.10.1 RabbitMQ – AMQP

RabbitMQ is an open-source message broker software that implemented the AMQP and other protocols. The Advanced Message Queuing Protocol (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 [11]. 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 [12].

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	46 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

Table 14 UC3 threats.

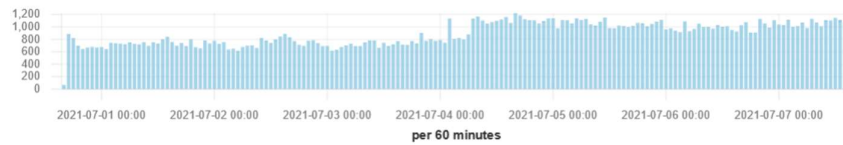
TTP ID	UC3-001
Title	Man-In-The-Middle Attack – Manipulation of transmitted data
Source(s)	https://attack.mitre.org/techniques/T1557/
Description	An attacker could position himself/herself between the data producer and the message broker as well as message broker and consumer to manipulate the transmitted 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	<p>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.</p> <p>Limit access to network infrastructure and resources that can be used to reshape traffic or otherwise produce MiTM conditions.</p> <p>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.</p> <p>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.</p>
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 [12]
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 [12]
Description	An attacker could obtain credentials of existing accounts and use them to get increased rights.
Mitigations	<ul style="list-style-type: none"> ▶ 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). ▶ 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.shiftright.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	<ul style="list-style-type: none"> ▶ 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
TTP ID	UC3-005
Title	Query injection attacks
Source(s)	https://blog.shiftright.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	<ul style="list-style-type: none"> ▶ 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

4.1.11 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 5.3.1) that documented threats to a public-facing service. Figure 14 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.

Logs
149,088



Threats
1,097

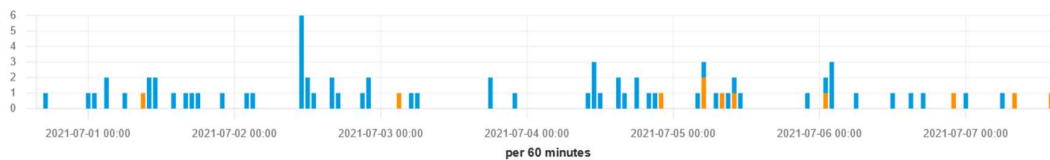
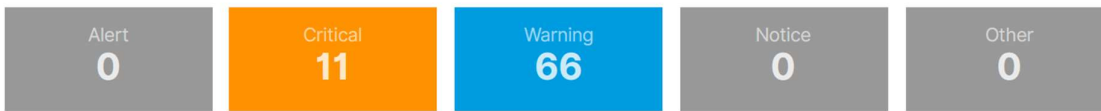
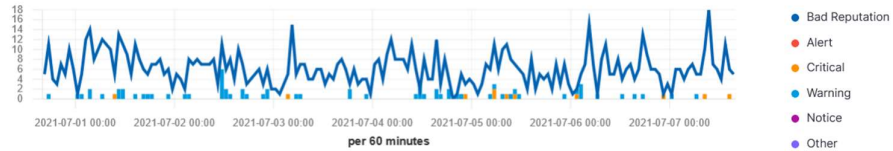


Figure 14 Basic honeypot statistics for a 7-day period.

Table 15 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 16 includes the analysis of the threats captured from the Pledger honeypot, as well as other known threats against cloud infrastructure in bibliography.

Table 16 Threats against the underlying Pledger infrastructure.

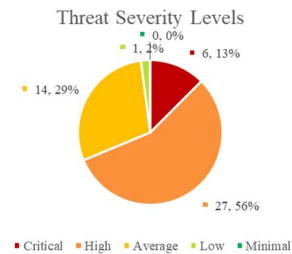
TTP ID	INFRA-001
Title	(Distributed) Denial of Service
Source(s)	MITRE, Pledger Honeypot
Description	Denial of Service is a type of attack that aims to exhaust resources or exploit protocol vulnerabilities to crash or slow down a victim server so that it denies access to real users. Within the Pledger 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. Blacklist policies: Refusing entry of traffic from IP addresses that have been blacklisted.
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. Blacklist policies: Refusing entry of traffic from low-reputation IP addresses that have been blacklisted.
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 Inclusion of Intrusion Detection and Prevention system

4.2 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 15).

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 15 Threat Severity basic statistics.

Error! Reference source not found. (a) provides the final results of the TTP prioritization, while (b) provides the selected countermeasures and mitigations, after ranking them according to utility and cost, green colour denotes measures that have been applied, while orange denotes measures that are considered as highly relevant, and yellow denotes measures that are out of scope of the project or low priority.

Table 17 (a) Threat prioritisation, (b) Countermeasure prioritisation.

TTP ID	Source	TTP	Threat Severity Level	Max TTP Score	Orchestration	Configuration	SLA	Decision Support	Big Data	Blockchain	CI/CD	UC1 Subsystem	UC2 Subsystem	UC3 Subsystem	Other
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										

TTP ID	Source	TTP	Threat Severity Level	Max TTP Score	Orchestration	Configuration	SLA	Decision Support	Big Data	Blockchain	CI/CD	UC1 Subsystem	UC2 Subsystem	UC3 Subsystem	Other
TH024	Articles/Bibliography	Threats to SLA subsystem/T&R components	High	3.4			3.4								
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								

TTP ID	Source	TTP	Threat Severity Level	Max TTP Score	Orchestration	Configuration	SLA	Decision Support	Big Data	Blockchain	CI/CD	UC1 Subsystem	UC2 Subsystem	UC3 Subsystem	Other
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			

5 Architecture of privacy, security and trust components

5.1 High level architecture

In addition to the planned security subsystem developed in T4.1, and based on the threat analysis, 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 16 provides an overview of the work performed to secure the Pledger Core and Use Cases.

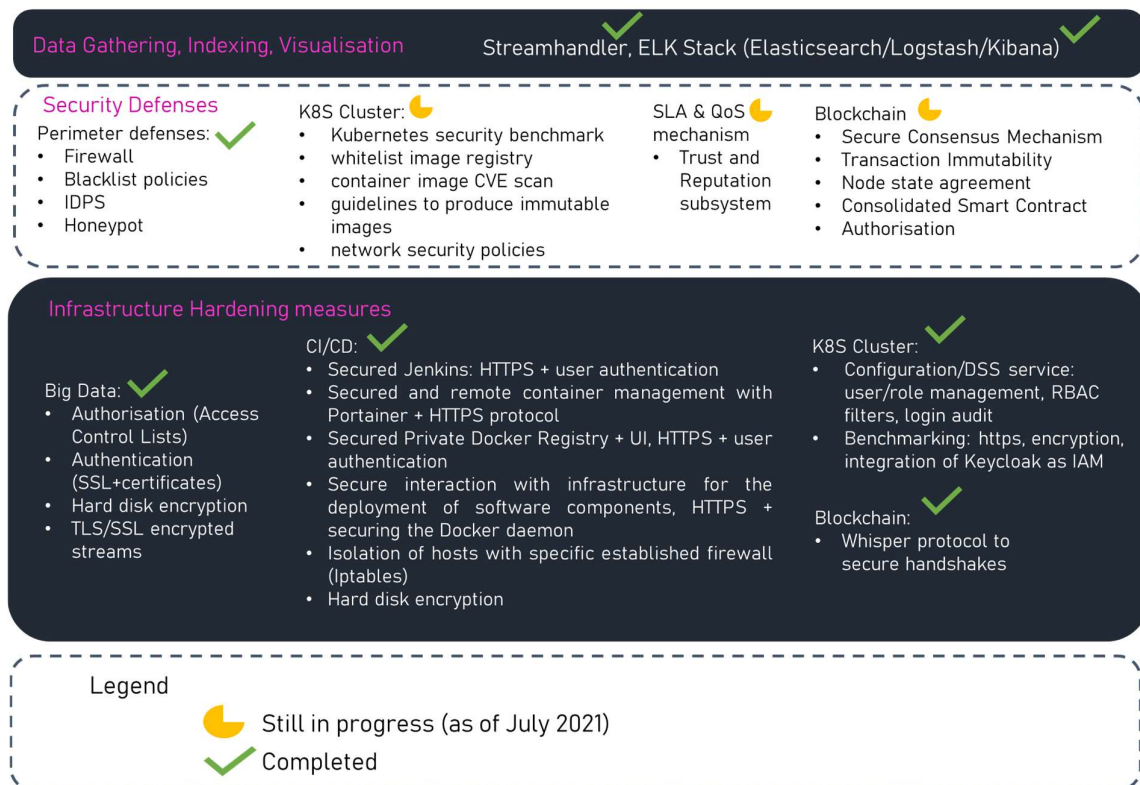


Figure 16 Pledger Security, Privacy and Trust tools and measures.

5.2 Infrastructure Hardening

5.2.1 Streamhandler Big Data Platform

In a standard Kafka setup¹, 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:

¹ Details on the platform and the data integration capabilities are provided within the upcoming D5.2.

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	56 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

- ▶ **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.

5.2.2 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.

5.2.2.1 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.

5.2.2.2 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.

Document name:	D4.1 Trust, Privacy and Security related tools				Page:	57 of 73
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status: Final

5.2.2.3 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.

5.2.2.4 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 (SYN flood attacks, smurf attacks, land attacks, attacks by malfunctioning 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.

5.2.2.5 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.

5.2.3 K8S Cluster

Within “Cloud Service Discovery” threat category, MITRE [13] identifies all the possible threads 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 thread 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

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	58 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

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 [14], 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.

In detail, such security activities include:

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 [15]. 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.

5.2.4 Blockchain security through the Whisper protocol

Whisper is a pure identity-based messaging system. It provides a simple low-level API and allows peer-to-peer communication between the nodes of Whisper network using the underlying DEVp2p^2 Wire Protocol.

All Whisper messages are sent to all whisper nodes. These messages are encrypted and then sent through the DEVp2p Wire Protocol, which in turn uses its own encryption, on top of the Whisper encryption. The messages could be decrypted by any node that has the corresponding key. What makes this secure besides the double encryption is that all nodes will accept and save the messages received whether they

² Pronounced devp2p.

are able to decrypt or not. This makes it nearly impossible to tell who the actual receiver of the message is. Furthermore, Whisper prevents DDoS attacks through proof-of-work (PoW) algorithm as messages would only be processed and further forwarded if their PoW exceeds a certain threshold.

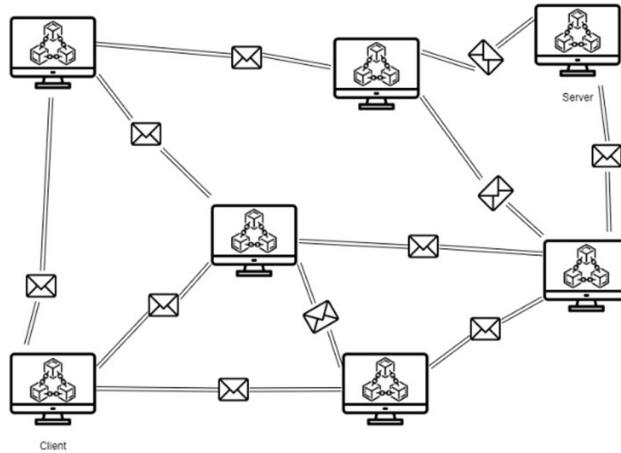


Figure 17 Whisper protocol architecture.

The Whisper protocol will be activated within the blockchain nodes. Two nodes would play the roles of a Server Node and a Client Node that would be exchanging handshake related information in the form of Whisper messages. There would be multiple other active Whisper nodes that would be connected to the two previous nodes in order to create a Whisper network (for increased security).

5.3 Deployed Security Assets

5.3.1 Distributed IDPS Cybersecurity Solution

5.3.1.1 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 preset 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 preset 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

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	60 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

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

5.3.1.2 Architecture

The architecture of the cybersecurity solution can be found in the following figure (Figure 18):

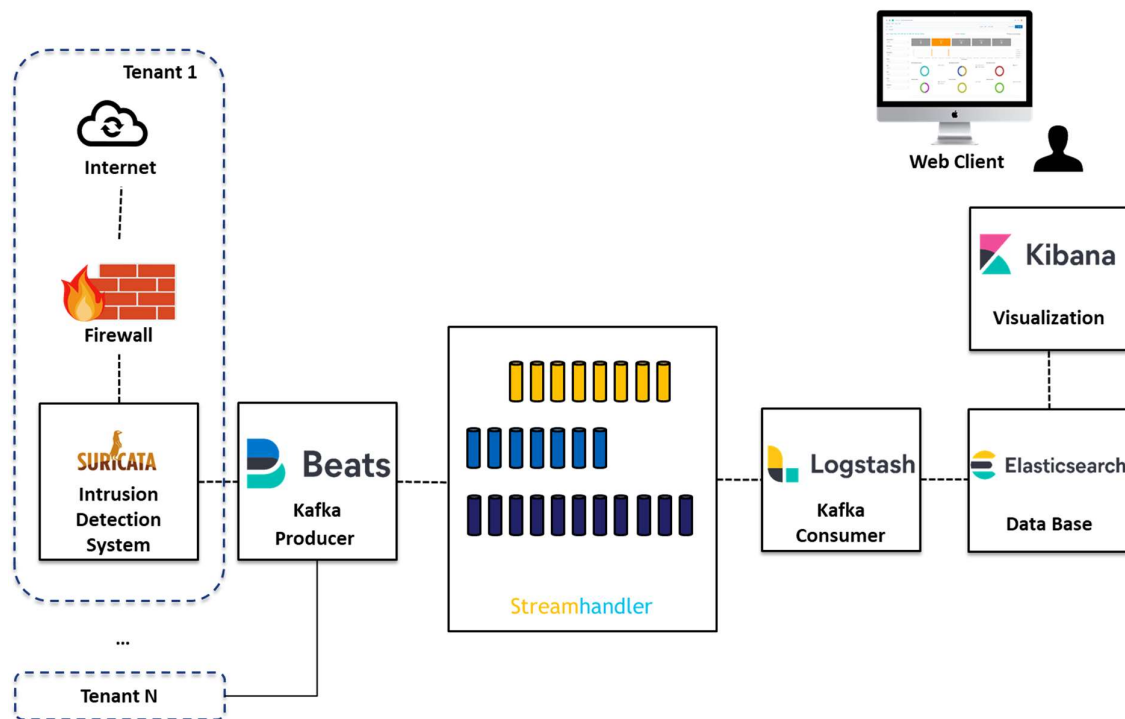


Figure 18 Distributed Cybersecurity platform architecture.

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

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	61 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

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 5.2.1, 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.

5.3.1.3 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 analyzes 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.

Document name:	D4.1 Trust, Privacy and Security related tools				Page:	62 of 73
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status: Final

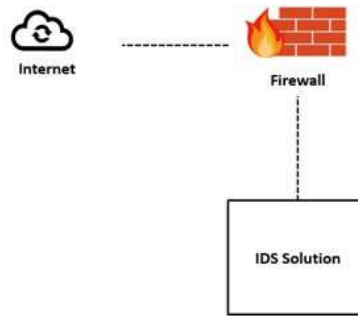


Figure 19 Threat Detection Mode – IDS.

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..

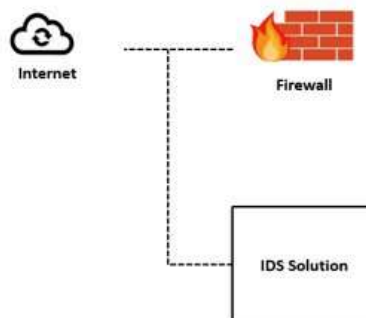


Figure 20 Honeypot Mode – IDS.

5.3.2 Trust and Reputation

Trust and Reputation (T&R) are two indices that can be defined and monitor to guarantee a minimum level of security between two entities of a distributed system that want to have a transaction or interaction. Many methods and mechanisms can be used to manage and model trust and reputation in systems such as P2P networks, ad-hoc ones, wireless sensor networks or even multi-agent systems. Such methods have been used in many 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

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	63 of 73
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

trustworthy (as identified by the system) participant to have an interaction with, preventing thus the selection of a fraudulent or malicious one.

In general, T&R management systems follow five general steps which are described by Marti and Garcia-Molina [16] (Figure 21):

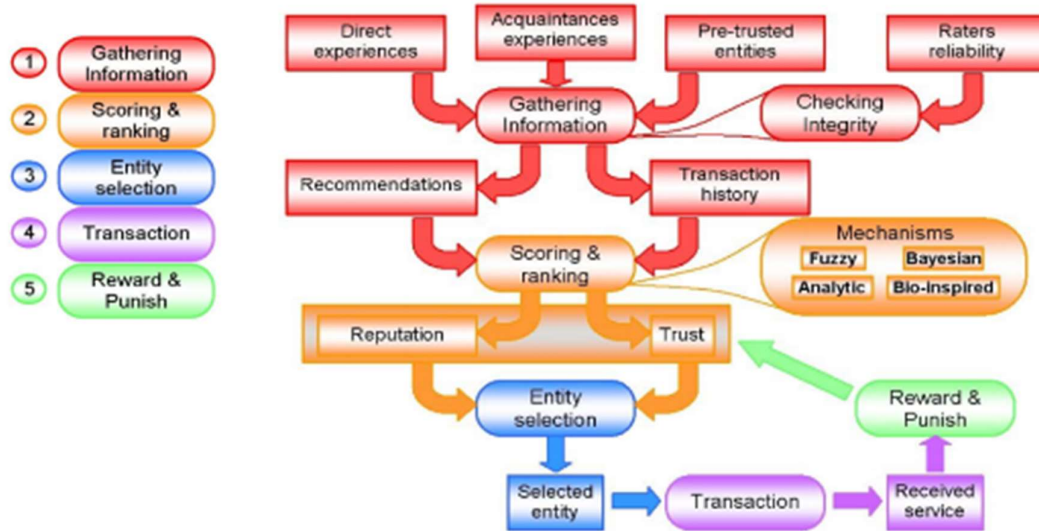


Figure 21 General steps followed in T&R models.

1. **Gathering information** about the behaviour of a certain participant.
2. **Calculating T&R indices** and computing a score for every peer in the network.
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. **Punishing or rewarding** according to the satisfaction obtained, adjusting consequently the global trust (or reputation) deposited in the selected service provider.

5.3.2.1 The Pledger T&R management system

It should be noted that the creation of a complete Pledger T&R management system is a work that requires integration between several components and subsystems of the Pledger Core System.

The first step in Figure 21 (gathering information) can already be 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 could be extracted from other UC-specific mechanisms, but this scenario will be considered at a later stage of the project.

The second and third step in Figure 21 is the one on which this subsection focuses. It is related to the definition of Trust & Reputation Indices and the mechanisms for the calculation of the said indices. Regarding the selection of entities per se, this should be done under the context of the Orchestration subsystem, and more specifically the “Recommender” component, as described in D3.2.

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	64 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

It is during the fourth step that the Pledger Blockchain comes into play. The interactions (and transactions) of the several peers can take place through this subsystem, while the a posteriori assension of the satisfaction for received services can be part of the Smart Contracts implementation.

Finally, step five is related to the policy that the T&R management system will follow to ensure that actors are rewarded or punished (e.g., getting blacklisted), based on their behaviour.

5.3.2.2 The Pledger T&R model

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) of OASIS, **there are no standards** for concepts like Trust and Reputation. In Pledger, the following definitions are considered:

- ▶ **Trust:** The expectation that an interaction will be satisfactory based on *our personal experience*.
- ▶ **Reputation:** The belief that an interaction will be satisfactory based on the experience of *our social circle*.

Node A will have a high Trust index for Node B if the services provided from Node B to Node A have been evaluated from Node A positively. Node A would have a high Reputation index for Node B if the services provided from Node B have been evaluated from an extensive “social circle” of Node A positively.

Let’s assume that actor-1 wants to find out some social characteristics of actor-2 for a specific service offered. The following terms can then be defined:

- ▶ **Popularity (P):** A counter which monitors how many times actor-2 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.
- ▶ **Trust (T):** The belief of actor-1 that actor-2 is going to deliver the correct service based on previous interactions of actor-1 with actor-2. The Trust Index of actor-2 provided by actor-1 is a property which states how many times actor-2 has successfully shared its services with actor-1. Trust is “subjective”, because it is estimated from perspective of the individual trustor (actor-1 in this case).
- ▶ **Reputation (R):** The belief of actor-1 that actor-2 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 actors (apart from actor-1) have on actor-2. In other words, this metric determines the belief of others on an actor and is useful especially when actor-1 does not have enough data to extract a Trust Index for actor-2 (because e.g. there are no interactions between the two actors yet).
- ▶ **Reliability (R’):** An absolute indicator of the performance of the actor that quantifies its efficiency to offer successfully its services relatively to its ideal or normal operation. The Reliability Index should be based on criteria like: response time upon request, ability to communicate, quality of service provided, etc. This is an index that could be extracted by other subsystems of Pledger, such as the SLA one.
- ▶ **Dependability (D):** A social measure combining all the above social measures. It can be simply derived by the expression $D = a \cdot T + b \cdot R + c \cdot R' + d \cdot P$ where a, b, c and d (non-negative integers) are the weights of the measures and $a + b + c + d = 1$. For this calculation, Popularity has to get normalized. By selecting the appropriate weights, we can provide the expression of the Dependability Index that we want. For example, when there are only a few interactions between actor-1 and actor-2, then the Trust Index should have a low weight and the Reputation Index should have a high weight. This means that the weights should change dynamically and be set according to the users or developers’ preferences.

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

- ▶ Actor-1 trusts actor-2 because Actor-2 has a good Reputation. This reflects that Reputation can be used to build Trust.
- ▶ Actor-1 trusts actor-2 despite the bad Reputation of Actor-2. This reflects that even if actor-1 knows the Reputation of actor-2, actor-1 has its own private knowledge (e.g. direct experience with actor-2) which is considered to be more important.

Generally, an actor can be evaluated only by information gathered from other actors. Its Dependability can be calculated by each and every other actor of the community (a subjective estimation) or by the whole system (a more, but not totally, objective estimation). Both big and small time-windows can be used to quickly detect malicious or unsatisfactory behaviour and avoid the fast redemption of blacklisted actors. Moreover, feedback from recent interactions should have a higher weight than this of older actions.

Benevolent actors should have more opportunities than newcomers. As a result, newcomers with 0 interactions with other actors will have Reputation equal to 0. However, an extra rule has to be applied to the model we have designed to give the opportunity to newcomers that have a low Reputation (because of the small number of interactions with other actors) to be chosen as service providers at some point and start building their Reputation. For example, 10% of the recommendations from the platform should introduce newcomers to the rest of the community. The same applies for actors which have low Reputation due to malicious or unsatisfactory behaviour in the past. In other words, this rule enables the **social integration** and **reintegration** of the actors to the system.

Depending on the scenario, most probably, different Trust and Reputation **scores** for different services provided by the members of the network will be considered. This feature helps us face quite many security threats (e.g., avoiding abuse of a high achieved Reputation).

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 assessment of the transactions (interactions) e.g. in the Blockchain. A malicious or faulty actor will quickly lose any trust. If the behaviour of the actor changes, the Social Reintegration feature of the system will allow it to build trust again.
- ▶ **Weight (w):** This is a value indicating how crucial the service is for the well-being of the actor. It is used in order to prevent a malicious actor from providing a minor service well and then exploiting the built Trust and providing a crucial service poorly. Due to this value, it is difficult for the Trust index to increase just because of minor services, whereas it can drop quickly in case of a crucial service with low quality.
- ▶ **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 its 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 looks into the appropriate Log Files and calculates the trust value as the weighted average of the log entries using:

$$\mu_i^k = \frac{\sum_{i=1}^N (s_i \cdot w_i \cdot f_i)}{W} \quad (1)$$

where W is the normalization co-efficient which ensures that the trust value will be between [0,1] and is calculated by:

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	66 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

$$W = \sum_{i=1}^N (w_i \cdot f_i) \quad (2)$$

The **mean value** (μ) is a measure of the overall observed behaviour of the actor and indicates the expected satisfaction value of the next interaction. However, it is needed 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 . Thus, 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:

$$\sigma_t^k = \sqrt{\frac{\sum_{i=1}^N (s_i^2 \cdot w_i \cdot f_i) \cdot W - \left(\sum_{i=1}^N (s_i \cdot w_i \cdot f_i)\right)^2}{W}} \quad (3)$$

Finally, we define Trust as:

$$T^k = \mu_t^k - \sigma_t^k \quad (4)$$

To sum up, μ shows the satisfaction that actor-1 should expect from actor-2, while σ shows how predictable the behaviour of actor-2 is. This means that if $T = 0.5$ then there is an 84% probability that the satisfaction for the service will be 0.5 or greater. 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 from the Blockchain.

5.3.2.3 Testing

In order to test our T&R model, at a first level we will use TRMSim-WSN, a simulator for T&R models.

The TRMSim-WSN [17] is a Java-based T&R models simulator aiming to provide an easy way to test a trust and/or reputation model over WSNs and to compare it against other models. The TRMSim-WSN is aimed at simulating algorithms for reputation and trust management in WSN systems, but the same principles can apply to IoT systems in general. The simulation can be run over a single randomly generated WSN or over a set of networks. The user is able to define parameters of the network, such as the percentage of clients and that of malicious nodes. Network topologies may also be loaded from and saved to XML files. Sample trust and reputation models have been included and an API is offered which provides a template for the users to help them easily load new T&R models to the simulator. For the tests, parameters that can be configured are:

- ▶ the number of executions,
- ▶ number of random networks to be tested,
- ▶ % of Malicious Actors,
- ▶ Collusion between Malicious Actors,
- ▶ Oscillating behaviour of actors, etc.

To evaluate the T&R model, we will compare it with the corresponding results of other T&R models (e.g. Eigentrust [18], PeerTrust [19], and PowerTrust [20]).

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	67 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

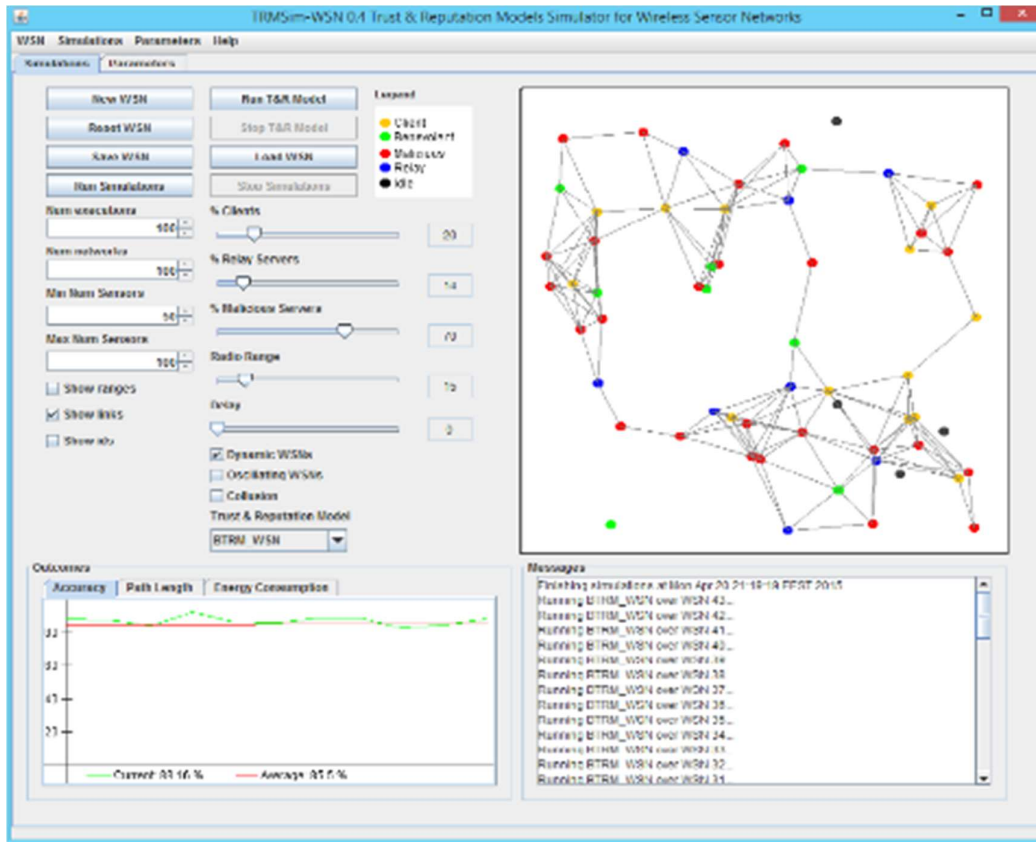


Figure 22 TRMSim-WSN.

6 Conclusions and Next Steps

6.1 Key results

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. During the period leading to July 2021, 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.

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 the 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 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 become necessary, in order to properly configure them to minimise security, privacy and trust risks. The use of Trust and Reputation mechanisms was also considered, in order to secure transactions among untrusted, unknown entities. The evaluation of trust and reputation mechanisms is on-going work. The Big Data, CI/CD, Blockchain and Kubernetes cluster for the project were sufficiently hardened against the uncovered threats, while work is ongoing to secure the rest of the infrastructure as well as the Use Case specific developments.

6.2 Plan for future activities

As the work in Pledger progresses, T4.1 will dedicate effort in the further development and testing of security assets (IDPS, Trust and Reputation, etc.) as well as the application of additional hardening measures. The threat analysis is considered as a living document and will be maintained during the project lifespan. In particular, the project expects that at least the following developments will take place:

For the Distributed Ledger:

- ▶ Private submissions on Pledger DLT to ensure privacy of specified data
- ▶ Secure Consensus Mechanism on Pledger DLT:
 - Transaction immutability
 - Nodes state agreement
- ▶ Pledger DLT migration to Orchestration:
 - leverage the latter's security
- ▶ Consolidated Smart Contract Deployment:
 - secure unified system

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	69 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

- ▶ Authorization mechanisms on the Wallet:
 - each party connects through their dedicated wallet

For the Kubernetes clusters:

- ▶ Kubernetes security benchmark
- ▶ whitelist image registry
- ▶ container image CVE scan
- ▶ guidelines to produce immutable images
- ▶ network security policies

Additional testing of the deployed security, privacy and trust mechanisms can then be performed. A tentative list of useful tools to simulate threats and monitor their effects on the Pledger infrastructure is herein presented.

Specialised operating system distributions contain pre-packaged and pre-configured penetration testing tools. Kali Linux is among the most well-known distributions and contains a variety of penetration testing tools. Furthermore, there are existing OS distributions packaged with a multitude of known vulnerabilities, that can be setup as “targets”. In Pledger’s case, a specialised OS is easily installed as a VM within any testbed and operated to launch attacks or as a “victim”.

Specialised software frameworks can be used to develop customised cyberattack vectors. The Metasploit framework is such an example; it provides tools for developing and executing exploit code against a remote target. Other frameworks provide a way to launch phishing attacks or use browser exploits.

Online cybersecurity databases often publish threat information or block lists. Block lists are collections of URL, URI, or IP addresses that are associated with known adware, malware or ransomware. Pledger plans to use these online resources to test functionality, performance and scaling. Threat information are published by many major cybersecurity enterprises.

Public, open-source code repositories such as GitHub offer collections of tools to simulate a variety of attacks (DoS, Phishing, Malware simulation etc). Furthermore, code to live malware, rootkits, worms etc. has been often released. Therefore, Pledger can use simulation tools or adapt existing source code and run malware in isolated forensic environments within its testbeds.

Network monitoring and visualization tools are also utilised to inspect the traffic that passes through specific Pledger components. It can be from low level traffic, flows monitoring (netflow, tstat) or high-level dashboards, etc.

Traffic generators such as iperf, httpperf, tcpreplay, etc can be used by Pledger to generate traffic with specific velocity, volume and variety characteristics (e.g., mixed with other malicious patterns).

The following table contains a list of publicly available tools that can be utilised by Pledger for penetration testing.

Table 18 Reference list of penetration testing tools publicly available.

Type	Tools
Specialised OS	Penetration testing: Kali Linux (Backtrack), WHAX, BlackArch, BackBox, Pentoo, Parrot Security OS.
	Target OS: Damn Vulnerable Linux (DVL), OWASP Web Testing Environment, Metasploitable
Software Frameworks	Metasploit, NMap, Burp Suite, BeEF – Browser Exploitation Framework, OWASP ZAP, w3af, OpenVAS, SpeedPhish, xfltrat, OWASP Xenotix
DDoS tools	Rate-based, network & transport layer: LOIC, XOIC, UDP Flooder, UDP Unicorn, hping3, tcpreplay, Dereil, DDOSIM

Type	Tools
	<p>Rate-based, application layer: OWASP Switchblade, TorsHammer, HULK, Saphyra, ExoFlood, DoSHTTP, GoldenEye, HOIC, AnonymousDOS, Dereil, DDOSIM</p> <p>Reflection/Amplification: XOIC, hping3, UDP Unicorn, arspooof, OffensivePython/Saddam, Tsunami</p> <p>Poisoning: arspooof, subterfuge, arpoison</p> <p>Fragmentation: Scapy</p> <p>Protocol-based, application layer: Slowloris, OWASP Switchblade, pyLoris, SlowDroid, XSSer, XSSProxy, DAVOSET</p>
Online threat databases	Malware Information Sharing Platform (MISP), Virustotal, Ransomware tracker, PhishTank, Malware Domains, Spamhaus Project
Botnets	SlowDroid, BoNeSi, Mirai, BASHLITE
Phishing and Identity theft	ClickJack (for UI Redress attack), LUCY, GoPhish, KingPhisher, SpeedPhish framework, Social Engineering Toolkit, BeEF (browser exploitation), CSRFdemo (Cross Site Request Forgery attack)
Malware source code	Available in GitHub: theZoo collection, Mirai, Zeus worm, Cypher, petya, bash-ransomware
Malware simulation	Stackhackr, Barkly, LUCY
Tunneling	<p>HTTP: chisel, corkscrew, httptunnel</p> <p>DNS: iodine, dns2tcp</p> <p>Other: multitun, proxytunnel, sstunnel, icmptx, fraud-bridge</p>
Remote execution & backdoors	Matahari, backdoorme, webshell, chromebackdoor, backdoor-apk, backdoorppt, BrainDamage
Cross-site scripting	Cross, BruteXSS, XSSYA, XSSer, Excess-XSS, OWASP Xenotix, XSSmh
Worms/Rootkits etc	OpenWorm, wormhole, PowerWorm, Wormz, wormbrowser ZeroAccess v3 (P2P malware), Zeus
Traffic generation, Monitoring and visualization	tcpdump, wireshark, LibreNMS, Grafana, OpenNMS, iperf, httpperf, moongen

7 References

- [1] PLEDGER. D2.3 - Pledger Overall Architecture. Voutyras, Orfefs. 2020
- [2] PLEDGER. D2.1 - Pledger Detailed Use Cases, Ochoa-Aday, Leonardo, Betzler, August. 2020
- [3] A. Shostack, Experiences Threat Modeling at Microsof.
- [4] Microsoft Corporation, "The STRIDE Threat Model | Microsoft Docs, [https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878\(v=cs.20\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)?redirectedfrom=MSDN) . Retrieved 2021-07-27.
- [5] Jackson E. Wynn, The MITRE Corporation, Threat Assessment and Remediation Analysis (TARA), (2014).
- [6] The MITRE Corporation, <https://www.mitre.org/>. Retrieved 2021-07-27.
- [7] B. Schneier, Beyond Security Theatre, New Internationalist, (2009).
- [8] Apache Foundation, Apache Kafka: a distributed streaming platform, <https://kafka.apache.org/>. Retrieved 2021-07-27.
- [9] E. Fernandez, Security in Data Intensive Computing Systems, in Handbook of Data Intensive Computing, Springer, (2011).
- [10] Good practices for security of smart cars, <https://www.enisa.europa.eu/publications/smart-cars>. Retrieved 2021-07-27.
- [11] G. C. M. Nebionne, Security of IoT Application Layer Protocols: Challenges and Findings., Future Internet. (2020).
- [12] 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).
- [13] Cloud Service Discovery, <https://attack.mitre.org/techniques/T1526/>. Retrieved 2021-07-27.
- [14] OpenStack security guide, <https://docs.openstack.org/security-guide/>. Retrieved 2021-07-27.
- [15] Pod Security Policies: <https://kubernetes.io/docs/concepts/policy/pod-security-policy/>. Retrieved 2021-07-27
- [16] S. Marti and H. Garcia-Molina, Taxonomy of trust: categorizing P2P reputation systems, Computer Networks 2006, Computer Networks, (2006).
- [17] F. Marmol and M. Perez, TRMSim-WSN, Trust and Reputation Models Simulator for Wireless Sensor Networks, in IEEE Intl Conference on Communications, (2009).
- [18] S. Kamvar, M. Schlosser and H. Garcia-Molina, The EigenTrust algorithm for reputation management in P2P networks, (2003).
- [19] L. Xiong and L. Liu, PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities, IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 7, pp. 843-857, (2004).

Document name:	D4.1 Trust, Privacy and Security related tools	Page:	72 of 73
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

[20] R. Zhou and K. Hwang, PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing, IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 4, pp. 460-473, (2007).

Document name:	D4.1 Trust, Privacy and Security related tools				Page:	73 of 73	
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status:	Final