Funded by the European Union

Horizon Europe Project TELEMETRY
HORIZON-CL3-2022-CS-01-02

TELEMETRY

Deliverable D2.1 – Analysis of Indicators and Metrics for Vulnerability Representation

Project Reference No	TELEMETRY - 101119747
Project Title	Trustworthy Methodologies, Open Knowledge & Automated Tools For Security Testing Of IoT Software, Hardware & Ecosystems
Work package	WP2
Type	R - Document, report
Dissemination Level	PU - Public (fully open)
Date	30/11/2024
Status	Final v1.0
Editor(s)	Steve Taylor, University of Southampton
Contributor(s)	Panos Melas, University of Southampton Aida Omerovic, SINTEF Ravishankar Borgaonkar, SINTEF Martin Gilje Jaatun, SINTEF Bernd-Ludwig Wenning, MTU Pasindu Kuruppuarachchi, MTU Antonios Mpantis, ATC George N. Triantafyllou, ATC Robert Seidl, Nokia Bell Labs Norbert Goetze, Nokia Bell Labs Jens Kuhr, Nokia Joerg Abendroth, Nokia Bell Labs Dmytro Prosvirin, Antonov Airlines Oleh V. Zaritskyi, WRCVE Andrey Kuznetsov, WRCVE Oscar Garcia Perales, i4RI Rafael Vidal Vila, i4RI Rosella Omana Mancilla, ENG

Reviewer(s)	Robert Seidl, Nokia Bell Labs
Document description	The scope of this deliverable is to analyse mechanisms, Indicators and metrics that can be used to represent the vulnerabilities and security posture of components and systems. As such, this deliverable focuses on the notion of "Indicators", what they mean, what uses they can be put to, and how they can be used.

Disclaimer

The TELEMETRY project is funded by the European Union under grant agreement ID 101119747. The information and views set out in this website are those of the TELEMETRY Consortium only and do not necessarily reflect those of the European Union or the European Research Executive Agency (REA). Neither the European Union nor the granting authority can be held responsible for them.

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
0.1	2024-11-01	Initial Draft	Steve J. Taylor
0.5	2024-11-25	Complete Draft	Steve J. Taylor with input from partners
1.0	2024-11-28	Release Candidate	Steve J. Taylor with input from partners

Executive Summary

The scope of this deliverable is to analyse mechanisms, Indicators and metrics that can be used to represent the vulnerabilities and security posture of components and systems. As such, this deliverable focuses on the notion of "Indicators", what they mean, what uses they can be put to, and how they can be used.

This deliverable has documented work undertaken so far in TELEMETRY regarding Indicators. We have defined the concept of Indicators, shown how Indicators fit within the (updated) TELEMETRY framework architecture that enables the different tools created by the project to interact and communicate, and this communication is facilitated by the Indicator concept. We have described the infrastructural components and how they support Indicators, tools that generate different types of Indicators, and other tools that consume Indicators, analyse them and either generate new Indicators or output decision support information to the operator of the TELEMETRY framework to guide courses of action. We have also given a brief example in one of the TELEMETRY use cases to illustrate Indicators' use within a real situation.

This deliverable has also taken an opportunity to provide an update to the TELEMETRY architecture, introduced in D1.1, and this is discussed with respect the concept and processing of Indicators. Within this architecture, TELEMETRY has created several tools that generate (and consume) Indicators, and these are discussed as updates to the tool descriptions in D1.1. Illustrative examples of uses of Indicators are provided by way of descriptions of how they are used by other TELEMETRY tools and in one of the TELEMETRY pilots.

The key purpose of Indicators has been established, which is to provide information for decision support in order to help practitioners, but in addition, the notion of Indicators has proved to be of great use within the project as a communication mechanism, and there is further work to be done regarding Indicators, in all tools that either generate or consume Indicators, as well as the infrastructural components that support their reporting, aggregation, storage and display. Some of this work is in interpreting different Indicators to enable mappings between the values reported in an Indicator and the component consuming it, as illustrated by the SSM consuming results of the ACRAM access control risk tool, which itself consumes Indicators from scanning tools. This work will continue and develop as part of WP2 as it contributes to future tasks such as integration of the TELEMETRY framework and user trials within the project's use cases and will be reported in subsequent deliverables.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Terms and Abbreviations

Abbreviation	Definition
ACRAM	Access Control Risk Assessment Methodology
ADI	Auditable Data Infrastructure
AI/ML	Artificial Intelligence / Machine Learning
BACON	Anomaly-Based Component for intrusion detectiON
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DLT	Distributed Ledger Technology
DUT	Device Under Test
NAD	Nokia Anomaly Detection
rNAD	Network Anomaly Detection
SBOM	Software Bill of Materials
SDL	Security Development Lifecycle
SIEA	Security Information and Event Acquisition
SSM	Spyderisk System Modeller
SUT	System Under Test
TEC	Trust Evaluation Category
UC	Use Case

1 Introduction

The scope of this deliverable is to analyse mechanisms, Indicators and metrics that can be used to represent the vulnerabilities and security posture of components and systems. As such, this deliverable focuses on the notion of "Indicators", what they mean, what uses they can be put to, and how they can be used.

This deliverable has also taken an opportunity to provide an update to the TELEMETRY architecture, introduced in D1.1, and this is discussed with respect to the concept and processing of Indicators. Within this architecture, TELEMETRY has created several tools that generate (and consume) Indicators, and these are discussed as updates to the tool descriptions in D1.1. Illustrative examples of uses of Indicators are provided by way of descriptions of how they are used in some of the TELEMETRY Pilots.

The deliverable is structured as follows. Section 2 defines the concept of "Indicators" as determined by the TELEMETRY approach, so as to contextualise the subsequent sections. Section 0 describing the evolved TELEMETRY architecture follows, along with how the notion of Indicators fits within it. Section 0 describes tools generating and consuming Indicators. Section 0 describes tools that consume Indicators undertake analysis and generate derived Indicators or provide decision support. Section 5.3 describes examples of indicator usage in TELEMETRY Use Case 2 (UC2) concerning Smart Manufacturing. Following this, there is a brief summary of the work done to date and next steps in Section 0.

2 Indicator Concept

2.1 Indicator Definition

Indicators represent observable information about the Device Under Test (DUT) or System Under Test (SUT) that potentially can underpin insight on its *cybersecurity posture*.

As a working definition of "cybersecurity posture", the following from Crowdstrike is appropriate: "*An organization's security posture represents the overall security status of its networks, systems, and procedures. It is a holistic snapshot of your security strengths and vulnerabilities across hardware, software, data, and user behavior.*"[1] RFC 4949 defines "indicator" as "*(N) An action -- either specific, generalized, or theoretical -- that an adversary might be expected to take in preparation for an attack. […] (See: "attack sensing, warning, and response". Compare: message indicator.)*"[2] This solely pertains to the potential for attack or threat, but TELEMETRY has a wider interpretation, regarding the overall status of the DUT / SUT, hence the preference for the Crowdstrike interpretation pertaining to security posture.

For TELEMETRY, Indicators comprise information of relevance for assessment of cyber security status or risk. A fundamental purpose of Indicators is to provide ***decision support*** information to help practitioners. Indicators may provide decision support directly by being presented on the TELEMETRY dashboard but can also serve as input to security analysis components in the

[1] https://www.crowdstrike.com/en-us/cybersecurity-101/exposure-management/security-posture/

[2] https://datatracker.ietf.org/doc/html/rfc4949

TELEMETRY framework, which results in decision support information for practitioners. At the current time, TELEMETRY has identified several subtypes of indicator, described as follows.

- **Threats** (*"Potential cause of an unwanted incident, which may result in harm to a system or organisation."*[3]) present in the SUT.
- Detected **Vulnerabilities** ("Weakness of an asset or control that can be exploited by one or more threats."[3]) in the SUT. Known vulnerabilities may be represented as CVEs (Common Vulnerabilities and Exposures)[4], but previously unknown vulnerabilities may be detected and reported.
- Confirmation of **expected state** / behaviour. This is an indication of normal operation, indicating the absence of anomalies, or incidents or misuse.
- **Incidents** (*"1. A security event that involves a security violation. [...] In other words, a security event in which the system's security policy is disobeyed or otherwise breached. 2. Any adverse event [that] compromises some aspect of computer or network security."*[2]) detected in the SUT.
- **Anomalies** (*"[...] activity that is different from the normal behavior of system entities and system resources. (See: IDS. Compare: misuse detection.)"*[2]) detected in the SUT. TELEMETRY tools provide different types of anomaly detection, including anomalies in component behaviour, network traffic and user behaviour.
- **Effects of Control measures** (*"Measure that is modifying risk. May include any process, policy, device, practice or other action"*[1], *"The management, operational, and technical controls (safeguards or countermeasures) prescribed for an information system which, taken together, satisfy the specified security requirements and adequately protect the confidentiality, integrity, and availability of the system and its information."*[2]) on the SUT.

Based on initial investigations, Indicators have the following properties.

- Indicators are types of signal that may be used as evidence in a decision making process that may result in corrective action being taken.
- Indicators may be derived from other Indicators. TELEMETRY tools may consume one or more Indicators and produce derived Indicators based upon them. A case in point is aggregation of time-series based Indicators to determine whether the behaviour over time is normal or anomalous.
- Indicators can be generated by some components and consumed by others. Indicators may be aggregated to provide empirical evidence, which is a composition of several pre-defined Indicators, for example, one event may not generate an important alarm, but the conjunction and correlation of different events cause an important alarm to be raised.

[3] ISO/IEC 27000:2018. Information technology — Security techniques — Information security management systems — Overview and vocabulary. https://www.iso.org/standard/73906.html

[4] https://www.cve.org/

- Indicators may be used in multiple ways: as direct evidence to the operator of the TELEMETRY framework; as input to TELEMETRY tools, where they may be processed and generate output as e.g. reports or derived Indicators. The operator of the framework is free to use them as they see fit.
- Indicators either pertain to an individual device (i.e. Device Under Test – DUT) or a larger system into which the device is deployed (i.e. System Under Test – SUT). The relationship between DUT and SUT is discussed in Section 3.1.
- Indicators are likely subject to change over time, due to a change in the SUT / DUT. The changed values need to be captured in timely way, in order to provide up to date information, and the trend over time may in itself be an indicator. Moreover, indicator values may need to be measured on demand in order to confirm an unchanged or assumed value (or state), and thus reduce uncertainty of a risk model.
- Indicators are assumed to potentially enrich the precision of the risk models. The Indicators are therefore, in most cases, traceable to one or more specific elements of a risk model. However, single Indicators may also be aggregated into higher levels, in which cases the "high-level", i.e. the aggregated Indicators support a risk model with empirical evidence which is a composition of several pre-defined Indicators. In such a case, the low-level Indicators are directly traceable to the indicator aggregation, while their aggregation is traceable to a risk model.

Overall, the Indicator concept has proven very useful, as both a decision support mechanism for practitioners but also as a communication means between TELEMETRY components. Later parts of this deliverable will illustrate this more fully. Uses of Indicators will evolve as more uses of the TELEMETRY tools are tested, so will be updated in future deliverables.

2.2 Indicator Specification

In TELEMETRY, we have proposed a customized two-fold template aimed to facilitate retrieval of cybersecurity Indicators, namely one template for the original baseline specification of the specific indicator itself, and one template for documenting each instance of measurement of the pre-specified indicator. Thus, for each identified indicator, there is a single (at any time valid) specification, and up to many documented measurement instances (each uniquely annotated and time-stamped). Indicator descriptions have been collected in templates describing the Indicator format and examples, and these are illustrated in Section 0 and 0. The two templates, one for specification and one for instance measurement documentation are shown in Table 1and Table 2, respectively.

Table 1: TELEMETRY indicator specification template

Telemetry Indicator Specification		
Field name	**Input**	**Explanation and guidance**
Unique indicator ID (mandatory)		Unique identifier of the indicator
Short name (mandatory)		A short name of the indicator

Definition (mandatory)		Definition of the indicator - qualitative/quantitative, as well as a definition of the variables/parameters involved. Also includes a specification of relationship with other Indicators and relationship with other parts of the architecture.
Purpose (mandatory)		Defines the purpose that the indicator serves.
Data source (mandatory)		Specifies where to retrieve the indicator values from. (we need to provide example, like firmware version number or tool software version)
Retrieval procedure (mandatory)		Specifies how to obtain the indicator values.
Expected change frequency (mandatory)		Specifies how often the indicator values are expected to change in a realistic setting.
Update/retrieval frequency (mandatory)		Specifies and recommends how often to retrieve the indicator values (given the expected change frequency).
Unit of measure (optional)		Specifies the unit of measure of the indicator.
Interpretation (optional)		Specified the interpretation of indicator values, e.g. which values or ranges of values are desirable, expected, acceptable, unacceptable, etc.
Scale (optional)		Specifies the measurement scale for the indicator.
Uncertainty (optional)		Specifies the uncertainty and the sources of uncertainty. Can e.g. be expressed in the form of intervals, confidence level, variance etc.
Author of specification (optional)		Name / role of the author of the specification.
Responsible for specification update (optional)		Name / role of the responsible for the updates of the specification.
Responsible for measurement (optional)		Name / role of the responsible for the indicator measurements.

Table 2: TELEMETRY indicator measurement template

Telemetry Indicator Measurement Results		
Field name	**Input**	**Explanation and guidance**
Unique Indicator ID (mandatory)		Unique identifier of the indicator, ref. specification.
Short name (mandatory)		The short name of the indicator, ref. specification.
Value (mandatory)		Obtained specific indicator value.
Measurement timestamp (mandatory)		Timestamp for retrieval of this specific indicator value.
Data source (optional)		Source of the obtained indicator value.
Interpretation (optional)		Interpretation of indicator value obtained, based on the guideline from the specification.
Uncertainty (optional)		Uncertainty of the obtained indicator value, expressed as required in the specification.
Follow up actions (optional)		Follow-up actions based on the measurement and its impact.
Measured by (optional)		Name / role of the one (system/person) responsible for obtaining this measurement.
Documented by (optional)		Name / role of the one (system/person) responsible for documenting this measurement.
Responsible for next measurement (optional)		Name / role of the one (system/person) responsible for obtaining next value of this indicator.
Responsible for follow-up actions (optional)		Name / role of the one (system/person) responsible for follow-up actions.
Comments (optional)		Further comments, e.g. needs for indicator specification update, relevant context changes, expected impacts, etc.

Each template consists of three columns: the name of the information field, the indicator-specific input to be provided, and an explanation and guidance column that interprets the purpose of each field. In the first column, next to each indicator name, it is indicated whether the field is mandatory or optional to provide input on. The differentiation between mandatory

and optional is a result of discussions in project meetings, and subject to updates based on lessons learned from future evaluation of the approach.

The templates are described in detail in the following subsections and are used in Section 0 when describing the results of TELEMETRY tools that generate Indicators.

2.2.1 TELEMETRY indicator specification template

The indicator specification template consists of the following fields:

- **Unique indicator ID** (mandatory): Unique identifier of the indicator
- **Short name** (mandatory): A short name of the indicator
- **Definition** (mandatory): Definition of the indicator - qualitative/quantitative, as well as a definition of the variables/parameters involved. Also includes a specification of relationship with other Indicators and relationship with other parts of the architecture.
- **Purpose** (mandatory): Defines the purpose that the indicator serves.
- **Data source** (mandatory): Specifies where to retrieve the indicator values from. (we need to provide example, like firmware version number or tool software version)
- **Retrieval procedure** (mandatory): Specifies how to obtain the indicator values.
- **Expected change frequency** (mandatory): Specifies how often the indicator values are expected to change.
- **Update/retrieval frequency** (mandatory): Specifies and recommends how often to retrieve the indicator values (given the expected change frequency).
- **Unit of measure** (optional): Specifies the unit of measure of the indicator.
- **Interpretation** (optional): Specifies the interpretation of indicator values, e.g. which values or ranges of values are desirable, expected, acceptable, unacceptable, etc.
- **Scale** (optional): Specifies the measurement scale for the indicator.
- **Uncertainty** (optional): Specifies the uncertainty and the sources of uncertainty. Can e.g. be expressed in the form of intervals, confidence level, variance etc.
- **Author of specification** (optional): Specifies name / role of the author of the specification.
- **Responsible for specification update** (optional): Specifies name / role of the responsible for the updates of the specification.
- **Responsible for measurement** (optional): Specifies name / role of the responsible for the indicator measurements.

2.2.2 TELEMETRY indicator measurement template

The indicator measurement template consists of the following fields:

- **Unique Indicator** ID (mandatory): Unique identifier of the indicator, ref. specification.
- **Short name** (mandatory): The short name of the indicator, ref. specification.
- **Value** (mandatory): Obtained specific indicator value.
- **Measurement timestamp** (mandatory): Timestamp for retrieval of this specific indicator value.

- **Data source** (optional): Source of the obtained indicator value.
- **Interpretation** (optional): Interpretation of indicator value obtained, based on the guideline from the specification.
- **Uncertainty** (optional): Uncertainty of the obtained indicator value, expressed as required in the specification.
- **Follow up actions** (optional): Follow-up actions based on the measurement and its impact.
- **Measured by** (optional): Name / role of the one (system / person) responsible for obtaining this measurement.
- **Documented by** (optional): Name / role of the one (system/person) responsible for documenting this measurement.
- **Responsible for next measurement** (optional): Name / role of the one (system / person) responsible for obtaining next value of this indicator.
- **Responsible for follow-up actions** (optional): Name / role of the one (system / person) responsible for follow-up actions.
- **Comments** (optional): Further comments, e.g. needs for indicator specification update, relevant context changes, expected impacts, etc.

3 TELEMETRY Architecture Update & Relation to Indicators

This section describes an update to the architecture presented in D1.1 that reflects the ongoing work in TELEMETRY to bring the components and tools together into a framework, where the tools may be used in different configurations and combinations to enable increased cybersecurity of the device / system under test. As part of this architecture, the Indicators are key communication components between the tools and the operator of the framework, as described in the following subsections.

3.1 Architectural Principles

Two relevant and related architectural principles (device and system under test) have been established as part of the architectural work, discussed next.

3.1.1 Device Under Test vs System Under Test

A key element of TELEMETRY's concept is the relationship between Devices and Systems, as they are the subjects of testing and monitoring of TELEMETRY. For the scope of TELEMETRY, when we consider a Device, it may be a software component, or an IoT device with hardware and firmware. Also, for the scope of TELEMETRY, the term "System" may be defined as *"a set of connected things or devices that operate together"*[5]. This implies that Systems are composed of Devices, which is certainly true, in that a Device may be interconnected and interact with other components in a wider System, for example an IP Camera working in a manufacturing environment interacting with a manufacturing robot in a smart factory System. However, given that IoT and software components are themselves composed of sub-components, e.g. hardware, firmware and software, which is highly likely to be built from third party libraries, it is reasonable to consider Devices as Systems in their own right, and the terms "Device" and "System" are applicable from the perspective of the context of concern and visibility. We therefore use the terms "Device Under Test" (DUT) and "System Under Test" (SUT) to describe the subjects of concern. In summary:

- A System is an interconnected set of components
- A Device can be a component within a larger system
- A Device can be a system in its own right

In some cases, a Device is a black box (e.g. when an IP Camera is a device bought from a supplier and deployed in a smart factory system – as in TELEMETRY use case 2). We are often unable to understand, monitor or control the inner workings of the Device and in many cases, we have to trust the manufacturer. Devices such as IP cameras are often cheap commodity goods with no guarantees of freedom from vulnerabilities or of any software updates should vulnerabilities be discovered. This has given rise to one of the challenges from Taylor (2024) – that the IoT Devices are deployed within Systems we are concerned about, but we have often little control over their vulnerabilities, and they may compromise other Devices in the System.

[5] https://dictionary.cambridge.org/dictionary/english/system

The notion of Software Bills of Materials (SBOMs) is certainly helpful here, as it helps a system deployer to understand the software composition of a Device within their System, but the critical challenge is that SBOMs are often not provided and are difficult to create by the user for some Devices. We may have control over other aspects of the system, for example we can monitor the network traffic from the IP camera and look for anomalous behaviour such as data transmission to unexpected destinations and block them if detected.

In other cases, if we are building the IP Camera, we are concerned with its composition of its sub-components, e.g. third-party libraries, bespoke code and hardware and their interactions so as to perform the function of the camera. Here, we are considering the IP Camera as a System in its own right. If we have this degree of visibility, especially if we are the creator of the Device, we will have made decisions about, and therefore understand, the supply chain of software and hardware that the Device is composed of.

3.1.2 Support for Security Development Lifecycle

The Microsoft Security Development Lifecycle (SDL) exemplifies a modern approach to security by design and in operation. The white paper on evolving the SDL (Ornstein and Rice, 2024) describes the lifecycle of the SDL. The concept of TELEMETRY, plus the function of the TELEMETRY tools and infrastructure directly supports the SDL, as described via the relationship between the TELEMETRY approach and the SDL, which is shown in Figure 1.

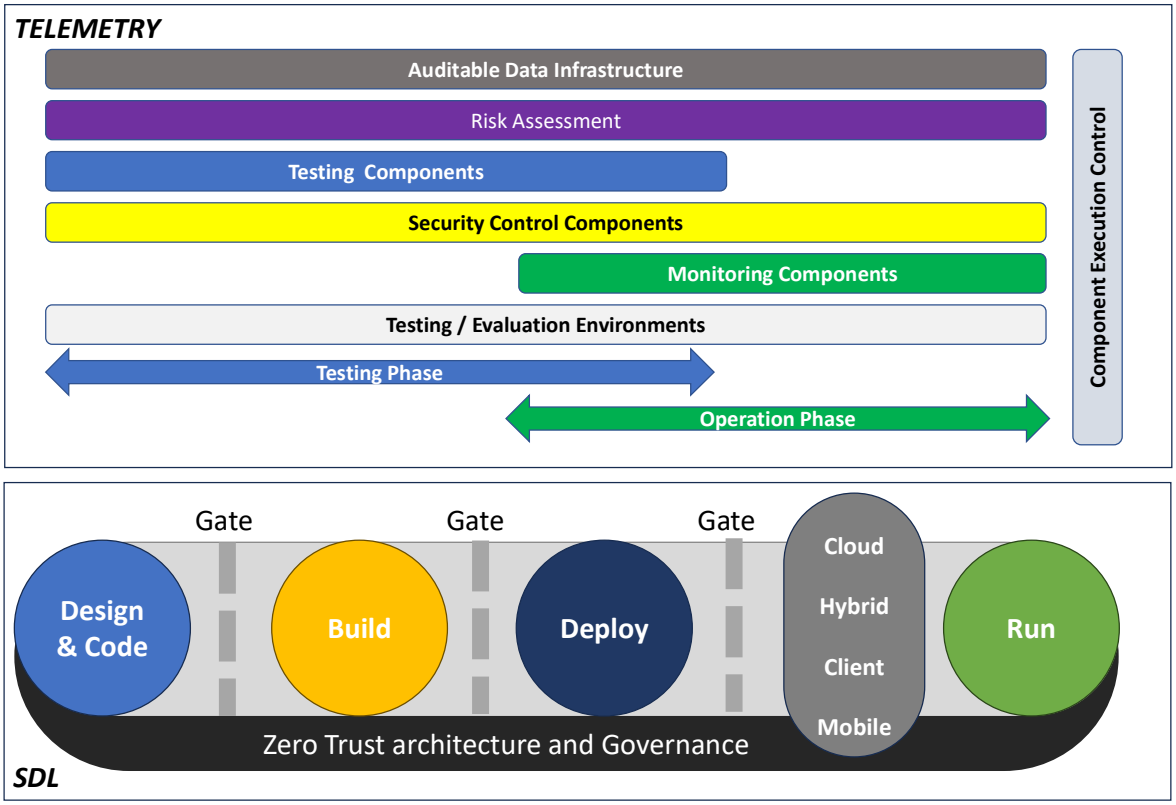

Figure 1: TELEMETRY & Microsoft SDL – SDL reproduced from Ornstein and Rice (2024)

The early phases of the SDL are where the Device is engineered (Designed and Built). Depending on the phase, the software may not yet exist (requirements, design), be incomplete

(security testing), or completely (or as near as) finished (penetration testing). This can include "conformance testing" of third-party hardware before inclusion in a System. The latter phases of the SDL involve Deployment of Devices into systems and operation (Running) of these Systems.

3.2 TELEMETRY Architecture

TELEMETRY supports the SDL via different tools & infrastructure that operate at different lifecycle stages of the DUT / SUT, fall into the following classes. TELEMETRY provides tools in each class, but additional third-party tools may be integrated into the framework. The updated TELEMETRY Conceptual Architecture is shown in Figure 2.

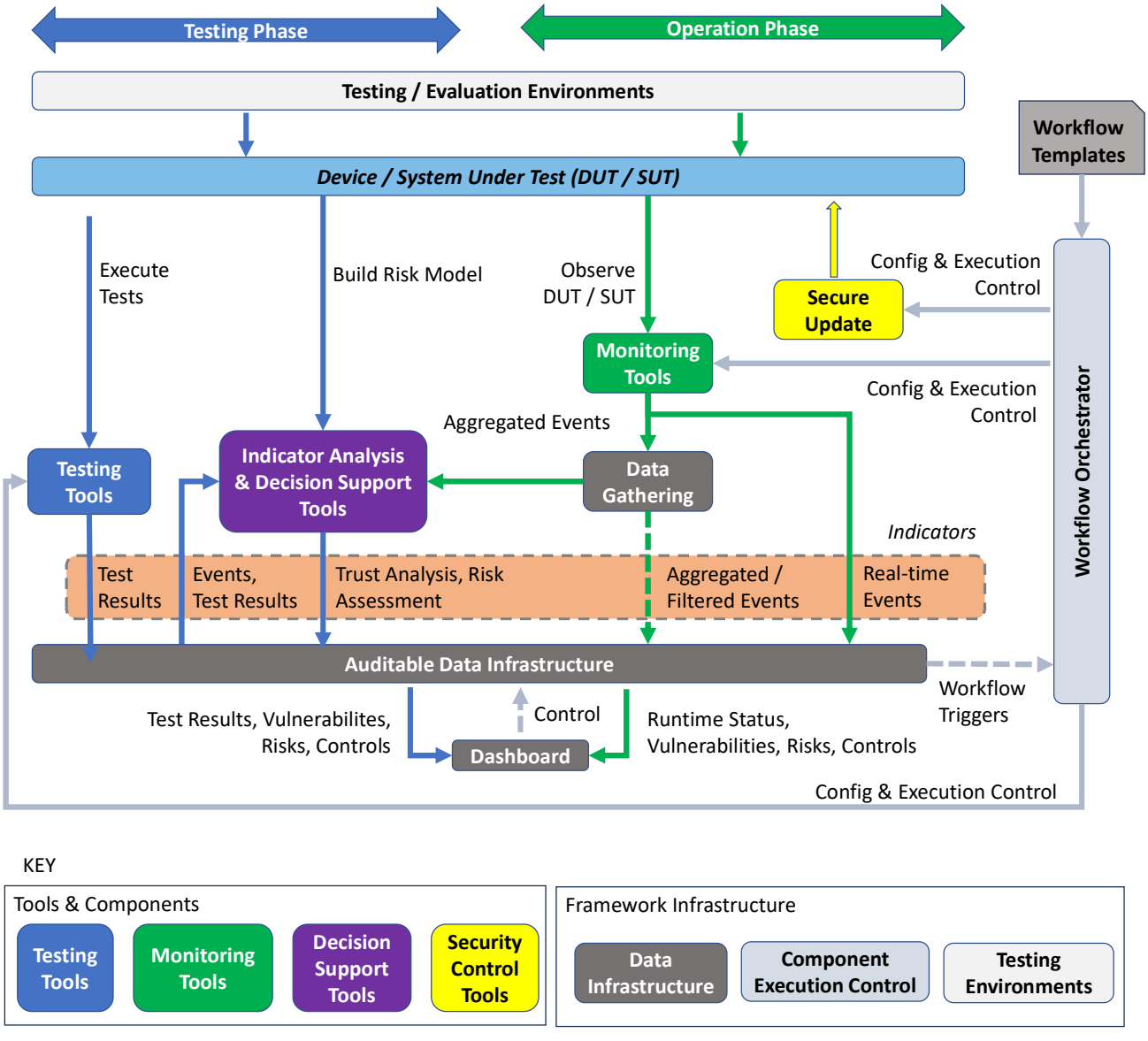

Figure 2: TELEMETRY Conceptual Architecture

Testing Tools (blue) are deliberately executed to test some characteristic of the DUT / SUT and with the expectation of specific outputs. These tools are typically used within the Design & Build phases for a device and its Deployment within a wider system, although testing may be undertaken within an operational system in the Operation phase also. Examples of TELEMETRY testing tools include Network Fuzzing, SBOM generation and Access Control testing.

Monitoring, Analysis and Detection Tools (green) observe the DUT / SUT as it is operating and raise events if specified conditions or anomalous conditions occur that enable us to detect bugs and flaws that we can feed back to the development phase for rectification - "feedback from the field" (McGraw, 2004). These tools are predominantly used with the Operation phase of a system with devices and may generate events that can be communicated to other TELEMETRY tools, such as Risk Assessment or inform additional testing. Such tools may include post-deployment penetration testing tools, intrusion detection tools, and various monitoring tools.

Security Control Components (yellow) may be applied to the DUT / SUT to manage risks identified. These components may be used at any stage of the SDL from Design to Operation, depending on the type of control. There are many examples of controls, from best practices to technical measures and TELEMETRY's focus in this area is focused on a technical mechanism for distribution of updates, so is primarily focused within the Operation phase of the SDL.

Indicator Analysis & Decision Support (purple) consume Indicators generated by other tools, analyse them and provide decision support, either directly to the dashboard, or via derived Indicators. A common thread amongst these tools is that they evaluate trust and risks from different perspectives, for example evaluation of impact and likelihood of compromises detected by the testing or monitoring & detection tools on the DUT or the SUT, along with recommendations of controls if the resulting risk level is unacceptably high. This can occur at any phase of the SDL and may consider device as a system, where, for example vulnerabilities in third party libraries or hardware affect the output of bespoke code. This is more likely in the earlier stages of the SDL, i.e. Design or Build. This analysis may also consider systems of devices at deployment time, i.e. when the system of devices is constructed, in which case, it functions is deliberately executed as a testing too. Alternatively, it may operate at Runtime where it may be informed by the output of monitoring or detection tools, to reactively compute any change in trust or risk of the operating Device or System based on detected events.

Testing / Evaluation Environments – (light grey) are dedicated testbeds, cyber ranges, emulation environments that enable testing under controlled but conditions representative of real deployment conditions. These are typically used at the earlier phases of the SDL (e.g. Design & Build) because they represent testing and evaluation environments as opposed to the real deployment environment, but may also be utilised in later Operation phases (Deploy and Run), especially if they represent Digital Twins of the real environment to enable analysis of dynamic events that occur at operation time.

Auditable Data infrastructure (dark grey) provides means to gather events, to aggregate those events, for tools and other components to securely and auditably exchange information and to interact with testing users. This is a key infrastructural element that operates at all phases of the SDL, as it supports both testing and monitoring / detection.

Component Execution Control (light blue-grey) is infrastructure that enables tools and other components to be configured and executed in different operational sequences depending on the needs of the situation at hand. This will be applicable at all phases of SDL.

3.2.1 Indicators Fit Within TELEMETRY Architecture

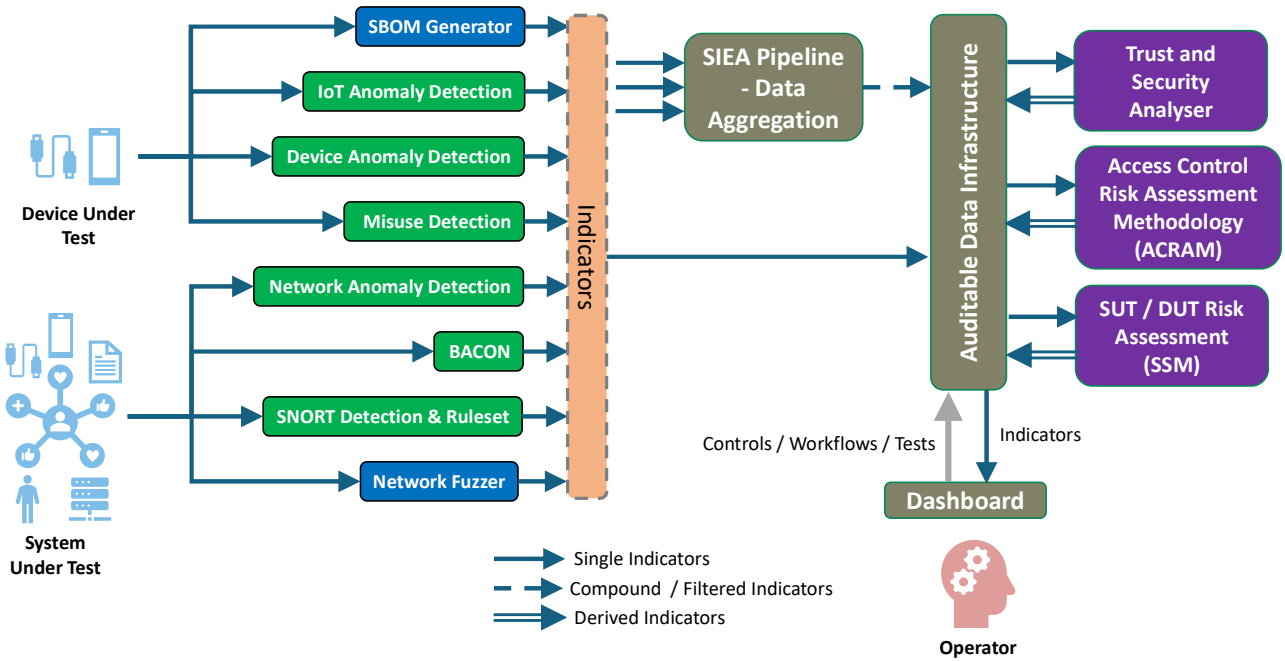

Figure 3: Indicator Generation / Consumption / Filtration / Aggregation / Usage

The relationship between the DUT / SUT, TELEMETRY tools and Indicators is shown in example form in Figure 3. This illustrates that there are tools that monitor (green) or test (blue) the DUT / SUT and produce Indicators. Where appropriate (e.g. in the case of time-series or frequent events), Indicators can be aggregated via the SIEA (Security Information and Event Acquisition) Pipeline to produce a compound or filtered Indicator that derives from the source Indicator. All Indicators (whether they are direct from their source tools or derived) are stored in the Auditable Data Infrastructure. This is a shared data infrastructure to which all components have access. Three tools (at the time of writing) consume Indicators, analyse them and generate decision support or derived Indicators. These are the Trust and Security Analyser, the SUT / DUT Risk Assessment (Spyderisk System Modeller - SSM) and Access Control Risk Assessment Methodology (ACRAM). The infrastructural components are described next, and the tools are described in the following sections, with specifications of the Indicators they produce.

3.3 Infrastructural Components

TELEMETRY has two key Infrastructural components (grey in Figure 2 and Figure 3).

3.3.1 Auditable Data Infrastructure - DLT based Data Sharing

As shown in the TELEMETRY architecture above, an Auditable Data Infrastructure is envisaged as an important element for data sharing among the tools, and as a place where a record of events, test results and actions is maintained. With the variety of tools contributing to the

overall TELEMETRY solution, the data infrastructure serves as a common repository where records are being kept of what was reported by the tools. In order to ensure that the records are reliable, trustworthy and meet requirements for auditability, they have to be resilient against manipulation. To address this TELEMETRY proposes a Distributed Ledger Technology (DLT) based data infrastructure as a data sharing and persistent storage solution for its tool ecosystem. The immutability feature of DLT facilitates tamper-proof records of events and actions. Its distributed nature adds resilience, and it also allows tools in different locations to contribute to the same records and share data such as test results or alerts among themselves.

The DLT based data infrastructure in TELEMETRY extends prior work on a platform known as SmartQC (McGibney et al, 2024) which was developed in a smart manufacturing context. SmartQC provides a JSON API for access to the data infrastructure and facilitates the definition of context and meta data structures that can be extended and validated prior to being committed to an underlying DLT layer. Any data that is committed to the ledger has to adhere to the defined contexts and metadata structures. The result is immutable, auditable storage and retrieval of records of events, actions, Indicators and reports that are generated by other tools.

The platform offers the benefits of DLT, in particular immutability of records shared in a distributed ledger, while abstracting from its complexity by featuring a data sharing JSON API on top, offering an abstraction layer that facilitates the interaction with the data sharing platform through relatively simple JSON messages while making use of DLT features in order to create reliable, auditable records. The abstraction layer further allows for flexibility in the choice of underlying DLT based on the features that are required, instead of being limited to one particular platform. In relation to testing tools such as the Network Fuzzer or the SBOM Generator, events and reports will be received directly from those tools. In the live operation stage, aggregated reports originating from anomaly detection tools will be received via the SIEA pipeline. Through the JSON API, tools such as the SSM or the Trust and Security Analyser can query the data sharing platform for reported data. Notifications will be triggered by incoming reports from testing phase tools and will be sent to inform the SSM of new content.

The existing platform SmartQC, which serves as a basis for the data infrastructure, was developed in an earlier project. It can accommodate a selection of underlying DLT flavours such as Hyperledger Fabric (Androulaki et al, 2018), IOTA, and BigchainDB (McConaghy et al, 2016). This SmartQC platform is modified and extended to address the requirements of TELEMETRY. In relation to functionality, one particular requirement that demands an extension to the current platform is the need for actions or notifications that are triggered by certain transactions. When new reports are received from testing tools such as the Network Fuzzer or the SBOM Generator, these may be of relevance to the risk assessment that is being done by the SSM. Hence, the SSM has to be notified of those new reports, meaning that the transaction that commits the report to the ledger should at the same time trigger such notification. This notification extension and other added functionality that is needed to facilitate integration in TELEMETRY is current work in progress. In addition to the extension of functionality, underlying context and meta data structures are defined in alignment with cybersecurity indicator specifications that are documented in this deliverable, so that there will be a mapping between indicator specifications and the data structures that are to be used in the data infrastructure. This mapping is discussed in more detail in the following subsections.

3.3.1.1 Mapping of Indicators to formats and contexts

While section 2 has outlined the definition of Indicators and the methodology that was applied in order to identify the Indicators that can be produced and consumed by TELEMETRY tools, their use within the tools ecosystem also has to be specified from a technical point of view, i.e., in terms of interfaces and message formats. This applies to their exchange between tools as well as to their storage in the data infrastructure.

As mentioned above, the data infrastructure's concept is built on prior work on the SmartQC platform (McGibney et al, 2024), which uses context and data transactions and the contexts in this concept represent a format definition. This is therefore a logical starting point for mapping or translating Indicators to formats. While not all tools report directly to the data infrastructure, but e.g. to the SIEA pipeline which then aggregates reports, they should still closely follow those formats for consistency and interoperability purposes within the TELEMETRY toolset.

3.3.1.1.1 Contexts and data transactions

A context in the sense in which it is being used in the data infrastructure is a kind of "schema" for content that is being sent to the DLT. Contexts are specified in JSON and describe data and metadata fields and formats, and also include the specification of permissions for who is allowed to submit content related to the context. Data fields are considered immutable, while metadata can be updated in subsequent transactions.

For TELEMETRY, any information that should be recorded in relation to an indicator should be reflected in the context. As such, there is a minimum set of information that has been identified to be required:

- Type of indicator
- Severity – A string containing one of a predefined set of values, e.g.: ON, OFF, GREEN, YELLOW, ORANGE, RED
- Source of the indicator – The entity that reports it
- Subject – The item (device) that the indicator refers to
- Timestamp – An ISO date-time string
- A generic free text field that allows any kind of supplementary detail as supplied by the tool (e.g., details on root causes for an alert)

Optional fields can be:

- Value – Numerical value, representing either a confidence or a numerical severity value.
- Time to live, indicating how long the indicator content is valid.
- Location (will be relevant for sensor data, for example, to identify which sensor readings the indicator is referring to).
- Derived Indicators (i.e. Indicators that are not direct observations but result from the aggregation of other Indicators) may also need some information on what they have been derived from.

The following shows an example of a basic, generic context for an indicator including the fields identified above:

```
{
    "data": {
        "context_data": {
            "type": {
                "name": "type",
                "description": "Type of the indicator",
                "type": "string"
            },
            "severity": {
                "name": "severity",
                "description": "Severity of the indicator, one of OFF, ON, GREEN, YELLOW, ORANGE, RED",
                "type": "string"
            },
            "value": {
                "name": "value",
                "description": "Numerical value, representing either severity or confidence",
                "type": "number"
            },
            "timestamp": {
                "name": "timestamp",
                "description": "Measurement timestamp, ISO date-time",
                "type": "string"
            },
            "subject": {
                "name": "subject",
                "description": "The subject the indicator relates to",
                "type": "string"
            },
            "source": {
                "name": "source",
                "description": "The entity reporting this indicator",
                "type": "string"
            },
            "supplementary-details": {
                "name": "supplementary-details",
                "description": "any further related details",
```

```
                "type": "string"
            },
            "time-to-live": {
                "name": "time-to-live",
                "description": "Period for which the indicator is valid,
optional field",
                "type": "object",
                "period": {
                    "name": "period",
                    "description": "TTL in seconds",
                    "type": "number"
                },
                "timestamp": {
                    "name": "timestamp",
                    "description": "TTL as timestamp",
                    "type": "string"
                },
                "expiry": {
                    "name": "expiry",
                    "description": "TTL as time until it is not newest",
                    "type": "string"
                }
            },
            "location": {
                "name": "location",
                "description": "Location of the subject, optional field",
                "type": "string"
            }
        },
        "context_metadata": {
            "follow-up-actions": {
                "name": "follow-up-actions",
                "description": "Field for follow-up actions to be taken",
                "type": "string"
            }
        }
    },
    "metadata": {
```

```
        "description": "Context definition of a generic indicator",
        "name": "GENERIC INDICATOR",
        "permissions": ["7juxGd7DMNBQokzxgNt5quko9QpdjX5bNCev4f2g1JZE"]
    }
}
```


A related data transaction submitted to the context can then look like this:


```
{
     "context_id":
"2428dd0b7dd87ee7d745ed68b0a9a9093d8e992eeda98c04aba59d3ecf47fbb6",
     "data": {
           "type": "Component level anomaly",
           "severity": "RED",
           "value": 0.85,
           "timestamp": "2024-11-25 12:34:56",
           "subject": "Residential gateway",
           "source": "SNORT tool",
           "supplementary-details": "... more details ...",
           "time-to-live": {
               "period": 86400
           }
     },
     "metadata": {
           "follow-up-actions": "TBD",
     },
     "public_key": "7juxGd7DMNBQokzxgNt5quko9QpdjX5bNCev4f2g1JZE",
     "user_id":
"535f03f0d7aa4903cc997cb421572460db369fbd91bc5bd15972d150f0f45355"
}
```


The context_id in the data transaction is an identifier assigned to the context when the context is created, and it is used to link the data to the context. The user_id identifies the user (which can be a person or a tool) who sends the transaction and is assigned to the user when it is created. The public_key is specific to the user identified by user_id and has to match one of the public keys listed in the context's permissions field. There can be more than one to allow for multiple users to submit data to the context. The follow-up-actions field, being metadata and thus updateable, can be used to recommend or track actions to be undertaken as a consequence of the issue reported in the indicator.

3.3.1.1.2 Mapping the Indicators to contexts

With regard to the mapping between Indicators and contexts, there is a choice between the following options:

- a one-to-one mapping, which means each indicator is mapped to exactly one context;
- a one-to-many mapping, meaning one specified indicator could be mapped onto several contexts, depending on further criteria such as source, subject or severity;
- a many-to-one mapping, which would lead to a very small set of generic contexts that cover a broad range of Indicators.

The choice among these mapping options also depends on other factors such as whether different tools which are reporting similar things use the same Indicators or whether each tool uses its own Indicators. Having indicator definitions per type of output rather than per tool means that the set of Indicators is small enough not to require a many-to-one mapping, essentially ruling out that option. Mapping one indicator to many contexts only adds complexity without any clear benefit, so the obvious choice of a one-to-one mapping remains. In section 2.2.2, an indicator measurement template was defined as part of the effort to collect indicator specifications from TELEMETRY tool owners. The fields in this template are now mapped to fields in contexts.

With a one-to-one mapping of Indicators to contexts, the following relations can be identified:

- The "**Unique Indicator ID**" from the template has its equivalent in the **context_id**.
- The "**Short name**" can be either the "**name**" in the context's metadata, or the "**type**" in the data.
- The "**Value**" is directly represented by the "**severity**" field and the optional "**value**" field in the context. The same applies for the "Measurement timestamp".
- The "**Data source**" can be represented by the "**subject**" field.
- "**Interpretation**" is something that is not reflected in the contexts. Including an interpretation of the indicator value in the message exchange between TELEMETRY tools is considered unnecessary as it can be expected that there is an existing understanding of the interpretation between producers and consumers of these within the context of TELEMETRY.
- "**Uncertainty**" is represented by the optional "**value**" parameter.
- "**Follow up actions**" is represented by a field of the same name.
- "**Measured by**" corresponds to the "**source**", i.e., the tool that reports the indicator.
- The fields "**Documented by**", "**Responsible for next measurement**" and "Responsible for follow-up actions" are not included in the context for now. This may be revisited in the further course of the project if a need arises to include them.
- "**Comments**" can be part of the "**supplementary-details**" field.

3.3.2 Security Information and Event Acquisition (SIEA) Pipeline

In the TELEMETRY framework, multiple upstream anomaly detection tools will report their findings during operation. It is a task of the 'Security Information and Event Acquisition' (SIEA) pipeline to ingest, aggregate, filter and prioritize these security related events and forward compact results to the data infrastructure or downstream tools. This is necessary because

some events (e.g. repeated status messages with the same information but only separated by time) flood the message exchange, effectively creating a benign or inadvertent denial of service attack. Therefore, there is a need to filter, aggregate or summarise these events.

The SIEA pipeline (Figure 4) consists of a set of applications: The *Aggregator* subscribes to a message broker and listens to alerts and insights reported by the upstream tools. The Aggregator is aware of the priorities mapped to each security-related event and can therefore control the flow of data to the Distributor. Next to this feature, the Aggregator can enforce a 'silence period' if multiple alerts of the same type are reported in high frequency by a 'noisy' upstream tool. The *Distributor* is a non-blocking web service capable of buffering events from the Aggregator in case a downstream tool is not ready for ingesting the next chunk of reports. The *SIEA* itself engages in the handshakes of downstream tools guaranteeing the correct processing of the compact reports without overloading the tools themselves. Each compact report relates to only one upstream tool. The applications are supported by a database which stores the status of the downstream tools and the corresponding SIEA application for forensics.

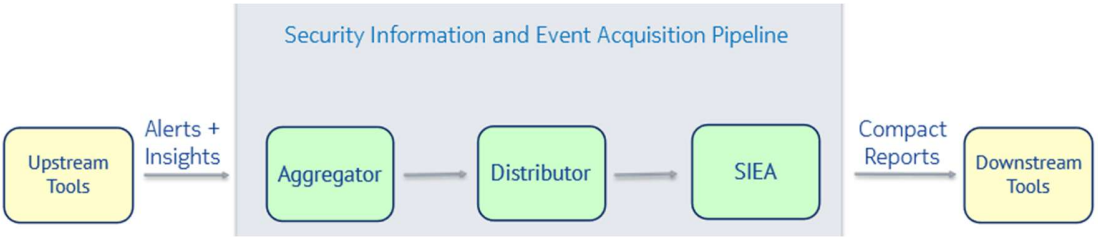

Figure 4: SIEA Pipeline

The SIEA pipeline subscribes to a Kafka broker which publishes events from upstream tools without changing their order. These events are pushed into three queues of different priorities, i.e. high, medium and low. Less prior events can be 'overtaken' by higher prior events. Example: Low-prior events are only forwarded downstream, if the high and medium queues are empty. In order not to flood the downstream tools, the SIEA pipeline groups, aggregates and, in some cases, suppresses recurring events. The SIEA pipeline controls the flow of events to downstream tools via RESTful handshake. The SIEA Pipeline operates in real-time and offers other upstream tools a controlled connection to its consumers. It outputs prioritized, aggregated and filtered events specific to the upstream tools - the pipeline only adds additional information to the JSON e.g. its priority, a potential repetition count and additional fields which the SSM might require.

Current Status: The SIEA Pipeline has been implemented with 4 applications. At this point in writing, all applications are dockerized. The next step is to connect the SIEA pipeline to the NAD pipeline using a Kafka Broker.

Lessons Learned: The Aggregator must be able to ingest messages from different upstream tools. Based on what they deliver, adaptations are required in the Aggregator itself. Analog to this, the SIEA must perform handshakes with different downstream tools. Since the SIEA is dockerized, the plan is to setup different customized instances of the SIEA for each destination tool. The advantages of this approach are a performance boost and an avoidance of blocking situations.

4 Indicator Generators

The tools & components of the TELEMETRY approach that generate Indicators are described in this section. To enable this deliverable to stand on its own, we have reprised some descriptive information from D1.1, supplemented with additional descriptive information, current status, lessons learned and a description of the Indicators generated. Each tool has tables (coloured blue and white, similar to the Indicator specification template in Table 1) following it describing the format of the Indicators generated. In some cases, this is a single type, and in other cases, multiple Indicator types are generated. Further, for some Indicators, examples are given of their measurements for illustration purposes, also in tabular form, coloured white with a green header (similar to the measurement template in Table 2).

4.1 Testing Tools

4.1.1 Network Fuzzer

The network fuzzer (following e.g. Miller et al., 1990) facilitates security testing of network interfaces, by assisting with the detection of unknown vulnerabilities. The tool achieves this by sending a large amount of specifically crafted requests to the interface under test,and observing whether it responds or behaves in an unexpected way. Such unexpected behaviour can indicate the presence of a vulnerability, which an analyst in turn can investigate further.

The main benefit of the tool is that it allows for an automatic detection of unexpected behaviour. By using this as a starting point, security analysts can greatly improve the efficiency of searching for new vulnerabilities. The tool builds on the open source boofuzz[6], which provides a well-documented framework for network fuzzing. Key extensions involve enabling the tool to generate test cases based on network captures from the interfaces of interest, reducing the need for manual configuration and set up, and implementing a standardized way of delivering/presenting test results.

Current Status: We have employed the network fuzzer in the Telco use case (UC3), both on the generic open source OpenWRT, and the specific device used by Telecom Italia.

Lessons Learned: Use of the tool still requires significant manual configuration effort on part of the tester, but in the right hands it produces very useful results. It was also interesting to note that among the differences between OpenWRT and the commercial Residential Gateway, the presence of an additional service was also the source of the principal vulnerability discovered, which was previously unreported.

Table 3: Network Fuzzer Indicator Specification

TELEMETRY Indicator Specification	
Unique indicator ID	Fuzz-1
Short name	Fuzzer
Definition	Running a Fuzzer on a network interface

[6] https://github.com/jtpereyda/boofuzz

Purpose	Find anomalous behaviour
Data source	The service running on the network interface
Retrieval procedure	Running the Fuzzer
Expected change frequency	Once in half a year
Measurement frequency	After every change (update)
Unit of measure	Number of crashes during a specific interval. The output of the tool should be a set of tuples, containing a description of the observed behaviour and an associated packet capture file with the packets required for reproducing the reported behaviour.
Interpretation	All numbers larger than 0 are unacceptable/undesirable. All crashes are treated as equally severe
Scale	Ordinal
Uncertainty	The output is uncertain, as it depends on the input.
Author of specification	SINTEF
Responsible for specification update	SINTEF
Responsible for measurement	Operator

Table 4: Network Fuzzer Indicator Measurement Example

Telemetry Indicator Measurement Results	
Unique Indicator ID	Fuzz-1
Short name	Fuzzer / iperf testing
Value	Crash found
Measurement timestamp	07/06/2024 09:00
Data source	testing iperf interface
Interpretation	Vulnerability found
Uncertainty	Low
Follow up actions	Determine vulnerability severity
Measured by	Lars
Documented by	Lars
Responsible for next measurement	Lars

Responsible for follow-up actions	Lars
Comments	Type of vulnerability unknown. May also allow for remote code execution in addition to denial of service.

4.1.2 SBOM Generator

A Software Bill of Materials (SBOM) is a structured overview of all external libraries or software components used to build a software program/system (see Jaatun et al, 2023). There are currently three major SBOM formats: SPDX[7] from the Linux Foundation; CycloneDX[8] from OWASP; and SWID as defined in ISO/IEC 19770-2[9]. The general idea is that software developers should create a distinct SBOM for every version of their software that they publish, enabling customers and users to quickly determine whether, e.g., a given vulnerability applies to the version they are using.

Even though provision of SBOMs has been mandated by the US Government[10] and mentioned in EU legislation[11], they are still far from ubiquitous and many software lacks SBOMs. We have thus explored to what extent we can create an "aftermarket" SBOM based only on analysis of a binary executable or firmware. We also map the discovered versions to the National Vulnerability Database[12] in order to flag any relevant vulnerabilities. This mapping can be performed periodically to catch any newly discovered vulnerabilities.

The SBOM generator gives an overview of the software components and libraries included in a software product. This will in turn allow the tool to list known vulnerabilities present in the software product and the vulnerabilities' severity.

The purpose of constructing an SBOM is to have an overview of the software components and libraries which are included in a given software product. Having such an overview is a prerequisite for determining which vulnerabilities affect your system, which in turn is essential when performing risk assessments or vulnerability management. While an SBOM is normally generated during the build process of a software product, this tool seeks to perform the same activity after the software has been released, only using the software package or binary file.

Current Status: SBOM generation is currently still a predominantly manual process using the package manager available on the firmware of the SUT. The mapping of library versions to CVEs in the NVD is however an automatic process.

Lessons Learned: The degree of success when applying the SBOM tool depends in a large extent on the properties of the SUT. In the cases where a full SBOM is provided by the manufacturer, the automatic CVE mapping feature will have a close to complete coverage.

---

[7] https://spdx.dev/

[8] https://cyclonedx.org/

[9] https://www.iso.org/standard/65666.html

[10] e.g. https://www.infosecurity-magazine.com/news/us-government-proposes-sbom-rules/

[11] e.g https://medium.com/@interlynkblog/eu-cra-and-sbom-5100c55752fa

[12] https://nvd.nist.gov/

Table 5: SBOM CVE Indicator Specification

TELEMETRY Indicator Specification	
Unique indicator ID	SBOM-CVE
Short name	S-CVE
Definition	Generate CVE from SBOM
Purpose	Find known vulnerabilities and software components in the firmware
Data source	Firmware file of a testing device
Retrieval procedure	Running CVE identification tool
Expected change frequency	Daily
Measurement frequency	Daily
Unit of measure (check if needed)	The tool generates two output files: an SBOM file and a list of identified CVE numbers. The file containing CVE numbers serves as a detailed reference as a unit of measure, providing unique identifiers associated with specific vulnerabilities. These identifiers, known as Common Vulnerabilities and Exposures (CVEs), are standardized labels that allow for the precise tracking and management of security weaknesses in software components.
Interpretation (check if needed)	N/A
Scale (check if needed)	N/A
Uncertainty (check if needed)	CVE numbers may be accurate, but software components listed in the SBOM file may not be accurate.
Author of specification	SINTEF
Responsible for specification update	SINTEF
Responsible for measurement	Operator

4.2 Operation (Runtime) Monitoring, Analysis & Detection Tools

4.2.1 BACON

In the context of cybersecurity and privacy protection, IoT devices have the potential to expand the landscape of risks related to both security and data privacy. As the number of IoT devices grows within an ecosystem, with devices communicating and collaborating not only internally but also externally, the lack of proper security can expose the entire ecosystem to significant risks, including privacy violations due to data leaks.

To mitigate these risks, the infrastructure, including both the devices and the broader IoT ecosystem, must be secured against intrusions and malicious activities. New types of attacks are constantly emerging, and recent research on intrusion detection systems is increasingly focusing on anomaly detection, which aims to identify potential new threats through the analysis and comparison of historical events. However, implementing effective anomaly detection can be challenging, as it may impact system performance and compromise privacy. Centralized analysis of traffic data may be more efficient given the large volume of data, but it risks exposing sensitive information without proper anonymization or data protection.

In this context, Federated Learning offers a promising approach to enhance IoT security while ensuring data protection. This decentralized method enables data to be processed locally on devices, allowing for threat detection without compromising privacy or requiring data to be centralized.

The BACON (Anomaly-Based Component for intrusion detectiON) tool trains and executes Federated Learning based models to detect anomalies in network traffic data based on behavioural patterns identified in historic usage scenarios such as normal activities of devices and infrastructure. The server side is designed for network traffic data based on federated learning approach, it is in charge of implementing the learning/training phase and receives the alerts from the client side, which is deployed on IoT devices/edges which use the training model to identify anomalous events (operational phase).

Application of Federated learning approach ensure (i) trains models locally, on client devices, while maintaining user data privacy, (ii) creates robust models based on data from various resources hence features and patterns, (iii) fosters collaborative security on device level, (iv) periodic, or automatic re-train of the detection model to ensure up-to-date data-analysis.

BACON takes as input a dataset of network traffic (pcap, csv) for offline analysis and/or real time network traffic; and generates alerts notifying anomalous events detected, including information related to the event and potentially the high-level type of the event.

Current Status: Initial version of the training phase and operational phase using known dataset.

Lessons Learned: The application of the training phase and the detection model definition can be influenced by several elements that need to be fined tuned based on the training dataset: type of ML technologies used, supervised and unsupervised. Additionally, the results must be interpretable. While the current version provides only binary classification, future iterations of the solution should offer more detailed and informative outputs.

Table 6: Irregular Traffic Patterns Indicator Specification

TELEMETRY Indicator Specification	
Unique indicator ID	BACON-ITP
Short name	Irregular Traffic Patterns
Definition	Deviation from normal activities in terms of communication with unrecognized sources/destinations
Purpose	Find anomalous behaviour that can lead to potential threats or attacks

Data source	Network Anomaly detection tools (BACON) or the network traffic data
Retrieval procedure	Running the abnormal detection models (inference)
Expected change frequency	Should be rare or occasional
Measurement frequency	measurement or analysis is done in real-time or semi-real time (or offline)
Unit of measure	{0,1} for each element of a given dataset; or alert for 1 or X packets
Interpretation	online analysis: an alert for each irregular pattern packet identified or for X packets; offline analysis: 0 means normal, 1 means irregular pattern in the packet
Scale	number of packets per hr?
Uncertainty	False positives and false negatives
Author of specification	ENG
Responsible for specification update	ENG
Responsible for measurement	Operator

Table 7: Unusual Traffic Volume Indicator Specification

TELEMETRY Indicator Specification	
Unique indicator ID	BACON-UTV
Short name	Unusual Traffic Volume
Definition	Deviation from normal activities in terms of packet quantity
Purpose	Find anomalous behaviour that can lead to potential threats like Distributed Denial of Service (DDoS) attacks or data exfiltration activities by detecting abnormal spikes in traffic
Data source	Network Anomaly detection tools (BACON) or the network traffic data
Retrieval procedure	Running the detection models (inference)
Expected change frequency	Should be rare or occasional
Measurement frequency	measurement or analysis is done in real-time or semi-real time
Unit of measure	packers per seconds or alert

Interpretation	based on training dataset, what is unusual volume should be notify and could potentially be a DDOS or attack
Scale	
Uncertainty	False positives and false negatives
Author of specification	ENG
Responsible for specification update	ENG
Responsible for measurement	Operator

4.2.2 Nokia Anomaly Detection Pipeline

The 'Nokia Anomaly Detection Pipeline' (NAD) aims to ingest sensor readings from IoT devices to predict anomalous behaviour of these devices in near real-time. It is based on training a machine learning (ML) model during normal operation of the DUT that describes expected sensor readings from the DUT: i.e. a fingerprint of the device. When this ML model is then applied during operation, any deviation from the expected sensor readings can be detected and reported. It uses past sensor measurements of a DUT to make predictions of expected behaviour during live operation and compares the prediction (target) with the measurement (actual) and makes a decision to report significant deviations in near-real-time to downstream components.

This tool takes a fingerprint of recorded sensor readings from the DUT operated during normal operation and can detect abnormal situations during live operation. It contains a machine learning application, which is tuned to predict the values of a chosen key performance indicator (cKPI) based on values of other key performance Indicators (KPIs). In this way, the tool is capable of detecting deviations not only from the cKPI but also from the other KPIs which are used to predict the cKPI.

Current Status: A feasibility study of the effectiveness of the NAD pipeline has been successfully launched in the Smart Manufacturing use case (UC2, covered later), when monitoring the sensor readings of a robot in operation. The tool chain has been tested in the use case for both stages of operation: the training phase and the (model) application phase. First results of the NAD Pipeline when the robot is forced to operate in anomalous speeds look promising. We can detect anomalies in the reported speed of joint 1; any deviation in the 6 reported angles of the robot; and anomalies in a fraction of the operation cycle of the robot.

Lessons Learned: It is a known fact that the quality of the training data impacts the performance of an ML-model significantly. We identified irregular gaps in the robot data when analysing plots and tables of this data. Normally, such gaps would be reported as anomalies. To overcome this problem, we crunch the robot data over a pre-configured timeslot via timeseries processing. With a timeslot of 2 or 3 seconds, the resulting data showed no gaps anymore and can be used for model building and model application.

Table 8: Nokia Anomaly Detection Pipeline Indicator Specification

TELEMETRY Indicator Specification	
Unique indicator ID (mandatory)	NAD-P
Short name (mandatory)	Nokia Anomaly Detection Pipeline
Definition (mandatory)	Near real time detection of anomalies in measurements from IoT devices
Purpose (mandatory)	Detection of anomalies
Data source (mandatory)	IoT Sensors
Retrieval procedure (mandatory)	Comparison with fingerprint recorded during normal operation
Expected change frequency (mandatory)	Ideally never, if IoT device operates normally. Worst case, every 2 or 3 seconds, if alarm state toggles during this time.
Update/retrieval frequency (mandatory)	Irrelevant, as NAD is always on during operation
Unit of measure (optional)	4 levels of alarms: off, yellow, orange and red. Alarms containing information on when an anomaly is detected mapped to an alarm condition scale (off -> yellow -> orange -> red) which indicates how long the anomaly persists. As soon as the DUT operates normally, the alarm condition is de-escalated (red -> orange -> yellow -> off). The output information also contains timestamps when transitions in alarm conditions occur.
Interpretation (optional)	off=normal operation / red=serious alarm
Scale (optional)	interval
Uncertainty (optional)	confidence level of alarm
Author of specification (optional)	Nokia
Responsible for specification update (optional)	Nokia
Responsible for measurement (optional)	Operator

Table 9: Nokia Anomaly Detection Pipeline Indicator Measurement Example

Telemetry Indicator Measurement Results	
Unique Indicator ID (mandatory)	Nokia Anomaly Detection Pipeline
Short name (mandatory)	NAD
Value (mandatory)	Alarm
Measurement timestamp (mandatory)	25.07.2024 10:26:45
Data source (optional)	ML model application insight
Interpretation (optional)	escalation by exceeding alarm threshold --> 2024-07-25 10:26:45 error:-9 upDownCnt:10 confidence:0.2767679908051982 2024-08-02 10:44:21,182 WARNING: 2024-07-25 10:26:45: raising status of alarm from OFF to RED
Uncertainty (optional)	based on confidence level (see above)
Follow up actions (optional)	must be performed by downstream components
Measured by (optional)	Norbert/Nokia
Documented by (optional)	Norbert/Nokia
Responsible for next measurement (optional)	Norbert/Nokia
Responsible for follow-up actions (optional)	Owner of upstream tool(s)
Comments (optional)	

4.2.3 Misuse Detection ML Toolkit

The *Misuse Detection ML Toolkit* is a set of runtime libraries that includes several AI/ML algorithms with the capability of training ML models. These ML models will be trained to detect the misuse by humans of software components & systems, developing ML for Intrusion Detection (Lee & Stolfo, 1998, Vinayakumar et al, 2019, Stibor et al. 2005) towards misuse detection based on baseline normal usage behavioural patterns. These patterns would be identified in historic usage scenarios such as normal activities of users and/or similar patterns on log files or data storages. The ML will learn from user-interaction and the detection of divergences in user behaviour from the norm, using similar principles to social engineering

for capturing user aspects such as user functional footprint, temporal behaviour and statistical data distribution. The Toolkit is composed of three main modules: *Model trainer*: allows training of AI/ML models; *Model manager*: allows the management and deployment of trained models; *Execution runtime*: allows the execution and serving of deployed trained models.

The Toolkit allows users with little or no analytical knowledge to train their own AI/ML models and deploy them in a central platform ready to be incorporated into either bigger and wider applications as a submodule or as a standalone callable service. It offers the capability of clustering the execution of several trained models so that their results can be viewed in the same graph so that they can be compared and analyse the right source of the abnormal behaviour detected. The Toolkit allows users with little or no analytical knowledge to train their own models by following a wizard approach that will guide the user across screens and options depending on the algorithm selected. In addition, the clustering feature allows the live comparison of the results of the results produced by all *clustered* trained models in the same view.

The input that the *Model trainer* is prepared to receive is any type of sensor-related (numeric-based) data from the DUT / SUT. The *Model manager* is capable of managing and deploying trained AI/ML models regardless these have been trained with the *Model trainer* or not. When executing a trained AI model through the *Execution runtime*, the input that any AI/ML trained model can receive will be sensor-related data from the DUT / SUT. These data will be relayed by the *Execution runtime* to the pod where the trained model will be containerized. The *Model trainer* provides the trained AI model, that can then be deployed in the target system through the *Model manager.* The *Execution runtime* publishes alerts referring to the abnormal behaviour detected from the cluster of trained models being executed. These alerts would then be relayed to the next downstream component.

Current status: The first version of the Model Trainer is being tested by the Smart Manufacturing use case (UC2) where historical baseline normal behaviour activity from sensors is analysed. These data is compiled all in a JSON file for triggering the training phase of several models, one for each element of the UC2 robot likely to be analysed. For the first attempts to train the UC2 datasets, a two-layered Convolutional Neural Network (CNN) from Keras is being used with 64 filters, activation *relu* and a hidden layer based on MaxPooling1D and GlobalMaxPooling1D configuration so that the normal behaviour of the robot can be modelled, and the anomalies fabricated by UC2 can be used to validate the training of the dataset. As such, six models per variable to analyse, namely speed, torque and position, are trained; so that, these six models can be clustered per each variable. The next development to be included here will be the ability to read SHAP-based charts that explain how the model works so that the final user is able to understand the model trained.

Lessons Learned: The model that was initially trained for UC2 has evolved by doubling the initial layers and adding more filters. In addition, continuous initial data collection has led i4RI and its Data Scientists to adapt the model to the changes in structure and frequency of the data provided.

Table 10: Misuse Detection

TELEMETRY Indicator Specification	
Unique indicator ID (mandatory)	MISS
Short name (mandatory)	i4RI Misuse Detection
Definition (mandatory)	Near real time detection of abnormal behaviour against normal recorded user activity in measurements from IoT devices
Purpose (mandatory)	Detection of misuse of devices
Data source (mandatory)	IoT Sensors
Retrieval procedure (mandatory)	Comparison with fingerprint recorded during normal operation
Expected change frequency (mandatory)	Best case scenario: never as device operates normally. Worst case scenario: as defined by the frequency of the readings per UC.
Update/retrieval frequency (mandatory)	Irrelevant, as MISS is always on during operation
Unit of measure (optional)	True / False
Interpretation (optional)	An anomaly detected/not detected
Scale (optional)	interval
Uncertainty (optional)	confidence level of alarm
Author of specification (optional)	i4RI
Responsible for specification update (optional)	i4RI
Responsible for measurement (optional)	SUT Deployer / Operator

4.2.4 SNORT Detection and Ruleset

The *SNORT Detection and Ruleset* aims to provide additional Indicators based on network traffic, based on different rules. Three types of rule set are being utilized. The basic rules, which are rules about general bad behaviour, such as blacklisted traffic, forbidden protocols, etc. The specific ruleset includes rules based known vulnerabilities or bad actors, specific rules point to CVE or explanation of why an alert is being sent. Finally, the advanced ruleset are rules based on research about early warning rules that map observed software development activities to actual network traffic. The advanced ruleset firings are Indicators and not as strong as e.g. the specific ruleset alerts. These Indicators can support the confidence of derived Indicators created by other downstream components.

Current Status: A feasibility study on the basic and specific ruleset, and research on the mechanisms required for advanced ruleset.

Lessons Learned: Due to early state of advanced ruleset only limited preliminary lessons learned can be offered. The public information and the data format required for the advanced ruleset for the TELEMETRY UC2 seems sufficient.

Table 11: Snort Basic Indicator Specification

TELEMETRY Indicator Specification	
Unique indicator ID (mandatory)	SNORTBas
Short name (mandatory)	Snort Basic
Definition (mandatory)	Rules running in the SNORT toolset
Purpose (mandatory)	Alarm of packets that are suspicious, known malicious and suspicious based on AI rulings
Data source (mandatory)	PCAP, mirror port
Retrieval procedure (mandatory)	network stream parsed and checked by SNORT
Expected change frequency (mandatory)	Every Second if an alarm has been found
Update/retrieval frequency (mandatory)	Every Second if an alarm has been found
Unit of measure (optional)	rules can be of different classes basic and each can have severity levels, such as suspicious (known malicious is the specific rule)
Interpretation (optional)	The interpretation should be clear, as the rules have a known reason for firing. E.g. a rule belongs to a specific threat or attack. The AI based rules will be linked to component and actor.
Scale (optional)	
Uncertainty (optional)	can be with very low uncertainty (e.g. known malicious linked to specific attack)
Author of specification (optional)	Nokia
Responsible for specification update (optional)	Nokia
Responsible for measurement (optional)	SUT Operator

Table 12: SNORT Indicator Specification

TELEMETRY Indicator Specification	
Unique indicator ID (mandatory)	SNORTSpec
Short name (mandatory)	SNORT Specification
Definition (mandatory)	Different rules running in the SNORT toolset
Purpose (mandatory)	Alarm of packets that are known malicious
Data source (mandatory)	PCAP, mirror port
Retrieval procedure (mandatory)	network stream parsed and checked by SNORT
Expected change frequency (mandatory)	Every Second if an alarm has been found
Update/retrieval frequency (mandatory)	Every Second if an alarm has been found
Unit of measure (optional)	rules can be of different classes basic, AI based and each can have severity levels, such as suspicious, known malicious
Interpretation (optional)	The interpretation should be clear, as the rules have a known reason for firing. E.g. a rule belongs to a specific threat or attack. The AI based rules will be linked to component and actor.
Scale (optional)	
Uncertainty (optional)	known malicious traffic linked to specific attack
Author of specification (optional)	Nokia
Responsible for specification update (optional)	Nokia
Responsible for measurement (optional)	SUT Deployer / Operator

Table 13: Snort AI Rule Alarms

TELEMETRY Indicator Specification	
Unique indicator ID (mandatory)	SNORTAIbased
Short name (mandatory)	Snort AI Rule Alarms
Definition (mandatory)	Different rules running in the SNORT toolset
Purpose (mandatory)	Alarm of packets that are suspicious based on AI rulings

Data source (mandatory)	PCAP, mirror port
Retrieval procedure (mandatory)	network stream parsed and checked by SNORT
Expected change frequency (mandatory)	Every Second if an alarm has been found
Update/retrieval frequency (mandatory)	Every Second if an alarm has been found
Unit of measure (optional)	rules can be of different classes basic, AI based and each can have severity levels, such as suspicious, known malicious
Interpretation (optional)	The AI based rules will be linked to component and actor, but are only Indicators
Scale (optional)	
Uncertainty (optional)	can be with very low uncertainty
Author of specification (optional)	Nokia
Responsible for specification update (optional)	Nokia
Responsible for measurement (optional)	SUT Operator / Deployer

4.2.5 r-Monitoring - Monitoring & Analysis of System Processes, Metrics and Network Traffic

The tool aims to enhance system security by providing comprehensive monitoring and analysis of system processes, metrics, and network traffic following patterns suggested by Shao et al. (2010). It includes dynamic file monitoring capabilities, which track changes to critical system files and directories in real-time. Any unauthorized or suspicious modifications are flagged and alerted to the system administrator as these could be indicative of malicious activities. Additionally, the tool continuously scans and evaluates running processes against known malware signatures and anomalous behaviour patterns to identify potential threats, ensuring proactive threat detection and response. This multifaceted approach fortifies the system's resilience against a wide array of security threats.

The tool provides comprehensive monitoring and analysis of system processes, monitoring metrics, file monitoring. Signature checking and network traffic:

- *Resource Monitoring* (CPU & memory consumption); *System Monitoring* (detailed process consumption);
- *File Monitoring* (monitor sensitive files for changes); *Hash Monitoring* (check hashes for altered processes);
- *Network capturing and Monitoring* (signature monitor); *Anomaly Detection on Network Data*; Record *Historical Data*; output to Dashboard.

The agent designed for collecting metrics, system parameters, and network traffic boasts a lightweight architecture that is compatible with various CPU architectures. Its efficient design makes it especially well-suited for devices with constrained resources (e.g. IoT), ensuring broad applicability without compromising performance. While several tools on the market provide some of the features listed above (e.g. Casola et al, 2019, Ghanem et al, 2013), it is rare to find a single tool that encompasses these capabilities comprehensively and even more for devices with constrained resources.

Current Status: The initial version of the monitoring agent has been successfully developed and tested on IoT devices, specifically ZTE router provided by the Telecoms Use Case (UC3). This agent effectively captures key metrics and network traffic data, which it then forwards to the monitoring application for further analysis.

Lessons Learnt: IoT devices are often constrained by limited resources, necessitating the development of the monitoring agent in a low-level language to efficiently manage these restrictions. Additionally, the diverse architectures present in IoT devices require thorough exploration and investigation. This is crucial for developing techniques that enable the application to operate seamlessly across all devices, achieving broad compatibility and performance optimization in a variety of hardware environments.

Table 14: Resource Anomaly Indicator Specification

TELEMETRY Indicator Specification	
Unique indicator ID (mandatory)	rMonitoring-Resources
Short name (mandatory)	rMonitoring Tool - Runtime System Level Monitoring Tool - Resources
Definition (mandatory)	Real time monitoring of IoT devices and computing devices
Purpose (mandatory)	Alarm of abnormal resource consumption
Data source (mandatory)	rMonitoring Tool agent
Retrieval procedure (mandatory)	measurements are parsed and checked by rMonitoring Tool agent
Expected change frequency (mandatory)	depends on abnormalities, techniques will be employed to reduce the frequency of these anomalies without affecting usability.
Update/retrieval frequency (mandatory)	depends on abnormalities, techniques will be employed to reduce the frequency of these anomalies without affecting usability.
Unit of measure (optional)	Each outcome will include description of status
Interpretation (optional)	Result will be binary alongside with the process that caused irregular resource consumption
Scale (optional)	
Uncertainty (optional)	N/A

Author of specification (optional)	ATC
Responsible for specification update (optional)	ATC
Responsible for measurement (optional)	SUT Deployer / Operator

Table 15: Resource Anomaly Indicator Measurement Example

Telemetry Indicator Measurement Results	
Unique Indicator ID (mandatory)	rMonitoring-Resources
Short name (mandatory)	rMonitoring - Runtime System Level Monitoring Tool - Resources
Value (mandatory)	status, process causes consumption
Measurement timestamp (mandatory)	Timestamp of measurement taken
Data source (optional)	measurements provided by rMonitoring agent
Interpretation (optional)	Identify abnormal measurements based on an established normal based on rules. Sudden change on resource consumption can indicate activation of malware or an attack.
Uncertainty (optional)	N/A
Follow up actions (optional)	report to Telemetry framework
Measured by (optional)	Antonis Mpantis (ATC)
Documented by (optional)	Antonis Mpantis (ATC)
Responsible for next measurement (optional)	Antonis Mpantis (ATC)
Responsible for follow-up actions (optional)	Antonis Mpantis (ATC)
Comments (optional)	

Table 16: Process Anomaly Indicator Specification

TELEMETRY Indicator Specification	
Unique indicator ID (mandatory)	rMonitoring-Processes
Short name (mandatory)	rMonitoring Runtime System Level Monitoring Tool - Processes

Definition (mandatory)	Real time monitoring of IoT devices and computing devices
Purpose (mandatory)	Alarm of sudden change on processes, known CVEs, malicious processes
Data source (mandatory)	rMonitoring Tool agent
Retrieval procedure (mandatory)	measurements are parsed and checked by rMonitoring Tool agent
Expected change frequency (mandatory)	depends on processes starts, stops and changes
Update/retrieval frequency (mandatory)	depends on processes starts, stops and changes
Unit of measure (optional)	Each outcome will include the process name and known CVEs, malicious flag
Interpretation (optional)	Result will be binary alongside with the process that caused irregular resource consumption
Scale (optional)	
Uncertainty (optional)	N/A
Author of specification (optional)	ATC
Responsible for specification update (optional)	ATC
Responsible for measurement (optional)	SUT Deployer / Operator

Table 17: Network Monitoring Indicator Specification

TELEMETRY Indicator Measurement Results	
Unique indicator ID (mandatory)	rMonitoring-Network
Short name (mandatory)	rMonitoring Tool - Runtime System Level Monitoring Tool - Network
Definition (mandatory)	Real time monitoring of IoT devices and computing devices
Purpose (mandatory)	Alarm of network traffic from or to malicious destinations or not whitelisted, keep track of network traffic type, signature malicious scan
Data source (mandatory)	rMonitoring Tool agent
Retrieval procedure (mandatory)	measurements are parsed and checked by rMonitoring Tool agent

Expected change frequency (mandatory)	If there is a large stream of packets tagged as abnormal, techniques will be employed to reduce the frequency of these anomalies without affecting usability. In some cases, this process can occur within milliseconds
Update/retrieval frequency (mandatory)	If there is a large stream of packets tagged as abnormal, techniques will be employed to reduce the frequency of these anomalies without affecting usability. In some cases, this process can occur within milliseconds
Unit of measure (optional)	Each outcome will include the source, destination, port, malicious flag
Interpretation (optional)	Result will include all necessary details for further actions
Scale (optional)	
Uncertainty (optional)	N/A
Author of specification (optional)	ATC
Responsible for specification update (optional)	ATC
Responsible for measurement (optional)	SUT Deployer / Operator

Table 18: File Monitoring Indicator Specification

TELEMETRY Indicator Specification	
Unique indicator ID (mandatory)	rMonitoring-File
Short name (mandatory)	rMonitoring Tool - Runtime System Level Monitoring Tool - File Monitoring
Definition (mandatory)	Real time monitoring of IoT devices and computing devices
Purpose (mandatory)	Alarm of changes on sensitive files
Data source (mandatory)	rMonitoring Tool agent
Retrieval procedure (mandatory)	measurements are parsed and checked by rMonitoring Tool agent
Expected change frequency (mandatory)	Depends on frequency of changes of sensitive file
Update/retrieval frequency (mandatory)	Depends on frequency of changes of sensitive file

Unit of measure (optional)	Each outcome will include the file that has been change, timestamp, user that changed (when it is possible)
Interpretation (optional)	Result will include all necessary details for further actions
Scale (optional)	
Uncertainty (optional)	N/A
Author of specification (optional)	ATC
Responsible for specification update (optional)	ATC
Responsible for measurement (optional)	SUT Deployer / Operator

Table 19: File Monitoring Indicator Measurement Example

Telemetry Indicator Measurement Results	
Unique Indicator ID (mandatory)	Runtime System Level Monitoring Tool - File Monitoring
Short name (mandatory)	rMonitoring Tool
Value (mandatory)	file name, timestamp, user
Measurement timestamp (mandatory)	Timestamp of measurement taken
Data source (optional)	measurements provided by rMonitoring agent
Interpretation (optional)	Changes on sensitive files can be considered as malicious actions or product of a malicious process
Uncertainty (optional)	N/A
Follow up actions (optional)	report to Telemetry framework
Measured by (optional)	Antonis Mpantis (ATC)
Documented by (optional)	Antonis Mpantis (ATC)
Responsible for next measurement (optional)	Antonis Mpantis (ATC)
Responsible for follow-up actions (optional)	Antonis Mpantis (ATC)
Comments (optional)	

4.2.6 r-Anomaly Detection

This tool is designed to monitor network traffic and identify unusual patterns that deviate from established norms (as established by a provided dataset of typical activity). The tool pinpoints specific features that contribute to each detected anomaly (e.g. Lundberg & Lee, 2017), using intrusion detection approaches exemplified by Sommer and Paxson (2010) and Fuentes-García et al. (2021) against a predefined baseline of typical activity. It employs machine learning algorithms (see e.g. Wang et al, 2021) to identify deviations and the underlying causes of these irregularities, that may suggest security threats or system malfunctions. The purpose is to enhance network security and reliability by promptly flagging potential issues. It utilizes a sophisticated algorithm to analyse a dataset representing healthy or typical network activity. By continuously comparing incoming traffic data against this baseline, the tool efficiently flags anomalies, which could indicate potential security threats or system failures. The r-Anomaly Detection tool is dedicated in interpreting network traffic, then detecting and assessing protocol-specific deviations (regarding the protocols used and monitored) from the reference datasets. It monitors network traffic, which consists of data packets, each containing parameters from various layers, and produces a JSON object that contains three values: one indicating whether an anomaly was detected, one describing the severity of the deviation from the defined baseline and a third defining the list of parameters that are considered abnormal.

Current Status: The current status is that we are analysing the data provided by Smart Manufacturing (UC2) and Telecommunications (UC3) to extract valuable insights. These insights are essential for designing the architecture of our model, and ensuring it is tailored to meet specific requirements and performance.

Lessons learnt: Network data is characterized by a variety of protocols, each with distinct features. Exploring and investigating methods to handle this diversity and anomalies occurring in each case presents a significant challenge. Tackling this challenge has taught us the importance of flexibility in model design to adapt to varied data types. Our pursuit to develop a robust model has emphasized the need to investigate advanced analytics and machine learning techniques. These methods are crucial for effectively handling and interpreting diverse network traffic, and for assessing its divergence from baselines, with the aim of improving reliability and consistency in our outcomes.

Table 20: Network Anomaly Detection Indicator Specification

TELEMETRY Indicator Specification	
Unique indicator ID (mandatory)	rNAD
Short name (mandatory)	Network Anomaly Detection
Definition (mandatory)	identifying traffic that deviates from the established baseline of normal behaviour. Such deviations may indicate malicious activities or actions that are not suitable or safe for the infrastructure.
Purpose (mandatory)	Alarm of packets that deviate from established normal
Data source (mandatory)	PCAP, mirror port

Retrieval procedure (mandatory)	network stream parsed and checked by rNAD
Expected change frequency (mandatory)	If there is a large stream of packets tagged as abnormal, techniques will be employed to reduce the frequency of these anomalies without affecting usability. In some cases, this process can occur within milliseconds
Update/retrieval frequency (mandatory)	If there is a large stream of packets tagged as abnormal, techniques will be employed to reduce the frequency of these anomalies without affecting usability. In some cases, this process can occur within milliseconds
Unit of measure (optional)	Each outcome will have severity measurement
Interpretation (optional)	Result will be binary alongside with the feature which caused anomaly
Scale (optional)	
Uncertainty (optional)	false positives, false negatives
Author of specification (optional)	ATC
Responsible for specification update (optional)	ATC
Responsible for measurement (optional)	SUT Deployer / Operator

Table 21: Network Anomaly Detection Advanced

TELEMETRY Indicator Specification	
Unique indicator ID (mandatory)	rNAD-A
Short name (mandatory)	Network Anomaly Detection Advanced
Definition (mandatory)	Analyse packets flagged as abnormal to gather additional information, such as the type of attack, known vulnerabilities, or indications of possible vulnerabilities.
Purpose (mandatory)	An alert with additional context about the detected anomaly, providing details such as the nature of the anomaly, its potential impact, and any related threats
Data source (mandatory)	mirror port
Retrieval procedure (mandatory)	network stream parsed and checked by rNAD

Expected change frequency (mandatory)	If there is a large stream of packets tagged as abnormal, techniques will be employed to reduce the frequency of these anomalies without affecting usability. In some cases, this process can occur within milliseconds
Update/retrieval frequency (mandatory)	If there is a large stream of packets tagged as abnormal, techniques will be employed to reduce the frequency of these anomalies without affecting usability. In some cases, this process can occur within milliseconds
Unit of measure (optional)	Each outcome will have description of situation
Interpretation (optional)	
Scale (optional)	
Uncertainty (optional)	false positives, false negatives
Author of specification (optional)	ATC
Responsible for specification update (optional)	ATC
Responsible for measurement (optional)	SUT Deployer / Operator

5 Indicator Consumers, Analysis & Decision Support

This section describes tools that consume existing Indicators, undertake analysis of their parameters and values, and provide decision support or derived Indicators.

5.1 Trust and Security Analyser

Trust is a subjective consideration shaped by the trustor's past experiences and perceptions. Therefore, it is crucial to establish a system that develops this knowledge and aids the trustor in making informed decisions based on trustworthiness. Trust analysis plays a vital role in risk assessment, particularly in domains such as cybersecurity, organizational governance, supply chain management, and system design. It assesses the level of confidence in the integrity, reliability, and performance of entities within a system or relationship, identifying potential vulnerabilities or risks arising from misplaced or excessive trust.

The Trust and Security Analyzer evaluates trust using seven distinct Trust Evaluation Categories (TECs). Five of these categories actively monitor system behavioural changes, while the remaining two utilize passive monitoring to provide the Trust and Security Analyser with updates based on their findings. The TECs include five categories derived from the Industrial Internet Consortium (IIC) trust framework[13], along with two additional categories specifically designed to address considerations from other support systems. The seven TECs are:

- Security
- Reliability
- Resilience
- Uncertainty and dependability
- Goal analysis
- Safety (Passive monitoring)
- Privacy (Passive monitoring)

As presented in Figure 5, once Trust and Security Analyser gathers information from the Indicators for TECs it will then aggregate and output the trust score in various formats based on the requirement of the use case.

[13] Industrial Internet Consortium: The Industrial Internet of Things Trustworthiness Framework Foundations. https://www.iiconsortium.org/pdf/Trustworthiness_Framework_Foundations.pdf

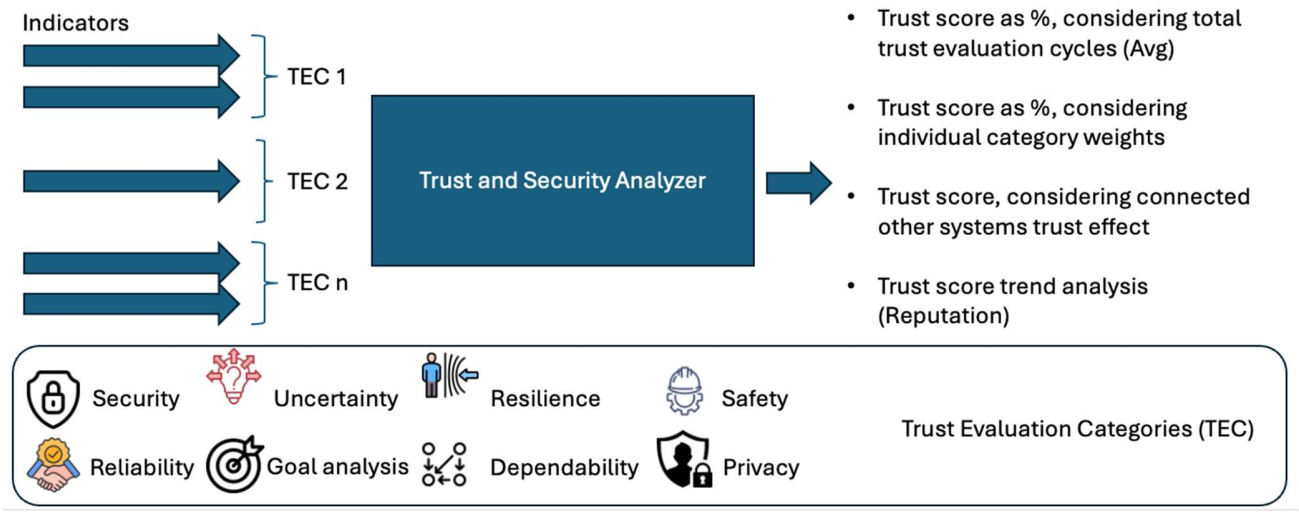

Figure 5: Trust and Security Analyser concept

In the context of TELEMETRY, it aggregates reported Indicators into a trust model, and it uses the score obtained through that model to assess if the SUT is operating in a manner that is deemed trustworthy. Trust is regarded as a medium to long-term metric here, and the trustworthiness of the SUT can assist decision-making when it comes to assigning workloads to the SUT. In UC2, for example, the trust score will apply to the overall "Factory in a Box" setup, and in a manufacturing scenario where multiple such "Factories in Boxes" are available, it can help in distributing the workload among these according to their trustworthiness. Also, it can serve as a metric that the owner of those factories can advertise to potential clients.

Table 22: Trust Assessment Indicator Specification

TELEMETRY Indicator Specification	
Unique indicator ID (mandatory)	TRUST
Short name (mandatory)	Trust Assessment
Definition (mandatory)	Trustworthiness value for a component of the SUT
Purpose (mandatory)	Indicate whether a component is trustworthy
Data source (mandatory)	Trust and Security Analyser
Retrieval procedure (mandatory)	Retrieving reports from data space and applying trust model
Expected change frequency (mandatory)	TBD (not very frequent)
Update/retrieval frequency (mandatory)	TBD (not very frequent)
Unit of measure (optional)	Unitless numeric value or percentage

Interpretation (optional)	High values indicate trustworthiness, low values indicate lack of trust
Scale (optional)	0 to 1 or 0% to 100%
Uncertainty (optional)	TBD
Author of specification (optional)	MTU
Responsible for specification update (optional)	MTU
Responsible for measurement (optional)	SUT Operator

5.2 WRCVE ACRAM (Access Control Risk Assessment Methodology)

Complex IT infrastructures may require a single comprehensive access control solution. This is especially relevant for networks that were scaled unevenly, without a pre-designed architecture, with a change in the main responsible in the process, as well as a number of software and administrative solutions that do not make up an optimal and coordinated system. A methodology is proposed, according to which Subjects (users) and Objects (services) are evaluated according to significant factors and with the help of a mathematical model based on fuzzy logic, the risk of providing access is assessed. Bakurova et al. (2024) and Lytvyn et al. (2024) provide full details of the approach, which is intended to help make more informed situational or system decisions for access management.

In this approach, values of Indicators and influence factors input are transformed into a set of "if-then" rules as output, where each rule establishes a correspondence between the factors and the level of risk. A set of such rules forms the knowledge base of the system. The values of the factors are processed by these rules, and at the output the model provides an integrated risk assessment. A set of significant factors is suggested in Bakurova et al. (2024) and Lytvyn et al. (2024) but can be modified by the administrator of the SUT. Each factor has 3 to 5 levels of significance, which are determined based on standards (such as CIS Benchmark, CVEs, etc.), infrastructure management practices, and administrator vision. Subjects are evaluated according to such factors as authentication level, access level, abnormality of behaviour, etc. Objects are represented by such factors as the level of vulnerability, the frequency of access, the level of data sensitivity, etc. Interaction factors such as attack vector or network type are also suggested. These are all combined, to determine risk levels considering probability of an attack and the level of seriousness of the attack.

The mathematical model of the methodology is based on fuzzy logic, which allows to consideration of ambiguity in the description of the state of the system and assessments of its vulnerabilities. By combining modern network monitoring tools, modern vulnerability libraries and comprehensive IT infrastructure data analysis using fuzzy logic, we achieve a more objective and effective risk assessment. This combination outperforms other existing approaches and methodologies or analyses that lack any of these components (Lytvyn et al, 2024).

Current status: Initial experiments are complete, as described in Lytvyn et al. (2024). The results of testing the proposed methodology can help to improve the applied access policies and the access control system itself, including at the human-machine level.

Lessons Learned: This approach allows to estimate how often an object is used, which can indicate its importance or potential vulnerability, but how exactly it will affect and what its meaning says will be clarified further in combination with other factors in the mathematical model. Further development of the work includes expanding the pool of Indicators to increase the application potential of the methodology, integration with widely used network management tools, and increasing the adaptability of the methodology through testing on real IT structures. Also considered is the improvement of the mathematical apparatus for more accurate estimates and the creation of a base for applied AI tools.

The general concept of the ACRAM methodology (Figure 6) involves the use of Indicators (signals) that characterize the state of the information system at a certain time: during system testing; during changes in both the system architecture and its components (subjects and objects), as well as during system operation (operational phase).

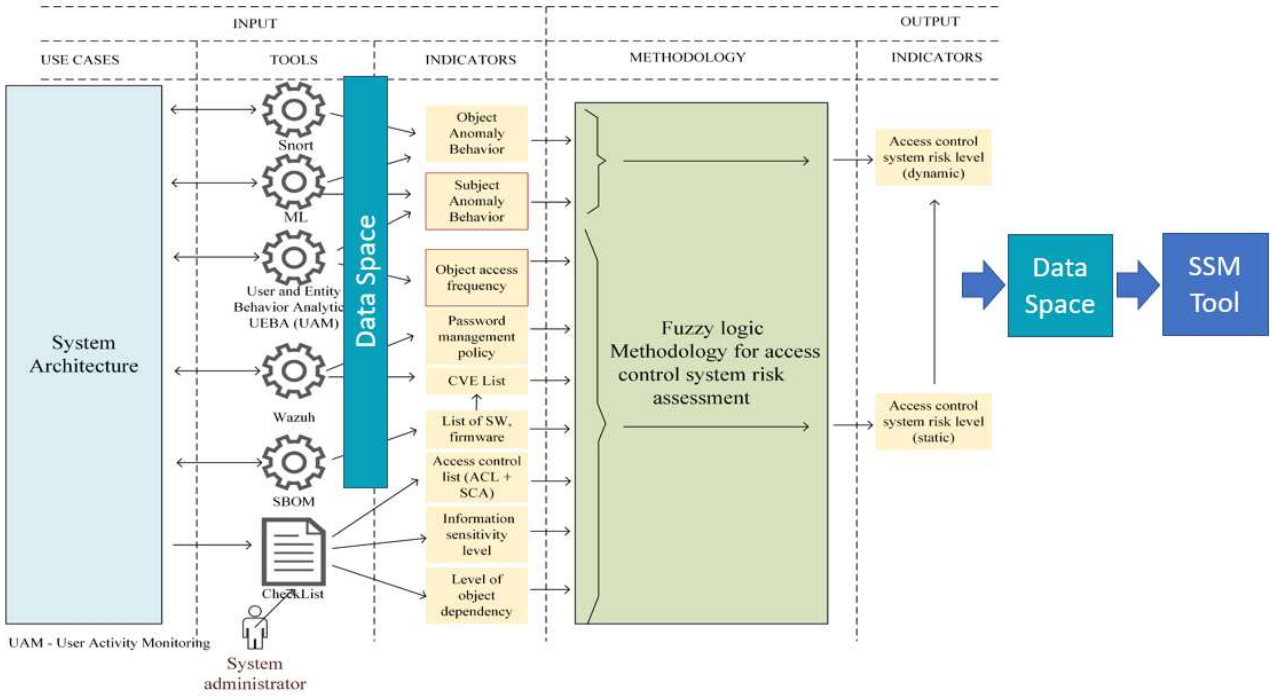

Figure 6: Concept of the ACRAM methodology

The main Indicators used as facts in the rules for calculating the level of risks, built on the basis of fuzzy set theory, are as follows:

I. Dynamic Indicators obtained using various tools in the project:

Object anomaly behaviour.

Subject anomaly behaviour.

Object access frequency.

Password management policy (can be obtained both using systems and from checklists, for example, an access control list)

CVE List or list of SW and firmware of subjects.

II. Indicators (static) that can be obtained using checklists:

1. Access control list.

Information sensitivity level of subjects, for example, different data bases with personal data etc.

Level of object dependency. This indicator is calculated based on the system architecture using graph theory.

A fragment of the integration of methodology rules and Indicators from different tools is shown in Figure 7. The figure presents several rules of methodology and a comparison of the facts of the rules with specific Indicators from the tools.

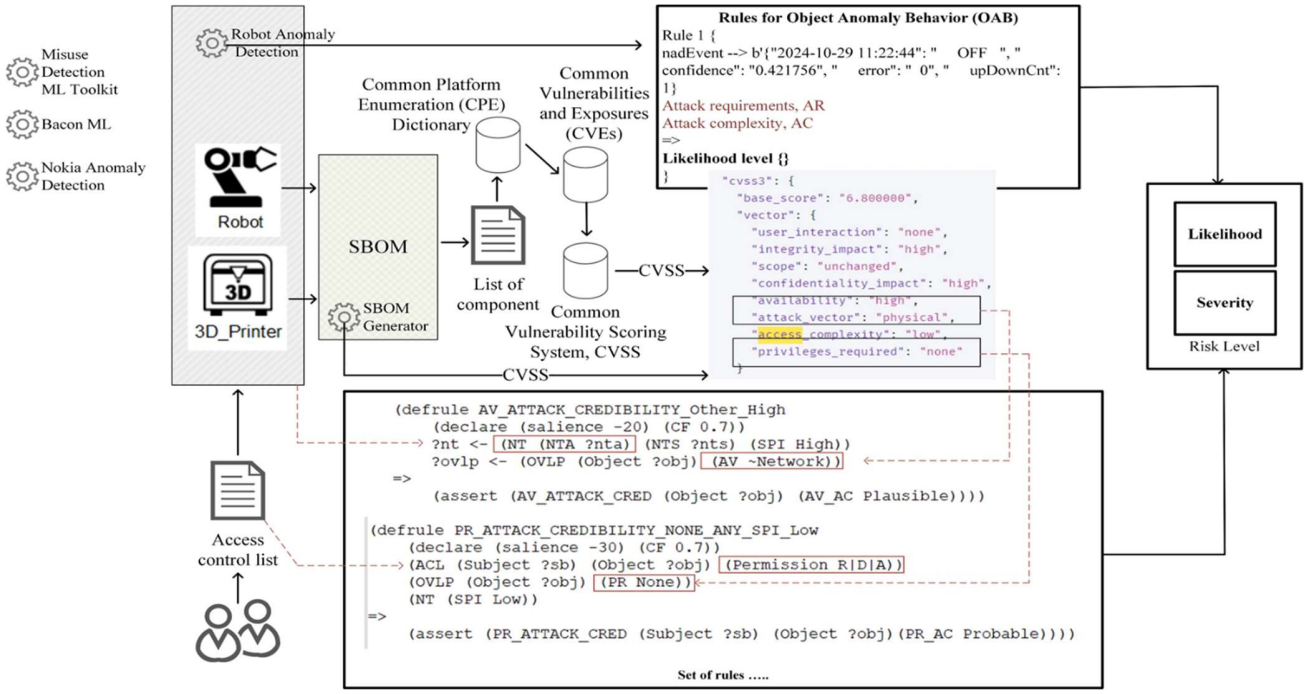

Figure 7: Integration of methodology rules and Indicators from different tools

Figure 8 presents the process of modelling and calculating the risk levels of the access control system in the event of receiving a signal from the data infrastructure regarding the occurrence of anomalous behaviour of a certain object (in this case, a robot). In this case, the context of the indicator is imported from the data infrastructure into the modelling environment, where the risk level is recalculated taking into account the new state of the system.

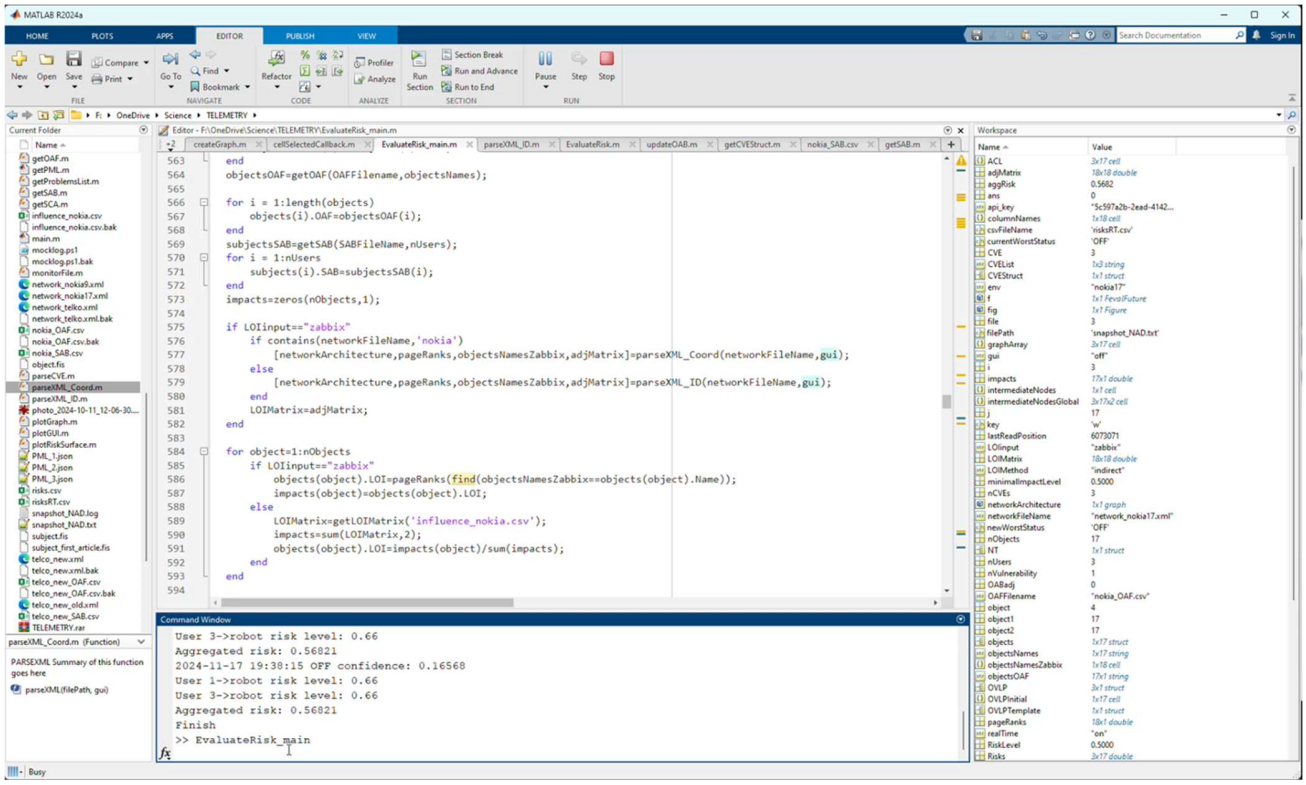

Figure 8: Modelling process for new system state due to anomaly signal

The modelling results are presented in Figure 9. Risk levels have different colours, as can be seen from the figure, the risk increases in the event of detecting an anomaly in the behaviour of the subject if such an anomaly is provided for in the vulnerability vector, i.e. the use of the exploit involves an anomaly (strange behaviour) in the behaviour of either the object or the subject of the system.

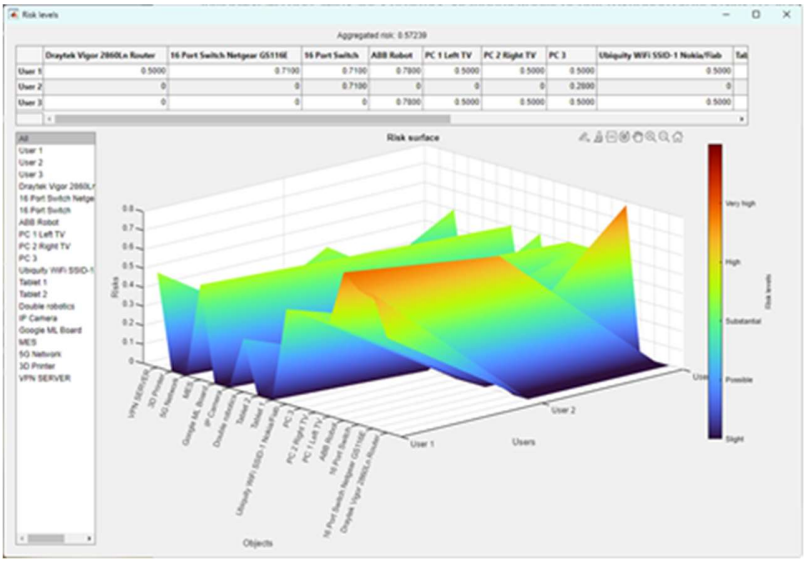

Figure 9: Result of risk level modelling

The algorithm for applying the methodology at different stages of operation of the information system of use cases is presented in Figure 10.

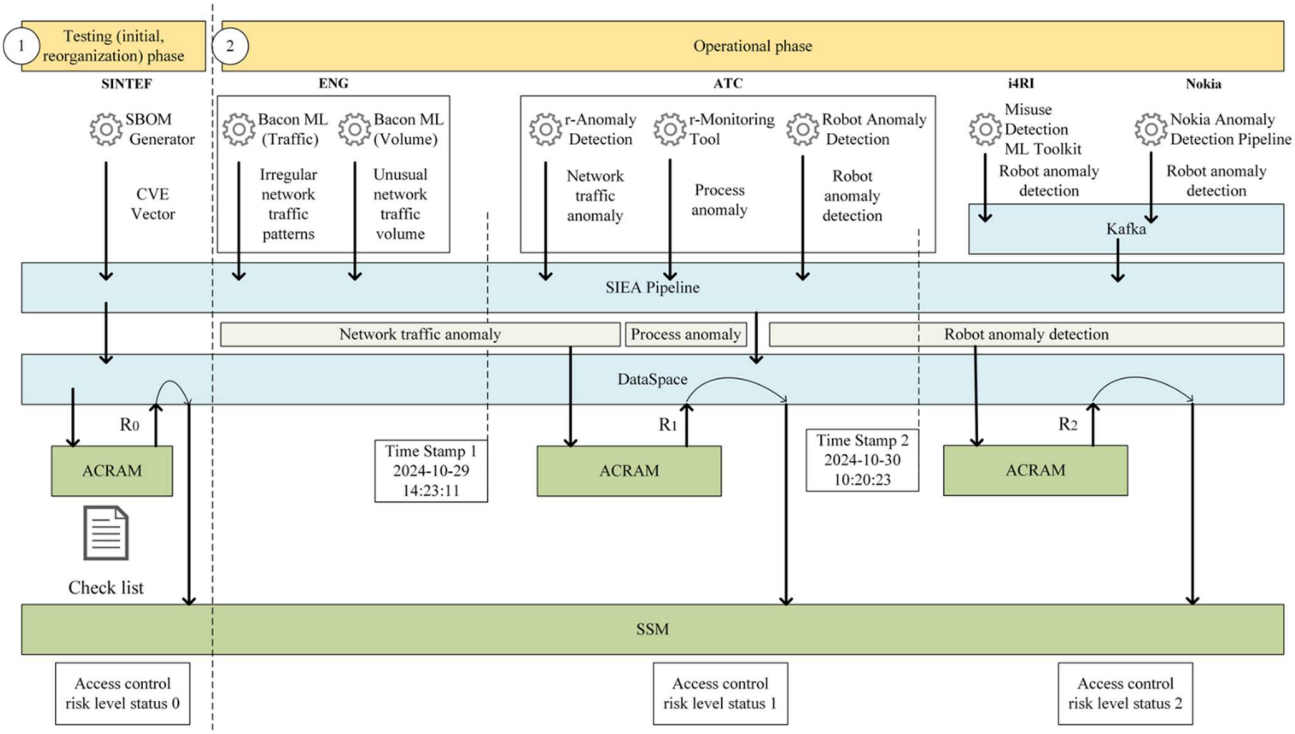

Figure 10: Algorithm for applying the methodology

The calculation of the risk level of the access control system is carried out at different time points. At the beginning of testing or deployment of the system, we estimate the initial risk level (R_0), during the operation of the information system, in case of receiving signals from the system status monitoring tools, the methodology recalculates the initial risk levels (R_1 -> R_2) taking into account the real state of the system (anomaly level). New data is obtained during the operational phase through the appropriate data exchange environments (Kafka -> SIEA Pipeline -> Data infrastructure). The initial data of risk levels is further transmitted to the SSM tool for consideration in the overall risk assessment of the system.

Table 23: ACRAM Indicator Specification

TELEMETRY Indicator Specification Template	
Unique indicator ID	ACRAM-00001
Short name	Risk_level
Definition	Quantitative/ measures the risk level of the access control system in terms of users and their access rights to system objects

Purpose	identify weaknesses in the access control system during the system initiation phase and calculate the risk level, as well as monitor changes in the system during its operation and calculate changes in risk levels depending on the actual state of the system objects
Data source	ACRAM methodology
Retrieval procedure	Running the ACRAM methodology, that uses other (partners) tools as an input data (for ex. Snort, ML etc.)
Expected change frequency	discrete or event-based
Measurement frequency	The methodology works at the following points in time: 1. During the initiation (deployment) of the enterprise information system. 2. In the event of a change in the architecture of the enterprise information system. 3. In the event of receiving signals about the detection of attacks or vulnerabilities from the partners' real-time tools.
Unit of measure	%
Interpretation	based on rules
Scale	0-1
Uncertainty	CF level = 0.0 -1.0
Author of specification	System administration
Responsible for specification update	System administration
Responsible for measurement	System administration

5.3 Spyderisk System Modeller (SSM) – DUT / SUT Risk Assessment

The Spyderisk System Modeller, abbreviated to SSM (Phillips, 2024) is a comprehensive automated risk management toolkit designed to enhance a system's security via the assessment of risks and recommendations of controls to lower the likelihood of risks with an unacceptably high level. SSM enables users to construct detailed system models (models of the SUT / DUT), identify cybersecurity and compliance risks, and determine the most effective mitigations. Its user interface is shown below in Figure 11.

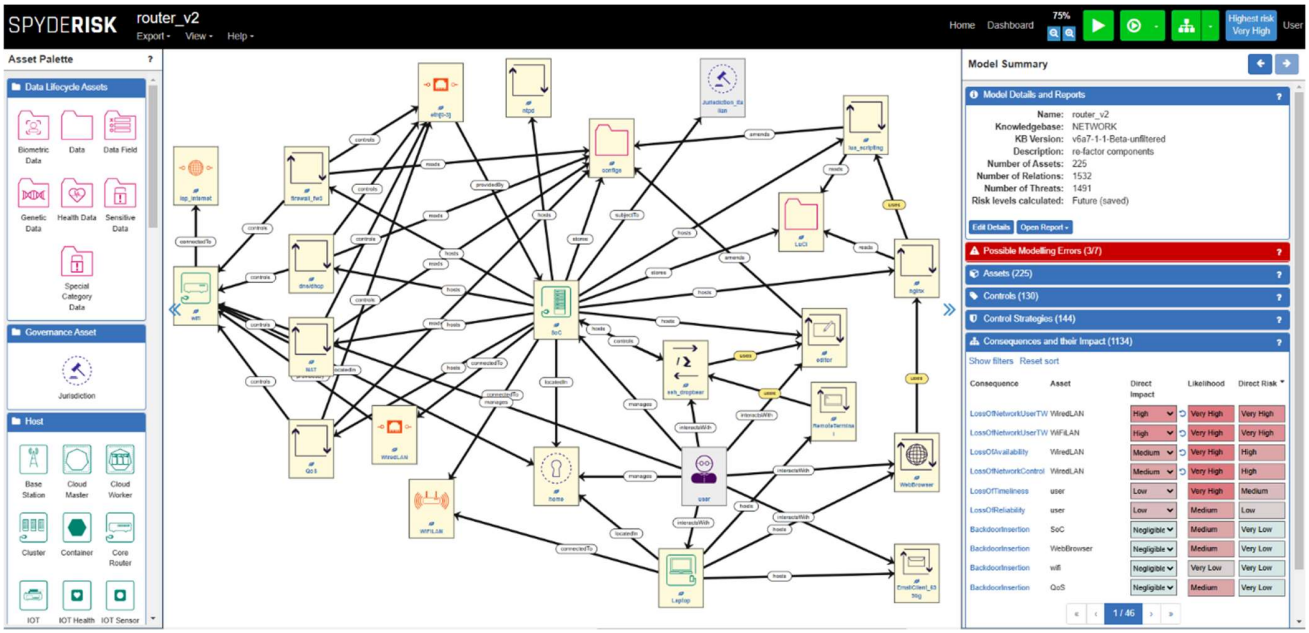

Figure 11: SSM Risk Modelling User Interface

A panel of asset icons is shown at the left, from which a system model can be constructed in the central panel, and at the lower right individual risks (named Consequences) are shown. Each risk can be explored to determine threats that cause it, and controls are recommended by the tool. The controls can be selected and the risk levels re-calculated to see the effect of the controls on the risks.

SSM is a knowledge-based risk assessment tool enabling system-level risk assessment at both design time and runtime. SSM system model concepts are compliant with the ISO standards 27000/27005. SSM enables users to construct detailed system models (models of the system under test), identify cybersecurity and compliance risks, and determine the most effective mitigations. Spyderisk enables modelling and analysing risk in information systems, based on ISO 27005, part of the ISO 27000 family for information security. It is at TRL6 and has been in development for 9+ years. Knowledge of vulnerabilities, threats, risks and controls to manage risk are encoded in a knowledge base designed to support automation using a cause-and-effect approach to risk modelling. The user constructs a model of their system under test, and the tool uses its knowledge base to identify relevant risks and threats and calculate risk levels, and to recommend controls to lower the likelihood of risks. It differs from existing methods of risk assessment in that other methods are largely based on checklists and human judgement, whereas Spyderisk is a simulator of the causes and effects of cyber threats in interconnected systems.

Dynamic context and threat propagation via interdependencies of assets and consequences offer a continuous monitoring and updating of risk assessments as new threats emerge or as changes occur in the system. Dynamic cybersecurity risk assessments are informed by the TELEMETRY Indicators. This adaptive feature enables the assessment of emerging risks at runtime along with by dynamically recommending and implementing necessary security controls, so they be addressed in a timely manner, enabling the maintenance of a robust defence.

SSM is deployed in Smart Manufacturing (UC2) and a process of mapping the Indicators from tools such as the anomaly detection tools to vulnerabilities in components in the risk model is underway. The SSM is also deployed in Telecommunications (UC3), where a risk model of the DUT (a domestic modem-router, known as the Residential Gateway) has been created.

Indicator Mapping

The key inputs are Indicators, representing outputs of all tools as described in Section 0. The semantics of these represent different aspects such as component vulnerabilities, anomalies detected, results of network scanning, etc, and so are many and varied. In order for them to be meaningful, they need to be mapped to elements inside a risk model, and the values reported translated into the SSM discrete scale. The SSM uses CVSS (currently v2 is supported but extensions to v4 are underway) to represent vulnerabilities in the risk model, which facilitates this mapping, but the mapping is highly domain and application specific. This work is in progress, but examples follow to illustrate the considerations and processes in mapping the Indicators to the risk modelling in SSM.

UC3 – Router: A model of the router DUT for TELEMETRY UC3 (Telecoms) has been created as a system in its own right, comprising constituent functional blocks of the router such as NAT, routing tables, switching hardware, DNS / DHCP, firewall, wifi module, ethernet connection, http server for configuration, etc. The SBOM generation tool that analysed the router DUT and provided an SBOM, which provides information of software libraries and components that represent the functional blocks of the router with versions, which can then be linked to vulnerabilities via queries to CVEs databases with the software libraries and versions as parameters. The results of these queries are CVEs with associated CVSS scores, which can then be mapped into SSM risk models via its support of CVSS. The result of this is that CVEs representing vulnerabilities are mapped into a risk model of the DUT's subcomponents. The values in the CVSS vector can then be applied to the SSM risk model, changing its vulnerability values for those affected components, and the risk levels for the whole of the router can be recalculated to see the effects that the vulnerabilities reported by the CVEs have caused.

ACRAM – Access Control Risk Management: A model of a system of interacting components including an access control system can be created, and the results of ACRAM can be mapped into this model. When ACRAM predicts a high risk of access control failure, the vulnerabilities associated with the access control system can be adjusted accordingly and the overall risks in the system recomputed, to show the propagated effects of the access control system failure.

r-Monitoring File Monitoring Indicator: In a similar manner to the previous example, a model of the SUT including the files being monitored can be constructed. Files are data objects and have vulnerabilities associated with their Confidentiality, Availability and Integrity. Should the r-Monitoring detect e.g. tampering or unauthorised access, the associated vulnerability in the file in question can be adjusted in the model, and the system risks recomputed.

Table 24: SSM Indicator Specification

TELEMETRY Indicator Specification Template	
Unique indicator ID	SSM-01
Short name	SSM Risk Vector

Definition	Qualitative risk levels, expressed as: • Type of risk • Impact of risk • Likelihood of risk
Purpose	Describe different types of risks in the SUT along with their assessed level
Data source	SSM Tool
Retrieval procedure	Running the SSM risk calculation. May be triggered by an incoming event that adjusts vulnerability levels in risk model, which will mean that the risk levels need to be re-calculated as a result of the change
Expected change frequency	Depends on frequency of notification reports
Measurement frequency	Upon receipt of notification of change of a specified vulnerability relevant in the risk model
Unit of measure	Discrete values: “Safe”, “Low”, “Medium”, “High”, “Very High”
Interpretation	The output represents the current status of the risk model in terms of different risk present in the SUT / DUT and their risk levels. Also included is a list of potential recommendations. Each recommendation specifies the controls that need to be implemented and details the expected risk reduction resulting from the implementation of these controls.
Scale	N/A
Uncertainty	N/A
Author of specification	UoS
Responsible for specification update	UoS
Responsible for measurement	Likely automatic but can be triggered by SUT Operator. Results interpreted by SUT Operator

6 Indicator Usage: UC2 – Smart Manufacturing

The Smart Manufacturing Use Case (UC2) is illustrated by the architecture in Figure 12. This shows the schematic structure of UC2, where anomaly detection tools for network traffic, as well as for operational data of a production machine, are employed.

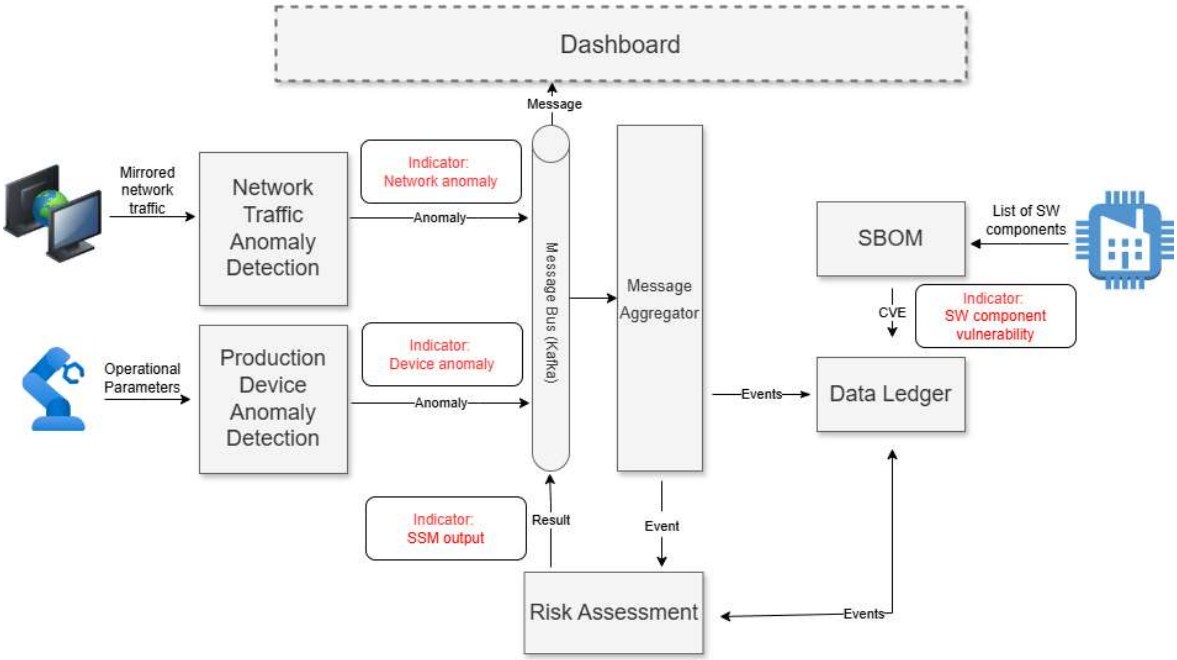

Figure 12: Use Case 2 Architecture

The monitoring tools generate Indicators based on the raw data with which they are fed. These Indicators are called "observable" Indicators because they are generated directly from the observed raw data.

The Indicators are delivered to a message bus (Kafka), from where they are picked up by the Message Aggregator, which is part of the SIEA, and forwarded to the Risk Assessment Tool and the DLT (Data Ledger).

On the right side of the image is the SBOM tool, which works offline and therefore delivers the indicator it generates directly to the DLT. This indicator is determined by comparing the software bill of materials (SBOM) of the smart factory with a CVE database; and is therefore also an observable indicator.

All Indicators are forwarded to the Risk Assessment Tool and processed to determine the overall risk of the smart factory and to generate suggestions for mitigating this risk. The indicator that is then generated by the Risk Assessment Tool, sent to the DLT, and displayed on the dashboard to the administrator is therefore a "derived" indicator, as it is not based on raw data, but on analysis of Indicators from other tools.

The description of the Indicators used in use case 2 is as follows:

Observed Indicators

1. **Device Anomaly**

a. **Generic description:** Detected abnormal operational behaviour of a device.
b. **UC2 usage:** Detection of abnormal operational behaviour of a production robot, e.g., moves too fast, moves too slow, moves differently than initially programmed.

2. **Network Anomaly**
   a. **Generic description:** Detected abnormal behaviour of a network component.
   b. **UC2 usage:** Detection of abnormal behaviour of a network component, e.g., a device starts transmitting more data than usual.

3. **Software Component Vulnerability**
   a. **Generic description:** One or more of the software components used in a system have CVE vulnerabilities of any severity.
   b. **UC2 usage:** One or more of the software components used in UC2 have CVE vulnerabilities of any severity.

Derived Indicators

4. **SSM Risk Assessment Output**
   a. **Generic description:** Aggregated assessment (e.g., in terms of risk) of observed Indicators.
   b. **UC2 usage:** Evaluation of the risk level of the smart factory (UC2 setup) based on the observed Indicators. Provides a risk level and suggestions for mitigation.

As it is proposed above (in section 3.3.1.1.1) that there should be a one-to-one mapping of Indicators to contexts, that means the following contexts should be defined:

1. a context for device anomalies,
2. a context for network anomalies,
3. a context for the SBOM output,
4. a context for the risk assessments.

The first two reach the ledger via the SIEA pipeline, meaning they are aggregated which makes them derived Indicators.

6.1 Example: Production Device Anomaly

We will now discuss the example of anomalies detected by a device anomaly detection tool. While the tool reports to the SIEA and not directly to the DLT, it makes sense to even follow the same formats here which makes for an easier aggregation. Hence, a context for a device anomaly indicator coming from the detection tool would look very much like the generic indicator specified above, with the location field being used to pinpoint the location of an anomalous event (e.g., a sensor on the robotic arm producing out of bounds torque values).

```
{
    "data": {
        "context_data": {
            "type": {
                "name": "type",
```

```
            "description": "Type of indicator",
            "type": "string"
        },
        "severity": {
            "name": "severity",
            "description": "severity of the indicator",
            "type": "string"
        },
        "value": {
            "name": "value",
            "description": "Value of the indicator",
            "type": "number"
        },
        "timestamp": {
            "name": "timestamp",
            "description": "Timestamp of detection",
            "type": "string"
        },
        "subject": {
            "name": "subject",
            "description": "The subject the indicator relates to",
            "type": "string"
        },
        "source": {
            "name": "source",
            "description": "System or person reporting the indicator",
            "type": "string"
        },
        "supplementary-details": {
            "name": "supplementary-details",
            "description": "any further details",
            "type": "string"
        },
        "time-to-live": {
            "name": "time-to-live",
            "description": "Period for which the indicator is valid, optional
field",
            "type": "object",
            "period": {
                "name": "period",
                "description": "TTL in seconds",
                "type": "number"
            },
            "timestamp": {
                "name": "timestamp",
                "description": "TTL as timestamp",
                "type": "string"
            },
            "expiry": {
                "name": "expiry",
                "description": "TTL as time until it is not newest",
                "type": "string"
            }
        },
```

```
            "location": {
                "name": "location",
                "description": "Location of the anomalous measurement",
                "type": "number"
            }
        }
        "context_metadata": {
            "follow-up-actions": {
                "name": "follow-up-actions",
                "description": "Field for follow-up actions to be taken",
                "type": "string"
            }
        }
    }
    "metadata": {
        "description": "Context definition of a device anomaly indicator",
        "name": "Device Anomaly Indicator",
        "permissions": ["7juxGd7DMNBQokzxgNt5quko9QpdjX5bNCev4f2g1JZE"]
    }
}
```


The corresponding JSON content of the message sent from the tool to the SIEA:


```
{
    "context_id":
"de5f4932be9e85d5e66f1309950425c3f11b059a6804b86fafade26d9b5ff783"
    "data": {
        "type": "Device Anomaly",
        "severity": "RED",
        "value": 0.47,
        "timestamp": "2024-11-26 12:34:56",
        "subject": "Robot",
        "source": "NAD",
        "supplementary-details": "...some statistics...",
        "time-to-live": 3600,
        "location": "Torque sensor 1"
    }
    "metadata": {
        "follow-up-actions": "Check the operation of the robot in the Factory In a
Box"
    }
    "public_key": "7juxGd7DMNBQokzxgNt5quko9QpdjX5bNCev4f2g1JZE",
    "user_id": "535f03f0d7aa4903cc997cb421572460db369fbd91bc5bd15972d150f0f45355"
}
```


From the SIEA to the DLT, a derived indicator can include additional information from the aggregation of observed Indicators. The simplest way to represent the aggregation would be to add a field indicating the frequency or the message count. With a message count field added, the context then can be:


```
{
    "data": {
        "context_data": {
```

```
            "type": {
                "name": "type",
                "description": "Type of indicator",
                "type": "string"
            },
            "severity": {
                "name": "severity",
                "description": "severity of the indicator",
                "type": "string"
            },
            "value": {
                "name": "value",
                "description": "Value of the indicator",
                "type": "number"
            },
            "timestamp": {
                "name": "timestamp",
                "description": "Timestamp of detection",
                "type": "string"
            },
            "subject": {
                "name": "subject",
                "description": "The subject the indicator relates to",
                "type": "string"
            },
            "source": {
                "name": "source",
                "description": "System or person reporting the indicator",
                "type": "string"
            },
            "supplementary-details": {
                "name": "supplementary-details",
                "description": "any further details",
                "type": "string"
            },
            "time-to-live": {
                "name": "time-to-live",
                "description": "Period for which the indicator is valid, optional
field",
                "type": "object",
                "period": {
                    "name": "period",
                    "description": "TTL in seconds",
                    "type": "number"
                },
                "timestamp": {
                    "name": "timestamp",
                    "description": "TTL as timestamp",
                    "type": "string"
                },
                "expiry": {
                    "name": "expiry",
                    "description": "TTL as time until it is not newest",
                    "type": "string"
```

```
                }
            },
            "location": {
                "name": "location",
                "description": "Location of the anomalous measurement",
                "type": "number"
            },
            "message-count": {
                "name": "message-count",
                "description": "Amount of similar messages aggregated in this
message",
                "type": "string"
            }
        }
        "context_metadata": {
            "follow-up-actions": {
                "name": "follow-up-actions",
                "description": "Field for follow-up actions to be taken",
                "type": "string"
            }
        }
    }
    "metadata": {
        "description": "Context definition of an aggregated device anomaly
indicator",
        "name": "Aggregated Device Anomaly Indicator",
        "permissions": ["7juxGd7DMNBQokzxgNt5quko9QpdjX5bNCev4f2g1JZE"]
    }
}
```


Accordingly, the JSON content of a corresponding message sent from the SIEA to the DLT would be:


```
{
    "context_id":
"de5f4932be9e85d5e66f1309950425c3f11b059a6804b86fafade26d9b5ff783"
    "data": {
        "type": "Aggregated Device Anomaly Report",
        "severity": "RED",
        "value": 0.47,
        "timestamp": "2024-11-26 12:35:12",
        "subject": "Robot",
        "source": "NAD",
        "supplementary-details": "...some statistics...",
        "time-to-live": 3600,
        "location": "Torque sensor 1"
        "message-count": "{RED:5, ORANGE:10, OFF:100}"
    }
    "metadata": {
        "follow-up-actions": "Check the operation of the robot in the Factory In a
Box"
    }
    "public_key": "7juxGd7DMNBQokzxgNt5quko9QpdjX5bNCev4f2g1JZE",
    "user_id": "535f03f0d7aa4903cc997cb421572460db369fbd91bc5bd15972d150f0f45355"
```

}

Additional raw data to underpin the report can be included in the "supplementary-details" field where needed, so that they can be kept on record even if downstream tools such as the SSM do not process them. It is advisable however that this is kept at a reasonable amount to not exhaust resources with vast amounts of raw data.

6.2 Other Indicators in UC2

For network traffic anomalies, contexts and data transaction formats can be defined and used analogous to the example shown for the production device anomalies. The "location" parameter in this case can be used to identify an IP address or a subnetwork where the anomaly occurs. For the software component vulnerability (i.e., the output of the SBOM tool), there should be additional fields included in the context, specifically for a list of CVEs and possibly the SBOM itself.

Derived Indicators such as the output of the SSM or the trust analyser should refer back to the Indicators that contributed to the assessment. For this purpose, the DLT allows for an additional field "related_id" in the data transaction, referring to previous transaction ids that contain the originally reported Indicators. These and other Indicators will be described more fully in D1.4.

7 Summary

This deliverable has documented work undertaken so far in TELEMETRY regarding Indicators. We have defined the concept of Indicators, shown how Indicators fit within the (updated) TELEMETRY framework architecture that enables the different tools created by the project to interact and communicate, and this communication is facilitated by the Indicator concept. We have described the Infrastructural components and how they support Indicators, tools that generate different types of Indicators, and other tools that consume Indicators, analyse them and either generate new Indicators or output decision support information to the operator of the TELEMETRY framework to guide courses of action. We have also given a brief example in one of the TELMETRY use cases to illustrate Indicators' use within a real situation.

The key purpose of Indicators has been established, which is to provide information for decision support in order to help practitioners, but in addition, the notion of Indicators has proved to be of great use within the project as a communication mechanism, and there is further work to be done regarding Indicators, in all tools that either generate or consume Indicators, as well as the infrastructural components that support their reporting, aggregation, storage and display. Some of this work is in interpreting different Indicators to enable mappings between the values reported in an Indicator and the component consuming it, as illustrated by the SSM consuming results of the ACRAM access control risk tool, which itself consumes Indicators from scanning tools. This work will continue and develop as part of WP2 as it contributes to future tasks such as integration of the TELEMETRY framework and user trials within the project's use cases and will be reported in subsequent deliverables.

8 References

Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." In Proceedings of the thirteenth EuroSys conference, pp. 1-15. 2018.

Anna V. Bakurova, Oleh V. Zaritskyi, Anatoliy A. Gritskevich, Pavlo V. Hrynchenko, Elina V. Tereschenko, Dmytro V. Shyrokorad, "ASSESSMENT OF THE VULNERABILITY OF THE INFORMATION SYSTEM BASED ON FUZZY MATHEMATICS", Combinatorial configurations and their applications: Materials of the XXVI International Scientific and Practical Seminar, Kropyvnytskyi–Zaporizhia–Kyiv, June 13-15, 2024, pp. 20-25.

Valentina Casola, Alessandra De Benedictis, Antonio Riccio, Diego Rivera, Wissam Mallouli, Edgardo Montes de Oca, A security monitoring system for internet of things, Internet of Things, Volume 7, 2019, 100080, ISSN 2542-6605, https://doi.org/10.1016/j.iot.2019.100080.

M. Fuentes-García, J. Camacho and G. Maciá-Fernández, "Present and Future of Network Security Monitoring," in IEEE Access, vol. 9, pp. 112744-112760, 2021, doi: 10.1109/ACCESS.2021.3067106.

S. He, M. Ghanem, L. Guo and Y. Guo, "Cloud Resource Monitoring for Intrusion Detection," 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, Bristol, UK, 2013, pp. 281-284, doi: 10.1109/CloudCom.2013.148.

L. A. Jaatun, S. Marie Sørlien, R. Borgaonkar, S. Taylor and M. G. Jaatun, "Software Bill of Materials in Critical Infrastructure," 2023 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Naples, Italy, 2023, pp. 319-324, doi: 10.1109/CloudCom59040.2023.00059.

Lee, W., & Stolfo, S. J. (1998). Data mining approaches for intrusion detection. In Proceedings of the 7th USENIX Security Symposium (Vol. 7, pp. 79-93).

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 4768–4777. https://dl.acm.org/doi/10.5555/3295222.3295230

Vasyl Lytvyn, Anna Bakurova, Oleh Zaritskyi, Anatoliy Gritskevich, Pavlo Hrynchenko, Elina Tereschenko and Dmytro Shyrokorad. Fuzzy logic-based methodology for building access control systems based on fuzzy logic. Proceedings of MoDaST-2024: 6th International Workshop on Modern Data Science Technologies, May, 31 - June, 1, 2024, Lviv-Shatsk, Ukraine. – ceur-ws.org/Vol-3723/paper7.pdf - https://ceur-ws.org/Vol-3723/

Trent McConaghy, Rodolphe Marques, Andreas Müller, Dimitri De Jonghe, Troy McConaghy, Greg McMullen, Ryan Henderson, Sylvain Bellemare, and Alberto Granzotto. "BigchainDB: a scalable blockchain database." White paper, BigChainDB (2016): 53-72.

A. McGibney, T. Ranathunga, and R. Pospisil, 'SmartQC: An Extensible DLT-Based Framework for Trusted Data Workflows in Smart Manufacturing', Feb. 27, 2024, arXiv: arXiv:2402.17868. [Online]. Available: http://arxiv.org/abs/2402.17868

G. McGraw, "Software security," in IEEE Security & Privacy, vol. 2, no. 2, pp. 80-83, March-April 2004, doi: 10.1109/MSECP.2004.1281254.

Miller, Barton P., Lars Fredriksen, and Bryan So. "An empirical study of the reliability of UNIX utilities." Communications of the ACM 33.12 (1990): 32-44.

David Ornstein and Tony Rice. Building the next generation of the Microsoft Security Development Lifecycle (SDL). March 7, 2024. https://www.microsoft.com/en-us/security/blog/2024/03/07/evolving-microsoft-security-development-lifecycle-sdl-how-continuous-sdl-can-help-you-build-more-secure-software/. White paper ato https://delivery-microsoft.sitecorecontenthub.cloud/api/public/content/142b1591af164f639663cb19eec03702?v=83974498

S. C. Phillips, S. Taylor, M. Boniface, S. Modafferi and M. Surridge. (2024). "Automated Knowledge-Based Cybersecurity Risk Assessment of Cyber-Physical Systems," in IEEE Access, vol. 12, pp. 82482-82505, 2024, doi: 10.1109/ACCESS.2024.3404264.

J. Shao, H. Wei, Q. Wang and H. Mei, "A Runtime Model Based Monitoring Approach for Cloud," 2010 IEEE 3rd International Conference on Cloud Computing, Miami, FL, USA, 2010, pp. 313-320, doi: 10.1109/CLOUD.2010.31.

R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2010, pp. 305-316, doi: 10.1109/SP.2010.25.

Stibor, T., Timm, I. J., & Han, K. (2005). An Evolution Strategy with Incremental Stochastic Covariance Matrix Adaptation for Misuse Detection. In "Proceedings of the 2005 IEEE Congress on Evolutionary Computation" (Vol. 3, pp. 2744-2751). IEEE.

Taylor, S.; Jaatun, M.; Mc Gibney, A.; Seidl, R.; Hrynchenko, P.; Prosvirin, D. and Mancilla, R. (2024). A Framework Addressing Challenges in Cybersecurity Testing of IoT Ecosystems and Components. In Proceedings of the 9th International Conference on Internet of Things, Big Data and Security, ISBN 978-989-758-699-6, ISSN 2184-4976, pages 226-234.

Vinayakumar, R., Soman, K. P., & Prabaharan Poornachandran, N. (2019). Evaluating shallow and deep networks for network intrusion detection systems in cybersecurity. "Future Generation Computer Systems, 107", 13-26.

S. Wang, J. F. Balarezo, S. Kandeepan, A. Al-Hourani, K. G. Chavez and B. Rubinstein, "Machine Learning in Network Anomaly Detection: A Survey," in IEEE Access, vol. 9, pp. 152379-152396, 2021, doi: 10.1109/ACCESS.2021.3126834.