D4.8 – FINAL VERSION OF CPSOS RUNTIME SECURITY MONITORING APPROACHES

Authors	Antonio Álvarez (ATOS), Miguel Martín (ATOS), Apostolos Fournaris (ISI), Aris Lalos (ISI), Francesco Regazzoni (USI), Wojciech Javorski (RTC)

Work Package	WP4

Abstract

This deliverable is of demonstrator type and describes the demonstrator for the final Security Runtime Monitoring and Management (SRMM) module of the CPSoSAware architecture. The SRMM detects and warns about any suspicious and malicious activity that may threaten the overall security properties of the system. The demonstrator proposes a three-layered architecture with SRMM at the three levels starting by lightweight versions in the lowest level and a full-capacity version on the top layer

Funded by the Horizon 2020 Framework Programme
of the European Union

Deliverable Information

Work Package	WP4 – CPSoSAware System Layer Design and adaptation of dependable CP(H)SoS
Task	T4.3 – CPSoSAware Security Runtime Monitoring and Management (SRMMI Design and Development
Deliverable title	D4.8 Final version of CPSoS Runtime Security Monitoring Approaches
Dissemination Level	Public
Status	Final
Version Number	1.0
Due date	M28 (30/04/2022)

Project Information

Project start and duration	01/01/2020 – 31/12/2022, 36 months
Project Coordinator	Industrial Systems Institute, ATHENA Research and Innovation Center 26504, Rio-Patras, Greece
Partners	1. ATHINA-EREVNITIKO KENTRO KAINOTOMIAS STIS TECHNOLOGIES TIS PLIROFORIAS, TON EPIKOINONION KAI TIS GNOSIS (ISI) * Coordinator 2. FUNDACIO PRIVADA I2CAT, INTERNET I INNOVACIO DIGITAL A CATALUNYA (I2CAT), 3. IBM ISRAEL - SCIENCE AND TECHNOLOGY LTD (IBM ISRAEL 4. ATOS SPAIN SA (ATOS), 5. PANASONIC AUTOMOTIVE SYSTEMS EUROPE GMBH (PASEU) 6. EIGHT BELLS LTD (8BELLS) 7. UNIVERSITA DELLA SVIZZERA ITALIANA (USI), 8. TAMPEREEN KORKEAKOULUSAATIO SR (TAU) 9. UNIVERSITY OF PELOPONNESE (UoP) 10. CATALINK LIMITED (CATALINK) 11. ROBOTEC.AI SPOLKA Z OGRANICZONA ODPOWIEDZIALNOSCIA (RTC) 12. CENTRO RICERCHE FIAT SCPA (CRF) 13. PANEPISTIMIO PATRON (UPAT)
Website	www.cpsosaware.eu

Control Sheet

VERSION	DATE	SUMMARY OF CHANGES	AUTHOR
0.0	30/03/2022	*Table of Contents and responsibilities*	Antonio Álvarez (ATOS)
0.1	05/04/2022	*Update in section 2.3.1*	Antonio Álvarez (ATOS), Miguel Martín (ATOS)
0.2	11/04/2022	*Updates in sections 2.1 and 3.1*	Antonio Álvarez (ATOS)
0.3	12/04/2022	*Adding section 2.2*	Antonio Álvarez (ATOS), Miguel Martín (ATOS)
0.4	13/04/2022	*Updating introduction to sections 3.3. Refining section 3.3.1. Updating section 3.3.3*	Antonio Álvarez (ATOS), Miguel Martín (ATOS)
0.5	18/04/2022	*Adding sections 2.3.2 and 2.4. Refinement of section 3.1*	Antonio Álvarez (ATOS), Miguel Martín (ATOS), Apostolos Fournaris (ISI)
0.6	20/04/2022	*Input to section 1*	Antonio Álvarez (ATOS)
0.7	20/04/2022	*Concluding section 1*	Antonio Álvarez (ATOS), Miguel Martín (ATOS)
0.8	21/04/2022	*Adding section 4. Adding acronyms list*	Antonio Álvarez (ATOS), Miguel Martín (ATOS)
0.9	25/04/2022	*Adding section 3.2. Adding section 5. Extending glossary table*	Antonio Álvarez (ATOS), Miguel Martín (ATOS)
0.10	26/04/2022	*Refinements in section 3.3.1. Adding executive summary*	Antonio Álvarez (ATOS), Miguel Martín (ATOS)
0.11	27/04/2022	*Adding abstract. Refinements in sections 2.3.1, 3.1, 3.3.3, 3.3.3.1, 3.3.3.2, 4.2, 4.3*	Antonio Álvarez (ATOS), Miguel

			Martín, Apostolos Fournaris (ISI), Francesco Regazzoni (USI), Wojciech Javorski (RTC)
0.12	28/04/2022	*Refinement in section 2.3.2*	Antonio Álvarez (ATOS)
0.*13*	10/05/2022	*Peer review and feedback processing*	Antonio Álvarez (ATOS), Apostolos Fournaris (ISI), Aris Lalos (ISI), Pekka Jääskelainen (TAU), Alessandro Zanella (CRF)
1.*0*	10/05/2022	*Version for delivery*	Antonio Álvarez (ATOS)

	NAME
Prepared by	Antonio Álvarez (ATOS)
Reviewed by	Pekka Jääskelainen (TAU), Alessandro Zanella (CRF)
Authorised by	Aris Lalos (ISI)

DATE	RECIPIENT
10/05/2022	Project Consortium
11/05/2022	European Commission

Table of Contents

List of Figures

List of Tables

List of Acronyms

Abbreviation / acronym	Description
ADAS	Advanced Driver Assistance System
AGV	Automated Guided Vehicle
APT	Advanced Persistent Threat
BEC	Business Email Compromise
BLE	Bluetooth Low Energy Protocol
CAN	Controller Area Network
CEP	Complex Event Processing
CISO	Chief Information Security Officer
CO	Confidential
CPS	Cyber Physical System
CPSoS	Cyber Physical System of Systems
CP(H)SoS	Cyber Physical (Human) System of Systems
CPU	Central Processing Unit
DCERPC	Distributed Computing Environment / Remote Procedure Calls
DoS	Denial of Service
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DNP3	Distributed Network Protocol 3
DNS	Domain Name System
ECU	Electronic Control Unit
ENIP/CIP	Ethernet/IP & Common Industrial Protocol
ENISA	European Union Agency for Cybersecurity
EPL	Event Processing Language
ERSPAN	Encapsulated Remote Switched Port Analyzer
FTP	File Transfer Protocol
GPS	Global Positioning System
GRE	Generic Routing Encapsulation
HIDS	Host Intrusion Detection System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IKEv2	Internet Key Exchange v2
JSON	JavaScript Object Notation
KRB5	Kerberos v5
LIDAR	Light Detection and Ranging
MiM	Man-in-the-Middle
MISP	Malware Information Sharing Platform
MPLS	Multiprotocol Label Switching
MQTT	Message Queue Telemetry Transport
NFS	Network File System

Abbreviation / acronym	Description
NIDS	Network Intrusion Detection System
NTP	Network Time Protocol
OBD	On Board Diagnosis
OSSIM	Open Source SIEM
PPP	Point-to-Point
PPPoE	Point-to-Point Protocol over Ethernet
QINQ	802.1.Q-in-802.1.Q
RDP	Remote Desktop Protocol
RFB	Remote Framebuffer
RIS	Remote Installation Sensor
RSU	Road Side Unit
SCTP	Stream Control Transmission Protocol
SDR	Software-Defined Radio
SIEM	Security Information and Event Management
SIP	Session Initiation Protocol
SMB	Server Message Block
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOC	Security Operation Center
SQL	Structured Query LAnguage
SRMM	Security Runtime Monitoring Module
SSH	Secure Shell
STIX	Structured Threat Information Expression
TCP	Transport Control Protocol
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VXLAN	Virtual Extensible LAN
V2X	Vehicle-to-Everything
WP	Work Package
WSN	Wireless Sensor Network

Executive Summary

Deliverable D4.8 is the final release coming from T4.3. In a nutshell, it compiles the relevant technical information about the Security Runtime Monitoring Module (SRMM). This module is based on XL-SIEM technology brought to the project by ATOS. The document addresses the internal architecture and operation, how it interfaces with the rest of the CPSoSAware architecture, and also proposes a practical application in the framework of the project. This demonstrator shows the strong potential of this technology for the resolution of real-world complex challenges that involve cyber-physical systems.

Going on details, it provides a threats and attacks landscape following a research work consulting documentation from relevant projects and initiatives. The top cyberthreats have been identified using a periodic report from ENISA. It builds on top of the concept of run-time security monitoring presented in D1.1 and further elaborates on it by presenting different types of sensors in charge of feeding with security information to a Security Information and Event Management (SIEM) system. Then the process of data collection, analysis, filtering and correlation to extract meaningful information issued under the format of an alarm is presented. It mentions the fact that alarms can be further correlated to raise alarms of a higher level and so on. The process works bottom-top and involves, on one side, the security analysis performed at sensor level and, on the other side, the one performed at SIEM level. The concept of security directive is presented and some examples are provided. How the information is reported and presented to the user, aiming at a quick reaction when needed, and some examples of meaningful security metrics, is another topic addressed in the document.

The Security Runtime Monitoring is performed in CPSoSAware by the component called SRMM (Security Runtime Monitoring and Management), which is, as said above, based on XL-SIEM technology. Its position in the CPSoSAware architecture and how it interfaces with other components (monitoring sensors and both CARLA and V2X simulators as inputs and the simulator themselves and other SRMMs placed in the architecture in the outputs) is specified in this document. The status of the requirements defined for the SRMM in CPSoSAware in the context of WP1 is updated. Also, both the internal architecture and the involved technology are described on detail. Data format in the different steps is addressed to provide some insights about how the information is processed to obtain relevant alarms to be presented to the user. To link it with demonstration scenarios, it is proposed a hierarchical architecture with collaborative SRMMs placed in three different layers. This architecture is implemented in practice within WP6 and is to be applied in the Automotive Pillar.

In connection with the mentioned architecture, this deliverable proposes a practical scenario in line with the Automotive Pillar of the project. The scenario aims at the detection of anomalies in real-time related to likely cyber-threats and attacks that could compromise the safety of the passengers in a car. The paramount goal is to preserve the correct and effective functioning and operation of the vehicle, punctually reporting security incidents and eventually enabling prompt corrective actions to mitigate their impact and minimize negative consequences. The document gives some examples of possible cyber-attacks that could be carried out in this context. Specific examples are proposed considering a distributed denial of service attack (DDoS), and firmware updates detection and mitigation at local, area and global levels.

1 Introduction

Deliverable D4.8, entitled *"Final Version of CPSoS Runtime Security Monitoring Approaches"* is the second and final delivery coming out of task T4.3 *"CPSoSAware Security Runtime Monitoring and Management (SRMM) Design and Development"*. T4.3 is part of WP4, entitled *"CPSoSAware System Layer Design and adaptation of dependable CP(H)SoS"*.

The document compiles the relevant technical information about the Security Runtime Monitoring Module (SRMM). This module is based on XL-SIEM technology brought to the project by ATOS. The document addresses the internal architecture and operation, how it interfaces with the rest of the CPSoSAware architecture, and also provides a demonstrator of practical application in the framework of the project. This demonstrator shows the strong potential of this technology for the resolution of real-world complex challenges that involve cyber-physical systems.

The document is structured as follows:

- Section 1 is the introduction to the document.
- Section 2 presents the Security Runtime Monitoring and Threat Detection functionality in the context of CPSoSAware.
- Section 3 is a description of the final version of the SRMM architecture. It also describes the XL-SIEM technology.
- Section 4 provides an updated application scenario to the Autonomous Vehicles case.
- Section 5 closes the document and presents a road ahead upon conclusion of T4.3.

Apart from the rest of tasks of WP4, T4.3 has connections and synergies with:

- T1.3, that defines the architecture and therefore provides input on the positioning of the SRMM within the CPSoSAware architecture
- T3.5, that provides monitoring sensors assets that produce logs which are processed by the agent with the help of the different plugins and subsequently injected to the SRMM for further filtering and correlation.
- T3.1 and T3.3 for the interfacing with the CARLA simulator
- T5.2 for integration purposes
- T6.2 and T6.3 for the application to the Autonomous Vehicles case

D4.8 is the continuation of the work presented in D4.3. We have adopted the approach of presenting a stand-alone document that contains the results of the work performed during T4.3 lifecycle. This way the reader does not need to visit D4.3, as all the relevant content is in D4.8. This means that some parts of the document have been left untouched, others have been modified to a greater or lesser extent, while some parts have been written from scratch. The following table highlights the differences between both documents.

Table 1. Comparison between deliverables D4.8 and D4.3

Section in D4.8	Section in D4.3	What is new
1. Introduction	1. Introduction	Extending the introduction, providing more details on how

		T4.3 interfaces with other activities in the project, and introducing a comparison table between D4.8 and D4.3
2. Security Monitoring and Threat Detection	2. Security Monitoring and Threat Detection	The section has been updated
2.1. Threats and attacks landscape	2.1. Threats and attacks landscape	Update introduced with information from ENISA Threat Landscape report for the period 2020-2021
2.2. Runtime Security Monitoring in the CPSoS context	2.2. Runtime Security Monitoring in the CPSoS context	No changes
2.3. Security analysis for detection of anomalies and threats	2.3. Security analysis for detection of anomalies and threats	The section has been updated
2.3.1. Security analysis performed at sensor level	2.3.1. Security analysis performed at sensor level	The table about CPSoSAware security monitoring sensors has been extended An example on how a sensor/agent is integrated, including the development of ad-hoc plugins, has been documented
2.3.2. Security analysis performed at SIEM level	2.3.2. Security analysis performed at SIEM level	A list of security directives identified for the project has been added
2.4. Reporting	2.4. Reporting	The list of security metrics has been extended
3. Security Runtime Monitoring and Management: final architecture design	3. Security Runtime Monitoring and Management (SRMM): preliminary architecture design	The section has been updated
3.1. Positioning of the SRMM in the CPSoSAware architecture and requirements	3.1. Positioning of the SRMM in the CPSoSAware architecture and requirements	Added figure from D1.4 in which SRMM and its neighbours are represented Explaining interfaces and their purpose Status of SRMM requirements updated at the time of the delivery of this document
3.2. SRMM internal architecture description	3.2. SRMM internal architecture description	The section has been updated in alignment with the content of D1.4 Discussing scalability
3.3. Technology description: XL-SIEM	4.1. Technology description: XL-SIEM	The section has been updated

3.3.1. Agents and plugins	4.1.2. Agents and plugins	The figure about the SRMM agent in CPSoSAware has been updated in line with the sensors identified. A fragment of a new plugin developed has been introduced as example
3.3.2. Input data and format	4.1.1. Input data and format	Change in the input-output example, just at the end of the section
3.3.3. XL-SIEM Storm topology: event processing and security analysis	4.1.3. Event processing and security analysis	More insights have been added on how rules are inspired and updates in examples
3.3.4. Alarms format, export and data sharing	4.1.4. Alarms format, export and data sharing	No changes
4. Demonstration scenario	4.2. Demonstration scenario	Fully updated demonstrator
4.1. Distributed Denial of Service (DDoS) Attack demonstration		Minor changes with respect to D4.3
4.2. Local firmware update detection and mitigation configuration		Added in D4.8
4.3. Area firmware update detection and mitigation configuration		Added in D4.8
4.4. Global firmware update detection and mitigation configuration		Added in D4.8
5. Conclusions and next steps	5. Conclusions and next steps	The section has been updated

2 Security Runtime Monitoring and Threat Detection

2.1 Threats and attacks landscape

Deliverable D1.1 [2] reviewed, in its section 7.2, the threat models, taxonomies and attacks classifications proposed by relevant projects and initiatives of relevance for the different CPSoSAware architectural domains: system, communications and device. The following table summarizes the work presented in the aforementioned deliverable and section.

Table 2. Threats and attacks landscape [32]

Domain	Related Asset / Component	Threats / Attacks
System		Physical attack (deliberate/ intentional) – Fraud Physical attack (deliberate/ intentional) – Sabotage Physical attack (deliberate/ intentional) – Vandalism Physical attack (deliberate/ intentional) Theft (devices, storage media and documents) Physical attack (deliberate/ intentional) - Information leakage/sharing Physical attack (deliberate/ intentional) Unauthorized physical access / Unauthorised entry to premises Physical attack (deliberate/ intentional) - Coercion, extortion or corruption Physical attack (deliberate/ intentional) - Damage from the warfare Physical attack (deliberate/ intentional) - Terrorist attack Unintentional damage / loss of information or IT assets Information leakage/sharing due to human error Unintentional damage / loss of information or IT assets Erroneous use or administration of devices and systems Unintentional damage / loss of information or IT assets Using information from an unreliable source Unintentional damage / loss of information or IT assets Unintentional change of data in an information system Unintentional damage / loss of information or IT assets Inadequate design and planning or improperly adaptation

		Unintentional damage / loss of information or IT assets - Damage caused by a third party Unintentional damage / loss of information or IT assets - Damages resulting from penetration testing Unintentional damage / loss of information or IT assets - Loss of information in the cloud Unintentional damage / loss of information or IT assets - Loss of (integrity of) sensitive information Unintentional damage / loss of information or IT assets - Loss of devices, storage media and documents Unintentional damage / loss of information or IT assets - Destruction of records Disaster (natural, environmental) - Disaster (natural earthquakes, floods, landslides, tsunamis, heavy rains, heavy snowfalls, heavy winds) Disaster (natural, environmental) – Fire Disaster (natural, environmental) - Pollution, dust, corrosion Disaster (natural, environmental) - Thunder stroke Disaster (natural, environmental) – Water Disaster (natural, environmental) – Explosion Disaster (natural, environmental) - Dangerous radiation leak Disaster (natural, environmental) - Unfavourable climatic conditions Disaster (natural, environmental) - Major events in the environment Disaster (natural, environmental) - Threats from space / Electromagnetic storm Disaster (natural, environmental) – Wildlife Failures/ Malfunction - Failure of devices or systems Failures/ Malfunction - Failure or disruption of communication links (communication networks) Failures/ Malfunction - Failure or disruption of main supply

		Failures/ Malfunction - Failure or disruption of service providers (supply chain) Failures/ Malfunction - Malfunction of equipment (devices or systems) Outages - Loss of resources Outages - Absence of personnel Outages – Strike Outages - Loss of support services Outages - Internet outage Outages - Network outage Eavesdropping/ Interception/ Hijacking - War driving Eavesdropping/ Interception/ Hijacking - Intercepting compromising emissions Eavesdropping/ Interception/ Hijacking - Interception of information Eavesdropping/ Interception/ Hijacking - Interfering radiation Eavesdropping/ Interception/ Hijacking - Replay of messages Eavesdropping/ Interception/ Hijacking - Network Reconnaissance, Network traffic manipulation and Information gathering Eavesdropping/ Interception/ Hijacking - Man in the middle/ Session hijacking Nefarious Activity/ Abuse - Identity theft (Identity Fraud/ Account) Nefarious Activity/ Abuse - Receive of unsolicited E-mail Nefarious Activity/ Abuse - Denial of service Nefarious Activity/ Abuse - Malicious code/ software/ activity Nefarious Activity/ Abuse - Manipulation of information Nefarious Activity/ Abuse - Misuse of audit tools Nefarious Activity/ Abuse - Misuse of information/ information systems (including mobile apps) Nefarious Activity/ Abuse - Unauthorized activities Nefarious Activity/ Abuse - Unauthorized installation of software

		Nefarious Activity/ Abuse - Compromising confidential information (data breaches) Nefarious Activity/ Abuse – Hoax Nefarious Activity/ Abuse - Remote activity (execution) Nefarious Activity/ Abuse - Targeted attacks (APTs etc.) Nefarious Activity/ Abuse - Failed of business process Nefarious Activity/ Abuse - Brute force Nefarious Activity/ Abuse - Abuse of authorizations Legal - Violation of laws or regulations / Breach of legislation Legal - Failure to meet contractual requirements Legal - Unauthorized use of IPR protected resources Legal - Abuse of personal data Legal - Judiciary decisions/court orders
Communication	SDN networks	Spoofing attacks Main in the middle attacks Tampering Repudiation Information disclosure Denial of Service – Flooding and Saturating attacks
	Wireless Sensor Networks (WSN)	Passive attack – Data Interception – Traffic Analysis Passive attack – Data Interception – Sniffing Passive attack – Data Interception – Key logger Active attack - Packets crafting – Replay attack Active attack - Packets crafting – Masquerading Active attack - Packets crafting – 0-day Active attack - Packets alteration – Main-in-the-middle (MiM)

		Active attack – Service compromising – DoS Active attack – Service compromising – DdoS Active attack – Service compromising – SQL Injection
	V2X Communication	DDoS attacks, doxing, website defacements Information theft, virtual sabotage, website parodies Whistleblowing Gathering information about network (reconnaissance) Man in the middle (MiM) Session hijacking Repudiation of actions Use crimeware, phishing, and spear-phishing Trojan Smash-and-grab, social engineering, business email compromise (BEC) scams, botnets, password attacks, malware, ransomware Interception of information Replay of messages Account hijacking Network reconnaissance Data exfiltration or privilege misuse Spear-phishing password attacks, social engineering, direct compromise, data exfiltration, remote access trojans, and destructive malware. Interfering radiation Cyber reconnaissance of critical infrastructure Defacements and claimed leaks Worm

		Spoofing
CPS	Device perception layer	Sensor Alteration, Data theft Sensitive information leakage Denial of Service Physical Attack on a Device
	Device Application layer	Malformed Firmware/Hardware Integrity Attacks Against Machine Learning Logging Mechanism alteration Application software functionality change

In deliverable D4.3 [3] we mentioned that ENISA analysed the top cyberthreats for the period January 2019 – April 2020, describing several trends, and reflected the results in a report [4]. According to this report, the top 15 cyberthreats in the period reviewed are: 1 - Malware, 2 - Web-based Attacks, 3 – Phishing, 4 - Web application attacks, 5 – Spam, 6 - Denial of service, 7 - Identity theft, 8 - Data breaches, 9 - Insider threat, 10 – Botnets, 11 - Physical manipulation, damage, theft and loss, 12 - Information leakage, 13 – Ransomware, 14 – Cyberespionage and 15 – Cryptojacking.

This information has been updated for the period April 2020 – July 2021 in [5]. The ranking has changed notably: 1 – Ransomware, 2- Malware, 3 - Cryptojacking, 4 – e-mail related threats, 5 – Threats against data, 6 – Threats against availability and integrity, 7 – Disinformation & misinformation, 8 – non-malicious threats, 9 – supply-chain attacks. The effect of COVID-19 pandemic and the change to a hybrid office model, which has dramatically expanded the attack surface, has led to a progressive exploitation of the weaknesses related to home office. The increasing number of cyber threats also has another explanation on the fact that infrastructures are shifting away from a rather traditional model to an online and cloud-based one. Breaking it down into sectors, ENISA reports that the public administration / government, the general public, the digital service providers and healthcare medical are the most affected ones.

2.2 Runtime Security Monitoring in the CPSoS context

Deliverable D1.1 introduced the concept of run-time security monitoring and identified the main components that should be part of it, namely: Intrusion Detection Systems (IDS), malware detectors and anomaly detectors, all of them connected and acting as source of security information for a Security Information and Event Management (SIEM) system. These elements should be tailored to the specificities of the technologies that compose a CPSoS system, but at the end of the day, two main functions should be provided:

- Collection of data (security-related information) from heterogeneous data sources deployed and monitoring each domain and layer of the CPSoS
- Analysis of the information collected from a security perspective, in order to detect anomalies, vulnerabilities or security incidents.

The CPSoSAware ecosystem model presented in D1.1 section 7.2 identified three major domains, namely System domain, Communication domain and CPS device domain; which should be adequately monitored

with specific components named security monitoring agents and sensors. These sensors and agents inspect configurations, data and behaviour of the elements of each domain to identify changes, detect unusual or suspicious behaviour and, in some more advanced cases, analyse these findings to report security events. Security events are correlated and analysed with the support of SIEMs, to detect threats and complex attack scenarios that involve different steps or actions taken by attackers at different levels in the system, in order to accomplish their malicious objective. When a series of security events collected from the monitored infrastructure matches a threat or attack pattern, SIEMs generate security alarms that inform the security operators of a potential security incident happening. Security alarms are evaluated by security incident handling teams to decide whether or not start an investigation, trigger automatic remediation or mitigation processes and also, are used to compute metrics that provide a view of the security situation of a system.

Security agents and sensors monitor the physical, virtual and software elements that integrate a CPSoS system. As already explained, these components report information, in the form of security events, to the SIEM system that processes these events, correlating and performing a security analysis that may result in security alarms. But security alarms can also be fed into a SIEM system, as any other security event, for cross-correlation of complex multi-level security run-time monitoring. In this way, a SIEM can be considered another type of security sensor that reports to a higher-level SIEM. This approach can be selected when we want to have specialized SIEMs monitoring a very specific sub-system and there are resource constraints that prevent from deploying a complete SIEM in the sub-system, as it may be the case of CPS systems.

Depending on the type of threats and attacks monitored and on the technological characteristics of the monitored infrastructure domain, in CPSoSAware we consider the following security runtime monitoring capabilities:

- **CPS-level security monitoring**: consists of observing the activity of the system within each individual CPS to collect security-relevant events, correlate them and generate information that serves to understand the security situation of the CPS at any point in time, and take local actions if needed. In this context, we can distinguish the following security monitoring capabilities, each one focused on a CPS architectural layer:
    - *Monitoring of the physical/device layer*: with a focus on the status and behaviour of the sensing and actuation devices of the CPS, e.g., GPS, Lidar, etc.
    - *Monitoring of the applications*: this capability focuses on monitoring security aspects related to the configuration and activity of the software services, applications and business processes running in the CPS.
    - *Monitoring of the data*: the objective in this case is to monitor security properties of the data stored, processed, and transmitted within the boundaries of the CPS.
    - *Monitoring of the intra-communication*: this capability monitors the communication interfaces of the CPS to detect anomalies in the intra-communication behaviour.

- **System-level security monitoring:** consist of observing the system as a whole, collecting security-related information from all the elements that compose the Cyber Physical System of Systems and correlating them at the same level. Thus, each individual CPS that belongs to the CPSoS is considered as another element of the system that is subject to fail or be attacked and because of that, security information (i.e., security alarms) is collected from CPSs and processed at system-

level. By doing this, it is possible analysing the security situation of the individual CPSs from a global perspective and detect more complex security anomalies. Besides this, the system-level security monitoring has the following capabilities:

- *Monitoring of the virtual/physical layer:* this capability focuses on monitoring security aspects related to the virtual and physical infrastructure that supports the CPSoS.
- *Monitoring of the application layer*: in this case the focus is on monitoring security aspects of the Cloud applications and services that permit operating, controlling and orchestrating the CPSoS. Similarly to the CPS-level monitoring, configuration files and activity logs are the main asset to be monitored in this case.
- *Monitoring of the data layer:* refers to monitor security properties of the data stored and processed by the Cloud applications and services used to operate, orchestrate and control the CPSs, including the data exchanged between the controller and the CPSs and the data exchanged between CPSs.
- *Monitoring of the inter-communication layer:* this capability focuses on monitoring security aspects related to the communications between the system and the edge nodes or CPSs.

2.3 Security Analysis for detection of anomalies and threats

2.3.1 Security Analysis performed at sensor level

Security monitoring sensors are pieces of software that observe a category of asset of the infrastructure, such as data or network communications, in order to collect specific information and possibly search for certain pattern match. The information collected is processed and the result is generated as an output that can be logged into a file or displayed in an output interface. In CPSoSAware, security monitoring sensors have capabilities to process the information collected about the observed asset and perform a security analysis that generates, as a result, security events.

It is worthwhile mentioning that not only security monitoring sensors, but also operational monitoring sensors (those focused on obtaining relevant parameters about platform operation and performance) have strong potential for the detection of security anomalies, and therefore it may make sense to integrate this type of data sources to feed the SRMM. The implementation of appropriate security directives is the way to make the most of this type of information.

Task 3.5 is devoted to the research and development of this topic. The table below shows a list of sensor categories used in CPSoSAware. Most of them are security sensors per se, but in some cases (like Nagios or the CARLA and V2X simulator sensors) operational information is provided.

Table 3. CPSoSAware Sensors

Security Sensor	Capabilities/Description	Monitored Layer / Asset Category	Security Events generated
Suricata (https://suricata-ids.org/)	Network Intrusion Detection System (NIDS) engine	Communication, both at System and CPS level.	Thousands of different security events grouped into

	Network Intrusion Prevention System (NIPS) engine Network Security Monitoring (NSM) engine Offline analysis of PCAP files Traffic recording using pcap logger Unix socket mode for automated PCAP file processing Advanced integration with Linux Netfilter firewalling	Support for packet decoding of: IPv4, IPv6, TCP, UDP, SCTP, ICMPv4, ICMPv6, GRE Ethernet, PPP, PPPoE, Raw, SLL, VLAN, QINQ, MPLS, ERSPAN, VXLAN, Geneve App layer decoding of: HTTP, HTTP/2, SSL, TLS, SMB, DCERPC, SMTP, FTP, SSH, DNS, Modbus, ENIP/CIP, DNP3, NFS, NTP, DHCP, TFTP, KRB5, IKEv2, SIP, SNMP, RDP, RFB, MQTT	categories and sub-categories, e.g. Suspicious – Network activity, Exploit – SQL Injection, Suspicious Scada Activity, Malware – Trojan, Recon - Scanner, etc.
OSSEC (https://ossec.net)	Host-based Intrusion Detection System (HIDS) Features: Log analysis, file integrity monitoring, Windows registry monitoring, centralized policy enforcement, rootkit detection, real-time alerting and active response. It runs on most operating systems, including Linux, OpenBSD, FreeBSD, MacOS, Solaris and Windows.	Applications and Data, both at System and CPS level. Monitor Integrity of Files and Logs from systems, devices and applications	Hundreds of different security events grouped into categories and subcategories, e.g. Authentication, System Information, Inventory change, etc.
Kismet (www.kismetwireless.net)	Kismet is a wireless network and device detector, sniffer, wardriving tool, and WIDS (wireless intrusion detection) framework.	Wi-Fi interfaces, Bluetooth interfaces, some SDR (software defined radio) hardware like the RTLSDR, and other specialized capture hardware.	Various events such as: Possible ap spoofing channel change Suspicious traffic Suspicious client Flood detected
Nagios (https://www.nagios.org/)	Nagios is a network/system status monitoring daemon that provides extended insights of the	It works both at network layer and system layer, monitoring hardware, services and applications, providing a set of metrics like CPU load, disk usage,	No security events per se, but it sends information about different metrics, and

	monitored IT infrastructure	number of current processes, memory usages and number of current SSH sessions, to name but a few.	this information can be processed following security directives at SRMM.
CPS Hardware Security Token	The sensor monitors the data integrity of exchanged messages between CPS The sensor identifies possible spoofing of the car GPS sensor. The accurate position of the GPS is calculating by fusing other modalities on the CPS Design space exploration of different integrity check techniques Computation monitors to detect anomalous processing activity (such as CPU load….)	Data at CPS level. TCP, UDP, HTTP/HTTPs CAN bus, deployed firmware GPS sensor data Computation	Integrity failure Authentication Failure GPS Spoofing GPS unavailability Excessive resource usage
CPS Communication Security Integrity	The sensor monitors if the communication interface identified that the communicating interface attempting to connect doesn't have the correct credentials (BLE, WiFi, ZigBee applicable)	Intra-CPS communication. BLE: Pairing credentials, WiFi: SSID/Password/MAC address, ZigBee: MAC address	WIFI authentication failure BLE authentication failure ZigBee authentication failure
CPS Communication Health status	Monitors if the communication interface identifies communicating link failure	Intra-CPS communication. In communication scenarios that responses are expected (bidirectional), they are not received after a specific time delay. In communication scenarios that responses are not expected (uni-directional), ping-like services can be deployed to monitor the link status	WIFI link failure BLE link failure ZigBee link failure

CPSoS authentication	Securely identify each individual CPS/CPHS system (e.g., ADAS Car, AGV, etc.)	Device/Edge at System-level. Identification/authentication token	Edge server unavailability Authentication Failure with Edge Server
CPS RIS	It is a daemon that periodically checks the version of the different software and firmware components at CPS level, generating an event when such a version changes.	It works at system level, detecting when version changes take place	New software / firmware version installed, indicating the version number and optionally providing more details
CARLA simulator sensors	GPS Sensor generating the actual position of the vehicle contaminated with noise.	It works at CPS level	A GPS spoofing attack can be detected when running the cooperative localization solutions. This can be reported at the simulated environment, by providing a feedback to the CARLA User.
V2X simulator sensors	ROS2 based simulator for modelling the communication layer between traffic agents and the infrastructure. Supports several wave propagations models and obstacle effects. Additionally implemented visualization model based on rviz2, that visualizes environment and all static/dynamic objects	Inter-CPS communication	No security events simulated in V2X simulator. Possible security related scenarios can be implemented in main AV simulator integrated with V2X Simulator

2.3.2 Security Analysis performed at SIEM level

SIEM systems collect information about a monitored complex infrastructure through the use of software agents, which are usually deployed at specific elements of the infrastructure that have access to the sources of information that are being monitored: network traffic, application logs, databases, etc. The information collected by agents is parsed, normalized and encapsulated in the form of events that follow a specific data format. Events are sent to the SIEM server for correlation, using predefined security directives or rules, in order to identify anomalous behaviours, discover possible threats and detect security incidents. When a specific set of received events matches a directive, a security alarm is raised and this, in turn may trigger actions according to predefined policies. Security alarms usually contain information about the threat or security anomaly detected, the affected infrastructure asset and the source of the security event (e.g. the source IP of an attack performed from an external actor). Besides that, security alarms may also contain information to determine the severity of the incident, such as reliability of the information collected, or risk associated to the asset affected by the incident. This type of information can be used to take adequate and proportionate actions to address or mitigate the incident. Some examples of these actions are notifying the security administrator (through email, dashboard, etc.) or the automatic or semi-automatic execution of certain reactions to reconfigure the system or implement more specific countermeasures.

SIEMs can be classified according to their features: data sources supported, data storage capabilities, processing capabilities, flexibility of the security directives, support for behavioural analysis, support for risk analysis, extensibility and interoperability through available APIs, resilience, visualization capabilities, reaction capabilities, deployment model, scalability or licensing, among others [30]. Other advanced capabilities of SIEMs are support for forensics and threat hunting, cloud readiness or support for advanced threat detection and response. Research and advisory IT organisations, such as Gartner [22], Forrester [23] or TechTarget [24], compare, classify, and evaluate SIEMs considering other business and market-related aspects too. But overall, SIEMs implement a general concept, which is depicted in Figure 1. In this figure, the different monitored infrastructure realms are depicted as sources of data of different nature at the bottom, communicating with the SIEM server through SIEM agents. At the top of the figure, the SIEM server stores the events collected from the monitored infrastructure and alarms triggered in a database and hosts the correlation engine and security intelligence processes that permit security administrators have an overview of the security situation of the monitored system at any point in time.

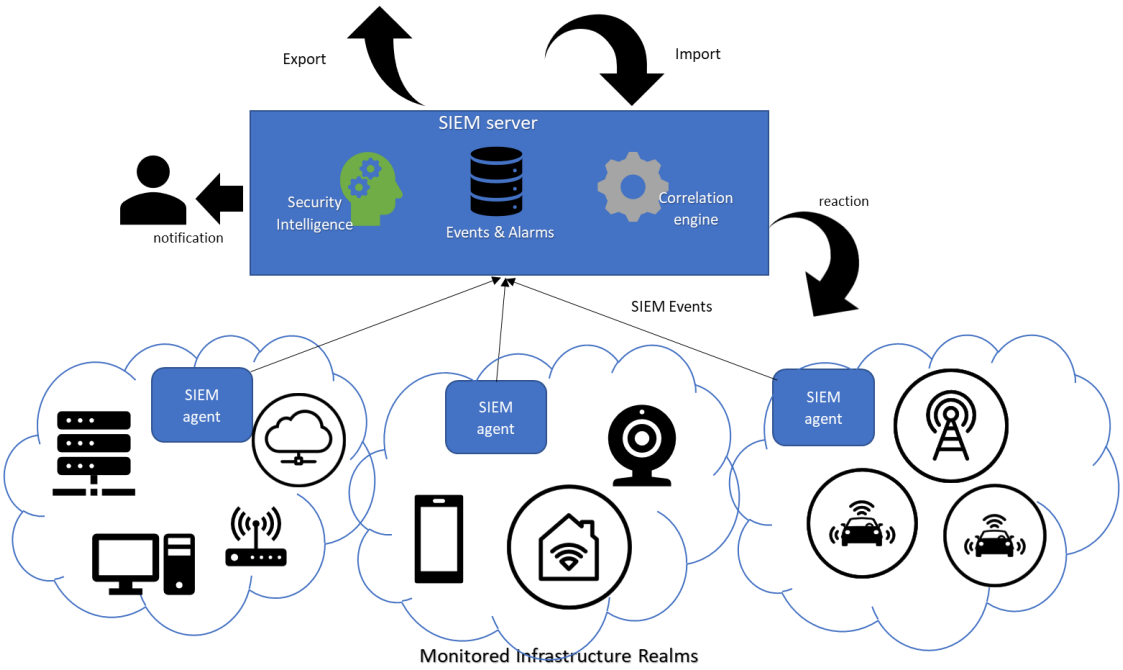

Figure 1. SIEM concept

SIEMs support the security analysis through different tasks. At design-time, prior to the deployment of the security runtime monitoring infrastructure, the following activities should be done to adapt the SIEM security analysis to the specifics of the monitored CPSoS context:

- **Identification and characterisation of the infrastructure assets**: considering the complexity of a CPSoS and the different nature of the elements that compose it, the first step is the identification of the assets that should be monitored and their characterisation, according to technical, business and security-related criteria. This activity permits establishing what are the critical assets in the system, with a higher business or security value, and design appropriate security directives and policies to protect them.
- **Identification and characterisation of data sources:** once the elements of the infrastructure that should be monitored have been identified, it is necessary to analyse the information that can be collected from them and select what is relevant from a security perspective. The result of this analysis is a list of data sources and event types, associated to these data sources, that will constitute the input for the correlation processes performed at the SIEM server.
- **Design and implementation of security directives:** this activity consist of analysing the characteristics of the CPSoS, the security requirements and the threats that may affect the system, in order to define possible attack/threat scenarios, suspicious or anomalous situations that should be monitored. These scenarios are translated into event patterns, which capture relationships of different type (temporal, causal) between events. Event patterns are codified as security rules or directives and are used by the correlation engine of the SIEM server to detect occurrence of such patterns in the events that are being collected at run-time from the monitored environment.

In the following, we are showing some examples of security directives identified within the context of the project. The EPL code is shown in a frame and the corresponding explanation comes under the frame.

CPS_HWVerification_Anomalous
==========
pattern [every-distinct(a.userdata4,60 seconds) a=CPS_HWToken_Verification_Failure -> (b= CPS_HWToken_Verification_Failure (b.userdata4=a.userdata4) -> c= CPS_HWToken_Verification_Success (c.userdata4=a.userdata4))]
Category: CPS Subcategory: Integrity
Reliability: 8 Priority: 1

Every *CPS_HWToken_Verification_Failure* event **'a'** with different **'userdata4'** followed by another *CPS_HWToken_Verification_Failure* event **'b'** with same **'userdata4'** followed by *CPS_HWToken_Verification_Success* **'c'** event with same **'userdata4'** produces a *CPS_HWVerification_Anomalous* alarm with reliability 8 and priority 1. The **userdata4** distinction is considered for 60 seconds, then the value is discarded.

CPS_HWVerification_Suspicious
==========
pattern [every-distinct(a.userdata4,60 seconds) a=CPS_HWToken_Verification_Failure -> NOT (b= CPS_HWToken_Verification_Success (b.userdata4=a.userdata4))]
Category: CPS Subcategory: Integrity
Reliability: 8 Priority: 2

Every *CPS_HWToken_Verification_Failure* event **'a'** with different **'userdata4'** not followed by another *CPS_HWToken_Verification_Failure* event **'b'** with same **'userdata4'** produces a *CPS_HWVerification_Suspicious* alarm with reliability 8 and priority 2. The **userdata4** distinction is considered for 60 seconds, then the value is discarded.

CPS_SystemBehaviour_Suspicious
==========
pattern [every-distinct(a.userdata4,60 seconds) a=CPS_HWToken_Verification_Success -> b= CPS_Abnormal_ResourceUsage (b.userdata4=a.userdata4)]
Category: CPS Subcategory: System Behaviour
Reliability: 8 Priority: 3

Every *CPS_HWToken_Verification_Success* event **'a'** with different **'userdata4'** followed by another *CPS_Abnormal_ResourceUsage* event **'b'** with same **'userdata4'** produces a *CPS_SystemBehaviour_Suspicious* alarm with reliability 8 and priority 3. The **userdata4** distinction is considered for 60 seconds, then the value is discarded.

CPS_SystemBehaviour_Abnormal
==========
pattern [every-distinct(a.userdata4,60 seconds) a=CPS_Abnormal_ResourceUsage -> ([2] b= CPS_Abnormal_ResourceUsage (b.userdata4=a.userdata4))]
Category: CPS Subcategory: System Behaviour
Reliability: 8 Priority: 2

Every *CPS_Abnormal_ResourceUsage* event **'a'** with different **'userdata4'** followed by 2 *CPS_Abnormal_ResourceUsage* event **'b'** with same **'userdata4'** produces a *CPS_SystemBehaviour_Abnormal* alarm with reliability 8 and priority 2.

At run-time, once the sensors, agents and SIEM server are deployed and running, the real-time analytics are done at two levels:

- **Real-time processing of security events:** consist of performing computing operations on the events received from the environment. Events can be received as streams or continuous flows and that is referred in literature as event stream processing. There are different platforms in the market that perform Data/Event Stream Processing, such as Hadoop, Spark, Storm, Kafka, Flume or Amazon Kinesis. As it is described in section 3.3.3, the technology that implements the runtime security analysis of the CPSoSAware SRMM is based on Apache Storm.
  The operations performed on events consist in filtering, aggregating, and correlating multiple events coming from the same or different sources, resulting in complex event computations, often referred as Complex Event Processing (CEP). These operations are executed in the SIEM server by the correlation engine, in accordance with the correlation rules contained in the security directives defined at design-time. Event processing systems can be classified into those that follow a query-based approach, a rule-oriented approach or a programmatic approach. As it is described in section 3.3.3, in CPSoSAware it is used a correlation engine technology that follows a query-based approach.
- **Analysis of security alarms:** alarms contain multiple useful information about the security anomaly, incident or threat detected, about the affected asset, but also contextual information of the environment around the detection that can be used for statistical analysis to identify trends and implement predictive algorithms. Moreover, the analysis of security alarms combined with forensic techniques can trace back until the root cause of the incident. Alarms usually contain reliability and risk values, which combined with the criticality of the affected asset can be used to perform an assessment of the severity of the incident and evaluate the impact in the overall security posture of the system. Last but not least, alarms can be exported into standard formats such as MISP or STIX, and feed third-party Threat Intelligence Analysis platforms or SOCs for further analysis and this way, contribute to the cybersecurity community.

2.4 Reporting

Security runtime monitoring sensors, in combination with SIEMs, collect and produce security-related information from a target monitored system and this information can be used for different purposes. As it is explained in section 2.3, security events and alarms help security administrators to be aware of security anomalies or incidents that may be happening in the infrastructure and respond to them promptly and adequately. But this information can also be used to compute security metrics that CISOs can use to assess the security posture of a complex system and take adequate corrective actions. As stated in [25] security metrics have the purpose of assessing *"security characteristics that are of interest for the operational and managerial security decision making"* and *"provide a framework for evaluating the security built into commercially available products or services and allow enriching the knowledge of the organization´s security and can provide information about the organization´s strengths, weaknesses and risks, with a global view of the organization security status".* Security metrics can be categorized in three

hierarchies: management, operational and technical metrics, which are addressed to the business management, the security management, and the security operation respectively [26]; but it is just one of the possible ways to categorize them.

In a CPSoS, the assessment of the security of the system can be done both at the individual CPS level and at the general system level. Reporting security information at CPS level permit evaluating whether the specific requirements of the CPS are met or not and thus, take adequate corrective actions and reconfigurations locally, in a fast and suitable manner. On the other hand, evaluating the overall situation of the system, considering each CPS from a global perspective, provide a more comprehensive understanding of all the factors that may influence the achievement of the system security goals, and apply general corrective actions that fit all the possible situations.

In the following, we are providing some examples of security metrics:

Table 4. Some examples of security metrics [25]

Metric	Definition	Input data	Output	Suggested frequency
Asset criticality	Represents the impact that the loss of an asset may have on an organization	List of assets with information about them	Asset criticality value, whether qualitative or quantitative	Monthly at the very least, but if there are variations in the list of assets, it should be refreshed as well
Number of known unresolved vulnerabilities per severity	Total number of known unresolved vulnerabilities by their severity levels	Known unresolved vulnerabilities per severity category	Number of unresolved vulnerabilities per severity level	Daily
Number of known unresolved vulnerabilities per vulnerability type	Open vulnerabilities per vulnerability type	Raw data about open vulnerabilities	Classified vulnerabilities per type	Daily
User activity	Top users with biggest number of failed login attempts	Raw events of users and failed login attempts	Ranking of users according to the number of failed login attempts	Daily
SOC´s percentage of effort time to resolve vulnerabilities and incidents	Metric that measures security team performance	SOC´s time devoted to vulnerabilities and incidents resolution and total SOC´s time	Percentage of time devoted to vulnerabilities and incidents resolution	Monthly
Top malware activity	Top malware detected in the organization by criticality	Reported incidents including malware activity	Top malware detected in the organization by criticality	Daily


C P S o S
A W A R E

3 Security Runtime Monitoring and Management: final architecture design

3.1 Positioning of the SRMM in the CPSoSAware architecture and requirements

As presented in deliverable D1.4, the SRMM has a clear positioning in the CPSoSAware architecture. This is illustrated in Figure 2 and Figure 3. On one side, the SRMM receives sensor data from the different sensors developed in the context of project task T3.5 and summarized in Table 3. The alarms resulting from the internal pre-processing, filtering and correlation process that takes place within the SRMM are published to a queue for further consumption. On the other side, the SRMM makes use of the REST API provided by the V2X simulator from project task T4.4, as well as the Multimodal Localization REST API provided from the CARLA simulator in WP3. In both simulators the communication will be bidirectional. The SRMM will receive events from the simulators and will send back relevant alarms for visualization in the simulators´ local environment

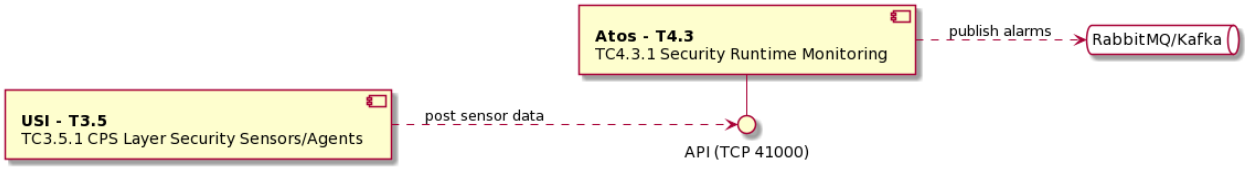

Figure 2. Positioning of SRMM within the CPSoSAware architecture [1]

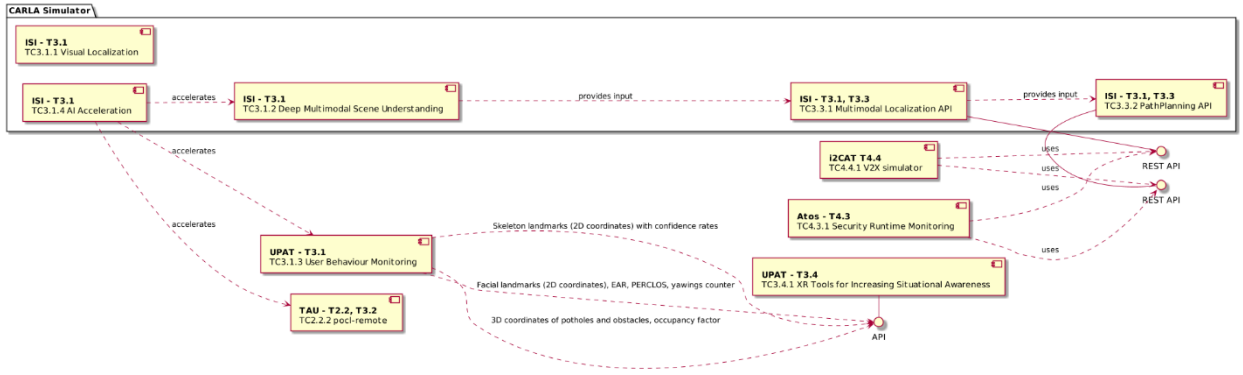

Figure 3. Positioning of SRMM within the CPSoSAware architecture [1]

In the following a formal description of these interfaces is provided:

Table 5. SRMM interfaces with the rest of CPSoSAware architecture

	Suricata	OSSEC	Kismet	Nagios	CPS Hardware Security Token	CPS Communication Security Integrity	CPS Communication Health Status	CPSoS authentication	CPS RIS	CARLA Simulator	V2X Simulator
SRMM	IN	IN	IN	IN	IN	IN	IN	IN	IN	IN/OUT	IN/OUT

Table 6. SRMM interfaces with the rest of CPSoSAware architecture

Interface	Data and description	Type of communication
SRMM <- Suricata	Different network layer security events grouped into categories and sub-categories	TCP 41000
SRMM <- OSSEC	Different host layer security events grouped into categories and sub-categories	TCP41000
SRMM <- Kismet	Several events referred to WiFi Security	TCP 41000
SRMM <- Nagios	Network and host layer metrics	TCP 41000
SRMM <- CPS Hardware Security Token	Information about data integrity of the exchanged messages between CPS	TCP 41000
SRMM <- CPS Communication Security Integrity	Information about intra-CPS communication correctly established (correct credentials)	TCP 41000
SRMM <- Communication Health Status	Communicating link failures in the communication interfaces	TCP 41000
SRMM <- CPSoS authentication	Authentication-related information	TCP 41000
SRMM <- CPS RIS	Detection of changes in software / firmware version	TCP 41000
SRMM <-> CARLA Simulator	Detection and mitigation of LIDAR/Camera adversarial attacks, GPS spoofing.	TCP 41000
SRMM <-> V2X Simulator	V2X communication details, messages, statistics of received packets etc.	ROS2. TCP41000 interface will be added.

In the following, the status of the requirements related to the SRMM, as defined in WP1, is updated:

Table 7. Status of SRMM requirements (updated from [1])

Req. ID	Description	Type	Source	WP	Target comp.	Target phase	Priority	Author	How addressed	Reported in	Status
TC4.3.1.R1	Input. The component must receive normalized security events through TCP/41000 from agents/sensors deployed remotely, in the infrastructure that is under surveillance. Events comply with a predefined JSON format.	Functionality & integration	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	ATOS	ATOS provides sensors and integrates them with the SRMM. If some partner provides sensors from their side, ATOS can integrate them mainly by normalizing the data format	D3.5, D4.3, D4.8	Achieved

TC4.3.1.R2	Configuration. The component should be configured using the component's graphical dashboard, to define the security monitoring infrastructure in use (topology of sensors/agents deployed and active), the security detection rules and the correlation directives.	Functionality & integration	End User & DoA	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	ATOS	Two possible ways: either using the existing interface developed by ATOS from other projects, and making needed adaptations, or developing a specific configuration interface in CPSoSAware. This remains to be decided	D2.2, D4.3, D4.8	Not achieved yet
TC4.3.1.R3	Events Processing. The component must process security events received as input, correlate them using the security detection rules configured, and generate security alarms as output, as defined in the correlation directives configured.	Functionality	End User & DoA	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	ATOS	We need to define the needed correlation rules to be applied in the SRMM	D4.3, D4.8	Achieved
TC4.3.1.R4	Output. The component should produce as output security alarms. Alarms comply with a predefined JSON format. Alarms can be configured to be persisted in a DB, logged into a file, transmitted to a third-party component (using a middleware such as Message Queue/Broker) and displayed in the SRMM graphical dashboard.	Functionality & integration	End User & DoA	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	ATOS	Alarms are produced following the internal correlation process of the SRMM. JSON format to be confirmed within the Consortium as it will have to be used by the CSAIE (T2.1). Message brokering technology needs to be confirmed. We have preference for AMQP or Kafka	D4.3, D4.8	Achieved
TC4.3.1.R5	Cross-correlation. Security alarms produced as output by the SRMM can be configured to be input into the SRMM correlation engine, for cross-correlation processes.	Functionality	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS8 - Final CPHs Architecture Design and Implementation - M36	C(ould)	ATOS	The capability of performing cross-correlation already exists. It is to be expected that new rules are produced in the context of the project	D4.3, D4.8	Achieved
TC4.3.1.NFR1	Scalability - of the SRMM correlation engine and data collection module	Maintainability	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS8 - Final CPHs Architecture Design and Implementation - M36	M(ust)	ATOS	It will be achieved by enhancements made to the assets during the project	D4.8	Achieved
TC4.3.1.NFR2	High-performance - of the SRMM correlation engine and the data persistence layer	Efficiency	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS8 - Final CPHs Architecture Design and	M(ust)	ATOS	During the project we will research on how to improve the	D4.8, D6.3	Partially achieved

						Implementation - M36			performance of the asset, which is currently high. We still have no information on how stressed the component will be during the piloting tests		
TC4.3.1.NFR3	Integrity - of the security events transmitted from sensors/agents to the SRMM component, and of the security alarms generated as output by the SRMM	Security	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	ATOS	Already achieved	D4.3	Achieved
TC4.3.1.NFR4	Confidentiality - of the security events transmitted from sensors/agents to the SRMM component, and of the security alarms generated as output by the SRMM	Security	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	ATOS	Already achieved	D4.3	Achieved
TC4.3.1.NFR5	Accountability - of the security events transmitted from sensors/agents to the SRMM component, of the correlation process and of the security alarms generated as output by the SRMM	Security	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	ATOS	Already achieved	D4.3	Achieved

3.2 SRMM internal architecture description

As a Security Runtime Monitoring of Cyber Physical System of Systems, the SRMM has to fulfil certain requirements that this type of system imposes, such as handling heterogeneous information sources or monitoring of distributed systems. To meet these requirements, the final design of the SRMM architecture is distributed and hierarchical. Each layer assumes the intelligence of a part of the system, considering the lower layers and supporting the higher ones. Where each node of the system is an instance of the XL-SIEM, which is *"an enhanced security data analytics platform with added high-performance correlation engine able to raise alarms"* [16]. The next section describes in detail this asset developed by Atos.

Lower layers of the system perform local logic, as you move up the hierarchy the logic becomes more general. This distribution of the intelligence allows the system to response to cybersecurity incidents even if on part is temporary disconnected from the rest of the system. This feature also allows the system to easily scale, deploying new nodes when is necessary and balancing the workload to a nearby node when one is overload. In addition, because the upper SIEMs receive security events from the SIEMs that are below, the last filter the information they send. This reduces the workload of the upper SIEMs, increasing the efficiency of the system.

The architecture proposed in this deliverable is determined by the *connected and automated vehicle pilot* where the SRMM system will be evaluated. This means that the architecture may be modified to be adapted to other projects. In this case, we propose an architecture with three layers:

- **Lightweight SRMM**: This is the lowest layer, deployed inside vehicles. Due to the limited environment, these SRMMs do not have much computational capacity. Therefore, these SIEMs only have a reduced set of rules related to vehicle domain. SRMMs usually receive security events from agents, which are installed inside the vehicles. These agents parse the logs from sensors, which monitor the vehicle and generate the security events. In addition, other nearby vehicles can be a source of events when they have this system installed. Finally, these SIEMs can receive security events and rule modification from the upper SIEMs.

  For example, a Lightweight SRMM may receive a decommissioning and update event affecting a particular firmware version. In that case, the SRMM must consider whether the vehicle where it is installed has the affected device and checks if the firmware version is affected by the rule; applying the associated action when fulfilled.

- **Area SRMM**: This type of SIEM is deployed on edge servers. They correlate rules related to an area, receiving security events from the Lightweight SRMMs, area agents, and other nearby Area SRMMs. The agents generate events related to the whole area such as communications or the edge server itself. These SIEMs filter and forward the events form the Lightweight SRMMs to Global SRMM and broadcast alarm events and rule updates from the Global SRMM to the Lightweight SRMMs.

- **Global SRMM:** There is only one SRMM of this type which is in the cloud. It has a global and complete vision of the system; hence it performs the global logic. It receives events from the Area SRMMs and global agents. The alarms that it raises are general and must be sent to lower SIEMs, as they are the ones who check local conditions to apply the associated actions.

Figure 4 presents the lower level of the SRMM architecture, showing some lightweight SRMMs connected to an area SRMM. At the bottom of the diagram are the sensors, which monitor the infrastructure. The agents process this information to yield security events that are sent to SIEMs. The latter correlate the events and send the generated alarms and relevant events to the Area SRMMs. In addition, alarms and relevant events are shared with nearby Lightweight SRMMs.

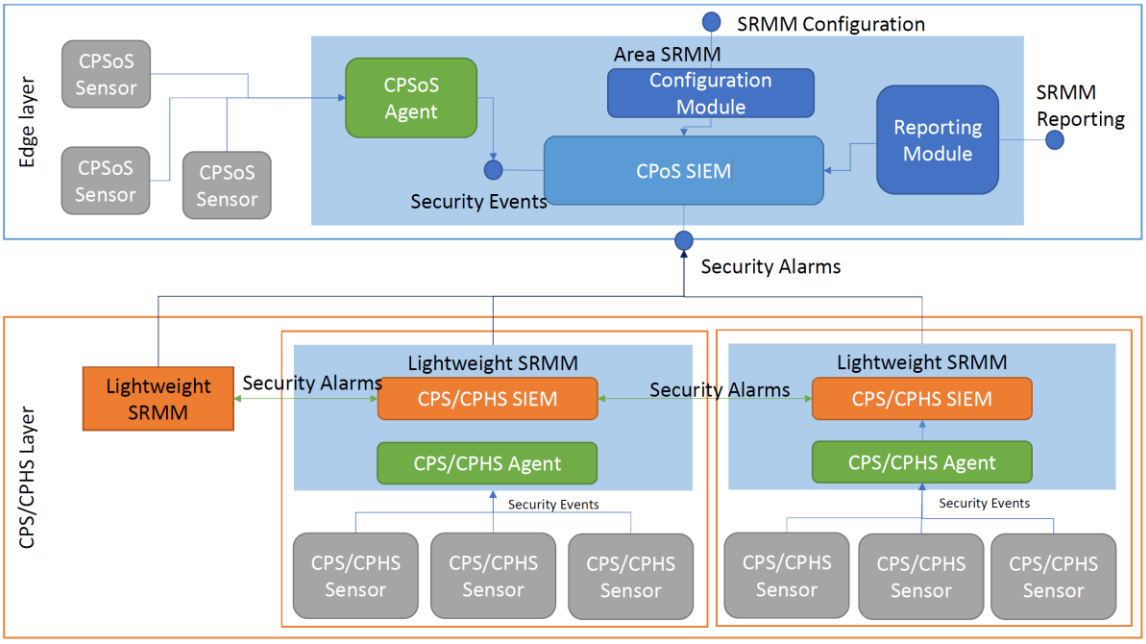

Figure 4. Lower level of SRMM architecture

Figure 5 depicts upper level of the SRMM architecture, where a similar scheme exists, with the lower SIEMs feeding the upper ones. In the bottom left part of the picture, there are some Lightweight SRMMs connected to an Area SRMM. While in the bottom right, there are other Area SRMMs with their interconnexions. About all of them, there is the Global SRMM.

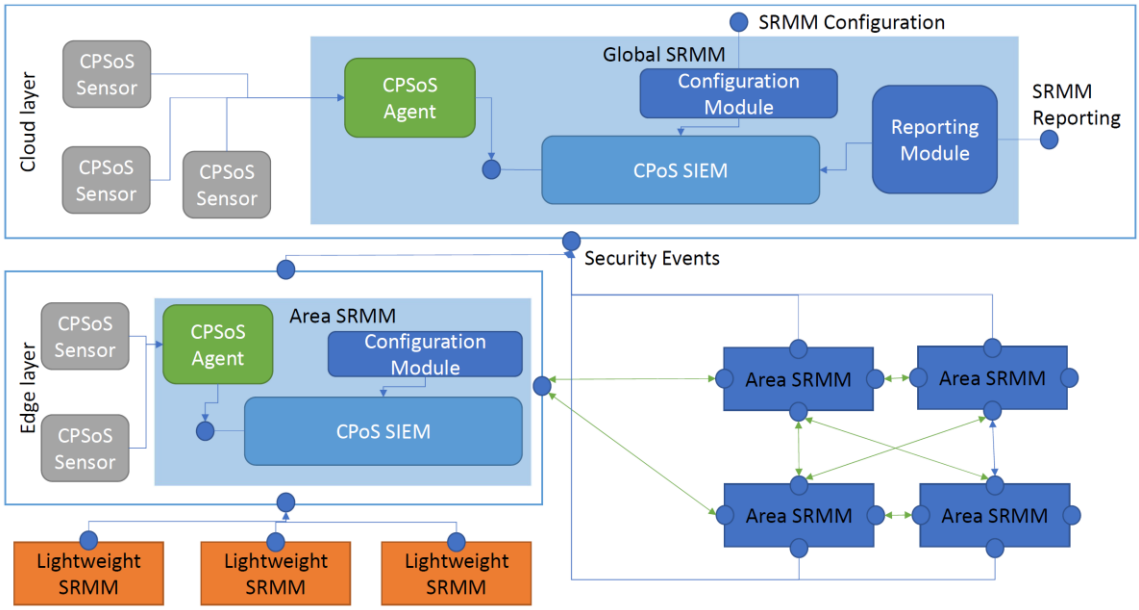

Figure 5. Upper level of SRMM architecture

Figure 6 shows the interaction between task T4.3 and other related tasks in the project. On one side, SRMM receives inputs from the different sensors provided by task T3.5. It also receives information which is relevant from the perspective of cybersecurity from two car simulators: CARLA (T3.1 and T3.3) and V2X

(task T4.4). On the other side, the alarms raised by the SRMM can be relevant inputs for other SRMMs existing in the architecture and this information can also be sent to CARLA and/or V2X so that it will be shown in the user interface of the corresponding/both simulator/s. In the figure, in the left hand-side, the configuration of the SRMM defines data sources, the topology of the infrastructure to be monitored and the security directives and policies to apply for the detection of anomalies and threats. Bottom up, the Events Collection is in charge of receiving input from CPS Sensors/Agents and the car simulators. Next, Events Processing applies filtering and aggregation rules before correlating the events according to the predefined security directives. Alarms generated by the correlation process are analysed and cross-correlated from a CPSoS perspective. The final results are reported to the abovementioned outputs of SRMM. On the right hand-side of the figure is depicted the storage of security events, alarms and logs for accountability and support forensic analysis.

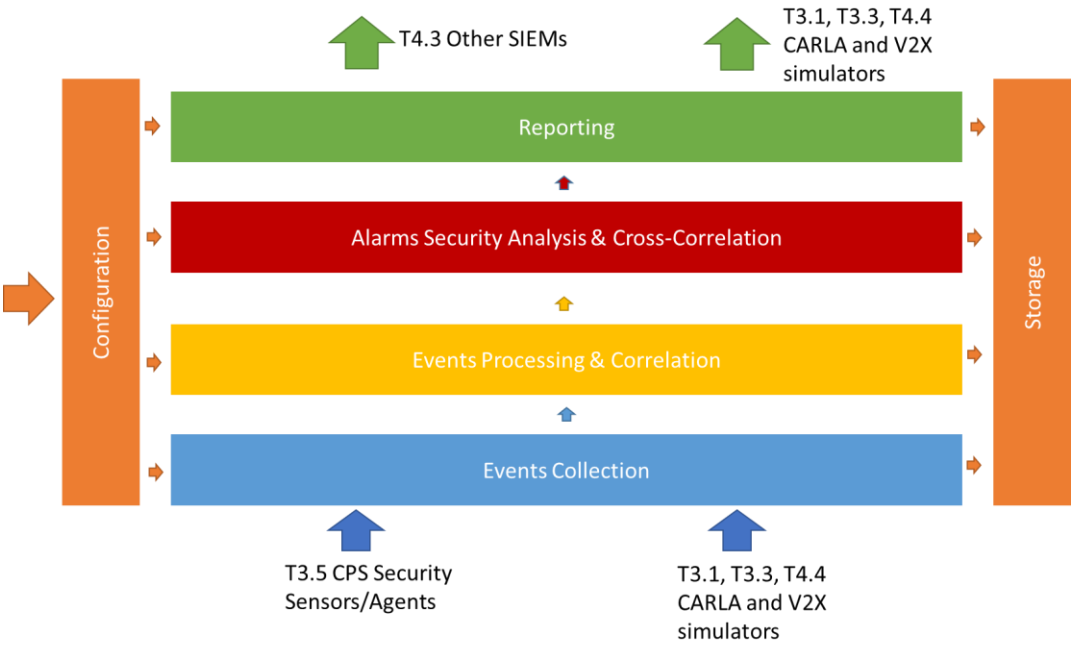

Figure 6. Security Runtime Monitoring in CPSoSAware: overall approach

Each SRMM is composed of:

- One or more **CPSoS Agents**, in charge of collecting and normalising security information produced by sensors deployed at the system level of the CPSoS. These sensors monitor assets such as the virtual/cloud infrastructure and servers, as well as the infrastructure network layer. The CPSoS agents generate as output security events that are pushed to the CPSoS SIEM through the corresponding *Security Events interface*.
- One **CPSoS SIEM**, in charge of the real-time security analysis of all the system-level security events received from the CPSoS Agents and cross-correlation of security alarms produced by the individual CPSs. These alarms are received through the *Security Alarms interface*.
- **Configuration Module** communicates with the CPSoS SIEM to implement the configurations received from other technical components of the CPSoSAware architecture through the SRMM *Configuration interface*.
- **Reporting Module** collects information from the CPSoS SIEM storage system to report results and statistical information to other technical components of the CPSoSAware architecture.

At each CPS, the SRMM is composed by:

- One or more **CPS/CPHS Agents**, in charge of collecting and normalising security information produced by sensors deployed at the CPS. These sensors monitor the assets that compose the CPS, including intra-communication monitoring. CPS Agents generate security events that are pushed to the local CPS/CPHS SIEM through the Security Events interface.

The communication between nodes is supported by the Task 4.2. Given the behaviour of the lower nodes, the vehicles, this task has to deal with a dynamic structure that is continually changing. This feature impedes the use of protocols with node identification and message addressing. Therefore, the Task 4.2 proposes to use an MQTT protocol [29]. This messaging protocol is based on two roles: publisher devices, which generate the information; and subscriber devices, which consume it. A broker manages the information, receiving the messages from publishers and sending to the subscribers. However, this communication can be bi-directional, allowing subscribers to send messages to publishers.

In this project, each vehicle is a publisher and broadcasts messages to all nearby nodes. These messages can reach another vehicle or the edge server via a Roadside Unit (RSU), installed in the road. When a vehicle changes coverage area from one RSU to another, the vehicle starts working through the new access without losing communication with the edge. Similarly, vehicles are subscribed to the information at the edge, so that when the Area SRMM publishes an alert, it reaches all vehicles in the area.

Although, the edge server and the cloud have a static position with IP addresses, the SRMMs at these positions use the same communication protocol. The Area SRMMs are the publishers and the Global SRMM is the subscriber. In addition, the Area SRMM is interconnected with nearby nodes to exchange security events.

This communication architecture allows for a dynamic configuration, where a new Area SRMM can be deployed and connected with the rest of the architecture almost transparently. After deploying a new node, it has to be connected to the Global SRMM. Then, a few nodes have to recalculate their near neighbours. Finally, the RSUs in the area controlled by the new SRMM are connected to the new edge server transparently to the lightweight SRMMs.

Figure 7 shows a diagram of what the communication flow looks like. Blue arrows represent the security events that the publishers send directly to higher-level nodes. The green arrows are the security events and alarms that a node broadcasts to nearby nodes. While orange arrows are the alarm events that the upper nodes broadcast to all their lower nodes.

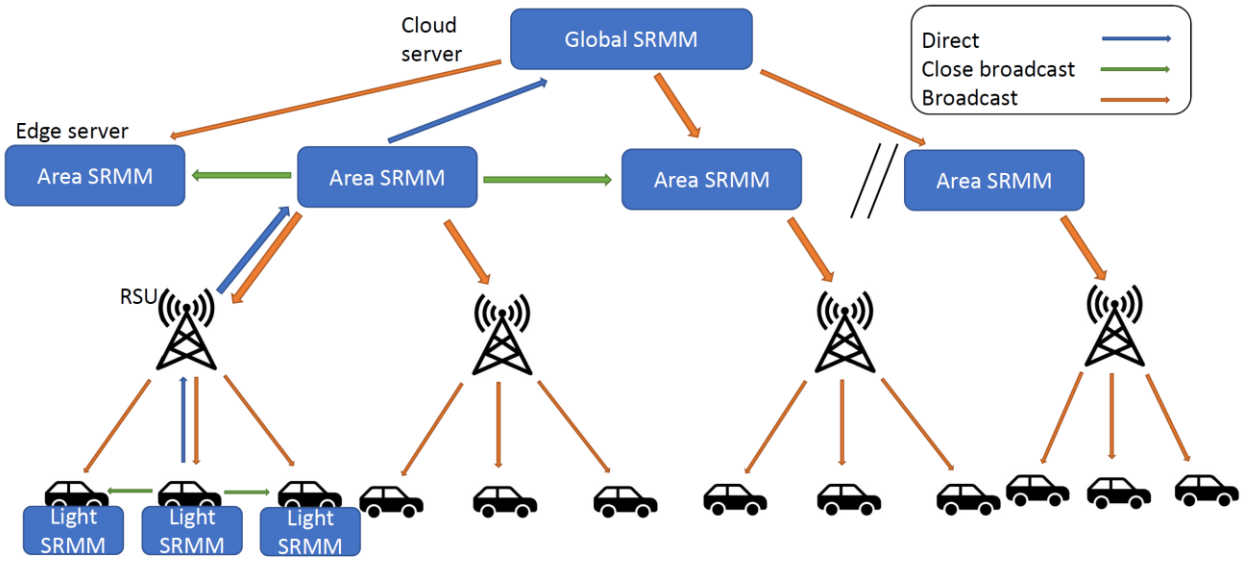

Figure 7. Communication flow of SRMM

3.3 Technology description: XL-SIEM

As defined in [31] *"Atos Cross-Layer SIEM (XL-SIEM) is a Security Information and Event Management (SIEM) solution deployed on top of the AlienVault's open-source SIEM OSSIM [15], with added high-performance correlation engine to deal with large volumes of security information"*. It provides scalability and distribution in security events processing through a cluster of nodes, and capacity to raise security alerts from a business perspective based on events collected from different data sources at different layers. These improvements, together with an extended support for data sources, a correlation engine, additional export methods and formats and reaction capabilities provides enhanced features compared to other open-source solutions available in the market. These enhancements, as well as a complete description of the architecture (see Figure 8), functionalities and implementation technologies of the XL-SIEM are detailed in the paper "*Towards an Enhanced Security Data Analytic Platform*" [16]. However, in the following sections and for self-containment of this document, we reuse some of the descriptions from that paper as well as from the online AlienVault OSSIM documentation [17], to briefly explain the main components of the XL-SIEM.

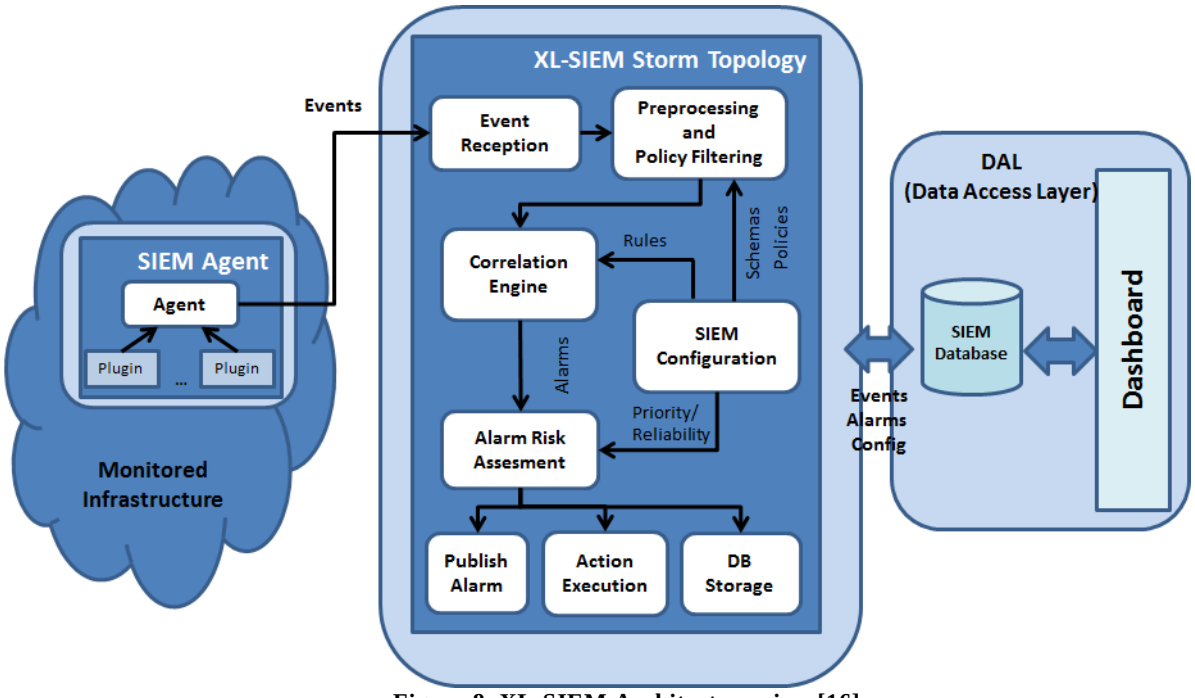

Figure 8. XL-SIEM Architecture view [16]

The XL-SIEM server improves the existing capabilities of the original OSSIM SIEM with a high-performance correlation engine, implemented with the complex event processing (CEP) Esper [18] (GPLv2 licensed) running in Apache Storm [19] cluster.

Esper is capable of processing 500,000 events per second with latency below 10 microseconds average with more than 99% predictability [30][31]. For more complex queries, these values are slightly reduced to a throughput of 120,000 events per second [3], keeping good performance capabilities for processing large volume of data. Security directives or rules are expressed using the Event Processing Language (EPL) [20], which is a declarative programming language that allows expressing security directives with rich event conditions and patterns in a simple way.

Apache Storm is a free and open source distributed real-time computation system that working together with Apache Zookeeper and RabbitMQ [21] allows processing the events in a scalable, distributed and fault-tolerant way. A Storm cluster is basically a set of nodes (hosts) where the processing tasks are distributed according to a predefined role. There are two different roles: master and worker. In Storm's terminology, the graph of real-time computation to be executed by the worker nodes is called topology. The latter includes not only the processing logic but also the links indicating how data need to be passed around between nodes. In the topology graph, spouts (sources of streams) and bolts (in charge of data processing) are connected with stream groupings. The XL-SIEM Storm Topology has defined three Spouts to handle input data (i.e. events) from different communication means: RabbitMQ and TCP Socket from agents, plus an additional input via DRPC from the Storm Topology itself. Fifteen Bolts are in charge of the processing, filtering, correlation and cross-correlation, writing to the database, handling internal and external communication and taking actions defined in policies [31].

This architecture introduces some advantages. Scalability, since the events collected are processed in parallel across a cluster of machines where the parallelism of the different parts of the topology can be scaled individually. Robust process management with the use of Storm and Zookeeper running together. And fault-tolerance, since tasks in a running topology heartbeat to the master node to indicate they are running smoothly. The Nimbus daemon in the master node monitors heartbeats and will reassign tasks

that have timed out. Additionally, all the tasks throughout the cluster that were sending messages to the failed tasks quickly will be sent to the new location.

After the processing of the security events in the correlation engine running in the Storm topology, the alarms generated are stored in the database.

3.3.1 Agents and plugins

Plugins are used to instruct agents on how to collect the raw data from security monitoring sensors and how to parse this data to extract the relevant information that will be included in the resulting XL-SIEM events. Plugins are executed by the XL-SIEM Agent which needs to be deployed and configured in a way that it can have access to the data generated by sensors (e.g., log files). For each sensor it is necessary to develop at least one plugin and configure the XL-SIEM agent to use it. Once this is done, the sensor is registered in the XL-SIEM server database as a data source, as it is explained in the OSSIM documentation [6]. This way, the events will be recognized and used appropriately in the XL-SIEM correlation engine. As described in section 2, it is required to develop one plugin for each of the sensors listed in Table 3 that will be used in CPSoSAware Security runtime monitoring infrastructure. Figure 9 shows the internals of an XL-SIEM agent. The figure is just a generalization, showing an agent that is able to parse information collected by all sensors used in CPSoSAware. However, in a real deployment, agents deployed at the system layer will only contain the plugins required to parse the data collected by sensors deployed at that level too. In the same way, agents deployed at each CPS will contain only those plugins required for the sensors deployed at the CPS. This way, resources used by the agents for parsing and event normalization are adjusted to the minimum necessary, to adapt to CPS resource constraints.

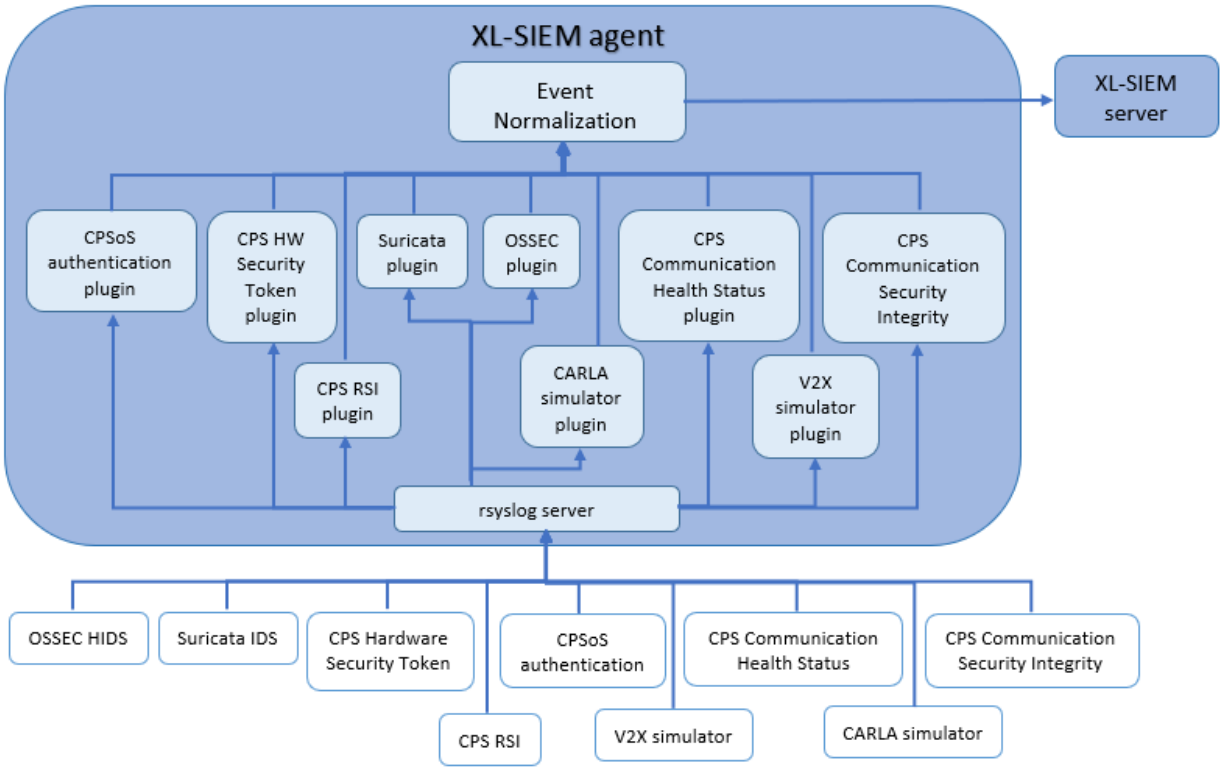

Figure 9. XL-SIEM agent in CPSoSAware

The XL-SIEM already contains a library of plugins, some are inherited from AlienVault OSSIM SIEM [7] and some others have been developed by Atos from previous deployments in other relevant research projects (such as FINSEC [8], CIPSEC [9] or ANASTACIA [10]) and business cases. To develop new plugins, a corresponding configuration file needs to be created, where among other things it is necessary to define the Regular Expression that parses the original raw data and translates into the corresponding event format fields, as is explained in deliverable D3.5 [11]. Figure 10 shows an excerpt from a plugin that parses the original raw data log produced by a CPS Hardware Security Token sensor, developed for this project, and assigns the relevant information extracted to the appropriate event fields.

```
1   source=log
2   location=/var/log/hst_syslog.log
3   create_file=false

4   [translation]
5   …
6   Memory Anomaly=5
7   CPU Runtime Anomaly=6

8   # Jun 23 08:42:39 887a7bd7e8b2 root [HSTSensor] {HostIP: 192.168.1.10, HostID: 03c92c24,
    User:0, HostState:admin, HSMid: 007f0101, timestamp: 1624010331, event:{ type: 5, failure:
    0, severity: 2}, comments: Memory Anomaly }
9   [04a - CPSOSAWARE - HST Event Log]
10  event_type=event
11  regexp="(?P<date>\w{3}\s{1,2}\d{1,2}\s\d{2}:\d{2}:\d{2})            (?P<sensor_device>.*?)
    (?P<syslog_username>.*?)                                          \[(?P<sensor_tag>.*?)\]
    {\s*HostIP:\s*(?P<HostIP>.*?),\s*HostID:\s*(?P<HostID>.*?),\s*User:\s*(?P<User>.*?),\s*Hos
    tState:\s*(?P<HostState>.*?),\s*HSMid:\s*(?P<HSMid>.*?),\s*timestamp:\s*(?P<timestamp>.*?)
    ,\s*event:\s*{\s*type:\s*(?P<type>.*?),\s*failure:\s*(?P<failure>.*?),\s*severity:\s*(?P<s
    everity>.*?)},\s*comments:\s*(?P<comments>.*?)\s*}"
12  plugin_sid={translate($comments)}
13  date={normalize_date($date)}
14  src_ip={resolv($HostIP)}
15  userdata1={$HostID}
16  userdata2={$sensor_device}
17  username={$User}
18  userdata3={$HostState}
19  userdata4={$HSMid}
20  userdata5={$type}
21  userdata6={$failure}
```

Figure 10. Excerpt of the CPS Hardware Security Token plugin configuration file

Table 8 shows an example of the input and corresponding output produced by the XL-SIEM agent for an event generated by the CPS Hardware Security Token. In the deliverable D3.5 [11] there are more details on how message logs are processed to generate events, using this same example.

Table 8. XL-SIEM input and output examples

Input to XL-SIEM Agents: CPS Hardware Security Token
Apr 12 08:00:32 233fa20390d9 root [HSTSensor] {HostIP: 192.168.1.10, HostID: 03c92c24, User:0, HostState:admin, HSMid: 007f0101, timestamp: 1624010331, event:{ type: 5, failure: 0, severity: 2}, comments: Memory Anomaly }
Output from XL-SIEM: XL-SIEM Event
{"event":{"type":"detector","date":"1649750432","device":"10.0.2.15","interface":"eth0","plugin_id":"15000 0","plugin_sid":"5","src_ip":"192.168.1.10","dst_ip":"192.168.1.10","username":"MA==","userdata1":"MD

NjOTJjMjQ=","userdata2":"MjMzZmEyMDM5MGQ5","userdata3":"YWRtaW4=","userdata4":"MDA3ZjAxMDE=","userdata5":"NQ==","userdata6":"MA==","userdata7":"Mg==","log":"QXByIDEyIDA4OjAwOjMyIDIzM2ZhMjAzOTBkOSByb290IFtIU1RTZW5zb3JdIHtIb3N0SVA6IDE5Mi4xNjguMS4xMCwgSG9zdElEOiAwM2M5MmMyNCwgVXNlcjowLCBIb3N0U3RhdGU6YWRtaW4sIEhTTWlkOiAwMDdmMDEwMSwgdGltZXN0YW1wOiAxNjI0MDEwMzMxLCBldmVudDp7IHR5cGU6IDUsIGZhaWx1cmU6IDAsIHNldmVyaXR5OiAyfSwgY29tbWVudHM6IE1lbW9yeSBBbm9tYWx5IH0g","fdate":"2022-04-12 08:00:32","event_id":"ba3611ec-aded-0242-ac11-0003a02ecf28"}}

3.3.2 Input data and format

The main input of the XL-SIEM technology is the XL-SIEM Event. Figure 11 shows the JSON data format of an XL-SIEM Event, based on the original OSSIM Event format [12], which is the output produced by the agents.

```
"a": {“type”: <string>,
"date": <string>,
"device": <string>,
"interface": <string>,
"plugin_id": <integer>,
"plugin_sid": <integer>,
"src_ip": <string>,
"dst_ip": <string>,
"src_port": <string>,
"dst_port": <string>,
"userdata1": <string>,
"userdata2": <string>,
"userdata3": <string>,
"userdata4": <string>,
"userdata5": <string>,
"userdata6": <string>,
"userdata7": <string>,
"userdata8": <string>,
"userdata9": <string>,
"log": <string>,
"fdate": <string>,
"tzone": <string>,
"event_id": <string>,
"username": <string>,
“password”: <string>,
"filename": <string>,
"organization": <string>
}
```

Figure 11. XL-SIEM event data: JSON format

Table 9 explains the meaning of each of the fields in the JSON data format.

Table 9. XL-SIEM event data: fields description

Fields	Description
Type*	Type of Agent: monitor, detector
Date*	Date and time of the event (*long* data type)
Device*	IP address of the agent that processed the event
Interface*	Network interface used by the agent
Plugin_ID*	Plugin ID used by the agent to parse and process the raw data
Plugin_SID*	Event type, as defined in the Plugin ID specification
Dst_IP	IP address for the destination of the event
Src_IP	IP address for the source of the event
Dst_Port	Destination port of the event
Src_Port	Source port of the event
Filename	Name of file associated with the event.
Username	The username associated with the event.

Password	The password associated with the event.
Userdata 1-9	User-created fields that can be used freely to log additional information
Organization	The name of the organization that owns the infrastructure where the agent is running
FDate	Full Date and Time the event was logged (ISO 8601 standard format)
TZone	Time Zone
Log	Raw log details that generated the event (Base 64 encoded)
EventID	Unique identifier of the event

3.3.3 XL-SIEM Storm topology: event processing and security analysis

The XL-SIEM triggers alerts based on correlation of different events. This task is divided in three main phases. First, the events that the SIEM receive are filtered by **Policies**. The events that pass policies are aggregated by **EPL statements** in general events. Then, **EPL directives** correlate the *statement events*. In this phase, correlation rules (EPL pattern) can consider fine-grained detail of the events, as source or destination IP or port. When a sequence of events meets the EPL directive pattern, the correlation engine raises the associated security alert.

3.3.3.1 Policy

Once the events are received from the XL-SIEM agents to the server and before their, the system verifies if the user has specified some conditions to filter the incoming events before they arrive to the correlation engine (e.g., source/destination IP, port, time/date range, type of event, or the SIEM agent where the event is collected). This is done with the definition of **Policies** which allow for example to have separated processing and correlation of events from different organisations or realms and comply with legal or business requirements.

Figure 12 shows an example of a policy used in the project. It accepts events from any source and port to any destination and port. In addition, it accepts any event type in any time. But it only accepts event from the agent-xlsiem-server agent.

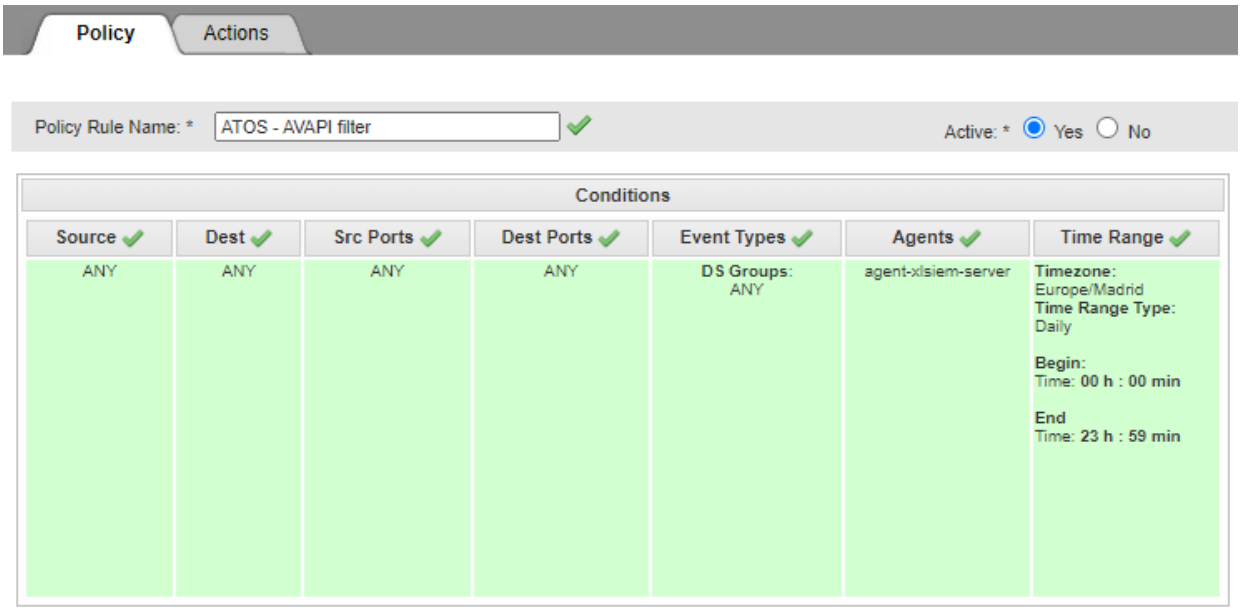

Figure 12. Filtering policy example

3.3.3.2 EPL Statement

As an intermediate phase between the Policies and the EPL Directives, the **EPL Statements** classify the events from the different agents and sensors into general events that the EPL Directive uses to raise security alarms.

Table 10 shows two examples of ELP Statements, which classify events form the CPS Hardware Security Token, plugin_id=150000. In the first case, events with plugin_sid equal to 1, 2, 3, or 4 are classified as CPS_HWToken_Verification_Failure. These events are related to the integrity and authentication of messages. In the second case, the events with plugin_sid equal to 5 or 6 are classified as CPS_Abnormal_ResourceUsage. This case covers the identification of abnormal use of memory or CPU runtime. For these examples, the meaning of the plugin_sid can be consulted in Figure 12.

Table 10. EPL Statements examples

ELP Statement	
CPS_HWToken_Verification_Failure	insert into CPS_HWToken_Verification_Failure select * from ossimSchema_default where (plugin_id=150000) and (plugin_sid in (1,2,3,4))
CPS_Abnormal_ResourceUsage	insert into CPS_Abnormal_ResourceUsage select * from ossimSchema_default where (plugin_id=150000) and (plugin_sid in (5,6))

3.3.3.3 ELP directive

The last phase of the correlation process is the application of the **EPL Directives**. These directives are composed by the name, which are also used as alert name; the EPL Pattern, which is a representation of the sequence of events that the directive identifies; category and subcategory of the directive; and the reliability the priority of the alarm.

Figure 13 shows an example of one EPL Directive developed for this project. This directive raises an alert when the system has an abnormal behaviour, *CPS_SystemBehaviour_Abnormal* alert. The `EPL pattern` considers every *CPS_Abnormal_ResourceUsage* events (a) with distinct `userdata4`, retaining the value of `userdata4` for 60 seconds. It searches a second *CPS_Abnormal_ResourceUsage* events (b) which happens after a `event` (`->`) with the same `userdata4` (`a.userdata4=b.userdata4`). This search is repeated 2 times (denoted by `[2]`). The EPL Directive `category` is *CPS* and `subcategory` *System Behaviour*, this classification is also applied to alarms. The `reliability` is 8 and the `priority` of the alarm is 2.

Directive SID *	103
Name *	CPS_SystemBehaviour_Abnormal
EPL Pattern	pattern [every-distinct(a.userdata4,60 seconds) a=CPS_Abnormal_ResourceUsage -> ([2] b= CPS_Abnormal_ResourceUsage (b.userdata4=a.userdata4))]
Category	CPS
Subcategory	System Behaviour
Reliability *	8
Priority *	2

Figure 13. EPL directive example

3.3.4 Alarms format, export and data sharing

When one or more events received at the XL-SIEM server match a certain security rule considering different events collected at different layers. These alarms are expressed through a predefined JSON format, and shared with other components, both internal and external, with respect to the organization who hosts the XL-SIEM itself. Figure 14 shows a list of the fields that can be found in the JSON associated to an XL-SIEM alarm:

```
{"AlarmEvent": {
        "DST_IP_HOSTNAME": <string>,
        "RELATED_EVENTS": <string>,
        "DST_IP": <string>,
        "PLUGIN_NAME": <string>,
        "SRC_IP": <string>,
        "PRIORITY": <integer>,
        "RELIABILITY": <integer>,
        "SUBCATEGORY": <string>,
        "USERDATA3": <string>,
        "USERDATA4": <string>,
        "PLUGIN_SID": <string>,
        "USERDATA1": <string>,
        "USERDATA2": <string>,
        "ORGANIZATION": <string>,
        "CATEGORY": <string>,
        "PLUGIN_ID": <string>,
        "USERNAME": <string>,
        "FILENAME": <string>,
        "BACKLOG_ID": <string>,
        "RELATED_EVENTS_INFO": {List of <Event>},
        "PROTOCOL": <integer>,
        "RISK": <integer>,
        "SRC_PORT": <integer>,
        "SENSOR": <string>,
        "SRC_IP_HOSTNAME": <string>,
        "SID_NAME": <string>,
        "USERDATA7": <string>,
        "DATE": <string>, ? YYYY-mm-dd HH:MM:SS
        "USERDATA8": <string>,
        "USERDATA5": <string>,
        "USERDATA6": <string>,
        "PASSWORD": <string>,
        "USERDATA9": <string>,
        "DST_PORT": <integer>,
        "EVENT_ID": <string>}
}
```

Figure 14. XL-SIEM alarms JSON data format

Some fields are self-explanatory (e.g., EVENT_ID, DATE, SRC_PORT, DST_PORT, DST_IP, SOURCE_IP…), but others require specific explanation:

- RELATED_EVENTS: XL-SIEM generates alarms considering one or more events who match a certain rule. This field contains the id of the events who led to the generation of the alarm
- PRIORITY: priority value evaluated by the XL-SIEM, associated to the raised alarm
- RELIABILITY: reliability value evaluated by the XL-SIEM, associated to the raised alarm
- RISK: risk value evaluated by the XL-SIEM, associated to the raised alarm. Risk calculation is based on this formula: *Asset Value * Event Reliability * Event Priority / 25 = Risk*
- RELATED_EVENTS_INFO: information about each single event that contributed to the alarm generation. There is an upper limit of the maximum number of events that can be inserted here.
- SID_NAME: high-level description of the alarm
- CATEGORY: category of the alarms
- SUB-CATEGORY: sub-category of the alarm

Besides the native OSSIM data format for Alarms, the XL-SIEM supports export alarms into MISP (Malware Information Sharing Platform) [13] and STIX (Structured Threat Information eXpression) [14] version 2.0 (STIX2), which are standard formats widely used Threat Intelligence Data Sharing.

4 Demonstration scenario

For this demonstration scenario we are assuming a set of autonomous vehicles, each one equipped with a set of sensors that improve the driving experience, e.g., in terms of usability, safety; and communication capabilities to connect to other vehicles or smart devices of their environment through V2X technologies. These technologies are very useful and convenient, enabling cooperative information sharing for streamlining traffic movement, improving road safety, etc. But on the other hand, these technologies make autonomous cars vulnerable, and a malicious actor exploiting these vulnerabilities may entail serious consequences in the safety of the driver and in the surrounding traffic context. For this reason, it is of paramount importance being able to firstly, monitor and detect anomalies in real-time, secondly correlate these with additional contextual information to determine if there is a security incident happening and thirdly, to assess the risk that this incident may pose to the safety, and potentially to other factors that guarantee the correct and effective functioning and operation of the autonomous vehicle. Being aware of these security incidents in real-time and the risk they pose, permits to warn and to take corrective actions promptly, in order to mitigate their impact and minimize negative consequences.

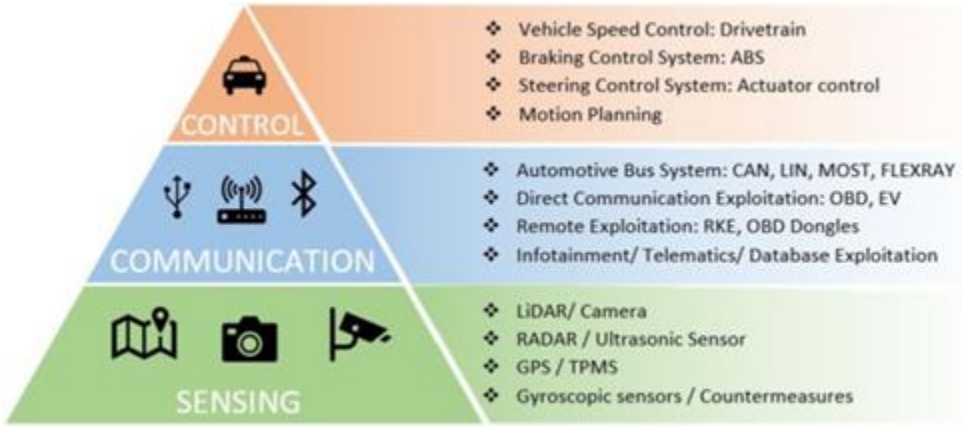

Figure 15. AV/ADAS vehicle – Security Monitoring Ecosystem [27]

Figure 15 which was already included in D6.1 [28], depicts the elements, or assets, that should be monitored and protected in an autonomous car scenario, grouped into three categories: vehicle control module, communication, and sensing. This landscape was already introduced in D6.1 but we include it here again for self-containment of this document. We should consider attacks and threats for each of the three categories of assets:

- **Threats/Attacks on the sensors.** Connected and automated vehicles depend on the collection of large volumes of sensors data and processing them to operate safely by maintaining the field of safe travel. Contactless sensors layer attacks can disturb operations of the perception layer of the AV/ADAS stack of the vehicle and can cause hazardous road situations. Practically all types of sensors can be attacked. The following is a non-exhaustive list of attacks that fall under this category:
    - *On-board camera exploit*: this benefits on common vulnerabilities that affect any smart IP camera and that permit attackers to control remotely the device for their own malicious purpose.

- *GPS sensor spoofing*: A location spoofing attack attempts to deceive a GNSS/RTK receiver by broadcasting incorrect satellite signals, structured to resemble a set of normal satellite signals (e.g., GPS, GLONASS, GALILEO, etc.).
- *Lidar sensor exploit - Adversarial attacks – data poisoning*: Lidar might be attacked by recording outbound optical signal and sending it back to optical receiver, camera can be disturbed with pointed laser beam (that can also permanently damage its CMOS/CCD sensors) or direct illusional attack on specific classification machine-learning algorithm and ultrasonic sensor can be jammed by generating ultrasonic noise, spoofed by crafted fake ultrasonic echo pulses or even quieted.
- *Relay attacks – Man-in-the-middle attacks:* can be used for sensor information (e.g. car positioning) stealing, modification and replay.

- **Threats/Attacks on the communication**, where we can distinguish:
  - Vehicle to Vehicle (V2V) and V2X attacks: with a focus on especially wireless access technologies and attacks detection including diverse types of attacks(e.g. masquerade, wormhole, man-in-the-middle) and assumes three main vectors of attacks for wireless communication: frequency of malicious communication, the effect of the attack on V2X (e.g. injection of fabricated message, message mutation or even preventing delivery of the message) and effect on the vehicle (e.g. compromising safety or loss of efficiency of the targeted cooperative application).
  - Intravehicular communication attacks, with a focus on CAN bus data manipulation.
  - Distributed Denial of Service (DDoS): consists in attacking a specific service of the system simultaneously and continuously from different sources and using various attacking mechanisms, with the objective of cancelling completely (or significantly degrade) the service. This attack may cause a disruption of the traffic flow, a collision of the vehicle or damages to the infrastructure.
- **Threats/Attacks on the vehicle control module**: in this category we distinguish side-channel attacks, fault-injection, and code-injection attacks, aiming at disclosing, altering and replaying sensible information used by the OBD or the ECU. ECU Firmware tampering or rogue updates have large implications as it can completely reprogram the vehicle's behaviour, resulting in it becoming a potential threat to public safety.

4.1 Distributed Denial of Service (DDoS) Attack demonstration

In this section, we are illustrating the use of the SRMM to monitor and detect a local threat/attack on the communication layer: a *Distributed Denial of Service (DDoS) attack.* In this type of attack, attackers can be vehicles connected to the network that send a volume of requests higher than what the system can handle, causing a downtime of the service. A *flooding attack* is a specific type of DoS that consist in generating traffic in order to exhaust network resources. In this demonstration scenario, we assume a set of compromised vehicles, controlled by an attacker, that use the flooding attack to disable the wireless network access point. For example, by sending large number of requests for establishing connection, therefore depleting the resources of the node. This will cause other vehicles in the network, who are already attached to it, to be (at least temporarily) isolated from the network.

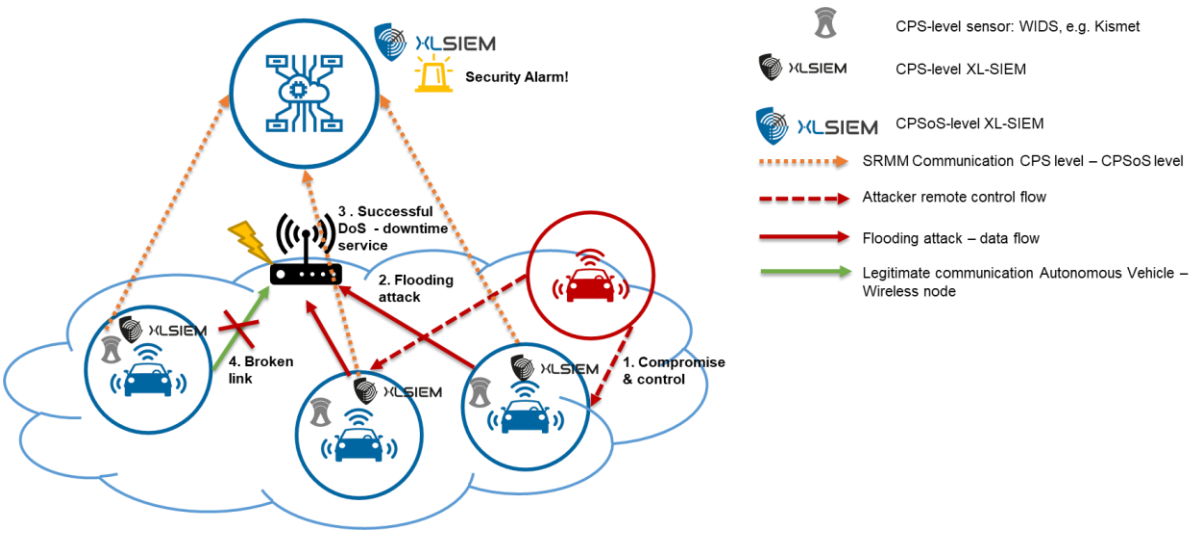

Figure 16. Demonstration scenario at CPSoS level [3]

In order to detect this type of attack in a network of autonomous vehicles communicating through wireless, we leverage the capabilities of the SRMM of CPSoSAware in the following way:

- Each autonomous vehicle is equipped with:
  - **Wireless Intrusion Detection System (WIDS)**, such as Kismet, acting as a security monitoring sensor.
  - **CPS-level XL-SIEM agent** that collects the security information generated by the WIDS sensor, normalizes it into security events, and forwards the events to the XL-SIEM server for correlation.
  - **CPS-level XL-SIEM server** that correlates all the security events received from the agent to alert of an attack that is happening. This instance of the XL-SIEM is the lightweight version that contains a subset of security rules specific to detect threats and attacks relevant for the autonomous vehicle context from the perspective of the individual CPS.
- The system-level SRMM is equipped with **a CPSoS-level XL-SIEM**. This XL-SIEM is a complete version of the technology and contains a full set of security directives relevant for the Autonomous Vehicle context from a wider perspective. For example, security directives that can detect attacks and threats that affect the inter-communication infrastructure, the cloud services, as well as specific rules designed to detect threat scenarios that cross-correlate alarms generated by more than one autonomous vehicle connected to the network and system-level alarms.

Figure 17 shows how the attack is performed at CPS-level

1- An attacker (depicted in red in the figure) gains access and control to one or more legitimate autonomous vehicles (e.g. through a malware installed or a exploit). A control flow is established between the attacker and the compromised vehicles that permit the attacker to use the legitimate connection between the vehicles and the Wireless AP node to perform malicious actions.
2- The compromised vehicle issues an unusual high number of requests to the Wireless AP (e.g. for disconnecting and re-establishing a connection), on behalf of the attacker.
3- The WIDS sensor running at the vehicle monitors the wireless communication and logs the unusual activity of the vehicle. This is detected by the CPS-level XL-SIEM, through several events collected and a security alarm warning of a potential data flooding attack is raised at the vehicle.

This alarm is also sent to the CPSoS level XL-SIEM for cross-correlation with other alarms raised by this vehicle or others in the realm.

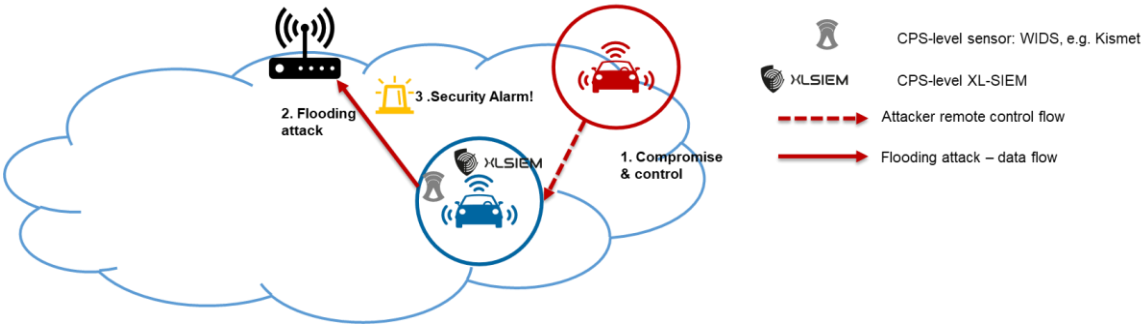

Figure 17. Demonstration scenario: at CPS level

At the CPSoS level, several alarms of the same type, i.e. "Wireless data flooding attack", are received from more than one vehicle in the realm (represented by an orange arrow in Figure 16). In the CPSoS XL-SIEM, there is a security directive that is activated when various alarms of type "Wireless data flooding attack" are received from different sources (i.e. different vehicles) and related to the same destination (i.e. Wireless AP) within a small time frame. If this behaviour is observed in the realm, a security alarm is triggered at the CPSoSAware XL-SIEM indicating that a "Wireless attack, successful denial of service against access point on GPS position".

Because the V2X technology approach cannot identify vehicles and has to send the message in broadcast mode, this type of attack only can only be identified. There is no way to block a broadcast on the communication channel. So, there is not possible response or mitigation.

4.2 Local firmware update detection and mitigation demonstration

In this demonstrator scenario, we consider that some vehicles have two *vulnerable* devices that an attacker can exploit. The first is a wireless connection device, such as Bluetooth, that could receive a message and resend through internal communication network to all other devices inside the vehicle. This first device does not check the message authority, relying on the security of the other devices. The second device allows to update their firmware from an URL passed as a command parameter. In this case, it relies on the message origin, the internal communication, to accept the update request.

With this scenario, an attacker can send a Bluetooth message to the fist devices that is resend to all devices inside the vehicle. While the most of devices ignore the message, the second vulnerable device updates its firmware from the URL passed in the attack message without any validation. The Figure 18 depicts the sequence diagram of the attack and the response of the Lightweight SRMM.

When the update is complete, the second device generates a *Successful update* event that triggers the validation action associated with updates. The Lightweight SRMM deployed on the vehicle captures this event and requests the CPS RIS to validate the new version. The CPS RIS takes the firmware fingerprint and compares it to the manufacturer's version list. If fingerprint is not in the list, as is this case, the CPS RIS raises an *Illegitimate update* event that the Lightweight SRMM receives and handles, escalating the event to the Area SRMM. The action associated with this alarm launches an update on the vulnerable

device with the latest legitimate firmware from the manufacturer's list. This produces a new *Successful update* event that the CPS RIS must validate again. But in this case, the update uses a legitimate firmware which raises a *Legitimate update* event, and it does not trigger any alerts in the Lightweight SRMM.

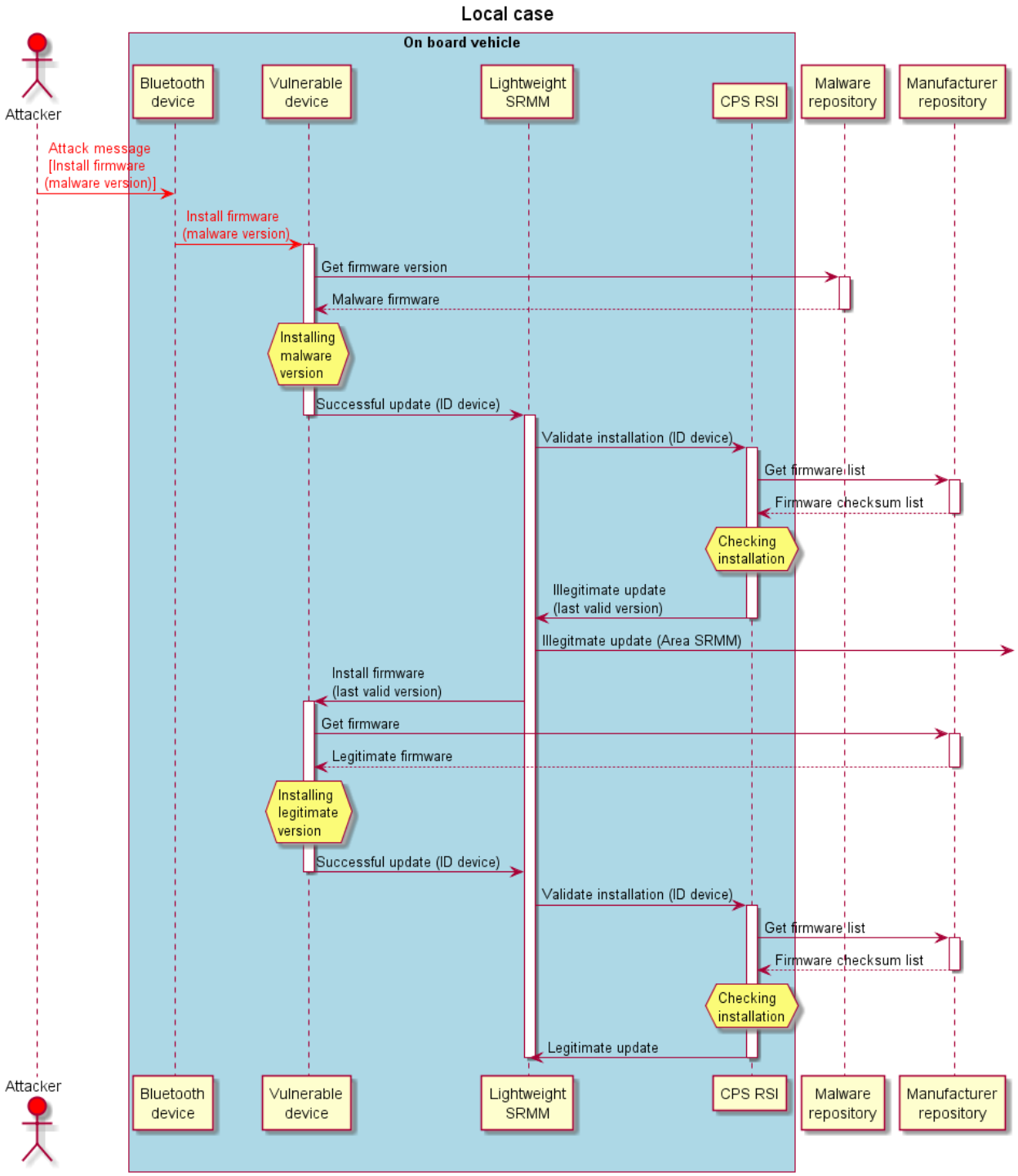

Figure 18. Local attack case

4.3 Area firmware update detection and mitigation demonstration

This scenario extends the previous use case where several attacks occur in the same area against several vehicles. In this case, the Area SRMM, located in the edge server, correlate the security events to decide to block the repository. This action is implemented by a new rule that is sent to Lightweight SRMMs, which are on the vehicles.

Figure 19 shows the sequence diagram where two vehicles receive malicious message, one of them is targeted twice. Locally, the Lightweight SRMM responses to the incident, mitigating the attack as is described in the previous case. Each time the Lightweight SRMM detects a local incident may raise a security event to upper SRMM.

The Area SRMM receives three *Illegitimate update* events, it correlates event information, such as the repository address. Then it sends to all vehicles in the area a new security event to block the communication with the malware repository. This event is received by the Lightweight SRMMs, triggering the associated action, which is an update of communication OBU rules to block the URL and IP of the malware firmware. As all vehicles under Area SRMM have to implement this rule, the attack is mitigated in the area.

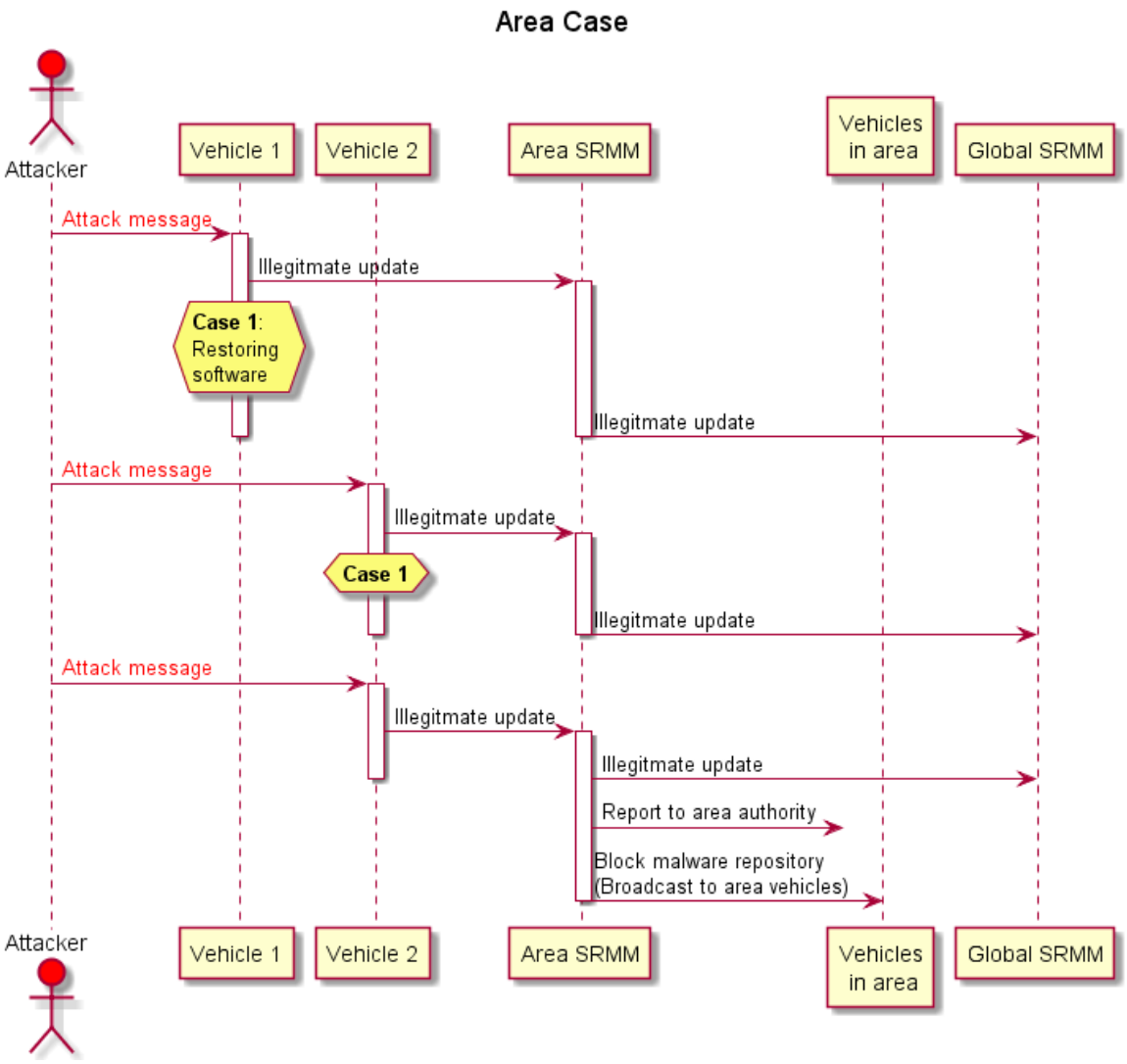

Figure 19. Area attack case

4.4 Global firmware update detection and mitigation configuration

This last scenario is considered the Global SRMM perspective. In this case, the SRMM correlates security events that receives from lower SRMMs, trying to achieve a general and global solution. Contrary to the previous case, which the SRMMs do not solve the problem and just response to attacks with a mitigation.

Figure 20 depicts what happens when several security events occur in different areas. When the Global SRRM receives 5 events, it sends to all Area SRMM a rule to block the malware repository, as do Area SRMM when the events occur on one area. This rule is broadcasted to Local SRMM where is implemented, the associated action modifies the communication OBU rules to block the connexion with the repository. This action mitigates the attacks, but it could be replicated with a different repository address. So, the Global SRMM generates a warning report that should be delivered to manufacturers to fix the detected vulnerabilities.

When the new firmware version is available, the Global SRMM broadcasts a rule update to Lightweight SRMMs through Area SRMMs. This rule compares the current firmware version against the new version. If they are different, the action associated to the rule triggers an update process that fixes the vulnerabilities.

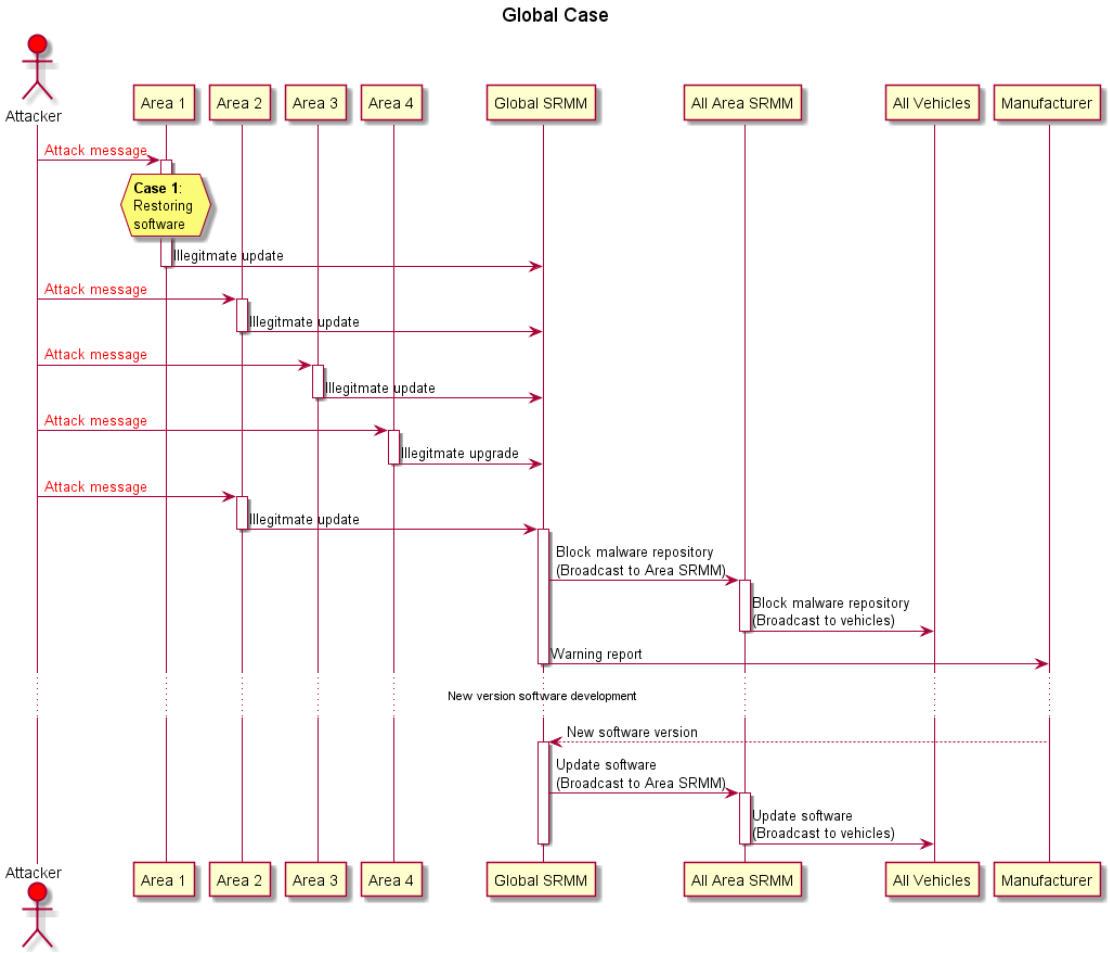

Figure 20. Global attack case

5 Conclusions and next steps

Deliverable D4.8 presents SRMM as a relevant component part of the CPSoSAware architecture. SRMM, based on XL-SIEM technology, monitors the CPSs to warn about any suspicious and malicious activity both at individual CPS and CPSoS level, which may have an impact on the security properties of the infrastructure and its assets. The document presents its internal modules, communication flow and operation. Also, its position in the reference architecture considering which other components it interacts with. The deliverable describes an example of usage within a specific demonstration scenario, that of the Automotive Pillar. It proposes a three-layered deployment of the SRMM, with a hierarchical architecture. The CPSs at the lowest levels of the architecture are monitored by a lightweight version of the SRMM that has reduced consumption and contains a very limited set of detection capabilities, that are increased in the upper layer while the SRMM on the top layer works at full detection capacity. This approach enables local security awareness inside the vehicle and permits the driver or the system to take action basing on this information. In addition, it allows distributing the workload and optimize processes, as the SRMM at the bottom layer will release the upper ones from processing thousands of events, and in the same way the top layer SRMM will not have to process all the elementary events coming from the individual CPSoS.

With the submission of this document task T4.3 concludes. WP5 and especially the Automotive use case within WP6 will be the main beneficiaries of the results of this task within next months

References

[1] CPSoSAware D1.4: Second version of CPSoSAware System Architecture

[2] CPSoSAware D1.1: Supporting, motivating and persuasive approaches, tools and metrics

[3] CPSoSAware D4.3: Preliminary version of CPSoS Runtime Security Monitoring Approaches

[4] ENISA Threat Landscape 2020. https://www.enisa.europa.eu/topics/threat-risk-management/threats-and-trends

[5] ENISA Threat Landscape 2021: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021

[6] https://cybersecurity.att.com/documentation/usm-appliance/plugin-management/tutorial-developing-new-plugin.htm

[7] https://www.alienvault.com/docs/data-sheets/usm-plugins-list.pdf

[8] FINSEC Consortium, "FINSEC: Integrated Framework for Predictive and Collaborative Security of Financial Infrastructures," May 2018. [Online]. Available: https://www.finsec-project.eu/. [Accessed 2022 March 18].

[9] CIPSEC Consortium, "CIPSEC: Enhancing Critical Infrastructure Protection with innovative SECurity framework," May 2016. [Online]. Available: https://www.cipsec.eu/. [Accessed 18 March 202].

[10] ANASTACIA Consortium, "ANASTACIA: Advanced Networked Agents for Security and Trust Assessment in CPS/IOT Architectures," January 2017. [Online]. Available: http://www.anastacia-h2020.eu/. [Accessed 18 March 2022].

[11] CPSoSAware Consortium, "D3.5 Modules for enabling Security and Trust," 2022.

[12] https://cybersecurity.att.com/documentation/usm-appliance/events/event-details-fields.htm

[13] http://www.misp-project.org/

[14] https://stixproject.github.io/

[15] https://cybersecurity.att.com/products/ossim

[16] G. Gonzalez-Granadillo, S. Gonzalez-Zarzosa and M. Faiella, "Towards an Enhanced Security Data Analytic Platform," in *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications*, Porto, Portugal, 2018.

[17] https://cybersecurity.att.com/documentation/resources/pdf/usm-appliance-user-guide.pdf

[18] http://www.espertech.com/esper/

[19] http://storm.apache.org/

[20] https://docs.oracle.com/cd/E13213_01/wlevs/docs20/pdf/epl_guide.pdf

[21] https://www.rabbitmq.com/

[22] Gartner, "Security Information and Event Management (SIEM) tools Reviews and Ratings". https://www.gartner.com/reviews/market/security-information-event-management

[23] Forrester, "The Forrester Wave™: Security Analytics Platforms, Q4 2020", December 2020.

[24] TechTarget – SearchSecurity, "A guide to SIEM platforms, benefits and features". https://searchsecurity.techtarget.com/buyershandbook/A-guide-to-SIEM-platforms-benefits-and-features

[25] DISIEM D3.1: Security Metrics and Measurements

[26] The CIS Security Metrics, v1.1.0.

[27] Zeinab El-Rewini, Karthikeyan Sadatsharan, Daisy Flora Selvaraj, Siby Jose Plathottam, Prakash Ranganathan: “Cybersecurity challenges in vehicular communications”. Vehicular Communications, volume 23, 2020, 100214, ISSN 2214-2096. https://doi.org/10.1016/j.vehcom.2019.100214

[28] CPSoSAware D6.1: Definition and planning of quantification trials

[29] https://mqtt.org/

[30] DISIEM D2.1: In-depth analysis of SIEMs extensibility

[31] FINSEC D4.1: SIEM infrastructure and tools I

[32] https://csiac.org/articles/evaluation-of-comprehensive-taxonomies-for-information-technology-threats/