Grant Agreement No.: 952697
Call: H2020-SU-ICT-2018-2020
Topic: SU-ICT-02-2020
Type of action: RIA

D1.3 OPERATIONAL SOS PROCESS MODELS & SPECIFICATION OF PROPERTIES

Revision: v.1.0

Work package	WP 1
Task	Task 1.3
Due date	31/08/2021
Deliverable lead	TUDA
Version	1.0
Authors	Jingru Wang (TUDA), Ferdinand Brasser (TUDA), Richard Mitev (TUDA)
Reviewers	Thanassis Giannetsos, Dimitris Papamartzivanos (UBITECH) Ioannis Avramidis (INTRASOFT)
Abstract	Deliverable D1.3 focuses on the definition of the system properties, per use case, that need to be considered for (run-time) attestation so as to produce the necessary security claims towards achieving the required level of assurance and trustworthiness. Such validation properties can actually capture the entire application stack of a device (from configuration properties to execution behavioural properties) depending on the types of attacks considered, the safety-critical nature of the services and any timing constraints. Towards this direction, ASSURED adopts both *activity*- and *artefact*-centric modelling approaches that enables the modelling of all software and hardware assets and their detailed system description so as to extract all necessary validation properties that set the scene for the evaluation of the novel set of attestation enablers to be designed in the context of WP3.
Keywords	Information & Process Modelling, Remote Attestation & Verification, Threat Modelling, System Validation Properties, Trust Modelling

Funded by the Horizon 2020 Framework Programme of the European Union

Document Revision History

Version	Date	Description of change	List of contributors
V0.1	01.05.2021	ToC was created and circulated where the focus was on Section 3.3 and Chapter 4 for starting the discussions with the use case partners on the considered threat model and the list of all possible validation properties	Jingru Wang (TUDA)
V0.2	19.05.2021	State-of-the-art Analysis on the type of modelling approaches that can be considered in ASSURED	Dimitris Papamartzivanos, Dimitris Karras, Sofianna Menesidou (UBITECH), Edlira Dushku (DTU), Ioannis Avramidis (INTRA), Ilias Aliferis (UNISYSTEMS)
V0.3	04.06.2021	First draft of the definition of all possible validation properties that can be considered in the context of remote attestation (Chapter 4)	Jingru Wang, Ferdinand Brasser (TUDA), Thanassis Giannetsos, Dimitris Papamartzivanos (UBITECH), Edlira Dushku (DTU), Liqun Chen (SURREY)
V0.4	25.06.2021	Definition of the threat model that can be considered in the context of an IoT-based "Systems-of-Systems" focusing on both host-based and network-based attacks (Section 3.3)	Jingru Wang, Ferdinand Brasser (TUDA), Thanassis Giannetsos, Dimitris Papamartzivanos (UBITECH), Edlira Dushku (DTU), Liqun Chen, Nada El Kassem (SURREY), Kaitai Liang (TUDE), Thanassis Giannetsos (UBITECH), Riccardo Orizio (UTRCI)
V0.5	09.07.2021	First draft of the mapping of the threat modelling in the context of the use cases (Chapter 5)	Liqun Chen, Nada El Kassem (SURREY), Jingru Wang (TUDA), Kaitai Liang (TUDE), Ilias Aliferis (UNISYSTEMS), Karthik Shenoy, Shantanoo Desai (BIBA), Stelios Basagiannis, Riccardo Orizio (UTRCI), Nikos Drosos (SPH), Dimitra Tsakanika, Ilia Christantoni (DAEM)
V0.6	23.07.2021	Update and finalization of the defined threat model highlighting also the threat taxonomy provided by OWASP (Section 3.3) Finalization of the attack trees per use case (Chapter 5)	Jingru Wang, Ferdinand Brasser (TUDA), Thanassis Giannetsos, Dimitris Papamartzivanos (UBITECH), Edlira Dushku (DTU), Liqun Chen, Nada El Kassem (SURREY), Kaitai Liang (TUDE), Thanassis Giannetsos (UBITECH), Karthik Shenoy, Shantanoo Desai (BIBA), Stelios Basagiannis, Riccardo Orizio (UTRCI), Nikos Drosos (SPH), Dimitra Tsakanika, Ilia Christantoni (DAEM)
V0.7	18.08.2021	Identification of the type of software and hardware assets to be considered per use case and mapping of the specific validation properties that can protect against all identified threats (Chapter 5)	Jingru Wang, Ferdinand Brasser (TUDA), Edlira Dushku (DTU), Kaitai Liang (TUDE,) Karthik Shenoy, Shantanoo Desai (BIBA), Stelios Basagiannis, Riccardo Orizio (UTRCI), Nikos Drosos (SPH), Dimitra Tsakanika, Ilia Christantoni (DAEM)
V0.8	27.08.2021	Final description of the trust assessment and modelling approach followed in the context of ASSURED (Chapter 2)	Dimitris Papamartzivanos, Dimitris Karras, Sofianna Menesidou (UBITECH), Edlira Dushku (DTU), Ioannis Avramidis (INTRA), Ilias Aliferis (UNISYSTEMS), Edlira Dushku (DTU)
V0.9	28.09.2021	Review the document	Thanassis Giannetsos, Dimitris Papamartzivanos (UBITECH), Ioannis Avramidis (INTRASOFT)
V0	22.02.2022	Polishing of the language of the deliverable and update of the validation properties. Submission of the deliverable.	Thanassis Giannetsos, Dimitris Karras (UBITECH), Jean-Baptiste Milon (MARTEL)

Funded by the Horizon 2020
Framework Programme of the European Union

Editors

Jingru Wang (TUDA), Ferdinand Brasser (TUDA), Richard Mitev (TUDA), Philip Rieger (TUDA)

Contributors (ordered according to beneficiary numbers)

Edlira Dushku, Nicola Dragoni (DTU)

Jingru Wang, Ferdinand Brasser, Richard Mitev, Philip Rieger (TUDA)

Liqun Chen, Nada El Kassem (SURREY)

Dimitrs Papamartzivanos, Thanassis Giannetsos, Dimitris Karras, Sofianna Menesidou (UBITECH)

Riccardo Orizio, Stelios Basayiannis (UTRCI)

Nikos Drossos (SPH)

Dimitra Tsakanika, Ilia Christantoni (DAEM)

Ilias Aliferis (UNIS)

Ioannis Avramidis (INTRA)

DISCLAIMER

The information, documentation and figures available in this deliverable are written by the "Future Proofing of ICT Trust Chains: Sustainable Operational Assurance and Verification Remote Guards for Systems-of-Systems Security and Privacy" (ASSURED) project's consortium under EC grant agreement 952697 and do not necessarily reflect the views of the European Commission.

The European Commission is not liable for any use that may be made of the information contained herein.

COPYRIGHT NOTICE

© 2020 - 2023 ASSURED Consortium

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:	**R**	
Dissemination Level		
PU	Public, fully open, e.g. web	✓
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to ASSURED project and Commission Services	

* R: Document, report (excluding the periodic and final reports)
DEM: Demonstrator, pilot, prototype, plan designs
DEC: Websites, patents filing, press & media actions, videos, etc.
OTHER: Software, technical diagram, etc.

Funded by the Horizon 2020
Framework Programme of the European Union

EXECUTIVE SUMMARY

Deliverable D1.3 focuses on the definition of the system properties, per use case, that need to be considered for (run-time) attestation so as to produce the necessary **security claims towards achieving the required level of assurance and trustworthiness**. Such validation properties can actually capture the entire application stack of a device (from **configuration properties** to **execution behavioural properties**) depending on the **types of attacks considered, the safety-critical nature of the services and any timing constraints**.

Recall that ASSURED core innovation is to be able to **enhance the security posture of a "Systems-of-Systems" environment** – comprising multiple, heterogeneous embedded devices – by **assessing dynamic trust relationships and defining a trust model based on which the necessary security claims can be produced for establishing trust through the entire service-graph chain**, for cooperatively executing safety-critical functions. In this context, ASSURED builds upon and expands the **Zero Trust concept** to tackle the issue of how to **bootstrap vertical trust from the application, the execution environment and device hardware** from one single component and continuing as such systems get connected to ever larger entities. This includes the design and employment of a new breed of attestation enablers for measuring and determining the integrity and origin of the system and all its internal (software) components. Trusted Execution Environments (TEEs), as sw- or hw-based security elements, will be essential to establish a verifiable chain of trust throughout the entire application stack of the host device, as well as protecting data in transit, at rest and in use.

However, trying to attest the **entire codebase raises concerns on the efficiency, scalability and robustness of these techniques** that question whether they can be applied in the real-world resource-constrained edge devices. Such limitations mainly stem from the fact that these types of operational assurance methods try to verify the integrity, during run-time, of the entire (untrusted) code base of commodity platforms and operating systems. Considering that competitive IIoT application markets will always produce innovative and large systems comprising diverse-origin software-based components, with uncertain security properties, the best one can hope for is that a **sub-set of such loaded software functions can be efficiently protected (in near real-time) against sophisticated run-time exploitation attacks**.

Therefore, it is of paramount importance to be able to identify only those core properties that are safety-critical to the operation of a device that we can continuously attest without affecting both the resources and behaviour of the target device. **This refers to the low-level properties that need to be attested for specific hardware, and can be enhanced by employing the modular protection profiles defined within ASSURED as part of the service graph representing the CPSoS**, and can be used to achieve the required **trustworthiness level of the entire system.**

To this end, D1.3 evaluates existing models, such as **Business Process Modelling Notation (BPMN)** and **Case Management Modelling Notation (CMMN)**, been proposed for all those safety-critical components that need to coexist and be securely executed in a platform with shared hardware and software resources (such as caches and central memory bus), and their connection to the overall system behaviour. Based on this, it develops a more suitable model for ASSURED that can represent the business and technical processes relying on **Cyber-Physical System (CPS)** environments. A model tailored for **Cyber-Physical Systems of Systems (CPSoS)** enables us to capture only those processes (such as algorithms, control, and device operational logic elements) with the highest criticality level, that need to be continuously verified in real-time. This will also act as the basis for the efficient, effective, and scalable attestation enablers.

By coupling the Zero Trust security principle with the need of "Never Trust, Always Verify" specific system properties, ASSURED bootstraps vertical trust for all users, devices and systems in a "Systems-of-Systems" environment by enabling continuous authorization and authentication prior to be granted access to data or resources. Through TPM-enabled security

Funded by the Horizon 2020 Framework Programme of the European Union

claims, assurances and verifiable chain of trust, ASSURED reaches its full potential: **not only does it mitigates risks stemming from the Zero Trust SoS environment but also ensures resilience.**

Funded by the Horizon 2020
Framework Programme of the European Union

TABLE OF CONTENTS

Funded by the Horizon 2020
Framework Programme of the European Union

Funded by the Horizon 2020 Framework Programme of the European Union

LIST OF FIGURES

Funded by the Horizon 2020 Framework Programme of the European Union

LIST OF TABLES

Funded by the Horizon 2020 Framework Programme of the European Union

1 INTRODUCTION

1.1 DYNAMIC TRUST ASSESSMENT IN SOS ENVIRONMENTS

Seeking to design successful supply chain services comprising millions of autonomous Cyber-Physical Systems (CPSoS), one has to cater to the **security, trust and privacy requirements** of all involved actors, ranging from the heterogeneous edge devices to the backend cloud systems. A key challenge is to establish and manage trust between entities, starting from **bi-lateral interactions between two single system components and continuing as such systems get connected to ever larger entities**. But the challenge ahead is how to *make sound statements on the security properties of single systems and transfer this to statements on the security properties of such hierarchical compositions of systems ("Systems-of-Systems")?*

Consider, for instance, the following security requirements in the context of the use cases considered in ASSURED:

- In the domain of the envisioned **"*Smart Aerospace*"** use case, where **Electronics Control Units (ECUs)** (e.g., Flight Management System, Environmental Control System, Cockpit Flight Instrument) in an aircraft get connected to subsystems, and subsystems are connected by gateways (Secure Server Router (SSR)) to form the complete in-aircraft electronic system, and as such an aircraft gets connected to the Ground Station Server (GSS) or other aircrafts.
- In the ***"Smart Manufacturing"*** use case, an **IoT Gateway** is used to connect the robots operating within a manufacturing infrastructure, so that the permitted movement of each robot can be determined centrally, taking into consideration the position of other robots or human workers.
- In the ***"Smart Cities"*** use case, IP surveillance cameras and smoke detectors, as well as a series of PLC controllers for controlling the surveillance systems, constitute the Serafio complex, which is controlled through an **Information and Communication Technology (ICT)** system and an operations center, which consists of a cloud-based infrastructure with various the networking components and computational resources.
- In the ***"Smart Satellites"*** use case, a **Ground Station (GS)** serves as a central unit, which is able to monitor, maintain and control various CubeSats, which are miniaturized satellites that perform specific missions in space. It is also possible for multiple CubeSats to collaborate in order to perform complex services and functions.

This kind of security requirements extends also to other industries and domains beyond the ones that are envisioned in ASSURED. For example, in the automotive industry, ECUs get connected to subsystems, and subsystems are connected by gateways, in order to form the complete in-vehicle electronic system of a vehicle. As such, a vehicle gets connected to an OEMs backend system in connected cars or linked to other vehicles in V2V communication, creating an environment where the **security level of a vehicle cannot be viewed in isolation**, but in tandem with the entire network of connected cars.

From all the above, it follows that *the notion of securing a single, isolated device is outdated, since it cannot guarantee the trustworthiness level required by organizations with a complex CPSoS*. However, given the increasing complexity of these systems, statements about **integrity of the overall system** or the **confidentiality of specific data items** are harder and harder to make. Therefore, the question that is raised and that we attempt to answer in this deliverable, and with the ASSURED attestation services in general, is the following: *How can we make sound statements on the specific security properties of single systems (that are*

Funded by the Horizon 2020 Framework Programme of the European Union

sufficient to depict the required level of trustworthiness) and transfer this to statements on the security properties of such hierarchical compositions of systems?

In order to reply to this question, we first need to refer to the type of attacks that pose a threat to “Systems-of-Systems” environments with complex CPSoSs. In ASSURED, we aim to address the ever-changing landscape of attackers and types of threats, by mitigating the most prominent threats to a CPSoS that can be addressed by the provided trust assurance services. For example, the **Open Web Application Security Project (OWASP)** [1] maintains a yearly updated ranking list of **Common Vulnerabilities and Exposures (CVEs),** which is formulated by taking into consideration factors such as Exploitability, Detectability, Likelihood and Technical Impact, and by performing community surveys on application security and development experts, as well as using practical data from various organizations regarding the severity and frequency of occurrence of each threat and vulnerability.

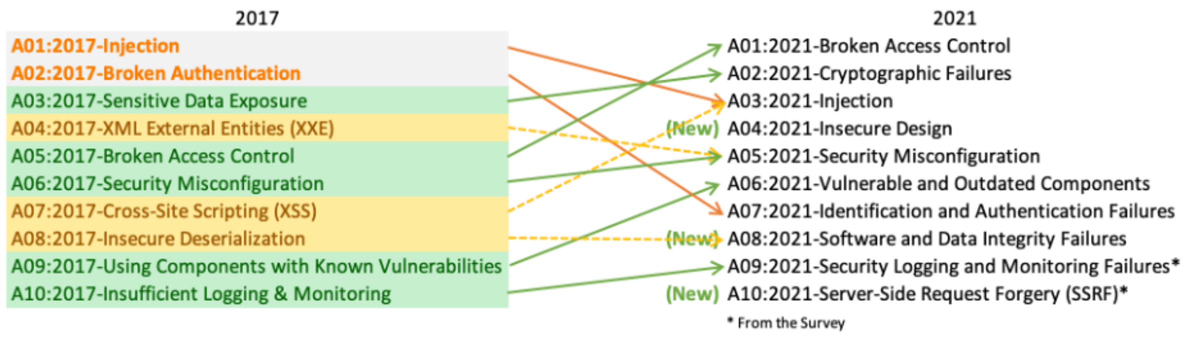
2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

FIGURE 1: OWASP CVE RANKING DIFFERENCES BETWEEN 2017 AND 2021

Figure 1 depicts the latest published OWASP ranking of CVEs for the year 2021, as well as a comparison with the corresponding list from 2017. By observing the evolution of the attack landscape, it becomes apparent that the concept of fulfilling security and privacy requirements for isolated assets or devices is becoming an increasingly outdated notion of security, and the construction of a security- and privacy-aware **Service Graph Chain (SGC)**, that takes into consideration the positioning of each asset in the hierarchical composition of a CPSoS is the way to move forward. In other words, the importance of classical network attacks is in a downward trend, while software-related vulnerabilities are becoming more prevalent and a lucrative target to be exploited by sophisticated adversaries. Specifically, some of the relevant attacks whose prevalence has increased significantly throughout the past few years and are relevant to the considered use cases are the following:

- **Broken Access Control**: Refers to unauthorized information disclosure, modification, or destruction of data or performing a business function outside an authorized user’s limits, due to an attacker tampering with the access permissions. Consider the Human Robot Interaction (CRI), where this issue is of critical importance, since unauthorized access to a robot may cause severe injury to a human worker. The attestation services provided by ASSURED together with the advanced identity management schemes (leveraging the concept of Verifiable Credentials) will address this, by verifying that the authorization status of the robots is in a trusted state.
- **Security Misconfiguration**: A malicious modification in the configuration of a device, enabling access by a malicious party. This is relevant to all use cases, since all devices need to be protected from threats resulting from an untrusted configuration. This is mitigated by employing the newly developed Configuration Integrity Verification (CIV).
- **Vulnerable and Outdated Components**: Outdated components are notoriously susceptible to security threats and attacks, since they may be loaded with firmware or application software that has not been patched in order to be protected from newly identified threats. This falls under the category of software vulnerabilities that can be

Funded by the Horizon 2020 Framework Programme of the European Union

mitigated by attestation services, verifying that the components possess the latest software versions.
- **Insecure Design**: Refers to architectural flaws related to design methodologies that cause vulnerabilities. Therefore, this category does not refer to a specific kind of vulnerability, but design and coding practices that make devices and systems more susceptible to various kinds of attacks. These can be detected with the help of attestation, so that they can be patched in future versions of the software.
- **Cryptographic Failures**: Encompasses failures and attacks related to cryptographic functions and cryptographic primitives. This is of particular importance in the smart satellite use case, which employs a variety of cryptographic schemes and protocols for communication of the base station with the deployed satellites and can be mitigated by the ASSURED traffic verification and attestation services.

From all the above, it follows that software-related threats and vulnerabilities pose a serious threat to the trustworthiness level of an ecosystem and have become increasingly prevalent over the past years. As it will become apparent throughout this deliverable, the integration of Trusted Computing technologies with novel attestation schemes provides an effective solution towards the mitigation of this kind of attacks, since such advanced assurance mechanisms can detect anomalies in the **device configuration** and **execution of processes** in each asset of a CPSoS (software or hardware asset). To this end, depending on the type of devices and services, we need to be able to map the properties that need to be attested with the corresponding devices. Note that this mapping can potentially be used as a guidance for the definition of **protection profiles**, or the enhancement of already existing ones; as have been proposed by the standards (ISO/IEC JTC1/WG13 and ISO/IEC JTC1/SC27). **This refers to the low-level properties that need to be attested for specific hardware, and can be enhanced by employing the modular protection profiles defined within ASSURED as part of the service graph representing the CPSoS**, and can be used to achieve the required **trustworthiness level of the entire system**. We will further expand on this notion in D2.2 [2].

The considered attack landscape, in tandem with the type of CPSoS considered within ASSURED, will also provide justification for the type of attestation schemes implemented as a part of the trust assurance services. Specifically, the schemes that will be employed and will be defined and expanded upon D3.2 [3] are the following:

- **Configuration Integrity Verification (CIV)**, which ensures the trustworthiness of the configuration of a device.
- **Control Flow Attestation (CFA)**, which ensures that the flow of actions performed by the software installed in a device is trustworthy.
- **Swarm Attestation**, which is used in cases where simultaneous attestation of multiple devices working in tandem is required.
- **Direct Anonymous Attestation (DAA)**, which provides platform authentication capabilities in a privacy-preserving manner.
- **Jury-based Attestation**, which ensures the correctness of the entire attestation process execution so as to capture any misbehaving entity; in the case where the Verifier essentially “lies” about the system measurements provided by another Prover device (Section 3.1).

1.2 SCOPE AND PURPOSE

The main goal of Deliverable 1.3 is twofold. First, to develop the **modelling notation suitable for ASSURED that is used to capture those safety-critical components and services**, comprising the Systems of Systems (SoS), and the interdependencies between them and the underlying platform, which will aid the work of identifying critical properties needed to be

Funded by the Horizon 2020 Framework Programme of the European Union

attested continuously. This is the second goal of this deliverable, namely, to determine the **validation properties** that are crucial for the operational correctness and security of complex SoS, not only platform configuration properties but also execution properties. Thanks to the modelling of SoS we can focus on those components and services with the highest criticality level. At the end, the output is then applied to four use cases envisioned in ASSURED, mapping their adversary models with corresponding attestation properties. In the following work packages, specifically WP3, we will develop attestation and validation schemes to enforce those identified properties.

To this end, this deliverable evaluates existing models, such as **Business Process Modelling Notation (BPMN)** and **Case Management Modelling Notation (CMMN)** and develops a model suitable for ASSURED that can represent the business and technical processes relying on **Cyber-Physical System (CPS)** environments. A model tailored for **Cyber-Physical Systems of Systems (CPSoS)** enables us to capture only those processes (such as algorithms, control, and device operational logic elements) with the highest criticality level, that need to be continuously verified in real-time. This is also the basis for the efficient, effective, and scalable attestation enablers.

The highlight of D1.3 is to **identify a set of properties that model the correct configurations and execution behaviours of critical components and therefore need to be continuously verified**. One of the core offering provided by ASSURED framework is the property-based attestation that corroborates the fundamental security and non-security properties of entities and extends this to a large SoS. These properties serving as a measure of trust should reflect even minor status changes associated with attacks and abnormal operations. On this basis, the properties form the basis for an evaluation of broader system-wide properties and assure the cyber resilience in SoS-enabled supple chains. Thus, identification of suitable attestation properties is a key for ASSURED project.

An **explicit specification of system model and adversary model** is a prerequisite for capturing suitable attestation properties, which is presented in D1.3 including the system trust anchor and the capabilities of adversaries at different levels. Depending on the system and application domain, the required security and non-security properties vary considerably. Thus, besides generic properties, we take full account of use-case specific properties for the envisioned four scenarios in ASSURED. Finally, D1.3 demonstrates the applicability of the model and attestation properties defined in this document towards the four use cases of ASSURED. This entails the analysis of each use case's threat model and security requirements, and mapping between threats and associated attestation properties and schemes. ASSURED framework will implement multiple attestation schemes corroborating various properties to cover threats within all phases of a device's execution.

1.3 RELATION TO OTHER WPS AND DELIVERABLES

As a subsequent document, Deliverable D1.3 directly gets input from D1.1, which defines the use case scenarios and functional specifications for the entire ASSURED framework, and D1.2, which specifies the ASSURED reference architecture including the components and the communicating interfaces. **This essentially sets the scene for the core building block of attestation protocols that serves as the central mechanism towards enhancing the operational assurance of the target SoS**. On the other hand, this vocabulary of system properties to be attested will also be embodied in the definition of the attestation-related data sharing behaviours that will be fleshed out in D1.4. More specifically, depending on the safety-critical nature of the service (and its subsequent properties to be attested) there might be different requirements when it comes to confidentiality, privacy and sharing attributes. This might relate not only **to privacy when it comes to the Device ID** but also to **attestation**

Funded by the Horizon 2020 Framework Programme of the European Union

evidence privacy in the sense that a Verifier should be able to attest to the correct state of a Prover device but without the need to know all the internal details of the Prover's state.

With the identification of the system validation properties to be considered for attestation in each one of the envisioned use cases, D1.3 also serves as a reference point for all the activities in WP3 revolving around the **design of the new breed of attestation schemes** as will be documented in D3.2, D3.3 and D3.6 and D3.7. Furthermore, these properties will set the scene for the experimentation and evaluation of all attestation enablers, in the use cases, to be documented in D6.2.

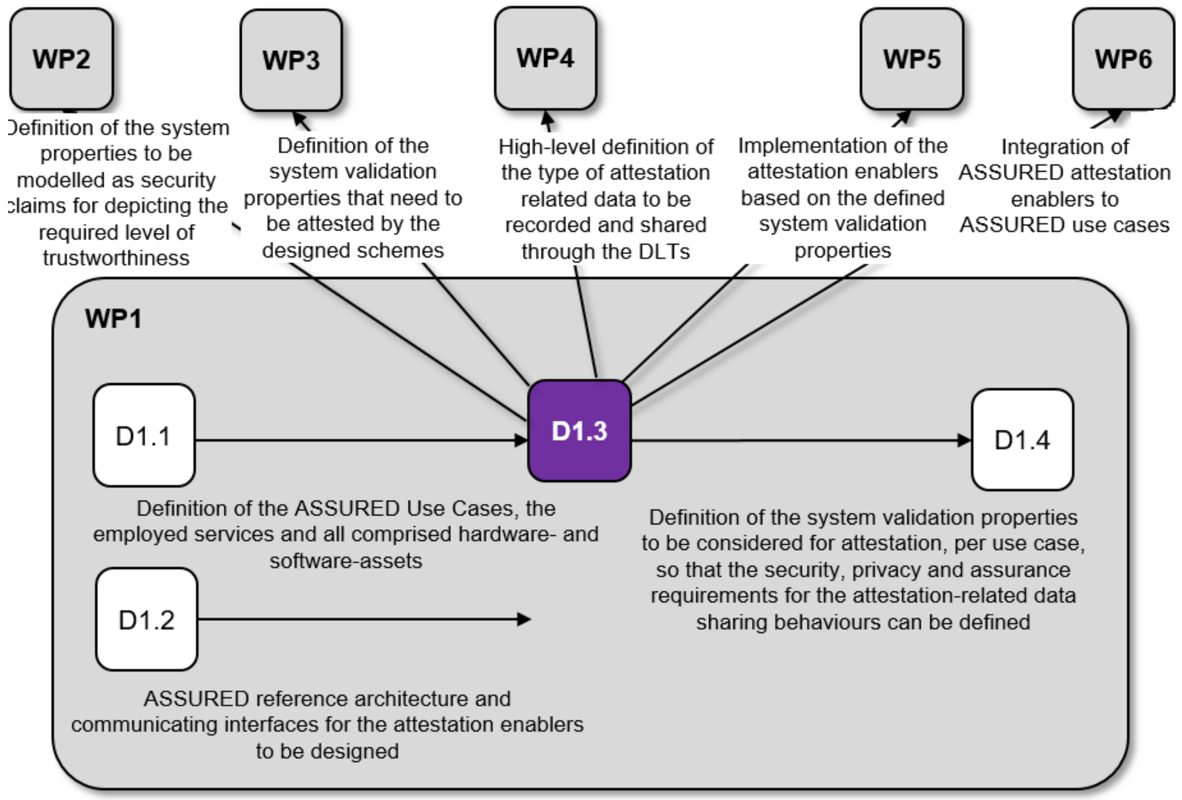
WP2
WP3
WP4
WP5
WP6
Definition of the system properties to be modelled as security claims for depicting the required level of trustworthiness
Definition of the system validation properties that need to be attested by the designed schemes
High-level definition of the type of attestation related data to be recorded and shared through the DLTs
Implementation of the attestation enablers based on the defined system validation properties
Integration of ASSURED attestation enablers to ASSURED use cases
WP1
D1.1
D1.3
D1.4
Definition of the ASSURED Use Cases, the employed services and all comprised hardware- and software-assets
D1.2
Definition of the system validation properties to be considered for attestation, per use case, so that the security, privacy and assurance requirements for the attestation-related data sharing behaviours can be defined
ASSURED reference architecture and communicating interfaces for the attestation enablers to be designed

FIGURE 2: RELATION OF D1.3 TO OTHER WPS AND DELIVERABLES

Finally, D1.3 acts as a starting point of technical reference for the later WP4 technical activities revolving around the design of the **ASSURED Blockchain infrastructure for the secure and auditable recording and sharing of all attestation-related data**. Since ASSURED will provide advanced crypto primitives for decentralized Attribute-based Encryption and Attribute-based Access Control for this type of data, the structure and format of all validation properties and system traces to be considered is of paramount importance so as to make sure of the modelling of an **international attestation data space model**.

1.4 DELIVERABLE STRUCTURE

This deliverable is structured as follows. In Chapter 2, we present a detailed explanation of how we build the model to be employed by ASSURED. We review the state-of-the-art trust management techniques in supply chain ecosystems, as well as the SoS modelling notations. We will explicitly define the criterion that a model needs to fulfil to be used in ASSURED. In the end of Chapter 2, we introduce the ASSURED model combining both artefact-centric modelling and data-driven activities that dictate the status of the edge devices.

Funded by the Horizon 2020 Framework Programme of the European Union

In Chapter 3, we elaborate on several key definitions that constitute the foundation of the ASSURED attestation enablers. Chapter 3 starts with the definition of attestation and verification. Next, the system model and the “as is” system properties are defined, which specify the safety, security, and privacy requirements that need to be fulfilled, even under attack. To understand the adversary capabilities and goals, we also provide a comprehensive adversary model for SoS. Given protection goals and potential attacks, we summarize a set of configuration and execution properties to be attested and verified to enhance the operational assurance and program correctness. In Section 4, we elaborate on the static and dynamic validation properties, which are used by the attestation mechanisms in order to determine that a system-of-systems is in a trustworthy state. Based on the model introduced in Chapter 3 and the attestation properties elaborated in Chapter 4, we demonstrate a detailed analysis of each use case scenario in ASSURED in Chapter 5. We apply the model to use cases to identify those safety-critical components, define the adversary model and select matched attestation properties. Finally, Chapter 6 summarizes and concludes the deliverable.

Funded by the Horizon 2020
Framework Programme of the European Union

2 APPROACH AND MODEL

We previously presented a list of the most prominent threats and vulnerabilities in the current cybersecurity environment, and we highlighted the threats whose prominence is in an upward trend, and the ways that these threats can be addressed by the attestation services provided in ASSURED. Taking this into consideration, we next aim to identify and define the **properties that should be attested**, as well as the **cyber-security and trust model**, so that we can consider that the target CPSoS can is in an **acceptable and trustworthy state**.

In general, this approach can be either **activity-driven** or **asset-driven**. In an activity-driven approach, cybersecurity assurance is evaluated based on the actions taken by the assets of the CPSoS, while in the asset-driven approach, the focus is placed on the assets or devices themselves, their configuration properties, and their current state. In ASSURED, we follow an asset-driven approach, which is also referred to as **artefact-centric**. Throughout this chapter we will expand upon this notion and provide justification for its selection.

To this end, we will define the asset modelling approach that will be employed, which will provide the basis for the extraction of the properties to be attested. We consider that each hardware asset can contain and run multiple software assets, and based on the asset cartography, we should be able to identify **what processes are executed** and **which attacks are more impactful**, so that we can subsequently define acceptable states. We will also define the notation that will be used in order to express these properties, so that the operational and security status of the system can be accurately represented with the required level of granularity. Also, note that artefact-centric modelling will provide the basis for the definition of the protection goals and attack settings per use case, which will be provided in Chapter **Error! Reference source not found.**.

When referring to “Systems-of-Systems” that are used in various domains in the industry, such as the use cases considered in ASSURED, this constitutes the secure continuum of the edge and the cloud working in tandem in a trustworthy manner. This is referred to as the **edge-cloud continuum digital trust**. In this context, ASSURED is conceptually positioned to be within the Industrial Reference Architecture (IRA) [4], which is presented in Figure 3, along with the positioning of ASSURED within the IRA. The IRA consists of the following layers:

- The **Field**, which contains the devices and assets of the system, as well as the various stakeholders.
- The **Edge Control**, which contains the ASSURED security mechanisms that are used to securely control and manage the devices of the system.
- The **Ledger**, which constitutes the secure medium that is used in order to exchange data and perform management operations.
- The **Cloud**, which is the operational centre, i.e., the backend of the system, where cloud and supply chain applications are executed.

Among the aforementioned categories, the focus in ASSURED is to secure the **Field** and the **Edge** (physical devices, sensors, and gateways), by identifying the appropriate security mechanisms that are contained in the **Ledger** and are managed by the **Cloud**. In this context, we aim to identify the processes that should be secured. Note that the types of data to be secured are further addressed in D1.4 [5].

Funded by the Horizon 2020 Framework Programme of the European Union

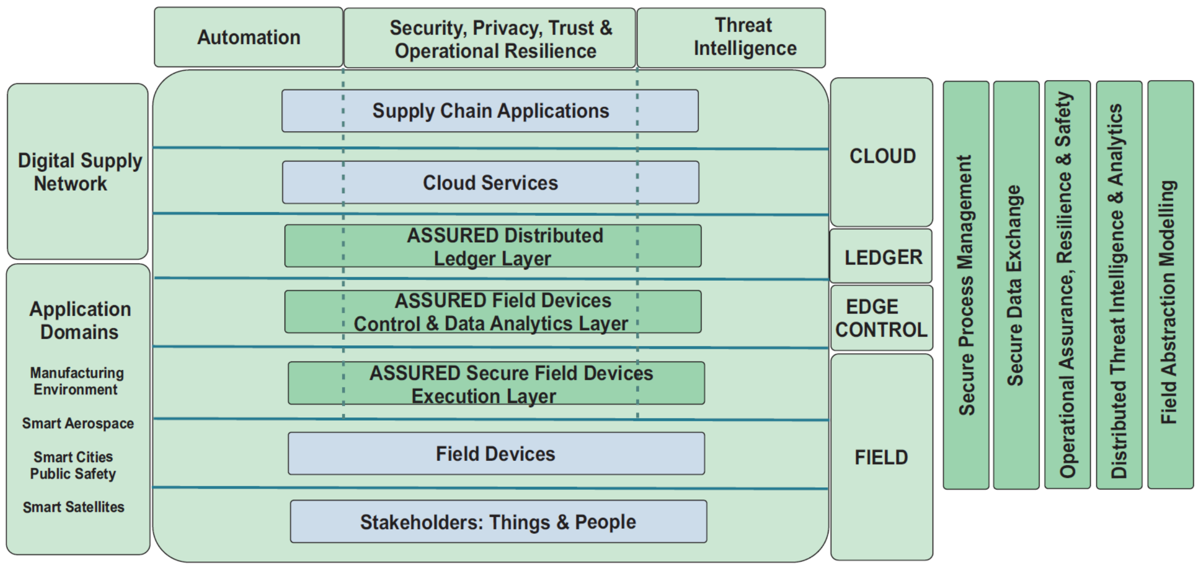
Automation
Security, Privacy, Trust & Operational Resilience
Threat Intelligence
Digital Supply Network
Application Domains
Manufacturing Environment
Smart Aerospace
Smart Cities Public Safety
Smart Satellites
Supply Chain Applications
Cloud Services
ASSURED Distributed Ledger Layer
ASSURED Field Devices Control & Data Analytics Layer
ASSURED Secure Field Devices Execution Layer
Field Devices
Stakeholders: Things & People
CLOUD
LEDGER
EDGE CONTROL
FIELD
Secure Process Management
Secure Data Exchange
Operational Assurance, Resilience & Safety
Distributed Threat Intelligence & Analytics
Field Abstraction Modelling

FIGURE 3: POSITIONING OF ASSURED IN THE INDUSTRIAL REFERENCE ARCHITECTURE (IIRA)

Towards this direction, ASSURED will adopt key technologies, in the field of **trusted computing and lightweight cryptographic trust anchors**, as enablers for the **secure deployment, and verifiable assurance** of safety-critical components running at the edge and the **secure communication**, between interacting entities, by enabling advanced key establishment mechanisms. This will include trust extensions, leveraging **remote attestation** schemes, guaranteeing the **trust relationships between all layers in the SoS run-time stack**.

2.1 MODELLING CYBER-SECURITY ASSURANCE AND TRUST IN COMPLEX SOS

When we refer to cyber-security assurance witihn ASSURED, we aim to go beyond the classical notion of cybersecurity, which focuses on the protection of individual assets from network attacks. Here, we aim to capture the interactions and security relationships within the assets and define a holistic security approach, so that we can eventually achieve the desired level of trustworthiness in an entire "System-of-Systems" environment. To this end, we employ assurance and attestation services that consider the security of the system as a whole.

By using such assurance and attestation services, ASSURED increases the level of trustworthiness and integrity of the overall SoS-enabled ecosystem.This does not only include the **integrity of the constituent hardware and software assets** (and the data they exchange), but goes beyond this notion to also capture the **security relationships and interactions between such system components in the context of a trusted service graph chain**. Particularly with respect to safety and security, components must be enabled to make and prove statements about their state and actions so that other components can align their actions appropriately and an overall system state can be assessed and security policies can be evaluated and enforced. Recall, for instance, the example scenario that was described in Section **Error! Reference source not found.** where single ECUs, in an aircraft, may be able to remotely attest to other devices that they are in untampered state of integrity and have up-to-date and valid input data available on which it bases its decisions. Based on such a proof, another ECU may then decide (based on a security policy) that it can accept commands from the earlier ECU without risking the safety and security of the overall system. For example, the landing stability control ECU may accept data from wheel sensors based on which it will control

Funded by the Horizon 2020 Framework Programme of the European Union

the vehicle's brakes. *This can be expanded to ensure the safety and security of an overall aircraft and then of systems of aircrafts communicating with the ground station*.

Such security relationships, between system components, are initially extracted from the operational goals of the service running and provided by the overall SoS, and complement the security – in a comprehensive and systematic way – by uncovering additional security objectives between related components to produce and integral security solution. They are usually defined by three concepts, namely **isolation, interaction and representation**.

- An *isolation relationship* exists when a component is partially or totally separated, i.e., isolated, by a second component, from other components located inside or outside a system. Examples can be considered software processes, running as part of the computing base of a system, whose execution must be done in isolated environments in] order to minimize the attack surface; or a data bus that isolates the information transmitted between two digital circuits. Such a relationship essentially dictates the very **strict integrity requirements on the input data needed by a component** towards progressing with its execution. For instance, in the "*Safe Human Robot Interaction*" scenario, as part of the Smart Manufacturing use case, the function for calculating the worker's position in a manufacturing floor is based on data coming from the location sensors that need to be either protected by a dedicated data bus or through advanced attestation mechanisms.
- An *interaction relationship* exists when two components interact or communicate in any way. The actual executed functions or the purpose of the communication are not relevant by themselves in the context of security, but only the location of the interactions for the purpose of identifying security requirements of the interacting components. For instance, in the previous scenario, examples of interaction relationships include the location calculation software process interacting with the underlying operating system, or the information transmitted between the location sensor's driver and the processing application software.
- A *representation relationship* exists between a system component acting on behalf of another component. Components participating on a representation relationship can be of any type, namely physical or digital without restrictions, and they enable the joint or gateway between different categories of components. Again, in the aforementioned scenario, an example of a representation relationship could be the collection of all system data, from the deployed robot arms to the Industrial Gateway, that represents the correct state of the entire manufacturing floor – in the context of safety environmental conditions with no fatal worker accidents.

By adopting such a cyber-security architecture, we claim that a **SoS can withstand even a prolonged siege by a pre-determined attacker** with known or unknown capabilities as the system can **dynamically adapt to its security and safety state**. This is substantially more flexible than traditional security mechanisms that often try to maintain and enforce pre-defined policies using rather static security mechanisms. We, essentially, provide a very high level of **operational assurance in integrity, security, and finally safety of the target SoS-enabled ecosystem as we actively manage the system states by permanently engaging with all involved devices and components in the context of a specific service graph chain**. This stems from converting all components to **roots-of-trust**, capable of providing verifiable evidence on their correctness, and using these roots-of-trust to establish and maintain trust relationships. Once a trusted chain is materialised, secure chain communications can be established and used to provide trusted chain-wide system updates. Thus, using the concept of a trusted service graph chain, trusted communities of services can be created within the "Systems-of-Systems".

To construct such "trusted chains of services and devices", we need to break them down to a composition of multiple heterogeneous devices in order to be able to identify the types of properties that need to be (periodically) verified per device (and their running software

Funded by the Horizon 2020 Framework Programme of the European Union

processes). However, towards this direction, it is also imperative to have a baseline of system specification common to all the edge devices.

2.2 IN THE SUPPLY CHAIN WE TRUST

Several cybersecurity assurance models have been proposed in the literature, in the context of various domains of the industry. Next, we present a state-of-the-art analysis of the methods that have been proposed, in the domains that refer to each of the use cases considered in ASSURED, i.e., the **smart city** domain, the **smart satellite communication** domain, the **smart manufacturing** domain and the **safe aircraft upgradability and maintenance** domain. Each of these is characterized by different needs and security requirements, which should be addressed by the cybersecurity assurance solution that will be applied.

Taking into consideration the state-of-the art that will be presented for all the considered domains, we can afterwards define the goal of the cybersecurity services provided by ASSURED. In the previous section, we introduced the types of relationships between assets (**isolation**, **interaction**, and **representation**). However, as it will become apparent throughout this section, the trust models proposed in the literature are obsolete in this regard, because the state-of-the-art in cybersecurity models does not capture this full set of relationships that need to be considered in order to achieve the desired level of trustworthiness of an entire "System-of-Systems", but only a subset of them. In ASSURED, we aim to cover this gap, by proposing **a security model that captures all three types of asset relationships**. Next, we describe how we aim to achieve this, by placing the ASSURED security services in the context of the envisioned use cases.

2.2.1 Smart City Domain

Smart City domain: Trust models for this category of systems have been introduced in the recent years [6][7], since the concept of smart cities is a rather new and has been launched after 2014. The key-factors in the models include [8][9]:

- Actors, users, management of interactions
- Applications, data
- Network, IoT, devices.

The main tool for trust management is the **Business Process Model and Notation (BPMN)**, applicable to diverse domains for the modelling of processes, annotations, data objects etc. and it has been applied also for smart city processes [10]. Recently **SecBMPN2BC** has been developed for research purposes and tested in case-studies including a city process [11]. SecBPMN2BC is an evolution of SecBPMN that DAEM has tested and used for the modelling for city processes in the H2020 project VisiOn "Visual Privacy Management in User Centric Open Requirements" H2020 – DS – 2014 – 1 – IA - GA 653642 [12][13]. The specific case-study for the application of SecBPMN on city processes referred to citizen data privacy by design during the consumption of a city service, more specifically, the issuance of a birth certificate. This case has been further researched for the evaluation of SecBPMN2BC that aims extend BPMN 2.0 for the design of business processes for Blockchain with a model driven approach.

2.2.2 Smart Satellite Communication Domain

In order to provide assurance in Smart Satellite Communication services, a **Trust Management System (TMS) for Satellite Flight Software (FSW)** tele-commanding that detects anomalous behaviour is proposed by Duncan [14], implementing the multiple trust

Funded by the Horizon 2020 Framework Programme of the European Union

mechanism TMS framework proposed by Zhao and Varadharajan detailing a unified trust management framework [15] and following an architecture similar to KeyNote TMS [16].

According to the TMS proposed by Duncan, Trust mechanisms allow trust relationships to be evaluated by system policies. The main trust mechanism proposed is the **Interaction Trust (I-Trust) mechanism**, which monitors the behaviour of entities in the system based upon interaction markers. The I-Trust mechanism consists of functions which calculate and maintain I-Trust values for entities communicating with the FSW. Each entity being tracked by the TMS can have multiple trust markers associated with it. A separate I-Trust value is calculated for each marker associated with an entity. These I-Trust values are later used to make policy determinations in the system. Credential Trust and Policy evaluation mechanisms are also mentioned along with a proposal for additional trust mechanisms that can be used (e.g., environmental trust mechanism).

According to Manulis [17], the trend for more part of communication stack to consist of SW instead of hardware opens the system to software threats and therefore they propose that future implementations should take in mind how to trust software to behave as intended to be.

Attacks can target the ground segment, the communications, and the space segment. Security threats for ground segment can include physical attacks, computer network exploitation, data corruption and outdated software deployed. Main security threats at communication level can include jamming, interception of data over a communication channel (eavesdropping), hijacking and spoofing. When it comes to the space segment, some threats can include software vulnerabilities and replaying recorded transmissions.

Some of the open challenges mentioned from Manulis for the satellite and ground segments are the following:

- Need for lightweight authentication and secure communications considering the CubeSat area and power consumption limitations.
- Key management in terms of scalability, group of dynamics (CubeSats entering and leaving constellations, assign & revoke key respectively), key protection and quantum key distribution.
- Software and firmware updates, handle introduction of vulnerabilities.

ASSURED can adress these above mentioned open challenges through the use of lightweight cryptoprimitives, key agreement protocol, software & control flow attestation and use of block chain technology. Specific properties to be attested by assured include the hash of software services (on Ground Station & CubeSats).

2.2.3 Smart Manufacturing and Smart Factory Domain

Security within the context of this domain is of utmost importance when it comes to integrating **Industrial Internet of Things (IIoT)** solutions to existing infrastructure. The notion of introducing technologies like Blockchain has been taken up by research as well as Open-Source / Proprietary providers, which in turn is providing a better overview of introducing Trust as well as Security within these smart manufacturing scenarios.

Each provider has a large set of tools and information that comprise of their ecosystem when it comes to providing IoT / IIoT Security with Blockchain Technology to form a trust model within a smart manufacturing unit. Large Enterprise solution providers such as IBM, SAP, Accenture provide their own proprietary solutions for introducing trust management via Blockchain through their own cloud platforms. The solutions are developed on a consultancy-based model and are generally highly narrow-focused based on the manufacturer’s criteria.

Funded by the Horizon 2020 Framework Programme of the European Union

Solution provider **MultiChain** [18] provides an **OSS (Open-Source Software) / Commercial solution**. The OSS variant provides basic features for smaller enterprises to get started on a simple adoption of Blockchain, and eventually fulfil increased requirements and demands by moving from the open-source solution to the commercial solution. Multichain provides a diverse ecosystem and software suites for external key management, permission consensus as a governance model, clustering and high availability and wallets for private key management. Their software suite is available online under the GPLv3 License [19] with commercial licenses and support offered on a consultancy basis. DoubleChain [20] provides a Blockchain-based platform for management of IoT Devices and assets, as well as other forms of asset and supply chain management. DoubleChain provides a patented authentication solution for IoT devices called **AEGIS** which uses Blockchain to provide trust management at the device level. **OriginTrail** [21] provides a Blockchain platform solution that focuses on data interoperability and increased collaboration between enterprises in the smart manufacturing area. OriginTrail relies on GS1 standard's **Electronic Product Code Information System (EPCIS)**, which provides a sound foundation for representing data in a fixed format, thus making systems and products in manufacturing systems more coherent and easily reliable and transparent through usage of Blockchain. A similar solution is provided by **WaltonChain** [22], where **Radio Frequency Identification (RFID)** systems can be integrated and transparently maintained through Blockchain integration.

A more mature ecosystem is **IOTA** [23], which provides a wide range of software suites and documentation for various purposes. The Industry Marketplace from IOTA [24] is a vendor as well as industry-neutral platform which focuses on automation of trading of physical and digital goods and services. This marketplace provides a transparent and reliable way to share process and manufacturing information through trust models and encrypted channels to necessary third parties and stakeholders. It provides **Decentralized Identifier Identification (DID)** for secure authentication in a decentralized manner.

2.2.4 Secure and Safe Aircraft Upgradability and Maintenance Domain

In this case, since we consider the safety of aircraft where issues may pose a severe risk to the wellbeing of human passengers and personnel, it is imperative to employ a comprehensive security solution, that takes into consideration all the components of the system, such as flight management modules, environment control systems and on-board wireless services.

One security solution for smart aerospace environments has been proposed by UTRC [25]. Specifically, the need to move to an automatic and remote approach to deal with the maintenance of both physical and cyber elements of the aircraft has been outlined throughout the document, particularly in sections III-D and III-F.

Also, in the **ARINC technical reports**, ARINC 827 [26] and ARINC 835 [27], the methods for to ensure safe software distribution in aerospace environments was discussed. Specifically, the use of a signature is proposed, which is only checked when installed on the device. While this is a good starting point, checking the signature only when the software is being installed is not enough and ASSURED would improve this by using run-time attestation mechanisms.

2.3 OVERVIEW AND COMPARISON OF EXISTING MODELS

Since ASSURED aims to develop a holistic framework targeting the cyber resilience in the supply chain domain, it is highly essential to be able to **represent business and technical operations** in such a way that allows **deterministic interpretation of acceptable states**. This essentially defines what constitutes a "normal" behaviour of an edge device that needs to be considered for attestation and verification, prior to making a decision on the correctness of a device's state. It is important to note that the criteria for defining acceptable states are also

Funded by the Horizon 2020 Framework Programme of the European Union

related to the properties that should be attested, taking into consideration the attack landscape that was presented in Section **Error! Reference source not found.**. The selected cyber-security assurance and trustworthiness model should take into consideration these factors in order to best address the requirements of a CPSoS operating within the ASSURED environment.

In the context of cybersecurity assurance**, Business Process Management (BPM)** models are frequently used in software development for understanding the behaviour of the users, their requirements, while resources are either human or non-human assets, e.g., software, apps, or IT devices. These models should provide a representation of business operations that includes **entities**, **entity types** (e.g., cyber, physical, human), **interactions**, **interaction types**, **protocols**, **attributes**, etc. As aforementioned, this modelling is the cornerstone of the ASSURED attestation mechanisms. **Several notations** are nowadays available to model business processes at different levels of granularity. Currently in the literature, the two most prominent types of BPM models are a) the **Business Process Modelling Notation (BPMN)** and b) the **Case Management Model and Notation (CMMN)**, and c) the **Decision Model and Notation (DMN)**.

BPMN is an activity-centric notation aimed at capturing business processes [28] and is under the supervision of **Object Management Group (OMG)** [29]. BPMN is the standard for modelling business processes and is widely used in both industry and academia. BPMN is based on flow-based modelling notations, such as UML activity diagram and IDEF [28]. It is designed to facilitate both communication between various process stakeholders [30] and for specifying requirements for software systems [31]. BPMN consists of tasks that represent the activities to be performed, gateways that route the sequence flow, events representing things that happen during the execution of a process, data capturing the flow of information and swim lanes depicting those responsible for the execution of the activities. Although BPMN **perfectly captures the control flow among the activities** and is also appropriate for modelling the **inter-dependencies** between manual and automatic tasks, it is **lacking an effective support in modelling the data management**, which is fundamental in the scope of security analysis in "Systems-of-Systems". To address this issue, it has been proposed to consider **smart devices as first-class citizens in the models** [32]**.** Such extensions aim to make BPMN suitable to capture the role of CPS in business processes.

In contrast to BPMN, **CMMN** is an **artefact-centric** and **declarative modelling notation**, designed to support case management and handling for dynamic changes [33]. The aim of CMMN is to assist with modelling the flexibility required for knowledge workers and the exceptions that occur in such declarative processes [34]. The main difference between CMMN and BPMN is the paradigm shift from procedural to declarative model [35][36]. In ASSURED, business process models will include **an enriched data model able to give the opportunity for the designer to increase the awareness with respect to the data managed securely by the CPS**. Focusing on data produced and/or consumed by the elements in the CPS gives the ability to the modeler to abstract from the intricacies of the underlying platform and to concentrate on the data. In this respect, ASSURED will also investigate the adoption of the CMMN notation, to **model knowledge-intensive business processes relying on CPS environments**, **where no specific control-flow** can be identified but most of the activities are driven by the changes of data status which can be driven by event occurring during the execution of the process. Also, it is important to consider that, availability or unavailability of data which depends on the status of the smart devices involved in their management, could affect a successful completion of the process. For this reason, according to the security-by-design pillar, **ASSURED will also investigate how to embed in the business process models information about data alternatives which can compensate the missing data, as well as sub processes which can be enacted in case original activities are compromised**. It should be clarified that models will be totally aligned with IIRA and RAMI 4.0 Patterns.

Funded by the Horizon 2020 Framework Programme of the European Union

Another model complementary to the BPMN and CMMN models is the **Decision Model and Notation (DMN),** that also introduced by the OMG group and converts the code behind complex decision-making into easily readable diagrams [37]. All the three models outlined in this section are called the **"triple crown" of business modelling notations**, intended to bridge between business and IT by providing graphical representations.

Even though the notion of BPM modelling has been extensively studied in the literature, **it has rarely been studied in tandem with the notion of trustworthiness**. One relevant research effort was presented in [38], where the integration of trustworthiness requirements in business process models using BPMN was proposed, and it was suggested that trustworthiness should be considered in the management of both human and non-human resources and in all stages of the business process life cycle (i.e., design, modelling, implementation, execution, monitoring and analysis). However, there is a lack of discussion on how to build the required level of trustworthiness. In addition, a conceptual four-stage model of trust-aware process design in BPM has been proposed in [39], which is based on reducing either uncertainty of specific process elements, or the vulnerabilities. A more recent work in [40] proposes **Trust Mining**, a business process modeling tool to analyse uncertainties and relationships based on previous work in [39].

2.4 SELECTION CRITERIA FOR THE ASSURED MODEL

From the above sections, it follows **artefact-centric modelling is more appropriate to be considered in the context of ASSURED**, since it allows us to concentrate on the specific cases and scenarios offered by the envisioned use cases. As artefacts we consider all the comprised devices – the difference in the ASSURED ecosystem is that we can use both the extraction of specific control-flows as well as activities that are driven by changes of the data/device status. Thus, this requires the identification of the core properties that need to be considered for attestation towards the **enhanced trust assurance of field elements**. Specifically, taking into consideration the requirements that have been outlined in the previous sections, the selection criteria for the cyber-security assurance and trust model employed by ASSURED are as follows:

- The model should be able to capture **all three kinds of security relationships** outlined in Section 2.1 (isolation, interaction, and representation), in order to cover the full range of asset relationships to fully capture the requirements of a CPSoS belonging to any domain. As it was previously mentioned, the existing modelling methodologies are outdated in this regard, therefore ASSURED should provide a solution in order to address this issue.
- The ASSURED model should be able to provide **trust assurance based on the properties that define acceptable states** (property-based attestation), meaning the properties that can be used in order to ensure that the assets comprising the CPSoS, as well as the entire system-of-systems, can be considered to be in a trustworthy state. These properties are defined by the threat landscape presented in section **Error! Reference source not found.**, where it was shown that software-based vulnerabilities are becoming increasingly prevalent.

2.5 ASSURED MODEL

The hybrid model adopted will consider an extension of the CMMN but combining both artefact-centric modelling and data-driven activities that dictate the status of the edge devices. This way ASSURED trust management includes both a) modelling of the interdependencies and

Funded by the Horizon 2020 Framework Programme of the European Union

interactions of the artefacts and b) modelling of the correctness and trustworthiness of the artefacts per se. The key aspect of ASSURED hybrid model is the role of enablers to extend the data sharing and security to the digital supply network. This capacity, jointly with the increasing demand by Industry 4.0 towards enhanced data sharing, will generate new opportunities to smoothly integrate ASSURED into the mainstream evolution of IIoT and Industry 4.0 Reference Architectures (i.e., RAMI 4.0 [41]). Towards this direction, ASSURED modelling is compatible with the reference architectures that have been proposed by the two key industrial consortia that are advancing the development of IIoT, namely Platform Industrie 4.0 and Industrial Internet Consortium, as shown in the figure below. These initiatives have developed RA models for Industry 4.0: the Industrial Reference Architecture (IIRA) and the RA Model for Industrie 4.0 (RAMI). RAMI 4.0 supplements the IIRA model with the axes "Lifecycle" and "Hierarchical Levels", while each of the four viewpoints outlined in IIRA reference architecture can be compared with the respective layers on the vertical axis of RAMI 4.0.

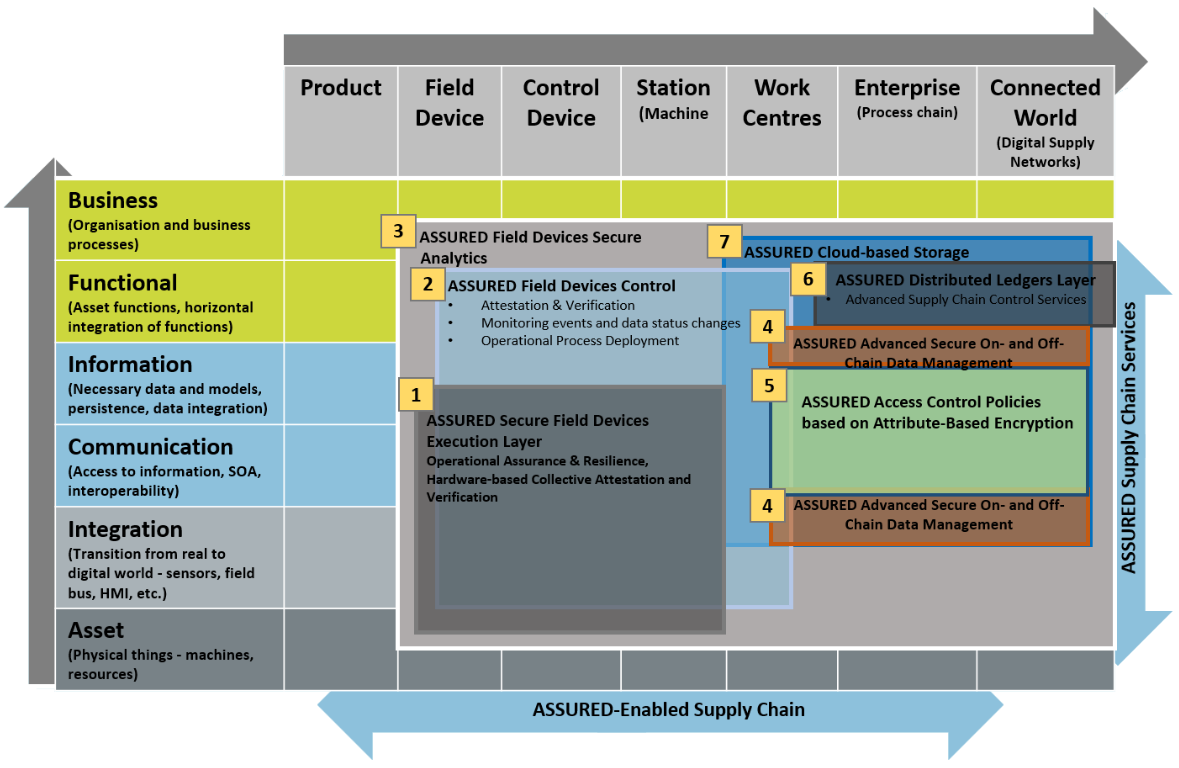
Product
Field Device
Control Device
Station (Machine
Work Centres
Enterprise (Process chain)
Connected World (Digital Supply Networks)
Business (Organisation and business processes)
Functional (Asset functions, horizontal integration of functions)
Information (Necessary data and models, persistence, data integration)
Communication (Access to information, SOA, interoperability)
Integration (Transition from real to digital world - sensors, field bus, HMI, etc.)
Asset (Physical things - machines, resources)
3 ASSURED Field Devices Secure Analytics
2 ASSURED Field Devices Control
Attestation & Verification
Monitoring events and data status changes
Operational Process Deployment
1 ASSURED Secure Field Devices Execution Layer
Operational Assurance & Resilience, Hardware-based Collective Attestation and Verification
7 ASSURED Cloud-based Storage
6 ASSURED Distributed Ledgers Layer
Advanced Supply Chain Control Services
4 ASSURED Advanced Secure On- and Off-Chain Data Management
5 ASSURED Access Control Policies based on Attribute-Based Encryption
4 ASSURED Advanced Secure On- and Off-Chain Data Management
ASSURED Supply Chain Services
ASSURED-Enabled Supply Chain

FIGURE 4: ASSURED MODEL REFERENCE ARCHITECTURE

ASSURED captures the operational processes and the various assets, of both RAMI 4.0 and IIRA and extend them for the case of digital supply networks comprising an ecosystem of collaborative manufacturing environments. The developed, enhanced and proved a trusted framework for attestation and system assurance is represented in Figure 4, where it is evident the alignment with the RAMI 4.0 and hence IIRA reference architectures. More precisely, the vertical axis represents the interoperability layers of RAMI 4.0, while the horizontal axis on top represents the hierarchy levels of RAMI 4.0. The following seven layers summarise how ASSURED hybrid model aligns with these reference architectures and how can be applied in the "Systems-of-Systems" used cases of Section 4.

1. **ASSURED Secure Field Devices Execution Layer: Operational assurance and resilience, hardware-based collective attestation and verification** – This layer covers the supply chain from physical assets to the produced information and applies to the hierarchy levels of RAMI 4.0 namely "Field Device", "Control Device" and "Station".

Funded by the Horizon 2020 Framework Programme of the European Union

2. **ASSURED Field Devices Control: Attestation and verification, monitoring events and data status changes, operation process deployment** – This layer covers the transition from the real to digital world to the assets functions and applies to the hierarchy levels of RAMI 4.0 namely “Field Device”, “Control Device”, “Station” and “Work Centres”.
3. **ASSURED Field Device Secure Analytics** – This layer acts as an umbrella and covers all the interoperability and hierarchy levels of RAMI 4.0, apart from the “Product”.
4. **ASSURED Advances Secure On- and Off- Chain Data Management –** This layer covers the transition from the real to digital world to the assets functions and applies to the hierarchy levels of RAMI 4.0 namely “Work Centres”, “Enterprise” and “Connected World”.
5. **ASSURED Access Control Policies based on Attribute-Based Encryption –** This layer covers the access to information in general and to the necessary data and applies to the hierarchy levels of RAMI 4.0 namely “Work Centres”, “Enterprise” and “Connected World”.
6. **ASSURED Distributed Ledgers Layer: Advanced supply chain control services** – This layer covers the asset functions and applies to the hierarchy levels of RAMI 4.0 namely “Work Centres”, “Enterprise” and “Connected World”.
7. **ASSURED Cloud-based Storage –** This layer covers the transition from the real to digital world to the organisation and business processes and applies to the hierarchy levels of RAMI 4.0 namely “Work Centres”, “Enterprise” and “Connected World”.

2.5.1 ASSURED Use Cases Artefact-Centric Modelling

Contrary to the traditional approach to industry in which a **hardware-based structure** with **a clear communication hierarchy was prevalent**, Industry 4.0 introduced flexible systems whose functions are not bound to hardware but distributed throughout the network. In these new systems internal communication can now be **observed across an organisation’s hierarchical levels**. **New types of interactions** have been introduced and external interactions between organisations have changed significantly and become more flexible. The result is that **Industrial Controls Systems (ICS) are a prime target**. These systems are **increasingly Internet-enabled for easier monitoring and control**. But moving to open systems with **IP addresses creates more avenues for attack –** especially if Internet access is poorly protected and ICS protocols for authentication are weak. Total financial losses attributed to security compromises jumped 38%. It could be argued that extracting monitoring data is a primary resource for cyber incident analysis.

Thus, to fully understand the ASSURED ecosystem, **one should identify the main Industry 4.0 components based on current RAMI 4.0** reference model for manufacturing Industrial IoT and supply chain environments. From a security point of view, such components can be placed into the following categories:

- **Industrial Control Systems**
- **IIoT End Devices – Sensors**
- **Control Systems Communication Networks and their Components**

Funded by the Horizon 2020 Framework Programme of the European Union

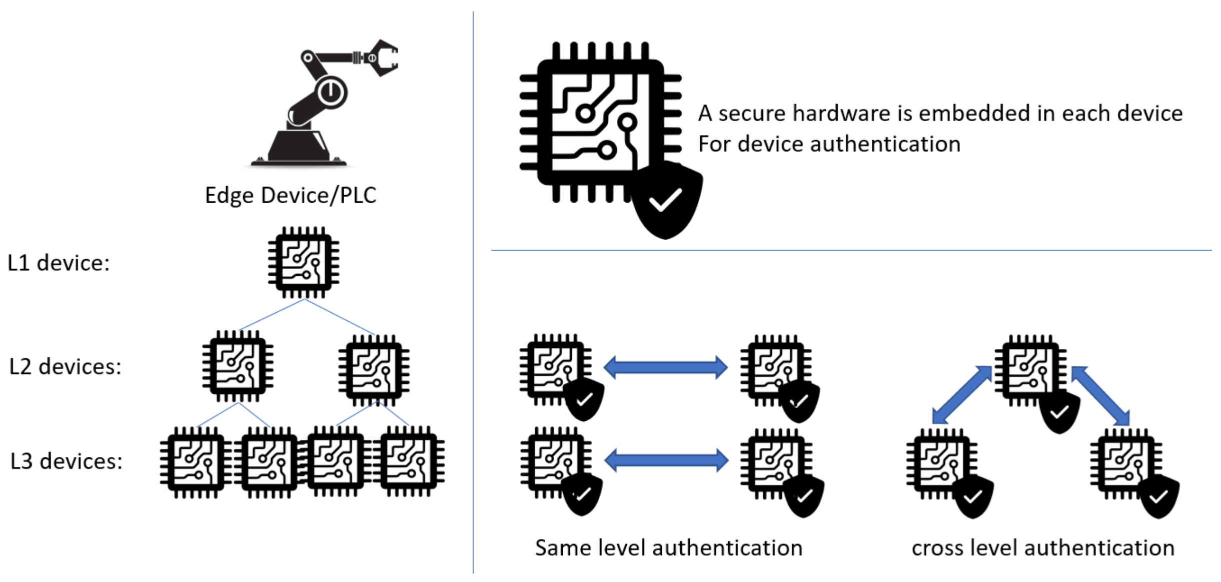
Edge Device/PLC
L1 device:
L2 devices:
L3 devices:
A secure hardware is embedded in each device
For device authentication
Same level authentication
cross level authentication

FIGURE 5: DEVICE LAYER STRUCTURE IN THE SMART MANUFACTURING USE CASE

Figure 5 depicts a conceptual overview of the layered structure of devices in the Smart Manufacturing use case (as an example), which demonstrates the types of relationships and interconnectivities between the assets that constitute the system architecture, as well as the levels of authentication that can be performed between these devices. For example, **same level authentication** can be performed between devices of the same level, where one acts as the Prover and the other acts as the Verifier. Conversely, in **cross level authentication**, attestation between two devices can be performed via a device belonging to a higher level, when a direct connection between the two devices is not available.

In Table 1, we provide a categorization of the devices and assets employed in the context of each use case into each of the categories mentioned above, in accordance with the RAMI 4.0 reference model, which will provide the basis for the artefact-based modelling methodology that will be followed.

TABLE 1: ARTEFACT-CENTRIC MODELLING IN THE ASSURED USE CASES

	Industrial Control Systems	IIoT End Devices – Sensors	Control Systems Communication Networks and their Components
Smart Manufacturing, Safe Human Robot Interaction (HRI)	The various robots and robotic arms operating on the manufacturing floor are connected and controlled by **Programmable Logic Controllers (PLCs)**. These can be accessed by an **OPC-UA Server** running on an **Industrial PC**. An **IoT Gateway** processes the collected data with collision detection algorithms.	The **robots** that constitute the manufacturing system can estimate the location of themselves and the human workers in the factory floor in order to avoid accidents, by using various **integrated location sensors**.	An **Ultra-Wide Band Wireless Location** System collects location information from the robots that operate on the manufacturing floor. The PLCs and robotic arms are connected over a **PROFINET network**.
Smart Aerospace	A wide range of operations is managed by various ECUs, which constitute the **Flight Management Systems**	Each of the on-board cyber-physical systems contains a set of **sensors**, which serve to	A **Secure Service Router (SSR)** gathers sensor data while flying, and transfers the data to

Funded by the Horizon 2020 Framework Programme of the European Union

	(FMS), **Environment Control Systems (ECS)**, **Cockpit Flight Instruments (CFI)**, and **on-board Wi-Fi systems**. These are all centrally controlled by a **Ground Station Server (GSS)**.	assist operations pertaining to flight management or environmental control, and measurement of flight data.	the GSS through **a cabled connection**, such as an **Ethernet connection**, when the aircraft reaches the ground.
Smart Cities	The edge devices and surveillance processes are controlled by **PLCs**. The system is centrally controlled by an **Information and Communication Technology (ICT)** system.	A set of **IP surveillance cameras** and **smoke detection sensors** are employed in order to generate data streams of video and sensory data, respectively.	A **cloud-based infrastructure** contains networking components to support the operations center.
Smart Satellites	A **Ground Station (GS)** serves as a central unit, which monitors, maintains, and controls the operation of the deployed CubeSats.	**CubeSats** are miniaturized satellites, which are deployed in order to perform specific missions in space, and are integrated with various hardware assets, such as **cameras** and **sensors** to collect telemetry data.	Communication between the GS and the CubeSats, or between two CubeSats, is performed via **secure channels**. It is also possible for the Ground Station to share data with external stakeholders.

Having identified the artefacts that constitute the operational system in each use case, the next step is to identify the **type of data that should be exchanged, as well as which are the security, privacy and trustworthiness requirements that should be fulfilled**. Also, a quantification of the attack vectors needs to be performed, which will afterwards lead to the mapping of attestation properties that correspond to each type of attack. Based on this mapping, we need to identify which are the possible types of attacks that need to be considered for each asset, and subsequently which types of properties need to be attested in order to mitigate these attacks. This notion will be expanded throughout this deliverable, and the categorization of these properties per asset and type of attack will be presented in detail for each use case in Chapter **Error! Reference source not found.**.

Funded by the Horizon 2020 Framework Programme of the European Union

3 ASSURED EDGE DEVICES ABSTRACTION MODELLING

Having defined the **model of trustworthiness** we want to achieve for a complex SoS-enabled ecosystem (to be further elaborated in D2.2 [2]), this Chapter proceeds with elaborating on the process for capturing **critical platform configuration and execution properties** that need to be considered – during attestation – in order to provide adequate detection measures against the most prominent types of attacks and vulnerabilities (Section 3.3). In addition, we present a generic explanation of those properties, as **system abstractions** (Chapter **Error! Reference source not found.**), to be considered when fleshing out such resources that need to be verified in the context of all four envisioned use cases (Chapter **Error! Reference source not found.**). This will set the scene for the definition of the optimal attestation policies and resources to be attested that need to be enforced to all hardware assets – of each use case asset cartography [2].

The intuition behind this mapping is to identify **what types of properties are violated by specific attacks**. ASSURED, as will be described in D3.2 [3], employs **novel, scalable attestation and verification schemes to corroborate the critical properties of edge devices** as a means to mitigate attacks and establish "*trusted chains of devices*" within a SoS-enabled ecosystem; from the **trusted boot and integrity measurement** of a CPS, enabling the generation of static, boot-time or load-time evidence of the system components' correct configuration (Configuration Integrity Verification (CIV)), to the **runtime behavioural attestation** of those safety-critical components of a system providing strong guarantees on the correctness of the **control- and information-flow properties** (Chapter **Error! Reference source not found.**), thus, enhancing the performance and scalability when **composing secure systems from potentially insecure components**.

In the following, first define the terms **attestation, verification, and events** (Section 3.1), as well as provide the **system specifications considered for the edge devices** (Section 3.2) comprising the SoS-enabled ecosystem. Since the concept of remote attestation is the core building block in ASSURED for guaranteeing the operational assurance of a system and hierarchical composition of systems, such a definition will enable us to better frame the concept of cyber-security assurance in such complex environments.

3.1 SETTING THE SCENE AND RELEVANT DEFINITIONS

As the name suggests, attestation is a trust establishment method that enables an entity (i.e., Prover) to gain trust of other entities (i.e., Verifier) by providing reliable and verifiable evidence on its **correctness and operational assurance**. In a typical attestation paradigm, as shown in Figure 6: Typical remote attestation paradigmFigure 6, the Verifier knows the expected legitimate configuration ***h'*** of the Prover. Additionally, Prover and Verifier are equipped with Attestation keys (AKs) used for protecting the integrity of the exchanged attestation reports. During the attestation process, the Verifier sends a challenge or a nonce ***N*** to the Prover (Step ①). This nonce is essentially a random value so as to guaratee the *freshness* of the attestation and protect against replay attacks. As will be described in D4.1 [42], in ASSURED, we aim for a **random but re-producible nonce** so that any entity (through the deployed smart contracts depicting the enforced attestation policies) can verify its correct generation; *based on the hash of the last block header from the ledger*. Upon receiving the challenge, the Prover measures its (software or configuration) state (Step ②), concatenates the measurement ***h*** with the challenge ***N***, signs the result with its AK and returns an authenticated response ***δ*** to the Verifier (Step ③). Since the Verifier knows the expected legitimate configuration of the Prover ***h'*** (*through* ***reference values,*** *that have been provided by the System Administrator, as system behaviour descriptions during the cartography of all software assets of the target SoS*) and the

Funded by the Horizon 2020 Framework Programme of the European Union

challenge ***N***, the Verifier computes ***δ′*** and compares it with the response ***δ*** received from the Prover (Step ④). If ***δ*** matches with

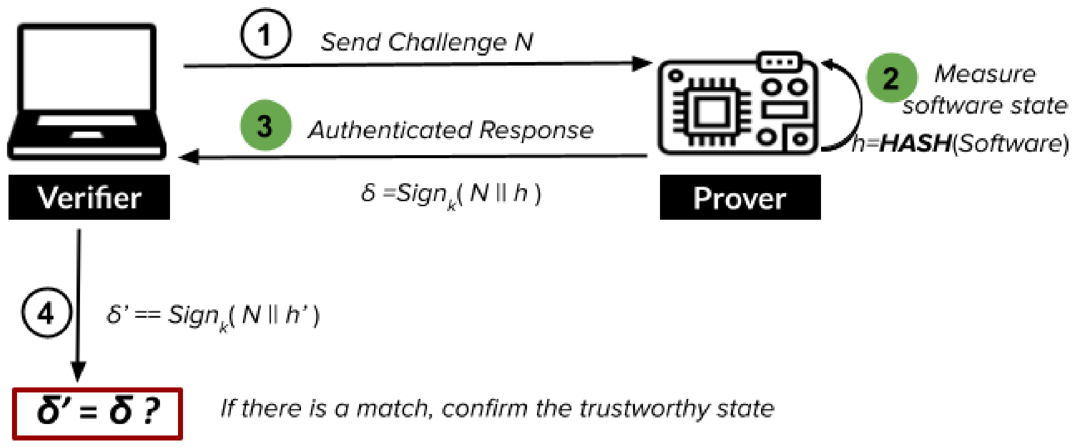
1
Send Challenge N
2
Measure software state
h=HASH(Software)
3
Authenticated Response
$\delta = Sign_k(N \parallel h)$
Verifier
Prover
4
$\delta' == Sign_k(N \parallel h')$
$\delta' = \delta$?
If there is a match, confirm the trustworthy state

FIGURE 6: TYPICAL REMOTE ATTESTATION PARADIGM

δ′, the Verifier claims that the Prover is trusted; **otherwise this is a strong indication of risk that the Prover is compromised and further investigation is needed**. In this context, all monitored traces, collected when extracting the device's measurement, are sent to the Attack Validation Component [43] for further analysis and identification of the exact intrusion point so as to then define any additional needed countermeasures.

Based on the Verifier's location, attestation schemes are classified into two categories: (1) **Remote Attestation**, in which Prover and Verifier reside in different devices and the remote Verifier triggers the Prover's attestation, as depicted in Figure 6, and (2) **Local Attestation**, in which Prover and Verifier may reside in the same device and the Prover self-triggers attestation. *ASSURED is envisioned to support both of them* through the definition [3] of remote schemes for attesting the executional and behaviour correctness of a system (i.e., Control-Flow Attestation, Swarm Attestation, Jury-based Attestation) and local schemes for privacy-preserving platform authentication (i.e., Direct Anonymous Attestation) or certifying the configuration correctness of a device (i.e., Configuration Integrity Verification) prior to allowing it to perform an action; i.e., *check the integrity of a loaded binary prior to the device been allowed to use the data outputed by this software binary*.

In this context, without loss of generality, in ASSURED, we define the following unified notions of **attestation and runtime verification**:

- **Attestation** only refers to the procedure of collecting (Verifier) and/or generating (Prover) attestation reports, and sending attestation reports to other entities (Steps ② - ③ in Figure 6). ASSURED specifies the attestation details as part of the attestation policies to be deployed and enforced through the use of smart contacts [42], e.g. *what kind of information needs to be collected, the types of system properties that need to be attested, whether the attestation report is sent to Verifier or threat intelligence engine*?
- **Verification** is a computing analysis paradigm based on obsering the system's behaviour, through the sequence of **events** performed as the system executes, and comparing it against what constitues the expected behaviour. Such events may, for instance, depict the **configuration sequence of a device** (e.g., list of loaded binaries) or the **software behaviour correctness by verifying the integrity of a specific control-flow** – *these are included in the collected attestation reports*. More specifically, runtime verification refers to the process where a Verifier validates the signature of an attestation report and verifies the Prover status through information included in the report (Steps ④ in Figure 6).

Funded by the Horizon 2020 Framework Programme of the European Union

This is needed for guaranteeing the integrity of the collected system states by the Prover – against compromised Provers which may try to impersonate the tracer processes running. Compounding this issue, we need to have strong mechanisms for **secure identify provisioning, integrity and authenticity of the collected measurements, and secure key management and distribution** all of which are provided in ASSURED through the newly designed TPM-based Wallet that can offer all the necessary trust properties as well as secure communication with the host tracer [44].

Recall that the aim of the ASSURED framework is the **cyber resilience of the entire supply chain** (Section 2.1) against an enhanced threat model, including advanced memory-related attacks like run-time control-flow attacks (Section 3.3.1). Therefore, it is imperative to capture **suitable behavioural and low-level concrete execution properties of a Prover** depicting both its overall state as well as the state of specific internal components (e.g., software processes) needed as part of the various identified security relationships with other systems. To this end, it is envisaged to perform both **Configuration Integrity Verification (CIV) and Control-Flow Attestation (CFA)** [3]: Not only **static properties** such as code and configuration integrity, but also **dynamic properties** like control-flow and data-flow information are captured by ASSURED attestation enablers. Therefore the attestation mechanism can disclose the corrupted integrity of mission-critical assets, and disrupted tasks executed on the edge devices.

In this context and in order to mitigate the common drawbacks of current attestation schemes when it comes to efficiency, scalability and robustness [45], ASSURED uses **property-based attestation on a specific set of critical configuration and execution properties of the target SoS and their comprised devices**. The intuition behind this is that most of the current limitations mainly stem from the fact that existing attestation mechanisms try to verify the integrity, during run-time, of the **entire (untrusted) code base** of commodity platforms and operating systems without also considering additional constraints posed by the existing relationships and interactions with other systems as part of the entire service graph chain. Considering that competitive supply chain application markets will always produce innovative and large systems that consist of diverse-origin software-based components, with uncertain security properties, it can be considered that a sub-set of such loaded software functions can be efficiently protected (in near real-time) against sophisticated run-time exploitation attacks. Hence, the adoption in ASSURED of the artefact-centric modelling notation, so as to frame the correct type of **system properties for representing the operational and security status** of core devices and highly critical services.

Critical platform configuration and execution properties include both **behavioural properties** and **low-level concrete properties about the platform's configuration and execution**, such as the current firmware version it is running, the version of its configuration file or presence of certain hardware properties, integrity of sensor measurements, execution paths to specific memory regions, ports and network interfaces, etc. It also includes abstracting these low-level values to higher level security properties or functions, e.g., custom algorithms running on edge devices. Following this concept and the type of abstract validation properties to be to considered (Chapter **Error! Reference source not found.**), we have identified the exact type of system properties that need to be attested (both for the hardware and software assets) in the context of the four envisioned use cases (Chapter **Error! Reference source not found.**) – using the model representation as put forth in Section 2.5 and the voccabulary of ISO/IEC/IEEE 24765:2017 on "*Systems and Software Engineering*".

The legitimate state of the static properties can be defined both during the design and run-time phases of a system's execution lifecycle, while most other critical execution properties will require run-time verification. For instance, the integrity of a loaded binary can be verified during boot up but might be also required to be attested when the system wants to be securely enrolled to a network (i.e., to be part of a smart manufacturing floor). On the other hand, the run-time verification of monitored control-flow events depicting the state changes of a software or hardware system requires run-time monitoring and attestation probes. Thus, ASSURED will employ **enhanced tracing techniques** [3][46] to monitor the configuration and behavioural properties of interest and collect the runtime evidence of those properties, which will be developed in context of WP3.

3.2 SYSTEM SPECIFICATIONS AND ASSUMPTIONS

In what follows, we document the **system model and adversary model** considered within the overall ASSURED architecture. More specifically, we will describe the **generic system model of devices**, comprising the target SoS-enabled ecosystems (will also constitute the baseline for the envisioned use cases), whose properties need to be attested, the **critical and non-critical system components**, and their requirements regarding **operation, safety, privacy, and security**. In addition, we present an explicit specification of the adversary model envisioned in ASSURED that covers different type of adversarial capabilities.

A high-level description of the system model, as part of the edge device architecture, is depicted in Figure 7. *More information is also given in D3.1* [44]*, where the exact modelling of the trusted computing base per device is described and in D3.2* [3]*, where the detailed break-down of the edge system architecture is put forth for enabling the execution of the ASSURED attestation enablers*. Besides common hardware components like commodity processors, communication busses, memory, and peripherals, we assume the existence of a trusted component, namely a Trusted Platform Module (TPM), which is in line with other attestation work. TPM contains a **unique device identifier** that is unforgeable and inaccessible for unauthorized entities. This identifier together with other **cryptographic keys** (Attestation Key, Secure Communication Key, Device Credentials, etc.) are safely stored in a secure storage that maintains integrity and confidentiality. Moreover, we assume that the hardware provides trustworthy and correct security engines such as a True Random Number Generator (TRNG), cryptographic hashes, and encryption algorithms. TPM is part of system's Trusted Computing Base (TCB), which means that TPM is trusted by default. Likewise, other core ASSURED components like the Tracer, the (TPM-based) Wallet and trusted OS functionalities (e.g., kernel functions) are crucial to the overall security process, thus, are considered to be trusted. These critical components are represented by grey boxes in Figure 7.

The white boxes in the overall system architecture represent the **code and data of the software processes running on the device**, including the Operating System. These programs perform the core operational functionality of the device, but can be prone to many cyber attacks, such as code injection attacks, runtime software attacks, malicious software updates, network-related attacks, etc. For instance, a code injection attack can exploit a software vulnerability on the device's software and compromise the program executed in the device with malicious code. Likewise, it can exploit a software vulnerability in the Operating System (e.g., buffer overflow exploitation) to compromise provided functionalities; from compromising buffers used for the tranmission of packet payloads to exploiting the entire stack for extracting secrets such key material [47]. Additionally, a runtime adversary can subvert the control-flow execution of a device's program without injecting any new code. A comprehensive discussion of the adversary model is presented in Section 3.3. Based on the capabilities of the modelled adversaries, the security of the non-critical components is provided by the critical components running on the device which, in turn, will be used for monitoring and tracing the system state events and properties that need to be verified towards the creation of trust- and privacy-aware service graph chains.

In ASSURED, the **critical components attest to the security state of the devices**, which include both the hardware- and the software-state. Specifically, the Tracer introspects the raw memory of the device and parses the memory to recover security-related information about the current state of the device. This information is used by the Verifier to detect whether a

Funded by the Horizon 2020 Framework Programme of the European Union

potential attack occurred and to update the risk level associated with this specific device. For example, the Tracer can recover the control-flow of a safety-critical program executing on the device; e.g., the worker's location calculation in the context of the HRI scenario in the Smart manufacturing use case. The recovered control-flow can be used in a control-flow attestation protocol to detect whether a non-critical component deviates from its expected control flow. The Tracer may also extract global device information, which is not related to a specific

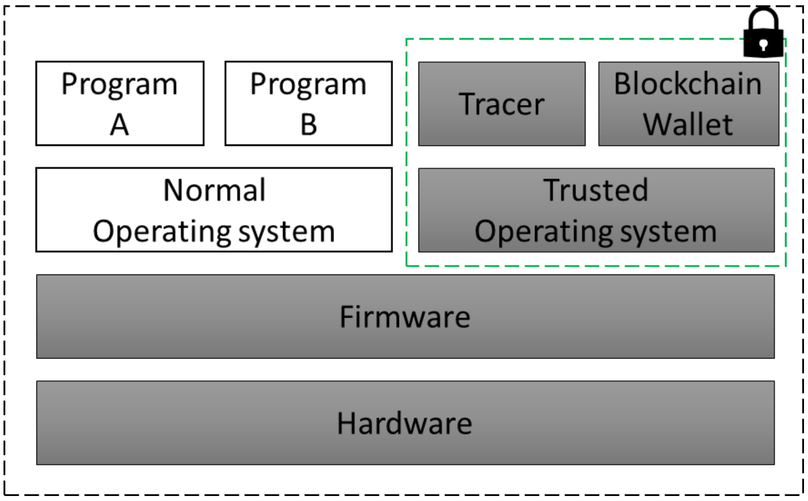

FIGURE 7: SYSTEM COMPONENTS, CRITICAL COMPONENTS ARE GREYED OUT. TRUSTED CODE IS EXECUTED INSIDE A TRUSTED EXECUTION ENVIRONMENT.

program such as the currently executing processes on the device and can be used for malware analysis.

The strong security guarantees offered by critical components makes them a prime target to attacks. Compromising them would effectively render the security mechanisms they provide invalid. Protecting the critical components is therefore fundamental, yet it is challenging: *Software-only protection mechanisms, such as memory safety incurs high-performance overheads, which may be unacceptable for use cases considered in ASSURED*. Instead, to secure the critical components we assume the existence of a Trusted Execution Environment (TEE) on the device (Figure 7). **The TEE protects the confidentiality, integrity, and freshness of code and data embedded within it by isolating it from the rest of the software executing on the device.** The critical components are deployed in the TEE (comprising the "*trusted world*" of an edge device), whereas the non-critical components are deployed outside the TEE denoting the "*untrusted world*". *We have to note, however, that increasing the code that is running as part of a device's TEE has an exponential impact on the performance of the device, since this will require multiple interactions between the trusted and untrusted worlds. In some cases, this may also require the use of additional data buses*. Hence, in ASSURED, we keep this **trusted codebase to the minimum**: essentially, we only consider as trusted the **Tracer which is responsible for the device runtime data and execution stream monitoring** and introspection as well as the **core building block of the Blockchain Wallet which essentially is the TPM**. For the latter, the accompanying TPM Software Stack (TSS) for interacting with the TPM is not part of the considered trusted world.

The TSS is a middleware that provides a multi-level API to applications for accessing the TPM. Through the APIs provided by the TSS, operating system and the users' applications can utilize the security functionality provided by TPM (in the context of ASSURED, the entire TPM-based Wallet). Several instantiations of TSS have been implemented in different languages, such as IBM TSS [48], Intel's TSS [49], Microsoft's TSS [50], Trousers TSS [51], Java TSS [52], and Daonity TSS [53] etc. More specifically, it handles all the data from and to the TPM, which includes marshalling/unmarshalling data, encrypting transactions for cryptographic sessions, parameter checking etc. The TSS consists of three API layers that provide three levels of abstraction, namely the SAPI, the ESAPI and the FAPI. The FAPI is the most feature rich layer, and its purpose is to cover most of the use cases of the TPM as it includes ready to use functions that require very little configuration. It is designed to be simple to use and make the development of applications as easy as possible. The ESAPI is a little more advanced API and is targeted to individuals who seek to have a little better control over what the TPM does. Finally, the SAPI is an interface that requires expert knowledge of the underlying TPM commands and architecture. The functions it contains can be directly mapped to almost every command that the TPM can execute and it allows for fine grained control of the module. This level freedom allows for misuse and errors, that is why it is of great importance to model all the possible threats that may appear. The TSS is closely related to the TPM as they are closely interconnected with the TSS being the gateway for all inbound and outbound communications of the TPM.

However, as part of all the ASSURED security mechanisms, when designed, we will also provide the option of ***weakening these trust assumptions*** *by outsourcing some of the trust calculations to external trusted entities such as the Privacy Certification Authority which is responsible for securely onboarding a device into the overall system network*.

3.2.1 Functional Properties of Systems and Components

Based on this Cyber-Physical System (CPS) architecture and in order to achieve the main vision of ASSURED towards the creation of **trust- and privacy-aware service graph chains**, in complex supply chain ecosystems, the first step is to model such CPSoS and break them down to a composition of multiple heterogeneous components. Each of these components can have their own functions and operational logic (as part of the entire service graph chain) that need to be protected through various **security and privacy-preserving policies**. Such policies can be established by identifying the security and privacy related objectives and relationships between the comprised sub-components. **Security and Privacy objectives** define what should be done about security, for targeted components, during the security process for the entire lifecycle of a CPS – *what type of system states need to be verified both during boot-up and run-time; type of data need to be exchanged for authenticating a component; privacy requirements when exchanging such data (even when verifying a system state, this might reveal information on the software processes running to a device/component), etc.* **Security relationships**, as described in Section 2.1, between system components complement the security and privacy properties, in a comprehensive and systematic way, by uncovering additional constraints and dependencies – between components – that need to be considered when designing and integral security solution.

In addition to security and privacy, all policies need to be aligned with **safety requirements** so as to be able to guarantee the **desired level of trustworthiness and correctness of the outcome of a service –** especially when, in many cases, we are dealing with safety-critical decisions (becoming critical for human lives) such as the HRI scenario in the Smart manufacturing application domain or the Secure Aerospace use case [54]. In almost all branches of industry, such highly automated and nowadays autonomous systems already perform several **safety-critical control functions and operations**, even for tasks they were not designed for [55]. In the future, their scope will increase even further to completely and fully autonomous safe operations; for instance, in the Secure Aerospace domain, autonomous flying, efficient engine control and cloud-based connected aircraft power distribution can be some examples of safety critical operations uploading securely data into the cloud, while their vital operations are crucial for safe human transportation. These systems will cause radical changes to secure and safe SoS, their supply chains, services, and business models in many transport industries beyond the aerospace reshaping the secure and safe industrial landscape in general [56].

Funded by the Horizon 2020 Framework Programme of the European Union

As safety-critical SoS become more dependent in big data analytics and machine learning approaches, their **safety requirements** become of paramount importance when exploring the security mechanisms to protect the SoS-enabled ecosystem. Research in safety verification of interconnected CPSoS has recently exploited standard analytic approaches in the form of black boxes to identify situations of error, like inefficient cutting processes [57], corrupted measurements [58], as well as falsification systems defining the safety of systems exploiting data-driven approaches [59][60][61]. At the same time, **security requirements** sitting aside with system safety requirements have to respect a series of additional **functional properties for system correctness, timeliness, liveness, fairness and accountability that will safeguard the SoS constituents, actors or devices**, from unexpected behaviors causing the loss of human lives. More recently SoS system designers are looking also into the **survivability requirements** of a safety-critical system ("*Provided that the system will suffice an external attack or an unexpected error, ensure that the system service will continue to operate with no more than a 25% time delay*").

In the context of ASSURED, the endmost goal is to address this **convergence of security, privacy, and safety** in a complex SoS ecosystem by **assessing dynamic trust relationships and defining a trust model and trust reasoning framework based on which involved entities can establish trust for cooperatively executing safety-critical functions**. This will enable the exchange of verifiable evidence, on the correctness and trustworthiness of all constituent devices and (hardware and software) assets, between entities that had no or insufficient pre-existing trust relationships. Thus, beyond the identification of security and privacy properties to be achieved, functional safety attributes also need to be considered as inherent part of the overall trustworthiness management of a SoS ecosystem.

3.2.1.1 Functional Safety Properties

Towards this direction, the focus on correct decision making (automated) operations should be governed by overarching properties [62]. The **Overarching Properties** are intended to define a sufficient set of properties for making **approval decisions that take into consideration system level safety and security requirements**. That is, when approval is sought for using a particular entity on, e.g., an aircraft, if the entity can be shown to possess these properties in their entirety, then granting approval for using that entity on an aircraft is appropriate. Hence the name: **properties because they encapsulate the "characteristic qualities" necessary to justify approval; overarching because they are intended to "encompass all" of the necessary properties.**

Overarching Properties are labeled using three different tags: **Intent, Correctness, and Innocuity**. Here are the statements of each.

- **Intent**: The defined intended behavior is correct and complete with respect to the desired behavior.
- **Correctness**: The implementation is correct with respect to its defined intended behavior, under foreseeable operating conditions.
- **Innocuity**: Any part of the implementation that is not required by the defined intended behavior has no unacceptable impact.

3.2.1.2 Security Properties

From a security standpoint, the following functionality should be also provided for the system to operate correctly. Such functionality will be used to map system level security requirements combined with safety properties when the system is safety critical. To that extend, forming overarching properties of and SoS ecosystem and verifying its behavior against them will be a continuous process, starting from the design phase and validated during its operation through correct decision making and policy enforcement.

Funded by the Horizon 2020 Framework Programme of the European Union

- **Secure boot.** Secure boot is a security standard that validates that a device boots with software trusted by the provider. Effectively, the signature of every software component is measured and verified to match the expected value. The check starts with the first loaded software from read-only memory (ROM) up to the operating system. If any of the measured signatures fails the check, the boot process fails, leaving the device in an unusable state.
- **Secure memory and storage.** Devices are equipped with TEE, such as ARM TrustZone, or other trusted components including a Trusted Platform Module (TPM) which would be the basis of the ASSURED Wallet component. **This essentially constitutes the security token for converting the device to a decentralized root-of-trust.** The Trusted Component (TC) provides strong isolation for the code and data in it. Furthermore, the strong isolation allows programs executing in the TC to securely manage cryptographic keys, which are used to encrypt, and sign content saved in the storage. This allows TCs to protect their data at rest even though it is not part of the TC itself.
- **Attestation.** Attestation allows remote users to detect a change in the expected system state. This state includes the hardware, firmware, software deployed, and the dynamic state of the system, such as the control-flow of executed programs. The TC software stack manages the attestation report. Therefore, it is trustworthy. The report is encrypted and signed before sending it to the remote user, which ensures the **confidentiality, integrity, and freshness of the report**.
- **Key management.** Devices are equipped with a platform key that is exclusively available to the TC. The TC uses this key to derive keys for different security purposes such as a sealing key, which is used to access the storage, or an attestation key that is used in the device attestation process. All key management trust attributes are defined in D3.1 [44].
- **Secure measurement.** The Tracer has sufficient privileges to access and measure the memory of all software components of the device. The measurement can be used to detect malicious activity in the device. The Tracer may be executed as part of the device's trusted computing base (Section 3.2). Thus, the Tracer is trusted to perform the measurements and analyze them correctly.
- **Safety requirements.** Together with security considerations, safety requirements should be also considered to be validated and verified. Different access control policies generated by the SoS high level requirements (e.g., the satellite or aerospace use-case) should enable safety critical directives/policies that the device or the network of devices should adhere to.

3.2.1.3 Privacy-Preserving Properties

The last branch in the definition of the overarching properties that we need to take into consideration when making approval decisions regarding the safety and security requirements of the system involves the **privacy-preserving properties**. The goal of these properties is to enable the execution of security functions, without divulging information regarding the configuration or status of the communicating devices, thus preserving their anonymity and privacy. The privacy properties considered in ASSURED are as follows:

- **Privacy-Preserving Platform Authentication.** Together with security considerations, privacy requirements need to also be managed – especially considering the sensitive nature of the information/data exchanged in safety-critical operations such as the one envisioned in the context of the four ASSURED-related use cases [5]. In this context, one of the core features would be for privacy-preserving platform authentication. This entails the use of appropriate credentials that can prove the authenticity and integrity of a device while hiding its identity. For example, consider the **BIBA Safe Human Robot Interaction (HRI)** smart manufacturing use case. When a hardware provider needs to attach its robot to the IoT gateway, this new device needs to be securely enrolled and registered into the system. In this case, the enrollment should be performed by ensuring that the device is in an authenticated and trusted state, but without requiring the device to disclose any information regarding its state or configuration, in order to preserve its privacy. Recall that ASSURED operates in a **Zero Trust mode which means that there are no trust assumptions on the state of a device**. Prior to communicating with any device, a trust relationship needs to be established based from security claims extracted from the execution of the appropriate attestation enablers. However, as will be described in D3.2, exchanging such information might entail data on the types of binaries loaded to a devices or the actual execution path details during its operation. This information has been shown in the literature that can pose a threat to the integrity of the device if shared with an attacker, hence, it is vital to be able offer **strong security and assurance claims but with attestation evidence data privacy**.

➲ **Privacy-Preserving Communication and Exchange of Data**. In the communication between devices, it should be ensured that a set of conditions needs to be satisfied with regards to the privacy of the communicating parties. For example, consider the Enhanced Public Safety use case, which involves the use of IP surveillance cameras and smoke detectors, as well as PLCs to control these components, as part of a city operation center. In this case, privacy is of particularly importance, because face detection and recognition algorithms are employed by the provided surveillance services which makes the protection of the identity of the monitored parties an issue of utmost importance. The privacy properties considered will be defined in D3.1 [44], and can be summarized as follows:

    - **Anonymity:** When a device receives network data, it should be ensured that the data originates from an authenticated source, depending on the system requirements. It should be possible to perform this authentication, without divulging configuration information of the transmitting device.
    - **Unlinkability:** When a device receives network data, it should not be possible to link the received data to the source of transmission. In the aforementioned use case, this is essential in order to protect the location of the monitored parties, so that the monitored data cannot be linked with a particular IP camera.
    - **Untraceability:** In the communication between devices, it should not be possible to trace actions and details regarding the movement of the data within the network. Similarly to the previous case, untraceability is an important part of the protection of the monitored parties' location and identity.
    - **Unobservability**: It should not be possible for an external party to observe the actions of a specific device belonging to a group of devices, such as the video streams created by each camera that is part of the city infrastructure.

3.3 ADVERSARIAL MODEL AND ASSUMPTIONS

Recall the ranking of most prominent threats and vulnerabilities that has been presented in Section **Error! Reference source not found.**, which provides some insight on the threat landscape that should be considered and addressed by the ASSURED attestation services. Taking this into consideration, we provide an initial definition of the adversarial model considered in ASSURED, according to the type and domain of the considered attacks. Note that this initial definition of the adversarial model will be further expanded upon in D2.1 [63].

With the large number of interconnected devices, many adversaries are targeting Cyber-Physical Systems of Systems (CPSoS) to access sensitive information of the devices, disrupt their normal operation, and even corrupt the data and software to violate their legitimate operations. Cyber attackers are increasingly using a complex set of tactics, techniques, and

Funded by the Horizon 2020 Framework Programme of the European Union

procedures to perform sophisticated malicious actions against cyber-physical resources, such as interconnected cyber-physical devices, networks, and software updates. The ultimate goal of an attacker is to compromise devices and evade detection from the security mechanisms deployed in the CPSoS. To deal with such an expanding attack surface, CPSoS infrastructure requires beyond-state-of-the-art security mechanisms to guarantee the reliability of devices.

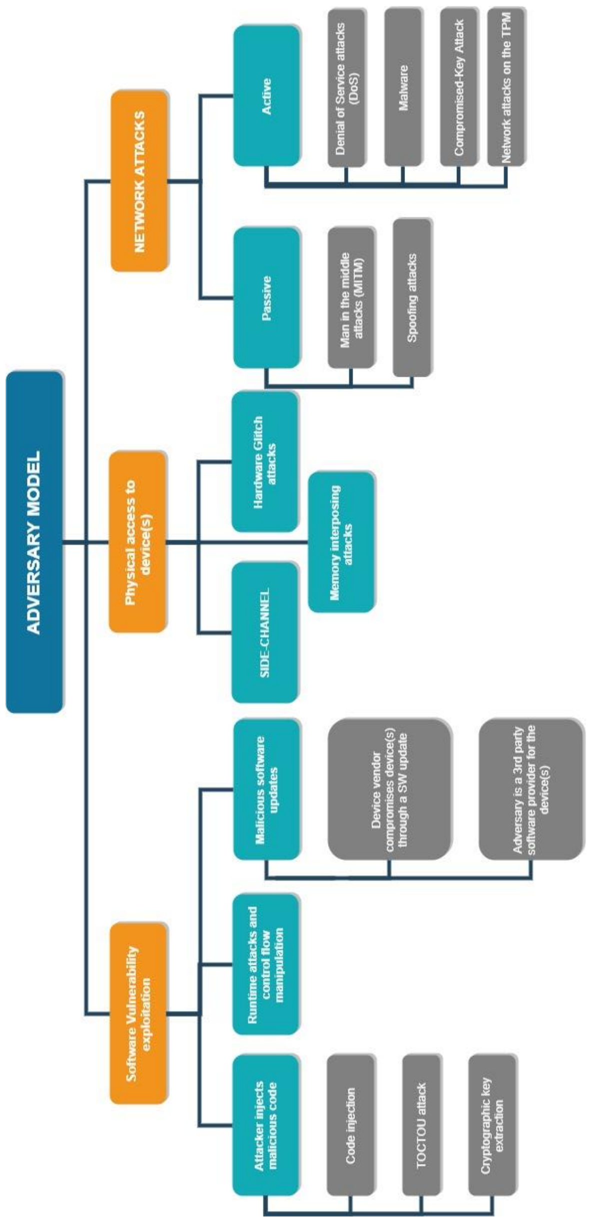
ADVERSARY MODEL
NETWORK ATTACKS
Active
Denial of Service attacks (DoS)
Malware
Compromised-Key Attack
Network attacks on the TPM
Passive
Man in the middle attacks (MITM)
Spoofing attacks
Physical access to device(s)
Hardware Glitch attacks
Memory interposing attacks
SIDE-CHANNEL
Software Vulnerability exploitation
Malicious software updates
Device vendor compromises device(s) through a SW update
Adversary is a 3rd party software provider for the device(s)
Runtime attacks and control flow manipulation
Attacker injects malicious code
Code injection
TOCTOU attack
Cryptographic key extraction

FIGURE 8: GENERIC ATTACK TREE GRAPH IN CONTEXT OF ASSURED

The main objective of adversarial modelling is to provide comprehensive and structured analysis of adversarial actions, including various factors such as entities, time, frequency, and attack stages. The analysis of this section aims to offer a holistic view of the adversarial modelling for SoS, as it considers all possible capabilities of the adversaries, while it spans across all the possible layers of SoS, considering Software and Network perspectives, and adversaries that can physically interact with systems. Such a model aims to identify the security threats, understand an adversary's goal in attacking a CPSoS infrastructure, and define the security properties that the novel security mechanisms should satisfy. In the following, we present a broad classification of the adversarial types. This classification is also in line with the adversary model described in the literature [64][65]. Note that, this section offers the holistic

Funded by the Horizon 2020 Framework Programme of the European Union

view of the adversarial model of the project and a use case-oriented documentation is the focal point of Section 4. In addition, it has to be stated that ASSURED aims to provide key attestation technologies which can protect against a finite range of attacks, leaving out of the scope some network attacks (such as jamming and eavesdropping) and physical attacks.

In **Error! Reference source not found.**, we summarize the main goals of an adversary along with the adversarial actions. The classification presented in the figure is reflected in the following respective subsections.

Before we proceed to the documentation of the attacks, we need to highlight the special case of privilege escalation attacks that could be the aftermath of any kind of attack that can grant access to a targeted system. More specifically, an adversary is possible to infiltrate to a system by exploiting a weakness or vulnerability of the untrusted stack of the system's architecture and then, penetrate further into the system by acquiring root privileges (escalation). Such an offensive action will enable the attacker to interact with the components of the TCB of the system through the manipulation of privileged processes that aim to interact with the trusted world of the system. ASSURED will be in position to defend against privilege escalation attacks that aim to undermine the operation of the critical components of the TCB (e.g., the tracer). More specifically, as described in Section and presented in Figure 7, in the context of ASSURED we will define those processes which will be responsible for bridging the normal operating system (the untrusted world) with the TCB. This definition will set specific limits and narrow down the scope of the monitoring process of ASSURED to only those processes which are allowed to interact with the TCB. Thus, through runtime attestation, ASSURED will be in position to detect prohibited interactions with TCB or to detect discrepancies in the operational profile of legitimate interactions, as a result of privilege escalation attacks.

3.3.1 Software-related Attacks

A software adversary is a remote malicious entity that disrupts the regular operation of a secure system by infecting it with malware. Typically, such an adversary violates the confidentiality and the integrity of the system. The most prominent types of attacks that have been identified are enumerated in the following sections.

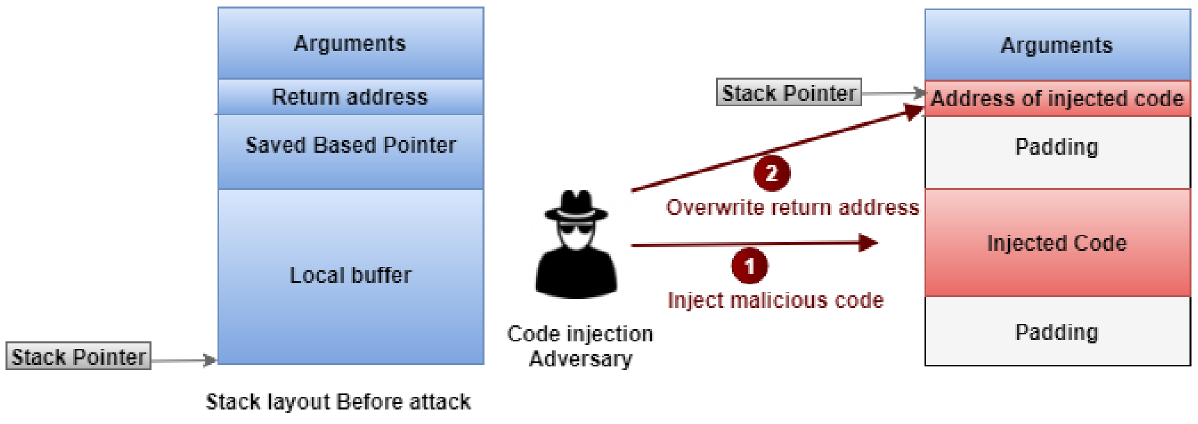

FIGURE 9: CODE INJECTION ATTACK REPRESENTATION

Code injection: A code injection attacker introduces and executes arbitrary code into the address space of a vulnerable application. In order to achieve this, the attacker exploits the missing memory bound check and forces the application code to exceed the memory buffer boundaries. To perform the code injection attack, an adversary typically sends to the target application the malicious code along with the input data. When the application does not validate the input length, the adversary sends larger input data than expected to overwrite the content of the stack above the local buffer, as shown in **Error! Reference source not found.**. The

Funded by the Horizon 2020 Framework Programme of the European Union

adversary sends input data, the malicious code, some padding that fills the rest of the buffer, and a code pointer. This input overflows the local buffer and overwrites the return address to point to the beginning of the malicious code. Thus the application reads the malicious address and executes the malicious code.

ASSURED will provide the means to defend against code injection attacks either by static or runtime attestation mechanisms. In case a code injection attack modifies the signature of the attacked application, this attack vector will be captured by static attestation mechanisms. On the other hand, if the attack does not modify the signature of the used binaries, the modified control flow of the program's execution will be captured by the runtime attestation mechanisms of ASSURED.

Cryptographic key extraction: This kind of attack considers an adversary in position of exfiltrating and locating cryptographic keys from the run-time environment of software-based services even when their software layout and data structures in memory are unknown.

Key secrecy is required in all stages of the management of its life cycle i.e., during creation, dissemination, storage, and usage. In fact, computing systems inherently require that any program, including its data and instructions, be loaded into the main memory before being run by the processor. Thus, while keys can be protected while stored [66], any conventional program that performs keyed cryptographic operations must, at some point, have the key material exposed [67] (known as the Key-Exposure Problem, KEP). In this context, adversaries are able to infiltrate microcontrollers [68] and to exfiltrate cryptographic keys, during system operation systematically, without affecting system operation. The authors in [69] exploited the causality between transceiver invocation and utilization of keyed cryptosystems to acquire timely memory extractions. In other words, since keyed cryptosystems inherently run post-reception (e.g., to verify or decrypt an incoming payload), it becomes feasible, by periodically exfiltrating conventionally used memory regions for storing run-time data (e.g., the memory stack), to capture data belonging to the keyed cryptographic function during the inevitable Key Exposure Window (KEW) caused by the KEP. In this way, an adversary could exfiltrate the cryptographic key residing in the memory.

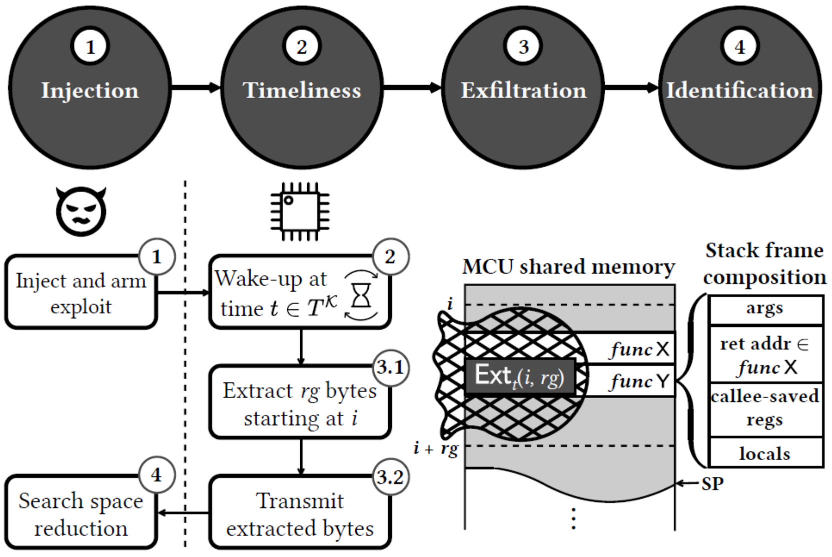

FIGURE 10: THE FOUR FUNDAMENTAL PHASES OF KEY ACQUISITION [69]

The runtime attestation offerings of ASSURED can form a defensive mechanism against this kind of attack. More specifically, using **Control Flow Attestation (CFA)** it is possible to identify those processes which are leveraged by the adversary to exfiltrate the required data from the

Funded by the Horizon 2020 Framework Programme of the European Union

memory region of the system. In addition, **Configuration Integrity Verification (CIV)** can also be used in order to mitigate attacks against cryptographic key extraction and to attest the correctness of the creation of a cryptographic key. For example, when a key generation function is executed, CIV can be used in order to check whether the cryptographic key has not been manipulated, by verifying that the hash of the key is equal to an already known and trusted hashed key value.

Time-Of-Check-Time-Of-Use (TOCTOU) Attacks: Remote Attestation (RA) techniques verify the remote device's state at the time when remote attestation functionality is executed, thus providing no information about the device's state before current RA execution or between consecutive RA executions. This implies that presence of transient malware may be undetected. In other words, if transient malware infects a device, performs its nefarious tasks, and leaves before the next attestation, its temporary presence will not be detected. This important problem, called Time-Of-Check-Time-Of-Use (TOCTOU), is well-known in the research literature [70][71] and poses major challenges in the context of remote attestation.

Unfortunately, current remote attestation architectures share a common limitation: they only measure a Prover's state at the time when remote attestation code is executed by the Prover. They provide no information about the Prover's state before attestation and execution or its state between two consecutive attestation executions.

ASSURED, in alignment with recent endeavours in the field [72], will aim to develop novel attestation mechanisms able to resist against TOCTOU attacks, deploying attestation techniques that can capture even the ephemeral presence of offensive actions that aim to evade the legitimate operational behaviour of systems. The new attestation techniques will be based on approaches that guarantee the freshness of attestation while Blockchain will be used to ensure that even ephemeral execution deviations can be captured by the attestation mechanisms.

Runtime attacks and control flow manipulation: Data Execution Prevention (DEP) [73] is a security feature within an operating system that enforces Writable⊕Executable (W⊕X) memory policy to prevent applications from executing code running on non-executable memory. Consequently, attackers cannot inject shellcode into target process's memory space on the fly. Although DEP helps in mitigating code injection attacks, it does not prevent code-reuse attacks where an attacker reuses and combines existing sequences of legitimate code already loaded on the memory. Figure 11 illustrates a code reuse attack that subverts the target program's execution control flow without injecting any new code. The target program is supposed to invoke either the *privileged* function or the *unprivileged* function, depending on the value of variable *auth*. However, an attacker may exploit a vulnerability found in the *unprivileged* function to overwrite the stored return address with the starting address of the *privileged* function. In this way, the program control flow is altered during run-time and a privileged function is erroneously executed.

1. **return-to-libc** [74]**.** Return-to-libc is a buffer overflow exploitation on a system that has enabled stack execution protection. When Data Execution Protection (DEP) is enabled, a standard buffer overflow attack does not work because DEP prevents the injection of arbitrary code into a process's address. To bypass DEP, a return-to-libc attack reuses existing code that already presents in target process's executable memory. After exploiting a buffer overflow vulnerability, the attacker modifies the return address stored in the stack to point to a function in standard *libc* library, thus realize a fake libc function call.

2. **ROP attacks** [75]**.** Unlike traditional return-to-libc attacks that utilize function calls, Return-Oriented Programming (ROP) attacks hijack program control-flow execution by stitching multiple ROP gadgets together, i.e., short instruction sequences ending with

Funded by the Horizon 2020 Framework Programme of the European Union

ret. Attacker needs to carefully search process memory for potential gadgets, either reusing the existing instruction sequence, or deliberately misinterpreting code bytes, e.g., from different starting points. The expressiveness of ROP attack is proven to be Turing-complete when the code space is large enough. In other words, the attacker is able to perform arbitrary computation.

3. **JOP attacks** [76][77]. Besides backward instruction like *ret*, forward instructions like indirect call and jump also allow attackers stealthily control the program execution and bypass defence techniques that harden the integrity of stack frames. Attackers search useful gadgets in memory that ends with forward instructions and chain them together with the aid of a special gadget called dispatcher to perform desired computation.

4. Apart from code reuse, runtime attacks are also an effective approach to extract information. A number of randomization-based schemes, such as address space layout randomization (ASLR), have emerged to mitigate code reuse attacks. The idea behind this kind of defences is to randomize the memory layout, thus make it challenging for attackers to reference gadgets. To overcome these defences, a new attack strategy, just-in-time ROP (JIT-ROP) [78], is introduced to reveal the address of code modules and generate ROP exploits on-the-fly. JIT-ROP starts with exploit the memory disclosure vulnerability to acquire a single runtime memory address, which reveals the content of corresponding code page. By recursively searching for pointers to other code pages, JIT-ROP cumulatively discover more code pages until enough gadgets are found. Another example of information leakage attack is Heartbleed attack [79] that is caused by a memory disclosure vulnerability found in OpenSSL and leads to the compromise of secret keys and confidential user data.

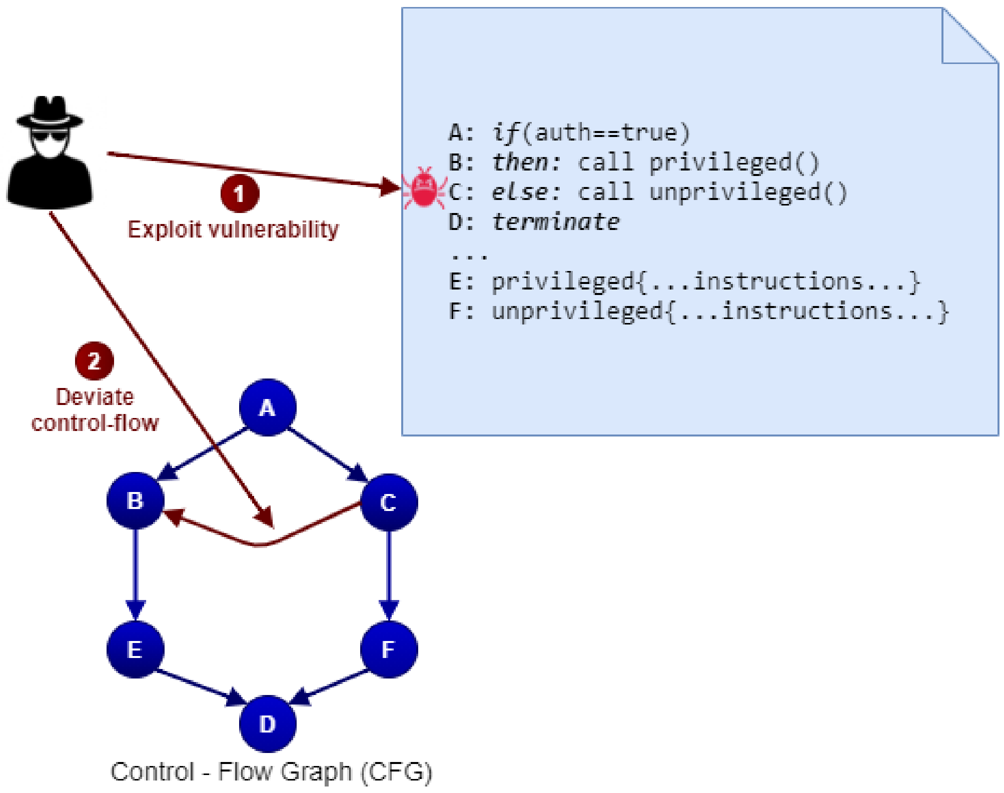
```
A: if(auth==true)
B: then: call privileged()
C: else: call unprivileged()
D: terminate
...
E: privileged{...instructions...}
F: unprivileged{...instructions...}
```

FIGURE 11: RUNTIME ATTACKS MANIPUATING PROGRAM CONTROL FLOW

Funded by the Horizon 2020 Framework Programme of the European Union

5. ***Non-control data attacks*** [80][81]: Most runtime attacks modify the target program's control data, such as return addresses, function pointers, and indirect branch targets, which leads to an execution path deviated from the original one and, therefore, can be detected by Control Flow Integrity (CFI) defences. However, non-control-data attacks may easily bypass these defences since this kind of attack has no explicit impact on program execution path. Taken the memory bit-flip as an example, flipping one bit of a security-critical non-control data, such as configuration data or user identity data, can lead to the corruption of access control mechanism, privilege escalation, or omitting variable initializations. The expressiveness of non-control data attacks is also proven in previous work. However, non-control data attacks are harder to construct than control flow attacks.

The aforementioned attack and its variations will be addressed by ASSURED using the **Control Flow Attestation (CFA)** mechanism, which can capture deviations on the execution profile of processes. Even though the attacker does not inject malicious code to subvert the system, the deviation of the execution flow of the application will be reflected in the captured traces and the attestation outcome.

Malicious software updates [82]: In the context of this attack, an adversary uses deceptive methods to fool a user or force an automated process to download and install dangerous code believed to be a valid update, but the update originates from a malicious and controlled source. The attack may look like a regular upgrade of update process of core application provided by the vendor of the system (or device) or by a 3rd party software provider. Although there are several variations to this strategy of attack, the adversary aims to position and disguise malicious content such that it masquerades as a legitimate software update which is then processed by a program, undermining application integrity. The latitude of this attack is immense as virtually all software requires frequent updates or patches, giving the attacker several opportunities to attack a system.

Adversaries usually perform phishing-assisted variations on this attack that involves hosting what appears to be a software update web site and then sending spam, phishing, or spear-phishing emails to the organization's users requesting that they manually download and install the malicious software update. Automated attacks involving malicious software updates require little to no user-directed activity and are therefore advantageous because they avoid the complex preliminary setup stages of manual attacks.

ASSURED will offer the technical means to defend against such kind of malicious software updates utilising both configuration integrity verification and runtime attestation. The ASSURED attestation mechanisms will attest the application binaries in order to guarantee that there are no malicious updates and the signature of the used binaries and configurations of the system reflect the correct operational status. In addition, through control flow attestation, ASSURED will be able to detect deviations in the execution profile of an application that might have been maliciously updated.

3.3.2 Network-related Attacks

Network attacks are unauthorized actions on the private channels within an organizational network. Malicious parties usually perform network attacks to alter, destroy, steal sensitive data or gain access to internal systems [83]. ASSURED Network security is therefore required to control the access which can be carried out by these network attacks. We will discuss some common network attacks to understand and to propose solutions to develop a network security that can give uninterrupted and secure services to ASSURED use cases to protect ASSURED networks and exchanged data that are vulnerable to different kinds of the network attacks. Network attacks are mainly classified into two types:

- **Passive:** when sensitive information is screened and monitored, potentially compromising the security of an organization.

- **Active:** when information is altered by a hacker or destroyed entirely.

Some of the most prominent network-oriented attacks that can threaten the Cyber-Physical Systems of Systems are the followings:

Man-in-the-Middle Attack: It is a type of network attacks where the attacker secretly relays and alters the communications between two parties communicating with each other. The attacker makes independent connections with the victims and relays messages between them and finally convince them that they are directly talking to each other over a private and secure channel, when in fact the entire conversation is controlled by the attacker. In ASSURED, secure communication between devices will be supported by the TPM. TPM will provide the cryptographic primitives to enable secure and authenticated communication and defend against MITM attacks.

Malware**:** Malware is malicious software such as viruses, ransomware and spyware, which consists of code developed by cyber attackers, designed to cause significant damage to the systems or can lead to unauthorized access to a network or a computer system. Malware can be transformed exponentially rapidly between connected devices, once a device becomes infected, it can connect to other devices via the internet and seeks access to more devices. ASSURED attestation mechanism will be used in order to detect nefarious interactions of the malware with safety critical process of the devices.

Denial of Service (DoS): A DoS Attack stops legitimate users from using a network, server, a service, or other resources. The attacker seeks to make a machine or network resource unavailable to its intended devices by interrupting the devices connected to the network. Denial of service is typically accomplished by flooding the targeted device with extra unneeded requests to overload the system and prevent some or all legitimate requests from being fulfilled [84]. The protection against DoS attacks is out of the scope of ASSURED.

Compromised-Key Attack: When an attacker obtains a user password that represents a secret key, this key is then considered a corrupted key. An attacker uses the affected/compromised key to gain access to secure communication without the attack being detected by the sender or recipient. The attacker may decrypt or alter the information by using the compromised key to generate additional keys to give the attacker access to any other secure communications. In the context of ASSURED, the management of the lifecycle of the cryptographic keys will be connected to the underlined Root of Trust to be used. Thus, the functionalities of ASSURED will be based on “proof of possession” for the keys generated by the underlined Root of Trust.

Spoofing Attacks [85]: Spoofing is the act of impersonating a communication or an identity so that it appears to be established with a trusted, authorised source. Some common spoofing attacks are email spoofing attacks that are used in phishing targets, and caller ID spoofing attacks that are often used to perform fraud. Attackers may also target more technical elements of an organization’s network, such as an IP address, domain name system (DNS) server, or Address Resolution Protocol (ARP) service, as part of a spoofing attack. In the context of the communication protocol and services of ASSURED, the identity of devices will be linked with the TPM. In this way, spoofing and impersonation attacks will be addressed by the intrinsic qualities of the TPM.

Network attacks on the TPM: The presence of the host with such physical proximity to the TPM, and the limited ability for the TPM to authenticate itself directly to a verifier, i.e., the TPM can only respond to the host’s commands, makes the secure and authentic communications

Funded by the Horizon 2020 Framework Programme of the European Union

between the TPM and an external verifier very challenging. The host, namely the TPM Software Stack (TSS), can be an active man-in-the-middle attacker, i.e., can read the messages between the TPM and an external verifier, delay or block the communication, modify the messages, or append messages. It can also coordinate a group of compromised TPMs to get credentials on compromised devices. The host can block the communication between the TPM and an external entity causing a denial-of-service attack that makes a machine or network resource unavailable to its intended users, by temporarily or indefinitely disrupting services of a platform connected to the Internet. The host may delete some message in the communication between the TPM and the verifier, such as in black hole attack when the host deletes some messages that are supposed to be forwarded either to the TPM or to an external verifier. The host can coordinate a set of malicious (compromised) TPMs to get credentials from a certificate authority on compromised devices, such as spoofing attack in which a program successfully falsify data, to gain an illegitimate advantage. A host may also change the destination of the data sent from a TPM to an external verifier or vice versa. For instance, a host may send some data that is supposed to be forwarded to a legitimate TPM to a compromised TPM instead. A host may also abuse privacy in anonymous signatures, such as Direct Anonymous Attestation (DAA) signed by the TPM, by simply disclosing the TPM identity.

Network attacks against the TPM will be partially addressed by the ASSURED artifacts. In fact, through the use of policies and sessions we will be in a position to regulate the interaction of the host with the TPM and the TCB. More specifically, in the context of D3.1 [44], we will define the trust models, requirements and assumption that will define the interactions between the host and the TPM. In this way, we will limit and concretely define the interactions and the conditions under which the communication with the TPM takes place.

3.3.3 Network attestation

Decentralization of communications and processing at the network edge have become over the recent years core design features of future secure networks as previously detailed. These changes come in effect as ever increasing rate/bandwidth requirements and diminishing latency, safety and security constraints rapidly pushed the limits of conventional, centralized architectures based on few, large, remote critical infrastructure in favor of those relying upon many, smaller, distributed radio, and network processing instances. Such measures not only benefit of better QoS due to closer range links given increased deployment density, but also provide cheaper, secure and more energy-efficient access to new revenue opportunities for MNOs and CPSs alike.

NFV is among the key technologies that enable such latter solutions, at the same time also opening the network architecture to third parties as well as providing security guaranties from a secure containerization or network slicing point of view. As a result, Virtual Network Functions (VNFs) and traditional Virtual Machine (VM)-based solutions are some of the key interoperable (standardized for instance under ETSI NFV Working Group) blocks for network security mechanisms alongside the Network Orchestrator (NO) [86][87] that could adapt attestation mechanisms to their functionality. Network Slicing is yet another virtualization mechanism enabling end-to-end secure slices physically running on the same infrastructure but satisfying different policies and constraints [88]. These not only enable full-depth logical network virtual separation of, network resources, but also compute and storage functionality. Embracing the whole framework from a security standpoint, a dedicated prover entity originated from ASSURED manages the several network slices, by assigning and monitoring them given the requests coming from different agents-roles (i.e., operators –end-users or companies providing their own services over-the-top) [89]. ASSURED will take however the secure emerging Edge Computing trend of networking to a new level and proposes a distributed security architecture based on attestation that can be rolled on commodity infrastructure (e.g., manufacturing, transport, surveillance devices and satellites) capable of serving a fully secure network under the high standards of next-generation applications and services. As portrayed earlier, such

Funded by the Horizon 2020 Framework Programme of the European Union

infrastructure not only provides a decentralized efficient approach to communications, but by its ubiquity is capable to support critical deployments and scenarios such as critical mission support, on-demand network secure self-configuration or attack-tolerant networking. To provision all the above, the network itself not only needs to be highly interoperable and virtualized, but also requires a high degree of autonomous intelligence under certain security policies.

Policy based secure networking is not a new concept [90], but to this point this has been usually based on link metrics and qualifiers such as secure QoS, available rate, bandwidth, expected latency etc. However, in the era of speed and increased mobility such qualifiers are highly dynamic and typical statistically driven policies tend to become obsolete and inefficient, yet aside their challenges to security requirements at the network level. The ambition of ASSURED is to utilize a secure ambient, context, connection and situational information distributed among the communication nodes running the attestation service in order to predict upcoming security-critical network behaviour and optimize the securely the network resources for increased coverage, allocation of bandwidth, while respecting security requirements.

3.3.4 Physical Attacks

This section considers an adversary with physical access to the hardware. The adversary attempts to violate the confidentiality and the integrity of the system. It must be stated that such physical attacks are out of scope of ASSURED. However, for purposes of completeness we elaborate on state-of-the-art attacks for such an adversarial model. Hardware-oriented attacks are left out of the scope of ASSURED as all the attacks require physical presence, which is hard to achieve in ASSURED because multiple devices work in tandem to serve requests or to support the operational goal of the use case deployments. In addition, these attacks are not hardware/software agnostic, i.e., the adversary tailors the attack for a specific device and its configuration. For example, the power signal measured during a Differential Power Analysis (DPA) attack is unique to a program's control flow and the hardware that it is executed on. In ASSURED, we consider heterogeneous devices, which makes mounting such attacks far more challenging. Next, we describe the state-of-the-art attacks possible for such an adversary in the context of ASSURED.

Side-channel attacks [91][92][93]. A prominent example of a side-channel attack is power analysis and particularly DPA, in which the adversary tracks the power consumption of a hardware device. The adversary relies on the fact that changes in voltages within the device reflect changes in the performed functionality. Therefore, by measuring current changes, the adversary learns a small amount of information about the manipulated data, violating the confidentiality security property.

Hardware Glitch attacks [93][94]. An adversary may mount hardware glitch attacks, for example, by glitching the voltage to the device. If timed correctly, this could affect the device's core functionality or even cause a change for fetching or evicting data to the main memory.

Memory interposing attacks [95]. An attacker may install an interposer between the DRAM and the DIMM socket before system boot. The interposer may then act as a man-in-the-middle for requests and responses to the main memory. It can either snoop memory accesses by duplicating the command bus signals and sends them both to the DRAM and a dedicated signal analyser. Alternatively, the interposer could completely change the requests made to the main memory, e.g., to read or write different values.

Funded by the Horizon 2020 Framework Programme of the European Union

4 SYSTEM PROPERTIES TO BE ATTESTED

The aim of this section is to flesh out the properties to be attested in context of ASSURED, which represent the trust level of a system. Specifically, a set of functional specification and validation properties that define the resources that need to be attested has already been documented in Section 3.2, in the form of system specifications and assumptions, functional safety properties, security properties and privacy-preserving properties. The successful attestation of these properties can be used in order to provide verifiable evidence towards the trustworthiness status of an entire system.

Taking the above into consideration, this Chapter is dedicated to providing definitions of all the Validation System Properties that should be considered in the context of attestation, in order to protect against the attacks that were presented in Section 3.3. This constitutes a first step towards mapping these properties to the particular system architecture and requirements of the use cases presented in Chapter **Error! Reference source not found.**.

4.1 VALIDATION PROPERTIES

Chapter 3 introduces the capabilities and "as-is" properties of the underlying system, as well as the adversary model of ASSURED, which is used in order to define the trust anchor of the ASSURED attestation services, the potential threats, and the security objectives that should be achieved. Compounding the output of Section 3.1 and the state-of-the-art analysis on attestation schemes conducted in D1.1 [96], we will extract a set of configuration and execution properties to be attested, that enables the operational, privacy, and security protection against attacks specified in our adversary model. For each use case scenario, depending on the potential threats and protection goals, a set of properties can be validated in ASSURED attestation schemes to fulfil the operational, security, and privacy requirements.

Based on the type of captured information, attestation protocols can be divided into two general classes: static attestation and dynamic attestation, also referred to as runtime attestation. Similarly, we divide the validation properties, i.e., the properties to be validated in attestation protocols, into static properties and dynamic properties. Static properties include some low-level concrete properties, such as the firmware version, binary signature, and the presence of specific hardware properties, which reflect the correct configuration of software and hardware deployed on devices as part of the supply chain ecosystem. Many attacks start with injecting malware or altering security configuration to gain unauthorized access or privilege in victim systems, such as static code injection and malicious software update attacks. As an example, these attacks, however, cause modified binary signature, violating the static property, ASSURED is able to tackle these attacks by validating static properties. Dynamic properties instead measure the correctness of operations and tasks executed in edge devices deployed in the supply chain ecosystem. Advanced attacks intend to perform adversarial computation by modifying the execution flow/control flow of legitimate software, so that classical defences based on static properties cannot detect and mitigate them. In this case, dynamic properties like control flow and data flow information play an important role, for instance mitigating various runtime attacks. Note that static properties are sometimes also needed in runtime attestation protocols as a measurement of trust in the correct state of the underlying devices. We will present a technical introduction of these validation properties in the rest of this section.

Prover captures the required information, as specified by the validation properties, statically or during runtime, and then generate an attestation report to be sent to verifier. The report may consist of captured states itself, which may be the memory snapshot, a list of taken branches, or a hash of binary. Alternatively, the prover can validate the captured state and only report the check result to verifier. The latter option minimises the verifier's workload and network

Funded by the Horizon 2020 Framework Programme of the European Union

traffic, but needs to be implemented under more strict assumption, namely the prover is unable to interfere validation process and forge check result.

Note that, the dependency between components needs to be considered as well when analyzing the system and determining validation properties. For instance, an application to be attested relies on an OS service. Even though the application passes the validation, an attacker can stealthily interfere in the application through the compromised OS service. Hence, each use case partner needs to meticulously analyze the dependency between components and define the TCB. Dependent components should either defined as part of the TCB, or they should be explicitly attested so that their trustworthiness is ensured.

4.1.1 Static Properties

Static properties refer to properties that indicate the static state or non-execution behaviours of systems, e.g., unmodified binaries, legitimate hardware components, or device configuration. These properties can be captured at any time during the operational lifecycle, prior to runtime, during, or after runtime. As the name suggests, static properties remain unchanged during a long timeframe. Therefore, there is no strict restriction to the capture the timing and duration of static properties. Besides, the size of the captured information is much smaller compared to the dynamic properties, since the static properties do not change frequently.

Multiple attacks included in the ASSURED adversary model can be mitigated by validating static binary, such as static code injection, malicious software update, and malware. Thanks to their time-invariant nature, static properties are generally easier to collect, report, and validate, but also limits its capability as static properties cannot disclose adversarial behaviours occurred during runtime.

In context of ASSURED, we measure the static state of edge devices deployed in supply chain ecosystem with the following properties.

Static binaries, including program code and static data, should be verified before being loading into the memory. Injection of malicious binaries through existing vulnerabilities on a system targeted by an attacker is a common method for malicious parties to steal confidential data, perform adversarial operations, or obtain unauthorized privilege. Therefore, we should assure that the program binary is provided by authorized partners. Also, it should be ensured that the binaries, the code, and the static data are unmodified, and the program version is up-to-date and not vulnerable, especially for the binary loaded into TCB, e.g., the tracer.

Dynamically loaded libraries, which can be utilized by attackers to evade static binary validation, need to be attested as well. Nowadays programs increasingly rely on dynamically loaded libraries in order to reduce the binary size and maximize code reuse. These libraries are linked to programs during runtime and are shared among multiple programs simultaneously, which avoids code base duplications, but also increases the attack vector. In order to mitigate library-based attacks, we should assure that the libraries are properly measured before being used by other programs. It should also be ensured that the version of libraries is up-to-date and not vulnerable.

Hardware components are also a crucial static property, especially for those hardware components served as the system trust anchor, such as the TPM, which is the cornerstone of device authentication and establishment of the trust chain. Otherwise, an attacker can easily bypass or compromise attestation schemes or secure communication within the Blockchain network by using an incorrect TPM or extracting secret credential from the TPM. Therefore, we should also attest that the installed hardware components are legitimate, for example via the serial numbers, and the firmware and secret credentials are valid and uncompromised.

Funded by the Horizon 2020
Framework Programme of the European Union

4.1.2 Dynamic Properties

In contrast to static properties, dynamic properties reflect the execution behaviour of prover devices and are able to disclose runtime attacks, thus attestation of dynamic properties requires to record program behaviours during runtime, for example the taken indirect branches, and memory accesses. Dynamic properties may drastically change over time, even in a benign case. The key challenge is to differentiate benign changes with malicious changes. One common approach is based on program Control Flow Graph (CFG) that specifies the legitimate execution paths of a program. Any execution behaviour that does not exist in CFG is considered to be incorrect.

Dynamic properties can make up for the deficiency of static properties, but they are harder to attest. When implementing an attestation protocol for dynamic properties, the way to capture and represent information plays an important role. Unlike static properties, capturing dynamic properties always incur a large number of recorded events, especially for complex software with long running times. To enable efficient verification, we need to carefully select dynamic properties based on the adversary model, and astutely design the representation form of captured data. Besides, TOCTOU attacks may also compromise the attestation of dynamic properties. In other word, an attacker deliberately chooses to perform malicious operations between two attestations.

Based on the state-of-art runtime attestation schemes and use case scenarios in context of ASSURED, we focus on the following dynamic properties, which enable the supply chain ecosystem protected by ASSURED to mitigate various attacks modifying the control flow and data flow of program, including code injection attacks, sensitive data extraction, code reuse attacks, non-control data attacks, as well as malicious software attacks.

Control flow information (CFI) is a dynamic property widely used in various attestation and detection techniques that aim to mitigate runtime attacks, such as runtime code injection attacks and code reuse attacks. Control flow information contains the addresses of code sections that are executed during runtime. If an attacker exploits the vulnerability to execute malicious payloads, either by executing injected code or by reusing existing code, the action will be recorded in the control flow report and give the verifier the chance to discover the attack. Usually, control flow information is represented in the form of a list of indirect branches, e.g., function calls, function return addresses, indirect jumps, whose destination addresses are calculated during runtime. However, as the size of program increases, the size of the control flow log increases dramatically, which poses a challenge on how to properly represent the control flow information and complicates the validation process. An attestation scheme can tackle this issue by restricting the size of program to be attested, or by using hashes to decrease the size of control flow log. Another option is to enforce certain policies, such as a CFI policy, on prover device and only report to verifier whether all control flow branches comply the CFI policy.

Data flow information (DFI) tracks the data stream within a program, including variable definitions, usages, and data dependencies between instructions and variables. For instance, data flow reveals whether the initial value of a variable originates from an untrusted source, and how a malicious input taints other instructions or variables. With the aid of data flow information, the attestation scheme may be able to detect non-control-data attacks that corrupt memory access operation by loading and storing instructions, without causing any unintended anomalies in the control flow [97]. Data flow information can be directly recorded and reported to verifier, for instance, in form of a list of memory access instructions, or checked on the prover side and only report the check result to verifier.

The correctness of program execution depends on multiple factors, such as the binary code, input data, relevant executables, and data objects [98]. Report containing relevant information can increase the chances of discovering runtime attacks. In context of ASSURED, we mainly consider the following relevant properties: **code and data measurement during runtime** aiming at disclosing runtime injecting attacks, **configuration files** fed to attested programs during runtime, **data exchanges with other entities** such as interactive input during runtime, as well as **system-level events** such as the device's software update history and the occurrence of system reboots.

Funded by the Horizon 2020 Framework Programme of the European Union

5 APPLICATION USE CASES

The aim of this section is to create the vocabulary of the system validation properties that need to be considered during attestation in the context of the envisioned use cases so as to be able to provide the necessary security claims. The set of validation properties that represents the trust status of system has been documented in Section 4.1, and will serve as a reference. Based on this, we elaborate on the instantiation of the four envisioned use cases of ASSURED and we showcase how the systems of these use cases will be protected against attacks by attesting configuration and execution properties. This output will help the **design and implementation of ASSURED attestation schemes** in D3.1 [44], as we figure out the adversary types, what kind of properties need to be attested to ensure the operational and security posture of supply chain ecosystem, and what kind of attestation schemes are necessary for the ASSURED framework. In addition, ASSURED attestation schemes need to be integrated into the use case demonstrators. This section will form the basis for the integration process.

In what follows, we apply the models defined in previous sections to the four use cases of ASSURED in order to identify the crucial services and attack scenarios that are most likely to have a severe impact on the trustworthiness of the system. Table 2, Table 4, Table 6 and Table 8 elaborate on the most critical attack scenarios for the four use cases, including details regarding the execution of these attacks, the affected use case components and services, the impact of each attack, and what type of attestation scheme can be used to mitigate each attack. In this direction, each attack scenario is further analyzed, by outlining the properties that are affected by each kind of attack, providing concrete descriptions of the properties in the context of each use case, and information on how to attest those properties. This is documented in Table 3, Table 5, Table 7, and Table 9.

5.1 SAFE HUMAN ROBOT INTERACTION (HRI) IN AUTOMATED ASSEMBLY LINES

The “Smart Manufacturing” Demonstrator located in Bremen, Germany, provides an infrastructure for Human Robot Interaction. The demonstrator for this use case is located in the demonstrator hall within the premises of BIBA and consists of three major components:

- **Ultra-Wide Band Wireless Location System**
- **Industrial Robotic Arms**
- **IoTGateway**

Figure 12 illustrates the major components that are deployed in the Smart Manufacturing scenario illustrating Human Robot Interaction. The Ultra-Wide Band Wireless Location System transfers the location information collected by the wireless tags to an MQTT broker that runs on a dedicated hardware, referred to as the Data Aggregator. The Robotic Arms are connected and controlled by **Programmable Logic Controllers (PLC)** over the PROFINET network and can only be accessed by an OPC-UA Server running on the Industrial PC. The IoTGateway is a flexible edge device that exists on the same network as that of the Industrial PC, and acquires live data from both the Ultra-Wide band wireless location system as well as the robotic arm system. The core function of the IoTGateway is to process the collected data with collision detection algorithms to avoid the collision between moving personnel or human workers, and a robotic arm located in the workspace. Thus, the operational assurance of IoTGateway plays an important role in the physical safety of the personnel in the context of the smart manufacturing use case.

Funded by the Horizon 2020 Framework Programme of the European Union

IoTGateway consists of distinct micro-services in the form of containerized software with dedicated software clients. We refer to the services needed to control the OT (Operations Technology) part as **South-Bound services**, which may include the services that control the movement of the robotic arms. Moreover, the IoTGateway offers **North-Bound services** for communication with private and public clouds, which enable access from external stakeholders. Note that IoTGateway supports secure communication channels, such as TLS channels, with South-Bound and North-Bound services.

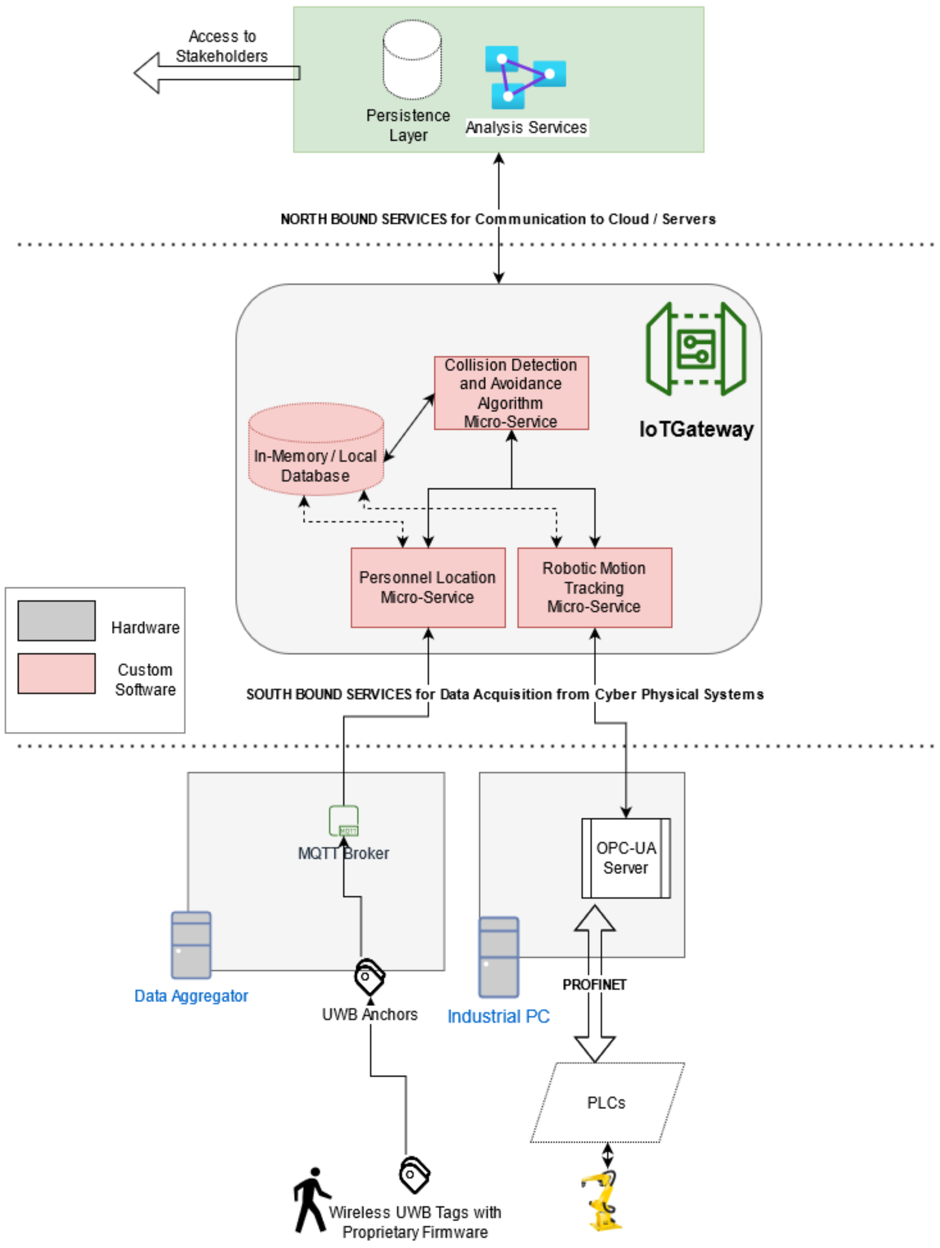
Access to Stakeholders
Persistence Layer
Analysis Services
NORTH BOUND SERVICES for Communication to Cloud / Servers
Collision Detection and Avoidance Algorithm Micro-Service
IoTGateway
In-Memory / Local Database
Personnel Location Micro-Service
Robotic Motion Tracking Micro-Service
Hardware
Custom Software
SOUTH BOUND SERVICES for Data Acquisition from Cyber Physical Systems
MQTT Broker
OPC-UA Server
Data Aggregator
UWB Anchors
Industrial PC
PROFINET
PLCs
Wireless UWB Tags with Proprietary Firmware

FIGURE 12: MAJOR COMPONENTS FOR SMART MANUFACTURING

5.1.1 Protection Goals and Attack Settings

The aforementioned system components contain a wide variety of services. An unidentified vulnerability can be exploited by an attacker in various ways, which may lead to severe impact on the integrity of the system. Figure 13 depicts the attack graph tree for the smart manufacturing scenario, which contains various potential adversaries.

Funded by the Horizon 2020 Framework Programme of the European Union

Software Attacks target any edge component of the system or network infrastructure that is able to execute code to control or collect information from connected sensors, actuators or other cyber-physical systems through dedicated APIs or network protocols, such as MQTT, OPC-UA, CoAP etc. In the smart manufacturing case, the IoTGateway plays a significant role in the collection of data, as well as in the execution of custom algorithms that are necessary for processing decisions made by applications at the edge level, such as sending control signals to the Robotic Arms. **Code Injection** adversaries aim to disrupt the normal execution of such algorithms and data collection firmware by injecting malicious code on the IoTGateway and causing safety hazards for personnel working in the workspace. Similarly, an adversary may inject malicious code into the data acquisition program to provide false information to the system, leading to false recognition of the control algorithms and the disruption of data integrity of the workspace. In case code injection is prohibited by some security features like DEP, the attacker can **manipulate the control flow** of collision detection and data acquisition programs and produce malicious behavior, by changing the execution order of the existing legitimate code loaded on the device's memory. One group of software attacks is **key extraction**, where an attacker exploits memory vulnerabilities to extract secret keys from edge devices, such as TLS keys.

Software Update Attacks involve tricking the system administrator into installing a malicious firmware update, or forcing an automated process to do so, in order to disrupt the Operations Technology part of the workspace. This may cause large downtimes, malfunctions in the robotic arms, or replacement of the services running on the IoTGateway with malicious code. One common method to execute such an attack would be to gain access to the IoTGateway and adapt the configuration of running micro-services on them. A third-party vendor may provide a malicious firmware update as well.

Network Attacks are commonly categorized into **passive** and **active attacks**. The former category refers to an attacker who passively screens and monitors the network data, which is performed by the Ultra-Wide Band wireless tags in the workspace. An external adversary however requires the introduction of a device into the physical workspace that can listen to network packets by eavesdropping on the network channels, as well as spoofing a device on the wireless network to inject false data on the UWB location system. The latter category refers to attackers who actively alter network communication, and can perform DoS attacks, spread malware over connected devices, gain access to secure communication channel with compromised keys, and perform network attacks against underlined TPM.

Physical Attacks require an adversary to have knowledge of the UWB location system and can introduce an UWB tag that can inject malicious data into the system or can extract a device's key from the tags. Common physical attacks include **side channel attacks**, **memory interposing attacks**, and **hardware glitch attacks**. Comparing with software attacks and network attacks, physical attacks are much more difficult to perform.

Among them, the most critical attack scenarios related to the smart manufacturing use case are listed in Table 2. Besides a general description of the attack scenario, we also state the components and services affected by attacks, their impact, and the protection goals we want to achieve in the context of ASSURED.

Funded by the Horizon 2020 Framework Programme of the European Union

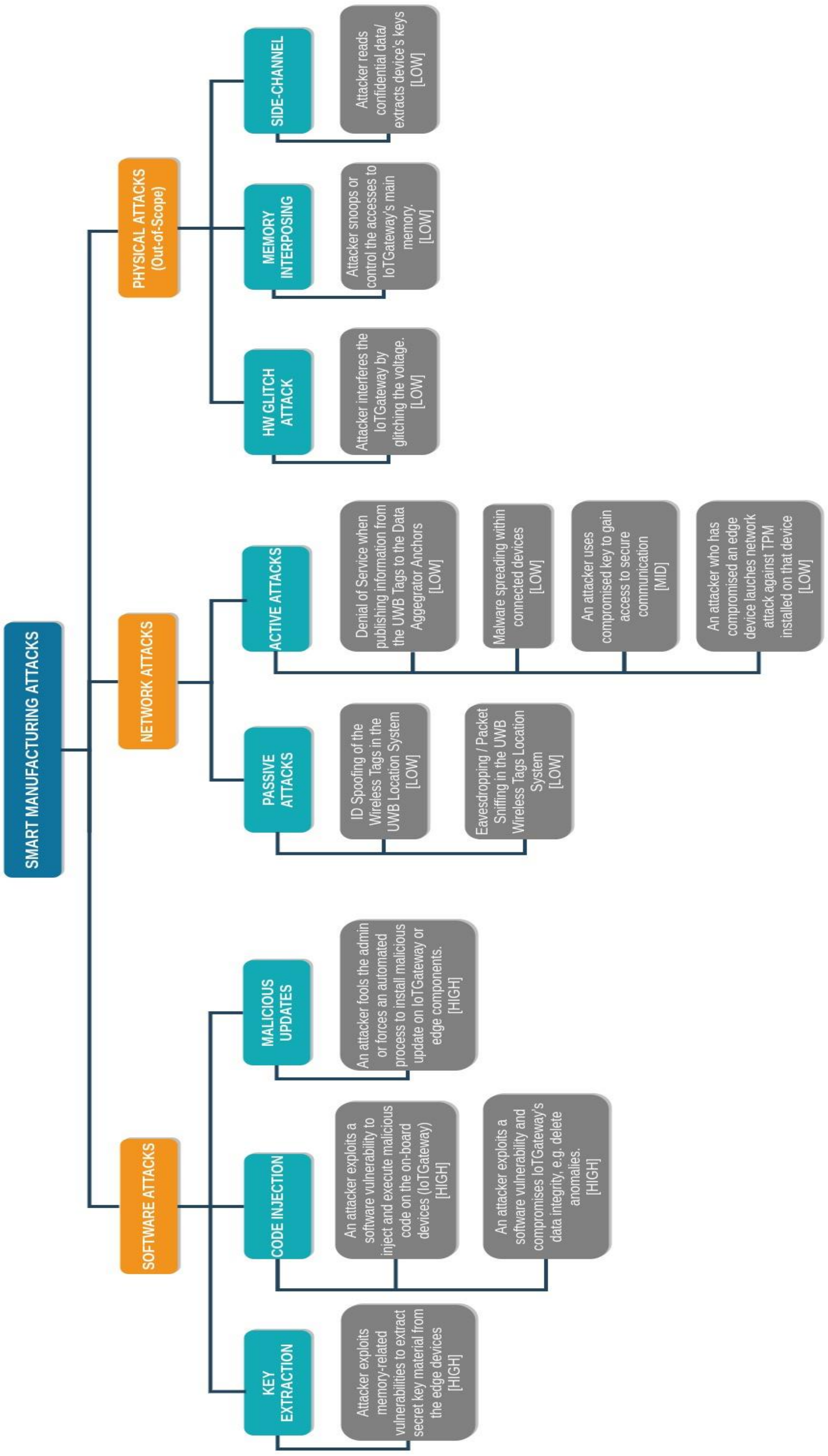
SMART MANUFACTURING ATTACKS
PHYSICAL ATTACKS (Out-of-Scope)
SIDE-CHANNEL
Attacker reads confidential data/ extracts device's keys [LOW]
MEMORY INTERPOSING
Attacker snoops or control the accesses to IoTGateway's main memory. [LOW]
HW GLITCH ATTACK
Attacker interferes the IoTGateway by glitching the voltage. [LOW]
NETWORK ATTACKS
ACTIVE ATTACKS
Denial of Service when publishing information from the UWB Tags to the Data Aggregator Anchors [LOW]
Malware spreading within connected devices [LOW]
An attacker uses compromised key to gain access to secure communication [MID]
An attacker who has compromised an edge device lauches network attack against TPM installed on that device [LOW]
PASSIVE ATTACKS
ID Spoofing of the Wireless Tags in the UWB Location System [LOW]
Eavesdropping / Packet Sniffing in the UWB Wireless Tags Location System [LOW]
SOFTWARE ATTACKS
MALICIOUS UPDATES
An attacker fools the admin or forces an automated process to install malicious update on IoTGateway or edge components. [HIGH]
CODE INJECTION
An attacker exploits a software vulnerability to inject and execute malicious code on the on-board devices (IoTGateway) [HIGH]
An attacker exploits a software vulnerability and compromises IoTGateway's data integrity, e.g. delete anomalies. [HIGH]
KEY EXTRACTION
Attacker exploits memory-related vulnerabilities to extract secret key material from the edge devices [HIGH]

FIGURE 13: ATTACK TREE GRAPH FOR SMART MANUFACTURING USE CASE

Funded by the Horizon 2020 Framework Programme of the European Union

TABLE 2: SMART MANUFACTURING CRITICAL ATTACK SCENARIOS

Attack Scenario	Description	Criticality	Counter-measure
Key extraction	**Scenario**: This attack targets the IoTGateway or edge components that have access to secret key material used for setting up secure communication with other entities. The attacker performs memory extraction to acquire the cryptographic key material exposed during the runtime of programs performing cryptographic operations. **Impact**: Compromised backward and forward confidentiality of exchanged data. **Protection Goal**: cryptographic key material can only be accessed by authorized partners and only in predefined ways.	**High**	Runtime attestation when accessing secret key material.
Code Injection	**Scenario**: This attack targets the IoTGateway or edge components that are connected to the cyber physical infrastructure such as Robotic Arms, PLCs controlling these arms, etc. A malicious code injection can cause downtime in the OT (Operations Technology) due to unauthorized calls to the CPS infrastructure and data manipulation causing a lapse in data security and personnel safety within the workspace environment. **Impact**: Compromised personnel safety, as well as production downtime, data security and integrity. **Protection Goal**: Avoid unauthorized code injection during runtime on edge components.	**High**	Static attestation when executing a micro-service running within a container e.g. OCI compliant container.
Control-flow	**Scenario**: This attack targets the IoTGateway or edge components that run specific software services related to control algorithms or data processing and accept user input during runtime. Control-flow manipulation can cause malicious operations in the OT (Operations Technology) and data manipulation causing a lapse in data security and personnel safety within the workspace environment. **Impact**: Unwanted operational behaviour of edge devices, compromised personnel and data integrity. **Protection Goal**: Abnormal control flow behaviour should be detected, and new protection policies should be deployed.	**High**	Runtime attestation for verifying control flow integrity.
Malicious Updates	**Scenario**: This attack targets the IoTGateway or edge components that run specific software services related to control algorithms or data processing that is required for decision making and signal control to CPSs. Any unverified or unauthorized software updates on the gateway can lead to downtimes or hazards to personnel safety within the workspace in which the IoTGateway is deployed. **Impact**: Compromise in personnel safety as well as production downtimes as well as ransomware causing lapse in system integrity as well as safety hazards.	**High**	Static attestation when performing updates on the Gateway for verifying binary signature, and runtime attestation for avoiding changes to control-flow logic.

Funded by the Horizon 2020 Framework Programme of the European Union

	Protection Goal: Software updates need to be verified and deployed only upon proper authorization to the edge components in the workspace.		

5.1.2 Model of Use-cases Properties

ASSURED offers multiple attestation schemes to attest various static and dynamic properties. The key is to select the appropriate properties according to the attack tree graph presented in the previous section. Table 3 documents the properties to be validated in the context of the smart manufacturing use case.

TABLE 3: MAPPING OF VALIDATION PROPERTIES FOR SMART MANUFACTURING USE CASE

Attack Scenario	Target Systems	Processes	ASSURED Security Enabler	Validation Property
Key extraction (Runtime)	IoTGateway	Key establishment	Static Properties	Static and dynamically loaded binaries responsible for secure communication and encryption of data. For example, X.509 TLS private key certificates on IoTGateway / special Access Tokens
Control-flow (Runtime)	IoTGateway	Collision Avoidance and Prediction Algorithm	Dynamic Properties	Control flow branch information within the container
Code Injection (Runtime)	IoTGateway	Micro-Services: • Robotic Motion Tracking • Personnel Location • Collision Avoidance and Prediction Algorithm	Static and Dynamic Properties	Configuration Files for each Micro-Service Changes to static code within a micro-service container Changes to Data Flow exchange with other micro-services
Malicious Updates	IoTGateway	Micro-Services: • Robotic Motion Tracking • Personnel Location • Collision Avoidance and Prediction Algorithm	Static and Dynamic Properties	Configuration Files for each Micro-Service Libraries / Dependency changes in the micro-services during updates

Funded by the Horizon 2020 Framework Programme of the European Union

5.2 SECURE COLLABORATION OF "PLATFORMS-OF-PLATFORMS" FOR ENHANCED PUBLIC SAFETY

The Athens testbed focuses on the topic of public safety and the protection of related city systems. Hence, a city operation centre is connected with several computational components residing either in the back-end infrastructure or at the edges of the network in order to offer public safety services. The infrastructure consists of a wide variety of heterogenous devices including, edge-devices, gateways, storage, and network infrastructure. The edge devices are crucial for meeting the operational goals of the use case as the system includes surveillance cameras and sensors that generate data streams of video and sensory data respectively.

As can be seen in Figure 14, the public safety ecosystem consists of the following Cyber Physical System of Systems:

- Edge devices located in strategic positions for the protection of the Serafio complex, and more specifically:
    - **IP surveillance cameras**
    - **Smoke detection sensors**
    - **PLCs for controlling the surveillance systems and processes.**
- The back-end **Information and Communication Technology (ICT)** system and cloud-based infrastructure which includes the networking components and computational resources to support the operations centre.

The edge devices communicate with the ICT infrastructure via secure communication channels. The secure communication is achieved based on the key establishment service that runs both on the edge devices and the back-end systems. In fact, this service runs first in order to securely instantiate the overall deployment and then proceed with the core safety critical operations. The operation of the IP surveillance cameras generates video data streams, from the edges of the network back to the ICT and cloud-based backend, in order to provide the necessary input to operation center and feed the object detection and face recognition processes. The collected data and the results of the video stream processing are meant to be shared with external stakeholders in the context of the data value chains of the ASSURED project.

In addition, the smoke detection sensors are critical for the safety of the Serafio complex. Specifically, these sensors transmit signals to the backend ICT infrastructure periodically to detect a fire and provide the necessary indications to the decision makers of the operation center to trigger the fire alarm and the established safety procedures (e.g., evacuation plans).

The ICT infrastructure is based on network components, such as IoT gateways and network switches, and servers that support the secure communication and the processing on the data streams, respectively. The network components support the bidirectional secure communication between the edge devices and the back-end systems. Specifically, the generated data streams are transmitted by the edge devices towards the back-end systems for further processing, while device management processes, such as software/firmware updates, device (re) configuration and commands, are pushed through the network to the edge devices.

Overall, in this context, the aforementioned Cyber Physical System of Systems of the public safety use case work in synergy to offer the following services that need to be protected:

➲ Video Data Stream Generation;

➲ Sensor Data Generation;

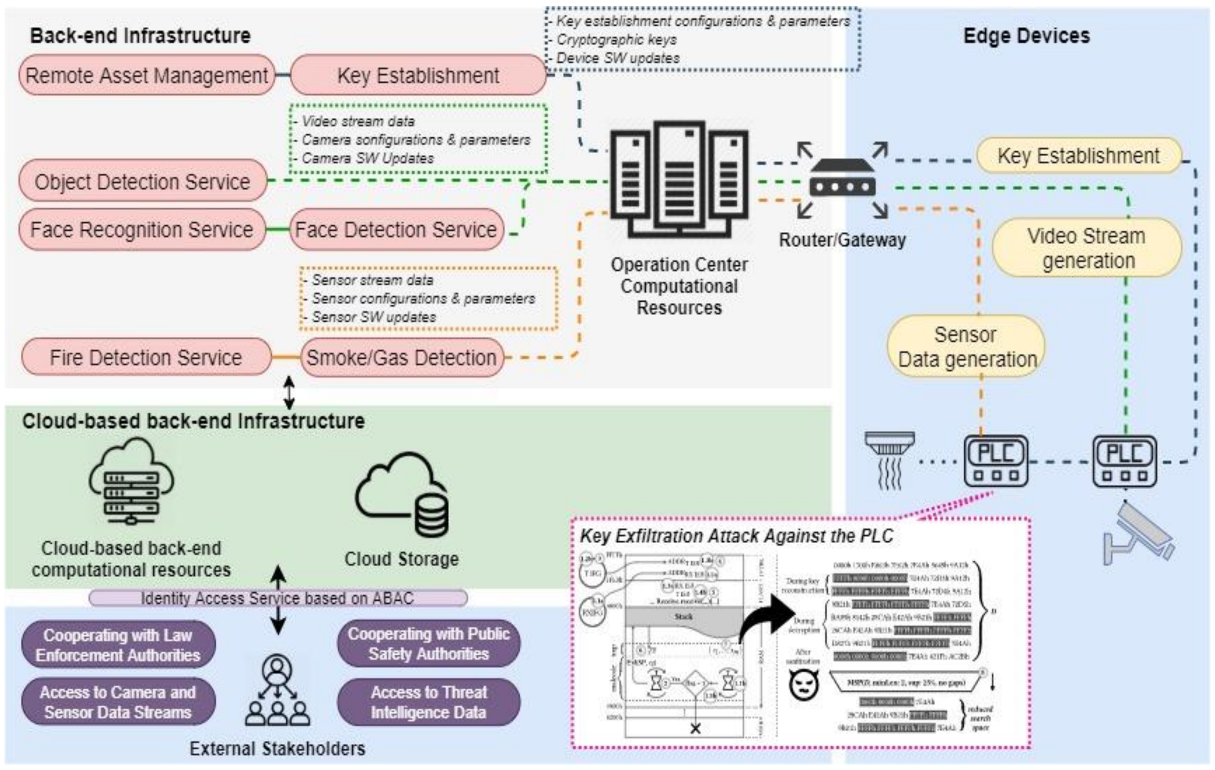
Back-end Infrastructure
Remote Asset Management
Key Establishment
- Key establishment configurations & parameters
- Cryptographic keys
- Device SW updates
Edge Devices
- Video stream data
- Camera sonfigurations & parameters
- Camera SW Updates
Object Detection Service
Face Recognition Service
Face Detection Service
Operation Center Computational Resources
Router/Gateway
Key Establishment
Video Stream generation
- Sensor stream data
- Sensor configurations & parameters
- Sensor SW updates
Fire Detection Service
Smoke/Gas Detection
Sensor Data generation
Cloud-based back-end Infrastructure
Cloud-based back-end computational resources
Cloud Storage
PLC
PLC
Key Exfiltration Attack Against the PLC
Identity Access Service based on ABAC
Cooperating with Law Enforcement Authorities
Access to Camera and Sensor Data Streams
Cooperating with Public Safety Authorities
Access to Threat Intelligence Data
External Stakeholders

FIGURE 14: PUBLIC SAFETY ECOSYSTEM

- Key Establishment (Cryptographic key management for secure communication)
- Remote asset management
- Face Recognition and Face Detection
- Object detection
- Fire detection and Smoke/Gas detection

The aforementioned services engage a wide variety of devices spanning from the edge to the cloud-enabled backend, and thus, posing a significant challenge of safeguarding the entirety of the safety-critical components of the system. As it will be documented in the following sections, these devices and the offered services can be targeted by adversaries in various ways. Figure 14 intuitively demonstrates a key exfiltration attack that can take place against the PLCs of the infrastructure, in alignment with the threat landscape that will be presented in the next section based on recent research endeavours in the field [69]. The goal of ASSURED is to provide a set of protection mechanisms that can guarantee the operational assurance of these devices and processes.

5.2.1 Protection Goals and Attack Settings

Attacks and types of malicious actions targeting the city system is depicted in the attack graph tree figure for the public safety scenarios and analysed below.

Software attacks target any edge component of the system and network infrastructure. The software-based attacks are applicable to devices and components of the system running backend codes, such as routers, gateways, servers, cameras, sensors etc. One group of software attacks refers to **key extraction** when an attacker exploits potential vulnerabilities on the memory of systems in order to extract secret keys from the edge devices, or additionally, to get access to edge components for accessing data related to memory.

Funded by the Horizon 2020 Framework Programme of the European Union

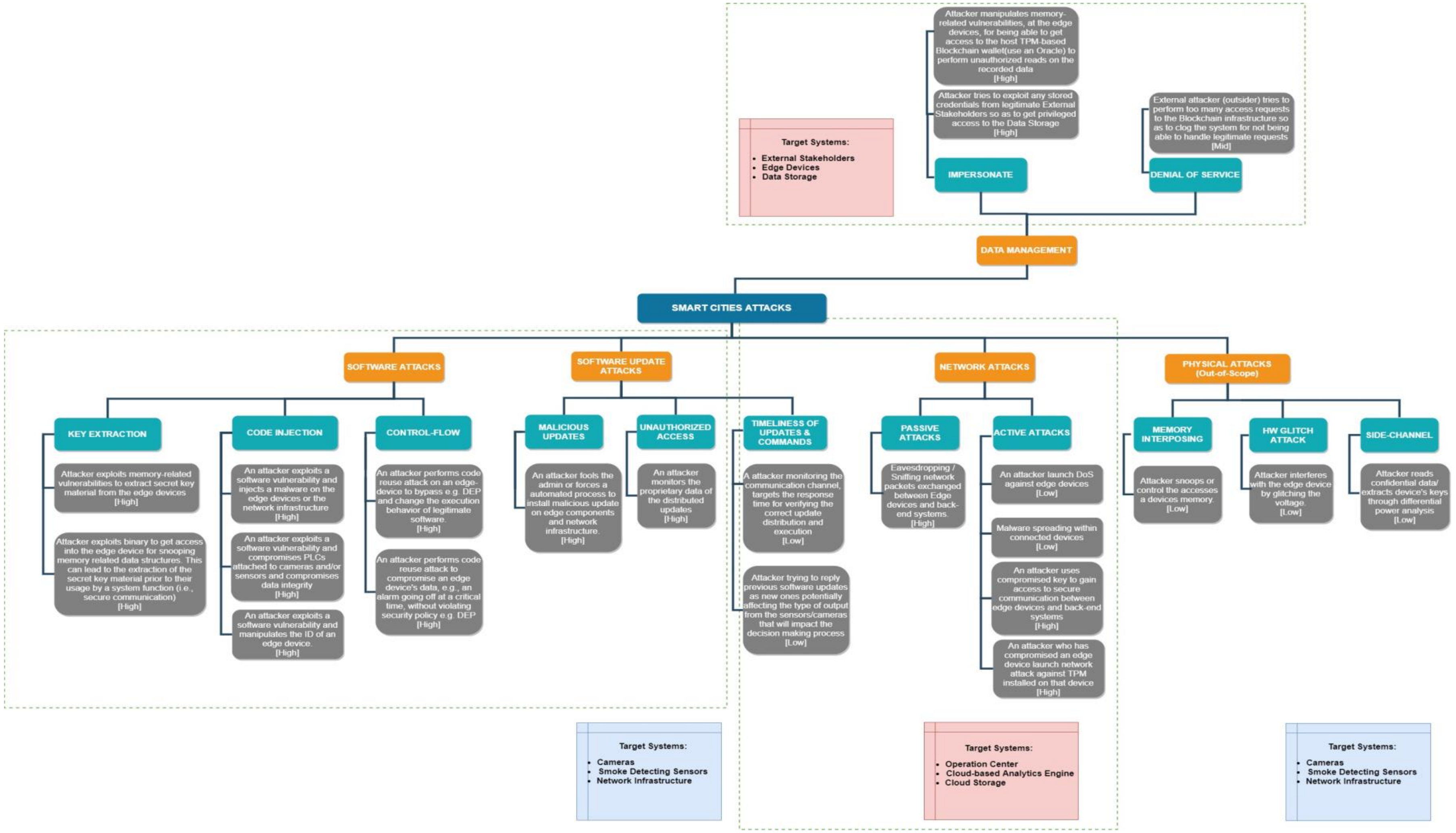
SMART CITIES ATTACKS
DATA MANAGEMENT
IMPERSONATE
Attacker manipulates memory-related vulnerabilities, at the edge devices, for being able to get access to the host TPM-based Blockchain wallet(use an Oracle) to perform unauthorized reads on the recorded data [High]
Attacker tries to exploit any stored credentials from legitimate External Stakeholders so as to get privileged access to the Data Storage [High]
DENIAL OF SERVICE
External attacker (outsider) tries to perform too many access requests to the Blockchain infrastructure so as to clog the system for not being able to handle legitimate requests [Mid]
Target Systems:
External Stakeholders
Edge Devices
Data Storage
SOFTWARE ATTACKS
KEY EXTRACTION
Attacker exploits memory-related vulnerabilities to extract secret key material from the edge devices [High]
Attacker exploits binary to get access into the edge device for snooping memory related data structures. This can lead to the extraction of the secret key material prior to their usage by a system function (i.e., secure communication) [High]
CODE INJECTION
An attacker exploits a software vulnerability and injects a malware on the edge devices or the network infrastructure [High]
An attacker exploits a software vulnerability and compromises PLCs attached to cameras and/or sensors and compromises data integrity [High]
An attacker exploits a software vulnerability and manipulates the ID of an edge device. [High]
CONTROL-FLOW
An attacker performs code reuse attack on an edge-device to bypass e.g. DEP and change the execution behavior of legitimate software. [High]
An attacker performs code reuse attack to compromise an edge device's data, e.g., an alarm going off at a critical time, without violating security policy e.g. DEP [High]
SOFTWARE UPDATE ATTACKS
MALICIOUS UPDATES
An attacker fools the admin or forces a automated process to install malicious update on edge components and network infrastructure. [High]
UNAUTHORIZED ACCESS
An attacker monitors the proprietary data of the distributed updates [High]
TIMELINESS OF UPDATES & COMMANDS
A attacker monitoring the communication channel, targets the response time for verifying the correct update distribution and execution [Low]
Attacker trying to reply previous software updates as new ones potentially affecting the type of output from the sensors/cameras that will impact the decision making process [Low]
Target Systems:
Cameras
Smoke Detecting Sensors
Network Infrastructure
NETWORK ATTACKS
PASSIVE ATTACKS
Eavesdropping / Sniffing network packets exchanged between Edge devices and back-end systems. [High]
ACTIVE ATTACKS
An attacker launch DoS against edge devices [Low]
Malware spreading within connected devices [Low]
An attacker uses compromised key to gain access to secure communication between edge devices and back-end systems [High]
An attacker who has compromised an edge device launch network attack against TPM installed on that device [High]
Target Systems:
Operation Center
Cloud-based Analytics Engine
Cloud Storage
PHYSICAL ATTACKS (Out-of-Scope)
MEMORY INTERPOSING
Attacker snoops or control the accesses a devices memory. [Low]
HW GLITCH ATTACK
Attacker interferes with the edge device by glitching the voltage. [Low]
SIDE-CHANNEL
Attacker reads confidential data/ extracts device's keys through differential power analysis [Low]
Target Systems:
Cameras
Smoke Detecting Sensors
Network Infrastructure

FIGURE 15: ATTACK TREE GRAPH FOR PUBLIC SAFETY USE CASE

Funded by the Horizon 2020 Framework Programme of the European Union

A **code injection** adversary aims to inject malware on the edge devices or the network infrastructure, compromises PLCs attached to the municipality cameras and/or sensors, in order to target the data integrity or manipulate the ID of an edge device. Finally, the attacks exploiting the **control flow** of DAEM's use case configuration either changes the control flow execution of legitimate software or manages to compromise the data of a sensor/camera/other edge device (e.g., Phishing on the Raspberry gateway).

In the context of remote asset Management, **software update attacks** may target any edge component of the system and network infrastructure similarly to software attacks. This category includes adversaries on **malicious updates** and **unauthorized accesses** that attack the municipal system as:

- A third-party vendor that provides a malicious firmware update
- An attacker that replaces legitimate software updates with malicious code
- An unauthorized user that performs software updates
- An attacker that monitors the proprietary data of the distributed updates

In addition, in the context of software update attacks, adversaries may target the **timeliness of updates & commands.** Such an adversary can potentially target the public safety domain by monitoring of the communication channels in order to attack the response time of devices for verifying the correct update distribution and execution. In addition, she can also insert previous software updates as new ones that can potentially affect the type of output from the sensors/cameras and, thus, impact the decision-making process.

Network attacks can also target any edge component of the system and network infrastructure. Those attacks **can be categorized into passive and active attacks**. The former category considers attackers that passively monitor the network by eavesdropping and sniffing network packets exchanged between the edge devices and the back-end systems. The latter category considers active attackers who can launch DoS attacks against devices and services, spread malware to the network devices, evade the secure communication by compromising cryptographic keys, and even compromise edge devices to interrupt the communication with the underlined TPM.

Data management attacks mainly target the systems of external stakeholders (LEAs, police etc.), data storage facilities and edge devices. An attacker could **impersonate** the behavior of a trusted actor or component of the city system, as a man-in-the-middle attack, in two potential ways:

- By manipulating memory-related vulnerabilities, at the edge devices, and getting access to the host TPM-based Blockchain wallet. This can lead to unauthorized reads on the recorded data.

- By exploiting stored credentials from legitimate External Stakeholders in order to get privileged access to the Data Storage.

The second kind of data management attack refers to **denial of service.** An attacker (outsider) tries to perform too many access requests to the Blockchain infrastructure to clog the system, so that it is not able to handle legitimate requests.

When it comes to **Physical Attacks**, those can be placed into three distinct categories. Physical access to the devices is required in all these categories, therefore they are outside the scope of ASSURED. However, for completeness purposes, we shortly elaborate on them. More specifically, **Memory interposing** attacks can be performed by an adversary by snooping or controlling the accesses to the main memory of edge devices. In addition,

Funded by the Horizon 2020 Framework Programme of the European Union

Hardware Glitch Attacks can be performed to interfere with the edge devices by glitching the voltage. **Side channel attacks** can be also performed, and an attacker could read confidential data or extracts device's keys through differential power analysis.

TABLE 4: PUBLIC SAFETY CRITICAL ATTACK SCENARIOS

Attack Scenario	Description	Criticality	Counter-measure
Key Extraction (Runtime)	**Scenario:** This attack targets the edge devices (e.g., cameras, smoke detection sensors) and their attached PLCs that are responsible for securely managing secrete key material used for the subsequent communication of the extracted data to the back-end Decision Support Engine. The attacker, by exploiting memory-related vulnerabilities (e.g., buffer overflow), snoops on the publicly accessible data structures (i.e., stack) for extracting secret keys prior to their usage. **Impact:** Compromise backward and forward secrecy and data confidentiality of exchanged data. **Protection Goal:** Compromise of an edge device should not allow the exploitation and/or leakage of any secret material used for the confidentiality and integrity of transmitted video and sensor data	**High**	**Runtime Attestation for verifying the integrity of loaded binaries**
Code Injection (Runtime)	**Scenario**: this attack exploits a software vulnerability and injects a malware on the edge devices or the network infrastructure. **Impact**: target data integrity or manipulate the ID of an edge device. **Protection Goal**: Compromising an edge device should not enable an attacker to access the core code and manipulate its ID.	**High**	**Runtime Attestation for verifying control flow integrity of safety critical services on the edge devices.**
Control-flow	**Scenario:** This attack targets the edge devices (e.g., cameras, smoke detection sensors) and their attached PLCs by changing the control flow execution of the legitimate software that runs on them. The attacker performs code reuse attack on an edge-device to bypass e.g., DEP and change the execution behaviour of legitimate software. **Impact:** Compromise the data of a sensor/camera/other edge device and divert the operational behaviour of safety critical application. **Protection Goal:** Compromise of an edge device and violation of its control flow integrity should be detected, and new protection policies must be deployed.	**High**	**Runtime Attestation for verifying control flow integrity of safety critical services on the edge devices.**
Malicious Updates	**Scenario**: An attacker replaces legitimate software updates with malicious code **Impact**: the attacker can intervene in the operation system functionality by deploying a malware in the format of a system or component operational update (e.g., a camera function) **Protection Goal**: Prevent remote access to assets of the system by external attackers	**High**	**Static attestation for the verification of the correct (or expected) signature of the application.**

Funded by the Horizon 2020 Framework Programme of the European Union

Unauthorised Access	**Scenario:** In the context of this attack, the adversary aims to get unauthorised access to the software update processes. The attacker is able to interfere with the software updates and get access to the parameter updates that might need to be circulated to the edge devices, e.g., periodicity of smoke data measurement collection, speed of angle change of the camera, etc. Thus, the attacker monitors the proprietary data of the distributed updates. **Impact:** Manipulation of the software update process which enables the attacker to compromise the operational behaviour of safety critical applications and modify the mode of operation of edge devices. **Protection Goal:** Establishment of secure communication between edge devices and the back-end systems in order to avoid attackers intervene in the software and parameters update processes. Provide guarantees on the integrity of binaries and configurations of edge devices.	**High**	**Runtime attestation and configuration integrity verification of the updated binaries**
Passive Network Attacks	**Scenario:** In the context of this attack the adversary is positioned in the middle of the communication channel of the edge devices and the back-end systems to perform eavesdropping and sniffing of the exchanged network packets. The attacker can take advantage of weak cryptographic primitives or the absence of identity authentication processes in the communication processes. **Impact:** Passive network attacks can be the first stage of more sophisticated attacks. Thus, the acquisition of critical information exchanged between edge devices and back-end systems can be useful for the attacker to compromise more safety critical applications. **Protection Goal:** Establishment of secure and authenticated communication between edge devices and the back-end systems in order to avoid attackers to be positioned in the middle of the communicating entities.	**High**	**Secure communication channel establishment based on the use of HW-based keys managed by the ASSURED TPM-enabled wallet**
Active Network Attacks	**Scenario:** In the context of this active network attacks the adversary can launch several attack variations with high impact. More specifically, an attacker can take advantage of compromised keys and decrypt the communication between the edge devices and the back-end systems. In addition, network attacks can be launched against the TPM installed on a device by exploiting the TPM Command Transmission Interface (TCTI) and, thus, interfering with, or disrupting, the communication between the host device and the TPM. **Impact:** Active network attacks can lead to manipulation or disruption of the communication between core edge devices and the back end-systems or between the host devices and the TPM that acts as the trust anchor of the overall deployment. Thus, active attacks can have great impact on the confidentiality of the communications and can threaten the safety critical services that capitalise on the trust qualities of the TPM. **Protection Goal:** Guarantee the secrecy of the cryptographic keys used to establish secure and	**High**	**<u>In case of Key Compromise:</u> Strong revocation of the cryptographic keys and credentials guaranteed by the ASSURED TPM-enabled wallet.** **<u>In case of compromised host device:</u> Use of policies- and sessions-related core TPM services to safeguard the communication**

Funded by the Horizon 2020 Framework Programme of the European Union

	authenticated communications based on TPM authorisation mechanisms. Ensure authorised and regulated interaction of safety critical processes with the TPM based on the principle of minimising the trusted computing base and adopting minimal trust assumptions.		**between the host and the TPM.**
Impersonation in data management	**Scenario:** In the context of impersonation in the data management process an attacker manipulates memory-related vulnerabilities, at the edge devices, for being able to get access to the host TPM-based Blockchain wallet to perform unauthorized reads on the recorded data. In addition, an attacker may exploit any stored credentials from legitimate External Stakeholders to get privileged access to the Data Storage. **Impact:** Such attacks can have great impact on the confidentiality of information stored and managed throughout the data value chains formed in the context of ASSURED. The malicious activity of the attacker can lead to data leaks of sensitive and operational data from the underlined infrastructures. **Protection Goal:** Guarantee operational assurance the TPM-based Blockchain wallet and ensure that unauthorised read of recorded data can detected though the use of control flow attestation.	**High**	**Runtime Attestation for verifying control flow integrity of the TPM-based wallet processes.**

5.2.2 Model of Use-cases Properties

In the previous section, we documented the attack scenarios which are relevant to the public safety use case by giving details on their impact, and the conditions that need to be met for these attacks to be performed in this use case. Using this as a baseline, this section offers a mapping among the identified attack scenarios with the systems and processes of the demonstrator by highlighting the properties that need to be validated by the ASSURED defensive mechanisms to meet the protection goals.

The public safety use case processes that should be protected are focused on the generation and sharing of privacy sensitive and safety critical data through the ASSURED supply chain. In this context, a core offering of the ASSURED framework is the attestation and verification of validation properties that has been defined in Section 4.1. Thus, the table below documents the properties to be validated against the most devastating identified attacks.

TABLE 5: MAPPING OF VALIDATION PROPERTIES FOR PUBLIC SAFETY USE CASE

Attack Scenario	Target Systems	Processes	ASSURED Security Enabler	Validation Property
Key Extraction (Runtime)	Edge devices (e.g., cameras, smoke detection sensors) and PLCs	• Key establishment	Static Properties	Static and dynamically loaded binaries responsible for the establishment of the secure communication. For instance, the binary for generating the necassary algorithmic material

Funded by the Horizon 2020 Framework Programme of the European Union

				and prameters for the generation of ECC-based keys in the context of Direct Anonymous Attestation leveraged for privacy protection. (See D4.3 [99])
Code Injection (Runtime)	Edge devices, Network components (e.g., IoT Gateway), back-end systems	• Video data stream generation • Sensor data generation • Key establishment • Remote asset management • Face detection/recognition • Object detection • Face Recognition and Face Detection • Object detection • Fire detection and Smoke/Gas detection	Static and Dynamic Properties	Attestation of statically and dynamically loaded binaries used to interact with core OS functionalities. Control flow information of all processes generating the services' data.
Control-flow	Edge devices, Network components (e.g., IoT Gateway), back-end systems	• Video data stream generation • Sensor data generation • Key establishment • Remote asset management • Face detection/recognition • Object detection • Fire detection and Smoke/Gas detection	Dynamic Properties	Control flow information of all processes generating the safety critical services' data.
Malicious Updates	Edge devices	• Video data stream generation • Sensory data generation • Key establishment	Static Properties	Attestation of statically and dynamically loaded binaries being updated in the context of the safety critical services.
Unauthorised Access	Edge devices	• Video data stream generation • Sensory data generation	Static properties	Attestation of statically and dynamically loaded binaries in the context of sofware upades that indicate the configuration paramaters of the data generation processes.

Funded by the Horizon 2020 Framework Programme of the European Union

Passive Network Attacks	Edge devices, Network components (e.g., IoT Gateway), back-end systems	• Key establishment • Remote asset management	Static properties	Attestation of statically and dynamically loaded binaries and the respecitve configuration of the responsible banaries for establishing the secure communication between edge devices and back-end systems.
Active Network Attacks	Edge devices, Network components (e.g., IoT Gateway), back-end systems	• Key establishment • Remote asset management	Static and Dynamic properties	Attestation of statically and dynamically loaded binaries that undertake credentials revocation process. Control flow information of the critical processes communicating with the TPM of a host device to detect control flow deviations.
Impersonation in data management	Edge devices	• Remote asset management	Static and Dynamic properties	Statically and dynamically loaded binaries of the TPM-based wallet. Control flow information of the TPM-based wallet service.

5.3 SECURE AND SAFE AIRCRAFT UPGRADABILITY & MAINTENANCE

The smart aerospace is a complex SoS-enabled ecosystem which consists of many onboard cyber-physical systems, such as **Flight Management Systems (FMS)**, **Environment Control Systems (ECS)**, **Cockpit Flight Instruments (CFI)**, and **on-board Wi-Fi systems**. A **secure server router (SSR)** is a real-time embedded device that enables the most important aircraft functionalities, e.g., on- and off-board communications. SSR collects critical data of the aircraft while in the air from the on-board edge devices and transmits the data collected to a Ground Station Server (GSS) when on the ground. To ensure the security of a smart aerospace, it is crucial to deploy a safe and secure data transfer between the SSR and the ground station server and perform secure remote updates on the SSR.

Figure 16 depicts the core components and services in smart aerospace use case. Currently the SSR communicates with the GSS through a cabled connection, specifically Ethernet connection, when the airplane is on the ground. SSR collects the sensor data while flying, and then transfers the data to the GSS through the cabled connection. Furthermore, as previously mentioned, the SSR offers several services on the airplane: separate Wi-Fi connections to the

Funded by the Horizon 2020 Framework Programme of the European Union

crew members and to the passengers enabled by cellular connections, cabled data collection and storage from the sensors and an ad-hoc access control module to be used when performing physical software updates. SSR runs over a proprietary implementation of Security-Enhanced Linux-based Operating System (OS). The GSS instead can be depicted as a simple data server, based on either a Linux or Windows OS, which stores the data received from the SSR.

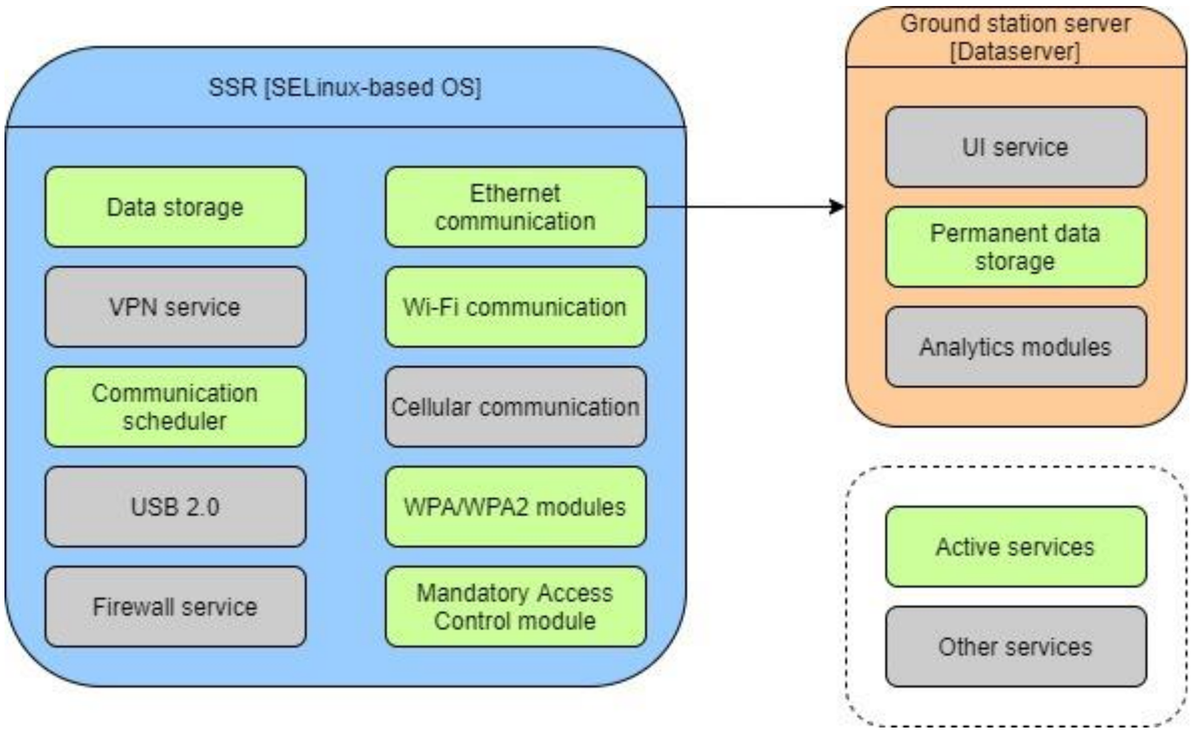

FIGURE 16 SMART AEROSPACE CORE COMPONENTS AND SERVICES

5.3.1 Protection Goals and Attack Settings

In aviation and aerospace industry, the trustworthiness on safety-critical components has high priority, otherwise countless lives could be lost. This is also the protection goal to achieve in context of smart aerospace, namely one can confidently say that all the activities performed by the system are justifiably and confidently trusted. The definition of trustworthiness on safety-critical components needs to include not only safety but also security. Therefore, ASSURED is needed to comply with the up-to-date security protection mechanisms of the aerospace systems.

The SSR is primarily used as an **Electronic Flight Bag (EFB)** interface and communication unit to enable pilots and crew to communicate over wireless interfaces. When deployed on the aircraft, the SSR wirelessly connects any EFB device, such as a pilot mobile device, to exchange data with other flight deck devices. The information exchanged varies widely by aircraft, airline, and EFB application, but EFB information may include pre-flight checklists, weather data, aeronautical charts, as well flight operation quality assurance data. Consequently, security of the router and data integrity is paramount. A successful compromise of the SSR software could potentially drive pilot confusion by providing inaccurate aeronautical data to the pilot. Moreover, given the SSR's ability to exchange flight operation quality assurance data, a compromise of the SSR would allow the attacker to potentially corrupt maintenance logs. While several advanced protection techniques are put in place around the SSR, the ASSURED framework aims to extend these conventional security measures with advanced runtime protection for additional assurance.

Funded by the Horizon 2020 Framework Programme of the European Union

The smart aerospace ecosystem can be vulnerable to different attack scenarios, shown in the attack tree in Figure 17. While the attack tree graph provides a complete view of the potential attack scenarios in the smart aerospace use case, the most crucial components of the ecosystem are the SSR, the GSS and the communication channel between them. This is due to the physical restrictions and composition of the ecosystem. For example, if we evaluate the network attacks centred on the SSR, the communication with the GSS is more vulnerable than the one with the sensors. On the one hand, the communication with GSS is performed through a wireless connection which does not require physical proximity for the attacker to either the SSR or the GSS, allowing him to perform physical-undetected wireless attacks on the connection. On the other hand, the communication with sensors requires the attacker to be physical connected to either the SSR or the sensors since these communications are fully cabled and not easily accessible to non-technical personnel.

If we consider the communication established between the SSR and GSS when performing either the secure data transfer or the secure remote software update, the SSR and GSS must first establish a wireless communication, attest each other's authenticity, and control the GSS authorization to ensure the confidentiality of the data transferred. If a software update is pushed to the SSR, starting from the same communication security requirements mentioned for the data transfer scenario, the GSS authorization must be verified and the integrity of the software must be checked, through static attestation at first and through dynamic attestation during run-time.

Starting from this nominal baseline, we identify several critical attacks that may severely affect the smart aerospace ecosystem, including malicious updates, unauthorized access, and passive network attacks.

Malicious update attacks target the SSR and may affect a wide variety of services that require updates or patches frequently. The attacker can use various approaches to masquerade malicious code as legitimate software updates, either firmware updates or updates of third-party software. This would open vulnerabilities on the SSR, allowing the attacker to gain control over the device, which would lead in having the whole ecosystem open to the attacker. **Unauthorized access attacks** aim at monitoring, accessing, and manipulating the proprietary data of the distributed software updates. This could lead to incorrect analysis of the data representing the status of the aircraft, followed by future incorrect mitigations which would be forced due to the data being compromised by the attacker when the sensors' data are transferred from the SSR. Finally, **passive network attacks** target the data exchanged between the SSR and the GSS. The attacker does not actively tamper the data, but just read and analyse the data in order to gain proprietary information on the overall system, on both software and hardware sides.

Besides the aforementioned critical attacks, there are other type of attacks that make less impact or are hard to be carried out. we will shortly introduce other potential attacks for purpose of completeness. Software attacks targeting SSR include **code injection attacks** and **control flow manipulation attacks** that inject malicious code or reuse existing code to perform malicious actions or compromise SSR's data. As a prerequisite, the attacker is able to send the exploit payload to the victim's applications along with the input data. One group of software attacks, referred to as **key extraction**, exploits memory vulnerabilities to extract secret keys from the devices may affect the SSR and GSS. In addition, **active network attacks** target the communication channel between the SSR and GSS. The attacker can launch DoS attacks, spread malware to the network devices, break the communication with compromised keys, and interrupt the network communication between TPMs. Finally, in the context of the smart aerospace use case, physical attacks require physical proximity for the attacker to the SSR, which renders this type of attacks unrealistic, such as **side channel attacks**, **memory interposing**, and **high glitch attacks**.

Funded by the Horizon 2020 Framework Programme of the European Union

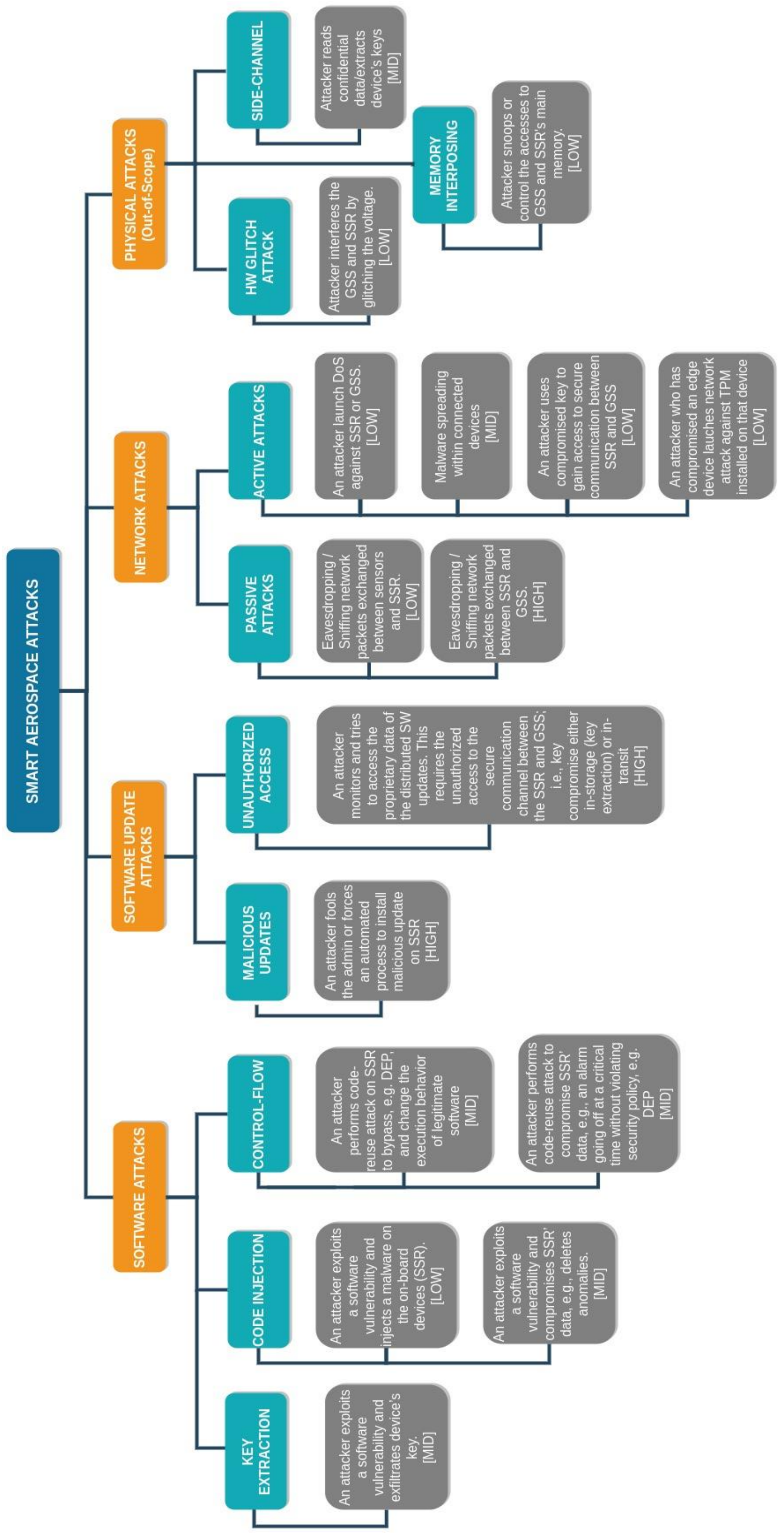
SMART AEROSPACE ATTACKS
PHYSICAL ATTACKS (Out-of-Scope)
SIDE-CHANNEL
Attacker reads confidential data/extracts device's keys [MID]
MEMORY INTERPOSING
Attacker snoops or control the accesses to GSS and SSR's main memory. [LOW]
HW GLITCH ATTACK
Attacker interferes the GSS and SSR by glitching the voltage. [LOW]
NETWORK ATTACKS
ACTIVE ATTACKS
An attacker launch DoS against SSR or GSS. [LOW]
Malware spreading within connected devices [MID]
An attacker uses compromised key to gain access to secure communication between SSR and GSS [LOW]
An attacker who has compromised an edge device lauches network attack against TPM installed on that device [LOW]
PASSIVE ATTACKS
Eavesdropping / Sniffing network packets exchanged between sensors and SSR. [LOW]
Eavesdropping / Sniffing network packets exchanged between SSR and GSS. [HIGH]
SOFTWARE UPDATE ATTACKS
UNAUTHORIZED ACCESS
An attacker monitors and tries to access the proprietary data of the distributed SW updates. This requires the unauthorized access to the secure communication channel between the SSR and GSS; i.e., key compromise either in-storage (key extraction) or in-transit [HIGH]
MALICIOUS UPDATES
An attacker fools the admin or forces an automated process to install malicious update on SSR [HIGH]
SOFTWARE ATTACKS
CONTROL-FLOW
An attacker performs code-reuse attack on SSR to bypass, e.g. DEP, and change the execution behavior of legitimate software [MID]
An attacker performs code-reuse attack to compromise SSR' data, e.g., an alarm going off at a critical time without violating security policy, e.g. DEP [MID]
CODE INJECTION
An attacker exploits a software vulnerability and injects a malware on the on-board devices (SSR). [LOW]
An attacker exploits a software vulnerability and compromises SSR' data, e.g., deletes anomalies. [MID]
KEY EXTRACTION
An attacker exploits a software vulnerability and exfiltrates device's key. [MID]

FIGURE 17 SMART AEROSPACE ATTACK TREE

Funded by the Horizon 2020 Framework Programme of the European Union

A summary of highly critical attacks for the smart aerospace use case is presented in Table 6.

TABLE 6: SMART AEROSPACE CRITICAL ATTACK SCENARIOS

Attack Scenario	Description	Criticality	Counter-measure
Malicious updates	**Scenario**: This attack targets the SSR and services running on it. An attacker replaces legitimate software updates with malicious code. **Impact**: Software integrity compromised, and system open to vulnerabilities. **Protection Goal**: Only legitimate software updates from known sources can be installed in the SSR.	**High**	Static attestation for verifying binary signature Authentic and confidential network communication
Unauthorized access	**Scenario**: This attack targets to the software update processes of SSR firmware and applications. The attacker is able to interfere with the update processes and access to the proprietary data of the distributed updates. **Impact**: Manipulation of proprietary data enables the attacker to compromise the operational behaviour of safety-critical applications and modify the mode of operations of SSR. **Protection Goal**: Confidentiality and integrity of software update processes.	**High**	Static attestation and runtime attestation for verifying the update processes.
Passive network attacks	**Scenario**: An attacker eavesdrops and sniffs the network packets exchanged between SSR and GSS. The attacker can take advantage of weak cryptographic primitives or the absence of identify authentication processes. **Impact**: Data confidentiality compromised. Passive network attacks can be the first stage of many other sophisticated attacks. **Protection Goal**: Retain the confidentiality of network data shared between SSR and GSS.	**High**	Secure communication channel established based on the use of HW-based key managed by ASSURED TPM-enabled wallet
Control flow	**Scenario:** An attacker injects malicious code to the device altering the software flow of commands expected to be executed. Data compromising the control flow execution could be instructions indicating faults or operational modes of the devices where no protection is present. **Impact:** Device inner-data captured and compromised. If additional commands can be injected to the device than more severe	**High**	Runtime Attestation for verifying control flow integrity of safety critical services on the edge devices.

Funded by the Horizon 2020 Framework Programme of the European Union

	attacks can occur (unauthorised privilege editing) **Protection Goal:** Code injection on the device and violation of its control flow integrity should be detected.		
Code injection	**Scenario**: An attacker injects malicious code at runtime or through infiltrating to firmware updates of the device **Impact**: target data integrity or manipulate the ID of the device. **Protection Goal**: Compromising the device should not allow an attacker to obtain access to core code and manipulate its ID or other core services	**High**	Runtime Attestation for verifying control flow integrity of safety critical services on the edge devices.

5.3.2 Model of Use-cases Properties

In this section, we make the mapping among the identified attack scenarios with the demonstrator processes by highlighting the properties needed to be validated by the ASSURED defensive mechanisms to meet the protection goals, as shown in Table 7.

TABLE 7: MAPPING OF VALIDATION PROPERTIES FOR SMART AEROSPACE USE CASE

Attack Scenario	Target Systems	Processes	ASSURED Security Enabler	Validation Property
Code injection	SSR	OS and all related services. All active services should be attested before their execution.	Static and dynamic properties	Binary attestation of the new software (device firmware) to be enforced into the device. Code and static data measurements on run-time. Control-flow and data-flow attestation.
Control-flow	SSR	OS and all related services. All active services should be attested before their execution.	Static and dynamic properties	Loaded libraries and code attestation. Code and static data measurements on run-time. Control-flow and data-flow attestation.
Malicious updates	SSR	OS and all active services on SSR	Static properties	Attestation of loaded binaries and configuration related to software update processes

Funded by the Horizon 2020 Framework Programme of the European Union

Unauthorized access	SSR	OS and all active services on SSR	Static and dynamic properties	Attestation of loaded binaries and configuration related to software update processes Control flow information of the update processes
Passive network attacks	SSR, GSS	All services responsible for establishing secure communication between SSR and GSS.	Static properties	Configuration integrity attestation. Attestation of loaded binaries responsible for establishing secure communication between SSR and GSS.

5.4 DIGITAL SECURITY OF SMART SATELLITES

The digital security of smart satellites use case ecosystem consists of the following Cyber Physical System of Systems, as shown in Figure 18.

1) **Cubesat** is a **miniaturized satellite** performing specific missions in space. There is cooperation among multiple Cubesats as well.
2) The **Ground Station (GS)** is the central unit, which monitors, maintains, and controls CubeSat operation.

CubeSats use KUBOS, an operating system specifically designed for CubeSat, which contains several specific services supporting necessary functions of the CubeSat. CubeSats also run custom software implementing the mission application and the integration with specific hardware payloads, such as cameras and sensors. Usually, CubeSats use hardware payloads to collect data such as telemetry data, and afterwards report the data to the Ground Station. Then, the mission software applications need to be updated regularly.

The Ground Station however runs a commodity OS (Linux or Windows) which is configured by an Admin and is operated by a CubeSat Operator. The Ground Station consists of the Gateway Service implementing the communication with CubeSats, third-party applications, backend systems (e.g., the ASSURED backend server) and other external parties (e.g. external stakeholders). The application layer of the ground station consists of specific services responsible for the control, monitoring, maintenance, and update of the Cubesats, such as audit service logging necessary information, and other services supporting the visualisation, sharing and notifications mechanisms of the system. More details can be found in Figure 18.

CubeSats communicate with the Ground Station and other CubeSats via secure channels. The data collected by CubeSats is transmitted to ground station periodically or upon request. Apart from the internal communication between CubeSats and the Ground Station, the data in some cases need to be shared (from the Ground Station) with external stakeholders.

© 2020-2023 ASSURED Consortium

Funded by the Horizon 2020 Framework Programme of the European Union

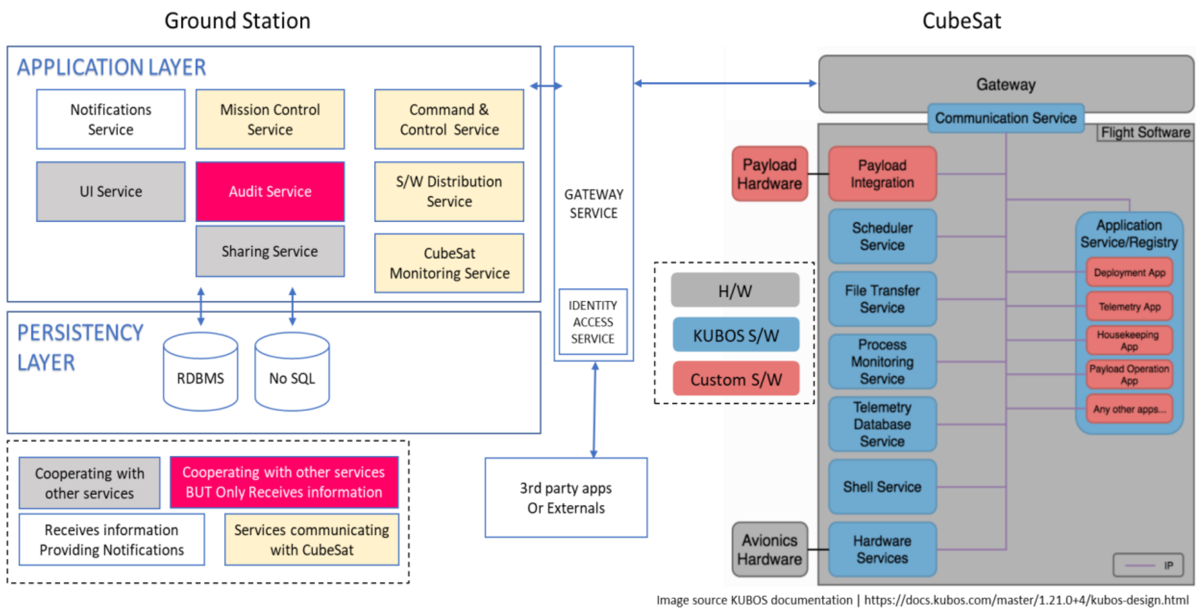

FIGURE 18 SMART SATTELITE COMMUNICATION SERVICES

5.4.1 Protection Goals and Attack Settings

Potential attacks targeting the smart satellites use case are depicted in Figure 19. The adversary's goal is to control a specific or a sub-group of CubeSats services in order to disrupt the normal execution of a mission application, or to cause data leakage or corruption of the data collected from the CubeSats. Given the launch process followed and the verification checks performed, as well as the operational environment of CubeSat, which limits physical access, the CubeSat can be assumed to be benign when firstly operating in space, and physical attacks are precluded. But the adversary can still compromise CubeSats through software update attacks. Specifically, the attacker uses deceptive methods such as phishing to fool the admin or CubeSat operator to install **malicious updates**, either CubeSat firmware or third-party software.

The Ground Station is a potential attack target as well. There are communication channels between the Ground Station and CubeSat, as well as among CubeSats. Attackers may leverage it to compromise more CubeSats after compromising the ground station first. The ground station, running on a common Windows/Linux OS, cannot be easily formally verified and is likely to contain exploitable vulnerabilities. An attacker can exploit a known or zero-day vulnerability to **inject code** into Ground Station in order to stealthily access the confidential data stored in Ground Station, or perform unauthorized actions such as delete anomalous logs, or send forged commands to CubeSats. Advanced adversaries can perform **runtime attacks** to manipulate the legitimate execution behaviours of critical services running in ground station, such as services communicating with CubeSats. This kind of attacks can bypass some security enforcement techniques like DEP. In addition, a special type of software attacks is **key extraction** that targets secret key materials stored or processed in ground station, such as TLS keys used to secure communication with CubeSats.

The aforementioned attacks belong to the category of software attacks that target software components with vulnerabilities. The adversary can leverage software attacks to gain access to the victim system, perform unauthorized operations such as modify the system configuration and access to sensitive data. On this basis, the attacker can use network attacks to compromise more edge devices. However, the network communications within the smart satellite use case are conducted through well-configured and secure channels, such as TLS, to ensure the data integrity, confidentiality, and authentication. DoS attacks are considered out

Funded by the Horizon 2020 Framework Programme of the European Union

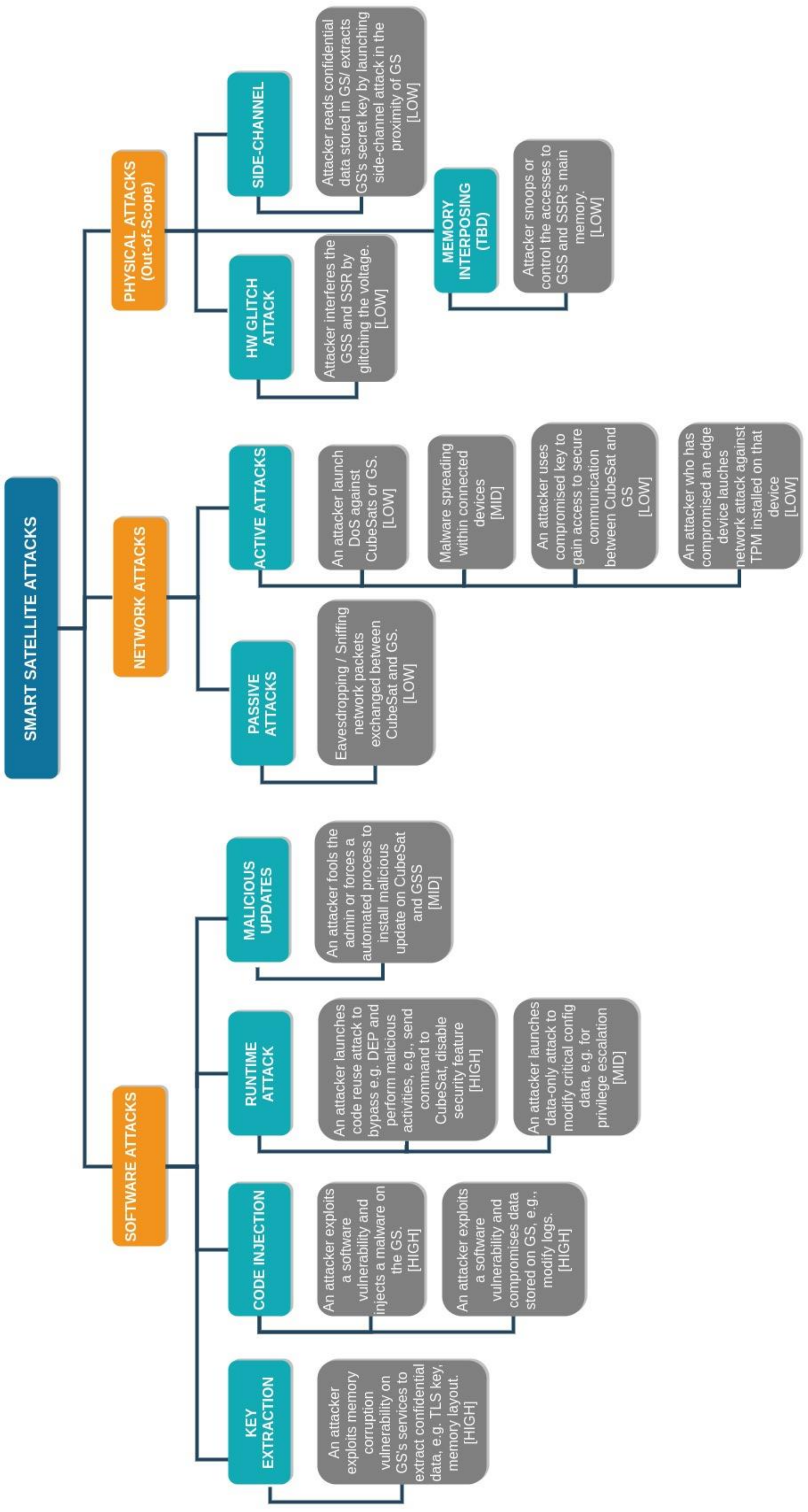
SMART SATELLITE ATTACKS
PHYSICAL ATTACKS (Out-of-Scope)
SIDE-CHANNEL
Attacker reads confidential data stored in GS/ extracts GS's secret key by launching side-channel attack in the proximity of GS [LOW]
MEMORY INTERPOSING (TBD)
Attacker snoops or control the accesses to GSS and SSR's main memory. [LOW]
HW GLITCH ATTACK
Attacker interferes the GSS and SSR by glitching the voltage. [LOW]
NETWORK ATTACKS
ACTIVE ATTACKS
An attacker launch DoS against CubeSats or GS. [LOW]
Malware spreading within connected devices [MID]
An attacker uses compromised key to gain access to secure communication between CubeSat and GS [LOW]
An attacker who has compromised an edge device lauches network attack against TPM installed on that device [LOW]
PASSIVE ATTACKS
Eavesdropping / Sniffing network packets exchanged between CubeSat and GS. [LOW]
SOFTWARE ATTACKS
MALICIOUS UPDATES
An attacker fools the admin or forces a automated process to install malicious update on CubeSat and GSS [MID]
RUNTIME ATTACK
An attacker launches code reuse attack to bypass e.g. DEP and perform malicious activities, e.g., send command to CubeSat, disable security feature [HIGH]
An attacker launches data-only attack to modify critical config data, e.g. for privilege escalation [MID]
CODE INJECTION
An attacker exploits a software vulnerability and injects a malware on the GS. [HIGH]
An attacker exploits a software vulnerability and compromises data stored on GS, e.g., modify logs. [HIGH]
KEY EXTRACTION
An attacker exploits memory corruption vulnerability on GS's services to extract confidential data, e.g. TLS key, memory layout. [HIGH]

FIGURE 19: ATTACK TREE GRAPH FOR SMART SATELLITES USE CASE

Funded by the Horizon 2020 Framework Programme of the European Union

of the scope of ASSURED. Thus, the criticality of network attacks is evaluated as low. Network attacks can be categorized into **passive attacks** like eavesdropping, and **active attacks**. The latter category includes spreading malwares to the network devices, evading secure communication by compromising cryptographic keys, and network attacks on the TPM.

In the context of the smart satellite use case, another potential attack target is the data processed and stored in the Ground Station. Attackers can reveal sensitive data, damage the integrity of data sent to other stakeholders, or disrupt the operation of ground station, e.g., generate false alarms about the system's security state. There are several available paths for attackers to achieve this goal. Besides the software attacks mentioned above, it is possible to perform physical attacks against the Ground Station. For instance, the attacker can perform **side channel attacks** to extract confidential data, or **hardware glitch attacks** to interfere with ground station by glitching the voltage. Alternatively, **memory interposing attacks** can snoop into or modify the memory by installing an interposer in ground station’s DRAM. However, based on the practical operational environment of the Ground Station, those physical attacks are not taken into consideration, since physical security protection like access control and surveillance is a much more effective and easy-to-deploy solution.

At the Table 8 below all attack scenarios evaluated of high impact are enlisted along with the respective countermeasures.

TABLE 8: SMART SATELLITES CRITICAL ATTACK SCENARIOS

Attack Scenario	Description	Criticality	Counter-measure
Key extraction	**Scenario**: This attack targets the ground station that run specific applications performing cryptographic operations with keys, such as software services communicating with CubeSat and external entities. The attacker acquires timely memory extraction to exfiltrate the cryptographic key material exposed during runtime of those applications. **Impact**: Compromise backward and forward data confidentiality of exchanged data. **Protection Goal**: cryptographic key material can only be accessed by authorized partners and only in predefined ways.	**High**	Runtime attestation for verifying the integrity of key access operations
Code Injection	**Scenario**: The attacker exploits software vulnerabilities and injects malicious code on the ground station. By means of compromised ground station, the attacker can further inject malicious code on the CubeSats. **Impact**: Perform unexpected operations on ground station and CubeSats. Compromise the data integrity and confidentiality of ground station and CubeSats **Protection Goal**: Unauthorized code should not be executed.	**High**	Static attestation and runtime attestation before executing a specific operation.
Runtime attack	**Scenario**: This attack targets ground station services. An attacker exploits memory corruption vulnerabilities on ground station's services to change the legitimate control	**High**	Runtime attestation for verifying control flow integrity before

Funded by the Horizon 2020 Framework Programme of the European Union

	flow of those services, in order to perform unauthorized operations and obtain unauthorized access to the memory, and extract information such as the TLS key. This attack helps the adversary to bypass common security enforcement, such as DEP. By compromising the Ground Station, the attacker can further attack CubeSats. **Impact**: Divert the operational behaviour of edge devices, compromise data confidentiality and integrity. **Protection Goal**: Abnormal control flow behaviour should be detected, and new protection policies should be deployed.		executing a specific operation.
Malicious updates	**Scenario**: This attack targets the ground station and CubeSats. An attacker can fool the admin and operator or force an automated process to install malicious updates on ground station or CubeSats. **Impact**: Compromise commands that are send to CubeSats. **Protection Goal**: Only legitimate software updates from known sources can be installed in ground station and CubeSats.	**High**	Static attestation for verifying binary signature before distributing updated version of the mission application.

5.4.2 Model of Use-cases Properties

In the previous section we specify the most critical attack scenarios for smart satellites use case and the relevant protection goals. This section will elaborate on how ASSURED meets each objective by attesting and verifying specific properties. Table 9 presents the mapping between the identified attacks and the corresponding validation properties.

TABLE 9: MAPPING OF VALIDATION PROPERTIES FOR SMART SATELLITES USE CASE

Attack Scenario	Target Systems	Processes	ASSURED Security Enabler	Validation Property
Key extraction	**Ground Station (GS).**	Ground Station software services communicating with CubeSats and external entities (during key establishment).	Dynamic properties	Control flow information of related processes before proceeding with key establishment.
Code Injection (Runtime)	**Ground Station (GS). CubeSat**	**Ground Station software services. All services included in a specific operation should be attested before proceeding with operation.** For example, the S/W distribution service (at the GS side) and File Transfer Service (at	Static Properties and dynamic properties	Attestation of loaded binaries. Control flow information of all processes. Configuration files of the services.

Funded by the Horizon 2020 Framework Programme of the European Union

		CubeSat side) should be attested before proceeding with the distribution of new version of mission software application. Also core services of CubeSats OS should be attested (KUBOS).		Changes to static code of the services Changes to data flow exchange with other services.
Runtime Attack	**Ground Station (GS), CubeSat**	**Ground Station software services. All services included in a specific operation should be attested before proceeding with operation.** For example, the S/W distribution service (at the GS side) and File Transfer Service (at CubeSat side) should be attested before proceeding with the distribution of new version of mission software application. Also core services of CubeSats OS should be attested (KUBOS).	Dynamic Properties	Control flow information of all processes. Configuration files of the services. Changes to static code of the services Changes to data flow exchange with other services.
Malicious update	**Ground Station (GS), CubeSat**	All Ground Station and CubeSats software	Static Properties	Attestation of the integrity of software updates. Configuration files of the services. Libraries / Dependency changes in the services during updates

Funded by the Horizon 2020 Framework Programme of the European Union

6 CONCLUSION

This final section will act as a synopsis of the deliverable and summarize its findings. The scope of this deliverable was to set the scene for the design of the novel remote attestation schemes [3] (e.g., Configuration Integrity Verification, Control-flow Attestation, Swarm Attestation and Jury-based Attestation) towards the creation of **trust- and privacy-aware service graph chains**. In this context, one important parameter to define is **the type of properties, of all the hardware and software assets of the SoS-enabled ecosystem, that need to be attested for achieving the desired level of trustworthiness**. Recall that one of the key contributions of ASSURED is the definition of a framework that will allow the use of property-based attestation to corroborate the fundamental security (and non-security) properties of single assets and extend this to a larger SoS. Essentially, the endmost goal is to provide the means *to reason on the overall security capabilities of complex SoS based on properties attested on the level of single components in order to assess the security of an SoS-enabled ecosystem both during* ***design- and run-time*** and enforce security policies based on such assessments.

Towards this direction, this deliverable provided a detailed **break-down of all static and run-time system properties** that, when considered for attestation, can provide verifiable evidence on the level of assurance of a system. These include resources ranging from **concrete configuration properties to low-level behavioural execution properties** covering all phases of a device's execution; from the **trusted boot and integrity measurement** of a CPS (e.g., list of allowed binaries loaded, type of hardware present, type of firmware present, etc.) to **run-time execution** (e.g., control-flow information, data-flow information, etc.) of only those safety-critical functions that have strong integrity and operational correctness requirements. This enables the vision of ASSURED towards employing advanced attestation schemes, for verifying the integrity (during both design- and run-time) of the target system, but **not of the entire (untrusted) code base** – which is rather impractical – but only of those properties that have the higher footprint on the overall SoS operation and level of assurance.

This compartmentalization also led to the identification of the **behavioural properties to be attested and the modelling of the critical software components in the context of all of the envisioned use cases** which will, in turn, dictate the resources that need to be considered for attestation when calculating the optimal set of attestation policies to be enforced [2]. This was also based on the instantiation of an **artefact-centric modelling notation** for representing all the technical operations, within a SoS-enabled ecosystem, in such a way so that we can extract the **acceptable state of all safety-critical operations**. Essentially, this is the model that can be leveraged by a System Administrator that wishes to identify all **relationships between the core assets in the target SoS** so that she can then capture **the control-flow among the activities and processes running within a system and in a hierarchical composition of systems**. This provides the **trusted reference values** of what is expected as a normal behaviour and against which real-time monitored states will be verified.

Overall, through this **codification of trust among computing entities** (that potentially are composed of possibly insecure – heterogeneous – hardware and software components), this deliverable puts forth a direct mapping of the system properties that need to be attested for protecting against specific type of vulnerabilities. This information will be further processed in the context of D2.1 [63] where a detailed threat analysis will be conveyed, in order to identify both the attestation tasks and the resources to be attested as a detection measure against the most prominent and impactful vulnerabilities. This, in turn, will be one of the input pipelines to the Policy Recommendation Engine [2] in order to calculate the optimal scheduling of attestation tasks that need to be enforced to all hardware assets towards achieving the desired level of trustworthiness – *not only for single systems as standalone components but also for the entire composition of systems capturing all the internal relationships and trust calculations*.

ABBREVIATIONS

Abbreviation	Description
ABAC	Attribute-based Access Control
ABE	Attribute Based Encryption
AK	Attestation Key
API	Application Programming Interface
ARP	Address Resolution Protocol
ASLR	Address Space layout Randomization
BFT	Byzantine Fault Tolerance
BGP	Byzantine Generals Problem
BPMN	Business Process Modelling Notation
CA	Certification Authority
CFA	Control-flow Attestation
CFG	Control-flow Graph
CFI	Control-flow Integrity
CIV	Configuration Integrity Verification
CMMN	Case Management Modelling Notation
CP-ABE	Ciphertext Policy Attribute Based Encryption
CPS	Cyber-Physical System
CRED	AK Credential
DAA	Direct Anonymous Attestation
DApps	Distributed Applications

Funded by the Horizon 2020 Framework Programme of the European Union

DEP	Data Execution Prevention
DID	Decentralized identifier Identification
DLT	Distributed Ledger Technology
DoA	Description of Action
DPA	Differential Power Analysis
DPos	Delegated Proof of Stake
Dx.x	Deliverable x.xl
ECC	Elliptic-Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EK	Endorsement Key
FMS	Flight Management System
HLF	Hyperledger Fabric
ICS	Industrial Control System
IoT	Internet of Things
JIT-ROP	Just-in-Time Oriented Programming
JOP	Jump-Oriented Programming
KDF	Key Derivation Function
KEP	Key Exposure Problem
KEW	Key Exposure Window
KP-ABE	Key Policy Attribute Based Encryption
MITM	Man-in-the-Middle
MSP	Membership Service Provider

Funded by the Horizon 2020 Framework Programme of the European Union

NO	Network Orchestrator
PBFT	Practical Byzantine Fault Tolerance
PCR	Platform Configuration Register
PLC	Programmable Logic Controller
PK	Public Key
RA	Remote Attestation
ROP	Return-Oriented Programming
SCB	Security Context Broker
SE	Searchable Encryption
SGX	Software Guard Extensions
SK	Secret Key
SoS	Systems of Systems
TCB	Trusted Computing Base
TEE	Trusted Execution Environment
TOCTOU	Time-of-Check-Time-of-Use
TPM	Trusted Platform Module
WPx	Work Package X

Funded by the Horizon 2020
Framework Programme of the European Union

REFERENCES

[1] «OWASP Top 10: Open Web Application Security Project,» [Online]. Available: https://sharedassessments.org/blog/owasp-top-10-open-web-application-security-project/. [Consultato il giorno 28 9 2021].

[2] «D2.2 – Policy Modelling & Cybersecurity, Privacy and Trust Policy Constraints,» *The ASSURED Consortium,* November 2021.

[3] «D3.2 – ASSURED Layered Attestation and Runtime Verification Enablers Design & Implementation,» *The ASSURED Consortium,* November 2021.

[4] «The Industrial Internet Reference Architecture v1.9,» Industry IoT Consortium, [Online]. Available: https://www.iiconsortium.org/IIRA.htm. [Consultato il giorno 2 2022].

[5] «D1.4 - Report on Security, Privacy and Accountability Models for Dynamic Trusted Consent and Data Sharing,» *The ASSURED Consortium,* August 2021.

[6] C. O. a. A. Ziółkowski, A. Orlowski, P. Kaplanski, T. Sitek e W. Pokrzywnicki, «Implementation of Business Processes in Smart Cities Technology,» *Trans. Comput. Collect. Intell.,* vol. 25, pp. 15-28, 2016.

[7] B. Re, R. Cognini, F. Corradini e A. Polini, «Modelling Process Intensive Scenarios for the Smart City,» 09 2014.

[8] P. Neirotti, A. de Marco, A. C. Cagliano, G. Mangano e F. Scorrano, «Current trends in Smart City initiatives: Some stylised facts,» *Cities,* vol. 38, pp. 25-36, 2014.

[9] E. Ortner, M. Mevius, P. Wiedmann e F. Kurz, «Design of Interactional End-to-End Web Applications for Smart Cities,» in *Proceedings of the 24th International Conference on World Wide Web*, Florence, Italy, 2015.

[10] I. Compagnucci, F. Corradini, F. Fornari, A. Polini, B. Re e F. Tiezzi, «Modelling Notations for IoT-Aware Business Processes: A Systematic Literature Review,» *Business Process Management Workshops,* pp. 108-121, 2020.

[11] J. Kopke, G. Meroni e M. Salnitri, «SecBPMN2BC case studies evaluation,» *Mendeley Data,* vol. 1.

[12] K. Angelopoulos, V. Diamantopoulou, M. Pavlidis, H. Mouratidis, M. Salnitri, J. F. Ruiz e P. Giorgini, «A Holistic Approach for Privacy Protection in E-Government,» in *ARES '17: Proceedings of the 12th International Conference on Availability, Reliability and Security*, Reggio Calabria, Italy, 2017.

[13] M. Salnitri, J. Jürjens, H. Mouratidis, L. Mancini e P. G. e. al., Visual Privacy Management. Design and Applications of a Privacy-Enabling Platform, Springer, 2020.

[14] M. C. Duncan, «Trust Management and Security in Satellite Telecommand Processing,» vol. 1380, 2011.

Funded by the Horizon 2020 Framework Programme of the European Union

[15] W. W. Zhao e V. Varadharajan, An Approach to Unified Trust Management Framework, ch. An Approach to Unified Trust Management Framework, 2009, pp. 111-134.

[16] M. Blaze, J. Feigenbaum, J. Ioannidis e A. Keromytis, «The KeyNote Trust Management System,» *Matt Blaze, Using the KeyNote Trust Management System,* 2001.

[17] M. Manulis, C. Bridges e R. a. Harrison, «Cyber security in New Space,» *Int. J. Inf. Secur.,* vol. 20, p. 287–311, 2011.

[18] «Multichain,» [Online]. Available: https://www.multichain.com/. [Consultato il giorno 2 2022].

[19] «MultiChain Github,» [Online]. Available: https://github.com/MultiChain. [Consultato il giorno 2 2022].

[20] «Doublechain,» [Online]. Available: https://doublechain.co.kr/platform.php. [Consultato il giorno 2 2022].

[21] «Origintrail,» [Online]. Available: https://origintrail.io/. [Consultato il giorno 2 2022].

[22] «Waltonchain,» [Online]. Available: https://waltonchain.org/. [Consultato il giorno 2 2021].

[23] «IOTA,» [Online]. Available: https://www.iota.org/. [Consultato il giorno 2 2022].

[24] «IOTA Industry Marketplace,» [Online]. Available: https://industrymarketplace.net/. [Consultato il giorno 2 2022].

[25] K. Sampigethaya e R. Poovendran, «Aviation Cyber–Physical Systems: Foundations for Future Aircraft and Air Transport,» *Proceedings of the IEEE,* vol. 101, n. 8, pp. 1834-1855, August 2013.

[26] «ARINC 827 ELECTRONIC DISTRIBUTION OF SOFTWARE BY CRATE (EDS CRATE),» *ARINC,* 10 August 2020.

[27] «ARINC 835 GUIDANCE FOR SECURITY OF LOADABLE SOFTWARE PARTS USING DIGITAL SIGNATURES,» *ARINC,* 2 January 2014.

[28] J. Becker, M. Indulska, M. Rosemann e P. Green, «Do process modelling techniques get better?,» in *Proceedings of the 16th Australasian Conference on Information Systems, Australasian Chapter of the Association for Information Systems*, Sydney, Australia, 2015.

[29] «Business proces modelling notation (BPMN) version 2.02,» Object Managment Group, 2014. [Online]. Available: https://www.omg.org/spec/BPMN/2.0.2/. [Consultato il giorno 2 2022].

[30] M. Rosemann, A. Schwegmann e P. Delfmann, «Vorbereitung der Prozessmodellierung,» *Prozessmanagement, Springer Gabler, Berlin, Heidelberg,* 2012.

[31] C. Ouyang, M. Dumas, W. Aalst, A. Hofstede e J. Mendling, «From business process models to process-oriented software systems,» *ACM Transactions on Software Engineering and Methodology,* vol. 19, n. 1, pp. 1-37.

Funded by the Horizon 2020 Framework Programme of the European Union

[32] R. Seiger, R. Kühn e M. e. a. Korzetz, «HoloFlows: modelling of processes for the Internet of Things in mixed reality,» *Softw Syst Model,* vol. 20, pp. 1465-1489, 2021.

[33] R. R. Mukkamala, T. Hildebrandt e T. Slaats, «Towards Trustworthy Adaptive Case Management with Dynamic Condition Response Graphs,» in *17th IEEE International Enterprise Distributed Object Computing Conference*, 2013.

[34] F. Milani, L. Garcia-Banuelos, S. Filipova e M. Markovska, «Modelling Blockchain-based business processes: a comparative analysis of BPMN vs CMMN,» *Business Process Management Journal,* vol. 27, n. 2, pp. 638-657, 2021.

[35] C. R. Carvalho, H. Mili, A. Boubaker, J. Gonzalez-Huerta e S. Ringuette, «On the analysis of CMMN expressiveness: revisiting workflow patterns,» in *5th Adaptive Case Management Workshop (AdaptiveCM), Colocated with EDOC*, 2016.

[36] T. Kala, F. Maggi, C. D. Ciccio e C. D. Francescomarino, «Apriori and sequence analysis for discovering declarative process models,» in *Proc. IEEE 20th Int. Enterp. Distrib. Object Comput. Conf. EDOC*, 2016.

[37] «Decision Model and Notation (DMN) Tutorial,» [Online]. Available: https://www.processmaker.com/blog/decision-model-and-notation-dmn-tutorial-examples/. [Consultato il giorno 2 11 2020].

[38] N. Gol Mohammadi e M. Heisel, «Enhancing Business Process Models with Trustworthiness Requirements,» in *Trust Management X*, Springer International Publishing, 2016, pp. 33-51.

[39] M. Rosemann, «Trust-Aware Process Design,» in *Business Process Management*, T. Hildebrandt, Boudewijn F. van Dongen; M. Roglinger, J. Mendling, 2019, pp. 305-321.

[40] M. Müller, N. Ostern, D. Koljada, K. Grunert, M. Rosemann e A. Küpper, «Trust Mining: Analyzing Trust in Collaborative Business Processes,» *IEEE Access,* vol. 9, pp. 65044-65065, 2021.

[41] «RAMI 4.1 Reference Architectural Model for Industrie 4.0,» InTech, [Online]. Available: https://www.isa.org/intech-home/2019/march-april/features/rami-4-0-reference-architectural-model-for-industr. [Consultato il giorno 2 2022].

[42] «D4.1 – ASSURED Blockchain Architecture,» *The ASSURED Consortium,* November 2021.

[43] «D2.7 – ASSURED Collective Threat Intelligence Analysis & Forecasting Framework,» *The ASSURED Consortium,* February 2022.

[44] «D3.1 – ASSURED Attestation Model and Specification,» *The ASSURED Consortium,* November 2021.

[45] N. Koutroumpouchos et al., «Secure Edge Computing with Lightweight Control-Flow Property-based Attestation,» in *IEEE Conference on Network Softwarization (NetSoft)*, 2019.

[46] «The ASSURED Consortium,» *D3.4 – ASSURED Real-time Monitoring and Tracing Functionalities,* February 2022.

Funded by the Horizon 2020 Framework Programme of the European Union

[47] H. B. Debes e T. Giannetsos, «Segregating Keys from noncense: Timely Exfil of Ephemeral Keys from Embedded Systems,» in *Distibruted Security Conference for Sensor Systems (DCOSS)*, 2021.

[48] «IBM's TPM 2.0 TSS,» IBM, [Online]. Available: https://sourceforge.net/projects/ibmtpm20tss/. [Consultato il giorno 2 2022].

[49] «Intel TSS,» Intel, [Online]. Available: https://github.com/tpm2-software/tpm2-tss. [Consultato il giorno 2 2022].

[50] «Microsoft TSS,» Microsoft, [Online]. Available: https://github.com/Microsoft/TSS.MSR. [Consultato il giorno 2 2022].

[51] «Trousers TSS,» [Online]. Available: http://trousers.sourceforge.net/. [Consultato il giorno 2 2022].

[52] «Java TSS,» [Online]. Available: https://sourceforge.net/projects/trustedjava/. [Consultato il giorno 2 2022].

[53] W. Mao, Y. Fei e C. Chunrun, «Daonity: grid security with behaviour conformity from trusted computing,» in *First ACM workshop on Scalable trusted computing*, 2006.

[54] F. Ma, J. Menéndez, M. Oliva e J. Ríos, «Collaborative engineering: An airbus case study,» in *International Conference of The Manufacturing Engineering Society*, 2013.

[55] «Multi annual strategic research and innovation agenda for the ecsel joint undertaking,» Technical report, ECSEL, 2015.

[56] J. Menéndez, F. Mas, J. Serván e J. Ríos., Virtual verification of an aircraft final assembly line industrialization: An industrial case., Advances in Manufacturing Systems, 2012.

[57] J. R. Walters e N. R. Nielsen., Crafting knowledge-based systems: Expert systems made realistic., John Wiley & Sons, Inc., 1988.

[58] B. Tang, «Toward intelligent cyber-physical systems: Algorithms, architectures, and applications,» 2016.

[59] T. Dreossi, A. Donzé e S. A. Seshia, «Compositional falsification of cyber-physical systems with machine learning components,» *Journal of Automated Reasoning 63.4,* pp. 1031-1053, 2019.

[60] K. R. Varshney e H. Alemzadeh., «On the safety of machine learning: Cyber-physical systems, decision sciences, and data products,» *Big data 5.3,* pp. 246-255, 2017.

[61] L. Sha, S. Gopalakrishnan, X. Liu e Q. Wang., «Cyber-physical systems: A new frontier,» in *IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing*, 2008.

[62] C. M. Holloway, Understanding the Overarching Properties, Hampton VA: National Aeronautics and Space Administration, Langley Research Center, 2019.

[63] «D2.1 – Risk Assessment Methodology and Threat Modelling,» *The ASSURED Consortium,* November 2021.

Funded by the Horizon 2020 Framework Programme of the European Union

[64] T. Abera, N. Asokan, L. Davi, F. Koushanfar, A. Paverd, A.-R. Sadeghi e G. Tsudik., «Things, trouble, trust: On building trust in IoT systems,» in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016.

[65] M. Ambrosin, M. Conti, R. Lazzeretti, M. M. Rabbani e S. Ranise., «Collective Remote Attestation at the Internet of Things Scale: State-of-the-art and Future Challenges,» in *IEEE Communications Surveys & Tutorials*, 2020.

[66] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz e A. Sahai, «Exposure-resilient functions and all-or-nothing transforms,» in *International Conference on the Theory and Applications of Cryptographic Techniques*, Berlin, Heidelberg, 2000.

[67] B. Kaplan, «Ram is key extracting disk encryption keys from volatile memory,» 2007.

[68] NVD, *CVE-2020-9395 Detail,* 2020.

[69] H. B. Debes e T. Giannetsos., «Segregating Keys from noncense: Timely Exfil of Ephemeral Keys from Embedded Systems,» in *Wireless Security Conference* , 2021.

[70] S. Bratus, N. D'Cunha, E. Sparks e S. W. Smith, «TOCTOU, traps, and trusted computing,» in *International Conference on Trusted Computing*, Berlin, Heidelberg, 2008.

[71] B. Larsen, H. B. Debes e T. Giannetsos, «CloudVaults: Integrating Trust Extensions into System Integrity Verification for Cloud-Based Environments,» in *European Symposium on Research in Computer Security*, 2020.

[72] Nunes, I. D. Oliveira, S. Jakkamsetti, N. Rattanavipanon e G. Tsudik., «On the TOCTOU problem in remote attestation.,» 2020.

[73] «Microsoft Corporation, Data Execution Prevention,» 2005. [Online]. Available: http://technet2.microsoft.com/WindowsServer/en/library/b0de1052-4101-44c3-a294-4da1bd1ef2271033. mspx?mf r=true.

[74] S. Designer., «lpr LIBC RETURN exploit,» 1997. [Online]. Available: http://insecure.org/.

[75] H. Shacham, «The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86),» in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007.

[76] S. Checkoway, L. Davi, A. Dmitrienko, A.-R. Sadeghi, H. Shacham e M. Winandy, «Return-oriented programming without returns,» in *Proceedings of the 17th ACM conference on Computer and communications security*, 2010.

[77] T. Bletsch, X. Jiang, V. W. Freeh e Z. Liang., «Jump-oriented programming: a new class of code-reuse attack,» in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, 2011.

[78] K. Z. Snow, F. Monrose, L. Davi, A. Dmitrienko, C. Liebchen e A.-R. Sadeghi, «Just-in-time code reuse: On the effectiveness of fine-grained address space layout randomization,» in *IEEE Symposium on Security and Privacy*, 2013.

[79] N. Mehta, «Heartbleed,» [Online]. Available: https://plus.google.com/+ MarkJCox/posts/TmCbp3BhJma, 2014.

Funded by the Horizon 2020
Framework Programme of the European Union

[80] S. Chen, J. Xu, E. C. Sezer, P. Gauriar e R. K. Iyer., «Non-Control-Data Attacks Are Realistic Threats,» in *USENIX Security Symposium*, 2005.

[81] H. Hu, S. Shinde, S. Adrian, Z. L. Chua, P. Saxena e Z. Liang, «Data-oriented programming: On the expressiveness of non-control data attacks,» in *IEEE Symposium on Security and Privacy (SP)*, 2016.

[82] «CAPEC-186: Malicious Software Update,» [Online]. Available: https://capec.mitre.org/data/definitions/186.html.

[83] M. Srivastava, An Introduction to Network Security Attacks, Springer, Singapore, 2021.

[84] R. H. Jhaveri, S. J. Patel e D. C. Jinwala., «DoS attacks in mobile ad hoc networks: A survey,» in *second international conference on advanced computing & communication technologies. IEEE*, 2012.

[85] H. Mustafa, W. Xu, A. R. Sadeghi e S. Schulz., «You can call but you can't hide: detecting caller id spoofing attacks».

[86] B. Han, ShreyaTayade e H. D.Schotten., «Modeling profit of sliced 5G networks for advanced nework resource management and slice implementation,» in *IEEE Symposium on Computers and Communications (ISCC)*, 2017.

[87] A.Ledjiar, E.Sampin, C.Talhi e M.Cheriet, «Network Function Virtualization as a Service for multi-tenant software defined networks,» in *4th Int. Conf. on Software Defined Systems (SDS)*, Valencia, 2017.

[88] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini e H. Flinck., « Network slicing andsoftwarization: A survey on principles, enabling technologies, and solutions.,» in *IEEE Communica-tions Surveys & Tutorials 20*, 2018.

[89] «5G-NORMA,» [Online]. Available: http://www.it.uc3m.es/wnl/5gnorma/.

[90] 5G-MoNArch. [Online]. Available: https://5g-monarch.eu/.

[91] P. Kocher, J. Jaffe e B. Jun., «Differential power analysis,» in *Annual international cryptology conference*, 1999.

[92] Y. Zhou e D. Feng, «Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing,» in *IACR Cryptol. ePrint Arch*, 2005 .

[93] O'Flynn, Colin, Zhizhang e D. Chen., «Chipwhisperer: An open-source platform for hardware embedded security research,» in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, 2014.

[94] S. Mangard, N. Pramstaller e E. Oswald., « Successfully attacking masked AES hardware implementations,» in *International workshop on cryptographic hardware and embedded systems*, 2005.

[95] D. Lee, D. Jung, I. T. Fang, C.-C. Tsai e R. A. Popa, «An off-chip attack on hardware enclaves via the memory bus,» in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

Funded by the Horizon 2020 Framework Programme of the European Union

[96] «ASSURED Use Cases and System Requirements,» *The ASSURED Consortium,* 8 3 2021.

[97] G. Dessouky, T. Abera, A. Ibrahim e A.-R. Sadeghi., «Litehax: lightweight hardware-assisted attestation of program execution,» in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018.

[98] L. Gu, X. Ding, R. H. Deng, B. Xie e H. Mei., «Remote attestation on program execution,» in *Proceedings of the 3rd ACM workshop on Scalable trusted computing (STC '08). Association for Computing Machinery*, New York, NY, USA, 2008.

[99] «D4.3 - ASSURED Blockchain-based Control Services and Crypto functions for Decentralized Data Storage, Sharing and Access Control - version 1,» *The ASSURED Consortium,* March 2022.

Funded by the Horizon 2020
Framework Programme of the European Union