



SDN-μSense

Project No. 833955

Project acronym: SDN-microSENSE

Project title:

SDN - microgrid reSilient Electrical eNergy SystEm

Deliverable D5.1

XL-SIEM System

Programme: H2020-SU-DS-2018

Start date of project: 01.05.2019

Duration: 36 months

Editor: ATOS

Due date of deliverable: 30/09/2020

Actual submission date: 30/09/2020



Deliverable Description:

Deliverable Name	XL-SIEM System
Deliverable Number	D5.1
Work Package	WP 5
Associated Task	T5.1
Covered Period	M5
Due Date	M17
Completion Date	M17
Submission Date	30/09/2020
Deliverable Lead Partner	ATOS
Deliverable Author(s)	ATOS, UOWM, CERTH, TECN, UBI, SID, OINF, CLS, AYE
Version	1.5

Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

CHANGE CONTROL
DOCUMENT HISTORY

Version	Date	Change History	Author(s)	Organisation
0.1	12/06/2020	ToC	Ruben Trapero	ATOS
0.1.1	23/6/2020	Changed ToC advised by QM	Ruben Trapero	ATOS
0.1.2	12/7/2020	Section 2, Section 3.1, Section 4.3	Ruben Trapero	ATOS
0.1.3	12/7/2020	Section 4.2.2	Orestis Mavropoulos	CLS
0.2	14/7/2020	Section 4.1.2	Panagiotis I. Radoglou-Grammatikis	UOWM
			Elisavet Grigoriou	SID
0.2.1	15/7/2020	Section 3.2	Angel Javier Jimenez Perez	AYE
0.2.2	15/7/2020 15/7/2020	Section 4.1.3, Section 4.1.4, Section 4.2.4, Section 4.2.5, Section 4.2.6, Section 4.2.7	Ioannis Spyridis	OINF
			Panagiotis I. Radoglou-Grammatikis, Panagiotis Sarigiannidis, Thomas Lagkas, Antonios Protopsaltis,	UOWM

			Stamatia Bibi, Anna Triantafyllou, Miltiadis Parcharidis, Giorgos Kechaidis	
			Elisavet Grigoriou	SID
0.2.4	16/7/2020	Section 4.1.4, Section 4.2.1	Marisa Escalante	TECN
0.3	20/7/2020	Section 5, Section 6	Ruben Trapero	ATOS
0.3.1	21/7/2020	Section 4.1.1, Section 4.1.6, Section 4.2.9	Eleni Ketzaki	CERTH
0.4	21/7/2020	First integrated version for milestone 1	Ruben Trapero	ATOS
0.5	23/7/2020	Section 1.3, Section 4.2.2, Section 7	Ruben Trapero	ATOS
0.6	30/7/2020	Section 1.1, Section 1.2, Executive summary, Conclusions	Ruben Trapero	ATOS
0.7	3/8/2020	Section 3.2	Angel Javier Jimenez Perez	AYE
1.0	24/8/2020	Ready to review	Ruben Trapero	ATOS
1.1	17/09/2020	Industrial revision addressed	Ruben Trapero	ATOS
1.2	18/09/2020	Section 4.1.5 added	Ruben Trapero	ATOS
1.3	22/09/2020	Academic revision addressed	Ruben Trapero	ATOS
1.4	24/09/2020	SAB revision addressed	Ruben Trapero	ATOS
1.5	28/09/2020	QM revision addressed	Ruben Trapero	ATOS

DISTRIBUTION LIST

Date	Issue	Group
21/09/2020	Revision	UOWM, IEIT, VETS, SAB, QM, TM
29/09/2020	Acceptance	UOWM, IEIT, VETS, SAB, QM, TM
30/09/2020	Submission	ATOS

SAB APPROVAL

NAME	INSTITUTION	DATE
Mr. Benito Caracuel Sillero	SCHN ES	21/09/2020

Academic and Industrial partner revision

NAME	INSTITUTION	DATE
Anna Triantafyllou Panagiotis Radoglou- Grammatikis	Academic partner: UOWM	21/09/2020
Maria Atanasova	Industrial partner: IEIT	17/09/2020

Valentin Kolev	Industrial Partner: VETS	18/09/2020
----------------	--------------------------	------------

Quality and Technical manager revision

NAME	INSTITUTION	DATE
Dimosthenis Ioannidis	CERTH	29/09/2020
Anastasios Drosou	CERTH	29/09/2020

Table of contents

Table of contents	5
Table of figures	7
Table of tables	7
Acronyms	8
Executive Summary.....	10
1 Introduction	11
1.1 Relation to other Tasks and Deliverables	11
1.2 Requirements analysis.....	12
2 XL-EPDS overview	13
3 Incident detection based on the XL-SIEM in SDN-microSENSE	15
3.1 XL-SIEM Deployment Considerations	16
3.2 Deployment considerations in SDN-microSENSE	21
4 Monitoring and Incident Detection in SDN-microSENSE	24
4.1 SDN-microSENSE protocols and threats	24
4.1.1 Modbus	24
4.1.2 DNP3	28
4.1.3 IEC 60870-5-104.....	29
4.1.4 IEC 61850.....	31
4.1.5 IEEE C37.118-2	33
4.1.6 Other relevant protocols within the EPES domain	35
4.1.6.1 MQTT protocol.....	35
4.1.6.2 NTP protocol	36
4.2 SDN microSENSE security sensors	37
4.2.1 EPES Honeypots	37
4.2.2 Network based incident detection: Enhanced Suricata.....	39
4.2.3 SDN based incident detection: Nightwatch	40
4.2.4 Modbus Intrusion Detection Sensor	41
4.2.5 DNP3 Intrusion Detection Sensor	41
4.2.6 IEC 60870-5-104 Intrusion Detection Sensor	42
4.2.7 IEC 61850 (GOOSE) Intrusion Detection Sensor	42
4.2.8 L-ADS.....	42

4.2.9	CERTH ML detector	43
4.3	Interfaces and data exchanged	45
4.3.1	XL-SIEM input mechanisms	45
4.3.2	XL-SIEM output mechanisms.....	47
5	<i>Event connectors.....</i>	51
6	<i>Security alerts</i>	53
7	<i>Unit Testing</i>	59
7.1	Unit Testing Environment	59
7.2	Unit tests.....	60
8	<i>Innovation Summary</i>	66
9	<i>Conclusions</i>	67
10	<i>References.....</i>	68

Table of figures

Figure 1. Links between D5.1 and the rest of deliverables and WPs.....	12
Figure 2. The XL-EPDS within the high level SDN-microSENSE architecture	13
Figure 3. XL-EPDS global architecture	13
Figure 4. T5.1 components within WP5 architecture	15
Figure 5. Gartner magic quadrant for SIEMS (taken [Gartner2020]) positioning the XL-SIEM (in orange)	16
Figure 6. XL-SIEM overview	17
Figure 7. Internals of the XL-SIEM engine	18
Figure 8. Docker containers deployed for an XL-SIEM installation.....	19
Figure 9. Comparison of requirements for deploying XL-SIEM containers	21
Figure 10. XL-SIEM infrastructure resources	21
Figure 11. ISA 95/98 standard - Industrial networks	23
Figure 12. SOAP/REST vs Streaming flow	24
Figure 13. Description of the Modbus protocol operation	25
Figure 14. IEC 60870-5-104 Protocol	29
Figure 15. Typical general view of a primary substation	31
Figure 16. IEC 61850 Data modelling. Source IEC/TR 61850.....	32
Figure 17. Synchrophasor communication system overview (taken from [Khan+16])	33
Figure 18. Message format of the IEEE C37.118-2 standard (taken from Khan+16)).....	34
Figure 19. The architecture of the CERTH ML detector	44
Figure 20. Exchange of information from detectors to the XL-SIEM agent using rsyslog.....	45
Figure 21. Status of a correctly configured rsyslog client based on TLS	47
Figure 22. Proof that the message was received by the client	47
Figure 23. SDN-microSENSE components using the output of the XL-SIEM Agent and Engine through RabbitMQ.....	48
Figure 24. Exchange queues at the RabbitMQ attached to the XL-SIEM.....	49
Figure 25. Consumer queues configured at the RabbitMQ server	50
Figure 26. Steps to create new correlation rules.....	53
Figure 27. Example of event definition for one of the honeypots	54
Figure 28. Example of an event classification for Modbus related events	54
Figure 29. Example of directive (correlation rule) for a Modbus related incident.....	55
Figure 30. Configuration panel where new directives are added to the correlation engine.....	56
Figure 31. XL-SIEM testbed in SDN-microSENSE	59
Figure 32. Evaluation strategy in SDN-microSENSE (extracted from T2.4)	60

Table of tables

Table 1. Connectivity dependencies between XL-SIEM containers.....	19
Table 2. Communication technologies comparison	23
Table 3. Description of the Function types, the Function names and the corresponding Function codes for the Modbus protocol [Knapp+14]	25
Table 4. Description of the attack tools and their corresponding threats for the Modbus protocol....	27

Table 5. Description of the attack tools and their corresponding threats for the DNP3 protocol.....	29
Table 6. Description of the attack tools and their corresponding threats for the IEC-104 protocol.....	30
Table 7. Description of the attack tools and their corresponding threats for the IEC 61850 protocol.	33
Table 8. Description of the attack tools and their corresponding threats for the IEEE C37.118-2 protocol	34
Table 9. Description of the attack tools and their corresponding threats for the MQTT protocol.	36
Table 10. Description of the attack tools and their corresponding threats for the NTP protocol.	37
Table 11. Log taxonomy for the IEC61850 honeypot	37
Table 12. Types of events for the IEC60870-5-104 honeypot.....	38
Table 13. Template to compile log taxonomies from detectors	52
Table 14. Fields of the security alert message exported by the XL-SIEM	57

Acronyms

Acronym	Explanation
APCI	Application Protocol Control Information
APDU	Application Protocol Data Unit
ARIEC	Cloud/based Anonymous Repository of Incidents
ASDU	Application Service Data Unit
CAPEC	Common Attack Pattern Enumeration and Classification
CF	Control Fields
CoT	Cause of Transmission
DA	Delay Attack
DNP	Distributed Network Protocol
DoS	Denial of Service
DRPC	Distributed Remote Procedure Call
ENISA	European Union Agency for Cybersecurity
EPES	Electrical Power and Energy System
FR	Functional Requirement
GOOSE	Generic Object-Oriented Substation Events
GR	General Requirement
IDPS	Intrusion Detection and Prevention Systems
IDS	Intrusion Detection System
IED	Intelligent Electronic Devices
IOA	Information Object Address
IoT	Internet of Things
IPS	Intrusion Prevention System
JSON	JavaScript Object Notation
KoD	Kiss of Death
L-ADS	Live Anomaly Detection Reasoner
M2M	Machine-to-machine

MIA	Message Integrity Attack
MISP	Malware Information Sharing Platform
MiTM	Man in the Middle
ML	Machine Learning
MMS	Manufacturing Messaging Specification
MQTT	Message Queuing Telemetry Transport
NTP	Network Time Protocol
OASIS	Organization for the Advancement of Structured Information Standards
PDA	Packet Drop Attack
PLC	Programmable Logic Controllers
RAF	Risk Assessment Framework
SCADA	Supervisory Control and Data acquisition
SDN	Software Defined Network
SMV	Sampled Measured Values
TLS	Transport Layer Security
UC	User Case
UR	User Requirement
VM	Virtual Machine
XL-EPDS	Cross-layer Energy Protection and Detection System
XL-SIEM	Cross Layer Security Information and Event Manager

Executive Summary

This deliverable is the first document of Work Package 5 (WP5) of SDN-microSENSE. WP5 focuses on cybersecurity protection in the EPES domain, including components for the monitoring of EPES infrastructures, detection of cyber incidents associated to application protocols commonly used EPES networks (T5.1), the development of components for detecting cyber incidents linked to these protocols (T5.2, T5.3, T5.4), and threat intelligence sharing capabilities within EPES (T5.4).

The content of this deliverable is focused on the description of the incident detection component included as part of the XL-EDPS module of the SDN-microSENSE architecture, which is based on the ATOS XL-SIEM. Within the XL-EDPS module, the ATOS XL-SIEM acts as a broker that links detectors deployed in the EPES infrastructure to monitor (IPS developed in T5.1, machine learning-based incident detectors developed in T5.2, access control monitoring developed in T5.4 and honeypots used in WP3), and the consumers of the XL-EDPS output, which is used by several components of the SDN-microSENSE architecture, such as S-RAF (developed in WP2), Honeypot Manager (developed in WP3) and threat intelligence sharing component like the ARIEC (developed in T5.4).

More specifically, the XL-SIEM has been enhanced to support the processing of events coming from detectors developed in T5.2, T5.2 and T5.4, supporting events associated to protocols commonly used by EPES, such as Modbus, DNP3, IEC 60870-5-104, IEC61850 and IEEE C37.118. These protocols and the detectors capable of detecting incidents associated to these protocols, are introduced in this document, with the intention to provide with a complete overview of the new application context introduced in the XL-SIEM. Those protocols and the corresponding details on detectors are thoroughly described in T5.2 and T5.3. Introducing support for these protocols to the XL-SIEM entailed integration efforts on the event processing side, which required the development of connectors capable of interpreting the new set of logs received from the new set of detectors developed in SDN-microSENSE. The correct interpretation of those logs and the information contained by them were key to add new intelligence capabilities to the XL-SIEM by means of new correlation rules that correctly interpret anomalous patterns associated to the logs received.

On the other side, attached to the XL-SIEM and as part of the XL-EDPS, a set of interfaces, based on messages queues, has been integrated to export both events received by the XL-SIEM and the security alerts produced by it. This information is consumed by several components of the SDN-microSENSE architecture, which triggers additional activities such as the automatic deployment of honeypots when zero day attacks are detected (WP3), the sharing of threat information through the ARIEC (T5.4) or the risk evaluation of incidents carried out by the S-RAF component (WP4).

1 Introduction

This document details the technical details of the developments carried out in incident detection component that is part of the XL-EPDS module in the SDN-microSENSE architecture. The main purpose of this component, which is based on the ATOS XL-SIEM, is to act as broker between the security detectors that are monitoring the EPES infrastructure and the components that consume information about security alerts detected by the incident detector. To this end, the XL-SIEM has been adapted to process events coming from detectors related to EPES specific protocols, including also the intelligence to interpret such information, correlate those events to produce the corresponding security alerts. A set of connectors have also been developed at the input side of the XL-SIEM, to interpret logs coming from new detectors, and output connectors to push verdicts about incidents detected that are used by several components of the SDN-microSENSE platform.

This document is structured as follows:

- Section 2 provides an overview of the XL-EPDS module of the SDN-microSENSE architecture and position it in the context of the rest of components of the platform.
- Section 3 provides with details related to the incident detection component used in the XL-EPDS module, together with deployment considerations in the context of SDN-microSENSE.
- Section 4 introduces the monitoring and detectors used in the XL-EPDS, including a high-level overview of the protocols and approaches to monitor them.
- Section 5 provides details about the inputs of the XL-EPDS component and the interfaces deployed to retrieve logs from detectors.
- Section 6 provides details about the output of the XL-EPDS component and the interfaces deployed to push security alerts to alerts consumers.
- Section 7 details the testbed deployed to check connectivity between XL-EPDS components and summarizes the unit testing carried out.
- Section 8 concludes the document and summarises the main innovations carried out in this task.

1.1 Relation to other Tasks and Deliverables

This document is related to the following deliverables (Figure 1):

- D2.2 [SDN22], where the requirements of the SDN-microSENSE platform are elicited
- D2.3 [SDN23], where the SDN-microSENSE architecture is described
- D2.4 [SDN24], from where it is taken the validation methodology
- D3.3 [SDN33], that describes the honeypots that will interact with the XL-EPDS
- D5.2 [SDN52], that describes the protocols, attacks and detectors and details the SDN based IDS
- D5.3 [SDN53], that describes the machine learning based detectors
- D5.4 [SDN54], that describes the privacy framework that interconnects to the XL-EPDS
- D5.5 [SDN55], that describes the ARIEC which uses the output of the XL-EPDS

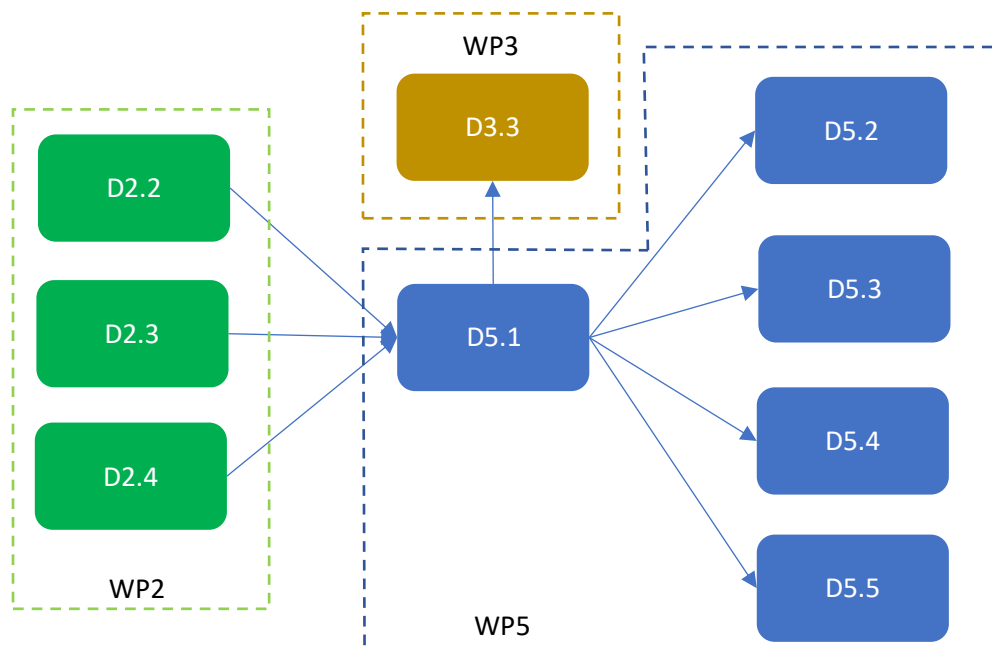


Figure 1. Links between D5.1 and the rest of deliverables and WPs

1.2 Requirements analysis

The following requirements elicited from D2.2 [SDN22] are covered by the components described in this document.

Functional Requirements General Requirements
FR-GR-03, related to the generation of alarms related to incidents. FR-GR-05, related to the ability to provide network flow metrics from network data FR-GR-09, related to the availability of interfaces for integrating external components such as detectors. FR-GR-10, related to the storage of events and incident information in a persistent database FR-GR-11, related to the availability of GUIs to visualize incident information FR-GR-12, related to the collection of security events
Functional Requirements User Requirements
FR-UR-03 to 14, which describe requirements related to the detection of different types of cyber attacks FR-UR-17, which describes que remote notification of incidents detected
Functional Requirements Use Case Requirements
FR-UC1-01 to 03, which cover cyber-attacks to SCADAs logical interface under the Use Case 1 FR-UC1-04 to 07, which cover cyber-attacks to the Station Bus network under the Use Case 1 FR-UC1-08 to 11, which cover cyber-attacks against the process control bus FR-UC3-01, which cover the defence against coordinated attacks scenarios in the Islanding Use Case 3.
Non-Functional requirements
All non-functional requirements refined in Table 12 of D2.2 are covered by this deliverable

2 XL-EPDS overview

The XL-EPDS subsystem of the SDN-microSENSE platform carries out the tasks related to infrastructure monitoring, incident detection and incident reporting. As shown in Figure 2, the XL-EPDS is located at the Application Plane. It directly interacts with the data and infrastructure plane, obtaining monitoring data (logs, network traffic, etc) from the components and network of this plane. It also interacts with the controller plane to obtain data from SDN controllers and infer incidents and anomalies based on the data obtained from this plane. Other elements of the Application Plane (i.e., S-RAF), use information from the XL-EPDS (such as security alerts). Further details of the XL-EPDS within the SDN-microSENSE architecture can be checked in Deliverable D2.3.

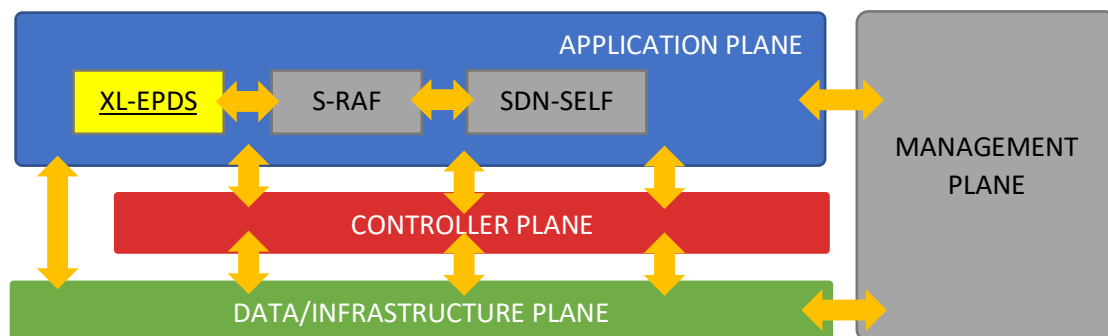


Figure 2. The XL-EPDS within the high level SDN-microSENSE architecture

The internals of the XL-EPDS consists on several components that cover the full cycle of the security management: monitoring, analysis, reporting, mitigating. The details of the XL-EPDS components, and their interrelation, are depicted in Figure 3 in the context of WP5 components and WP5 tasks.

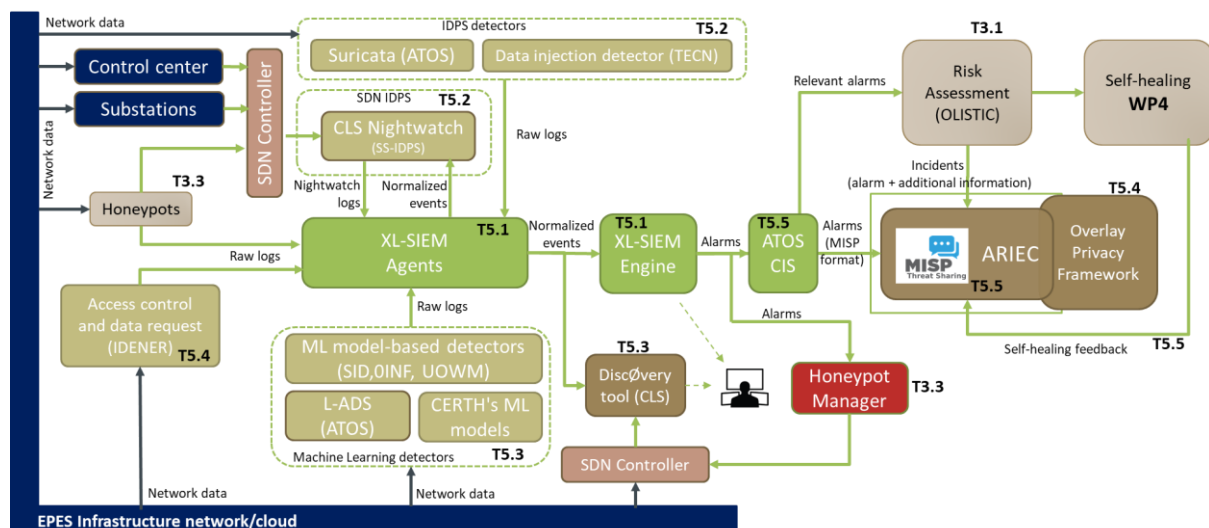


Figure 3. XL-EPDS global architecture

Four main parts can be identified in such figure:

- EPES Infrastructure, which corresponds to the Data/Infrastructure plane and Controller plane of the SDN-microSENSE architecture. Control centres, substations and SDN controllers are part of this group.
- Detectors, which correspond to all those components that provide with logs, evidences, and information relevant to be used to infer security alerts. We include here IDPS detectors and SDN IDPS (part of T5.2), machine learning based detectors (part of T5.3), honeypots (part of T3.3), and components for reporting about access control and data requests (part of T5.4).
- The reasoning engine of the XL-EPDS are the XL-SIEM components (part of T5.1) which are composed of XL-SIEM agents and the XL-SIEM engine.
- Consumers of security alerts, including here:
 - Risk assessment (by the OLISTIC component at the S-RAF), part of T3.1
 - Intelligent sharing components, such as the ATOS CIS and the ARIEC, part of T5.5
 - Visualization components, such as the Discovery tool developed in T5.3
 - Honeypot manager for the dynamic deployment of honeypots based on incidents detected and being developed in T3.3.

The following sections describes the details of the T5.1 components.

The core of the XL-EPDS subsystem is the ATOS XL-SIEM (highlighted in Figure 4), which interfaces with detectors, which provide input data, and three main parts of the SDN-microSENSE as consumers of the XL-SIEM output: OLISTIC component, ARIEC and the Discovery System.

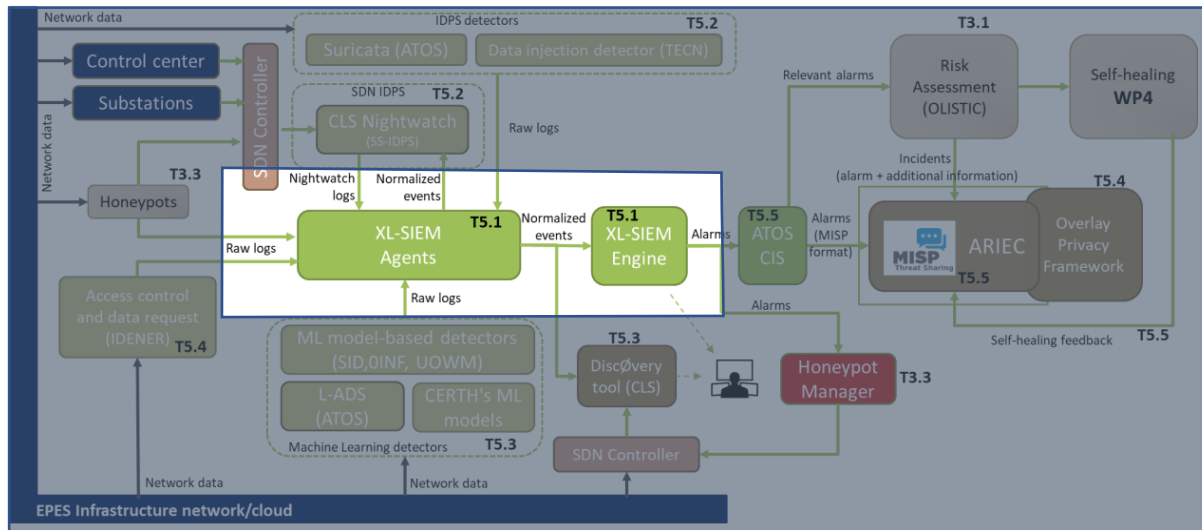




Figure 5. Gartner magic quadrant for SIEMs (taken [Gartner2020]) positioning the XL-SIEM (in orange)

To this end, for its adaptation to the SDN-microSENSE environment, it has been required an effort of adaptation to new communication protocols, which are typically not considered in the current SIEM solutions. The innovations carried out to be able to use the XL-SIEM within SDN-microSENSE are detailed in Section 8. Atos is the developer and owner of the XL-SIEM which has been used in T5.1 and is the umbrella that interconnects the rest of the components of WP5.

The following subsection details the internals of the XL-SIEM.

3.1 XL-SIEM Deployment Considerations

The XL-SIEM is composed of two main parts: one or more agents and one correlation engine.

XL-SIEM agents are light software components that are deployed in any part of the infrastructure. These agents will receive, from other components (sensors, detectors, or any other component deployed in any infrastructure) logs, events or any other information that can be used to infer an anomaly or a security incident. XL-SIEM agents aggregate, filter and normalize the logs received Figure 6.

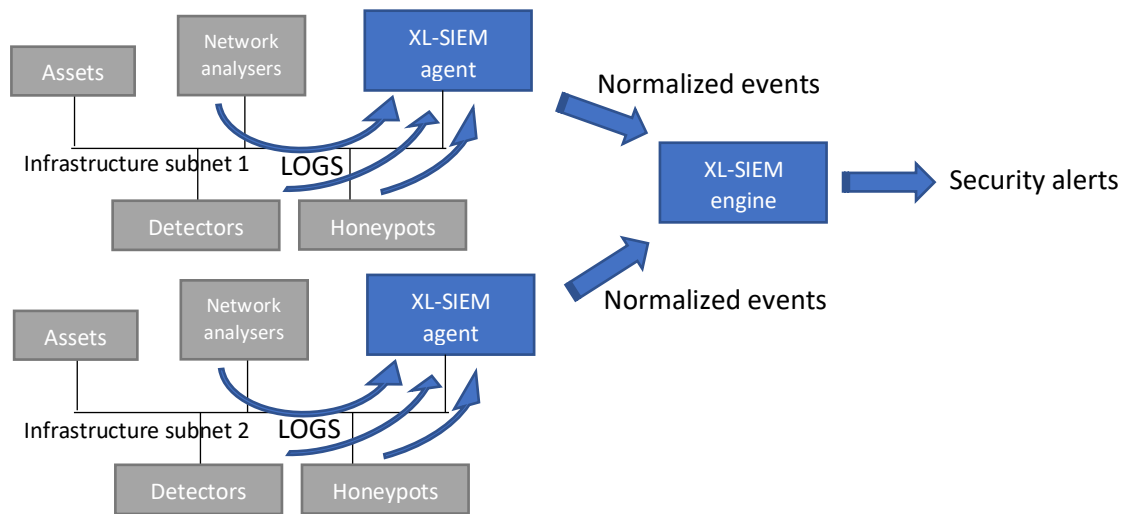


Figure 6. XL-SIEM overview

The mechanisms used to provide inputs to the XL-SIEM agents and to provide normalized events as output will be described in Section 4.2.8.

The other part of the XL-SIEM is the engine, which can be considered the core of the XL-SIEM. The internal representation of the XL-SIEM engine is represented in Figure 7. The XL-SIEM engine is based on two main technologies: Apache zookeeper and Apache Storm. The former is used to coordinate and control the execution of the rest of the components of the XL-SIEM engine. The latter is used to run the Storm topology that supports all the activities carried out by the XL-SIEM. As part of the Apache Storm used at the XL-SIEM engine there are three main subcomponents: workers are the actual elements that perform specific tasks, such as interacting with databases or applying correlation rules. Those workers are instantiated by supervisors, which can be deployed even at different machines and thus enabling a distributed deployment. This is one of the main advantages of using Apache Storm, because, depending on the requirements and resources available, it is possible to deploy supervisors distributed in different machines, instantiated or withdrawing supervisors dynamically as long as they are required. On top of these supervisors, there is an additional node, called Nimbus, which controls the tasks assigned to supervisors, controlling the deployment of supervisor nodes and the correct schedule and cooperation across them.

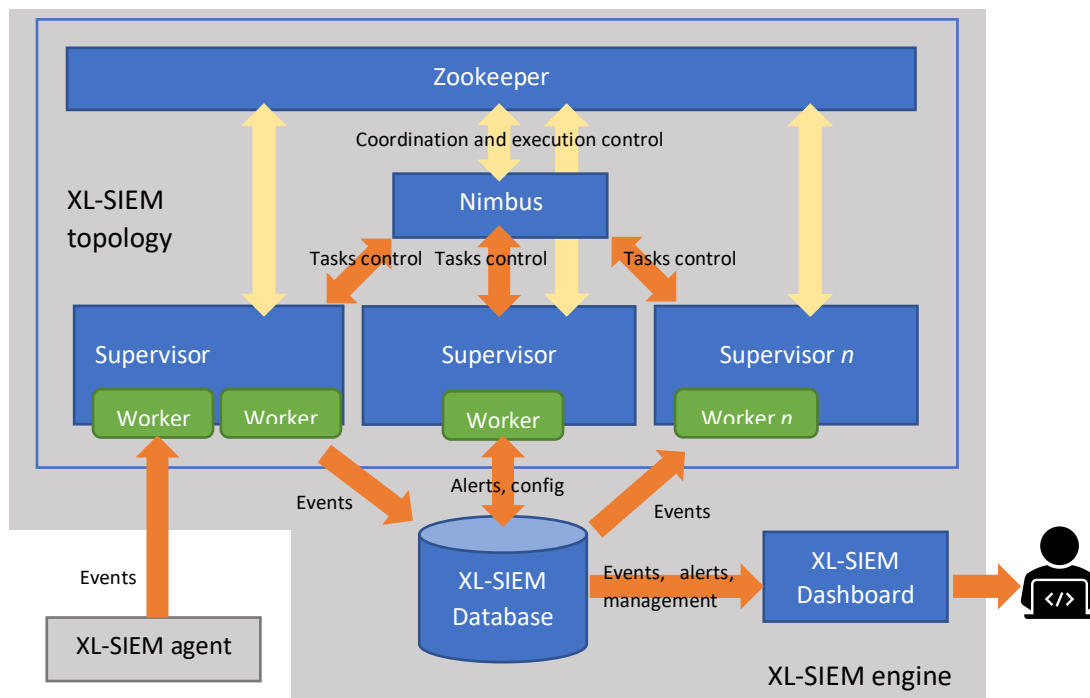


Figure 7. Internals of the XL-SIEM engine

The XL-SIEM has evolved as part of the activities carried out in SDN-microSENSE. The main improvement leverages the adaptability to the characteristics of the infrastructure where it is deployed. While in the past the XL-SIEM was built on a monolithic distribution to be instantiated on top of a Linux based operative system running on a dedicated machine, the latest improvements use Docker based containers to separate the different subcomponents of the XL-SIEM. This provides several advantages:

- Optimization of resources required to run the XL-SIEM Engine. The resources required to execute the XL-SIEM Engine has dropped significantly (up to half in terms of CPU and RAM) with the migration to Docker containers.
- Easy distributed deployment, with the possibility to deploy Docker containers for Nimbus, Supervisor and Zookeeper in different machines.
- Scalability, being able to easily deploy additional supervisors or withdraw them when they are not required.
- Easy adaptation to new requirements which allows to easily tailor the characteristics of the XL-SIEM Engine to specific requirements of the infrastructure where it is deployed.

Figure 8 shows the containers deployed for the installation of an XL-SIEM agent, representing the following containers:

- Storm container, which executes the Storm topology that supports the XL-SIEM engine. There are four containers in this group:
 - Supervisor_in, the container that executes the Storm supervisor, which manages the workers that execute specific tasks. This container uses a dedicated socket port, which is used to receive events from the XL-SIEM agents.

- Storm_ui (optional), a supporting container used for managing the processes of the topology, check performance, etc. A dedicated TCP port is used for this purpose.
- Nimbus is the node of the Apache Storm topology that control que activities carried out by supervisors (in this case, just one supervisor). This is done through a dedicated TCP port.
- Drpc, a container that manages the communication between the storm nodes, using a dedicated TCP port.
- Zookeeper container, which controls the status of the nodes of the Storm topology, using several ports for this purpose.
- A web container (xlsiemi_web_SDN-microSENSE) which contains the XL-SIEM dashboard for managing it and visualizing events and alerts, using the port 80 (8080 for HTTPS)
- A RabbitMQ container (optional) which can be used to support communications to and from the XL-SIEM with other components (for example, to export alerts). Details about this container will be given in Section 4.2.8
- A database container, used by the storm nodes to store events, alerts, configuration, etc, based on MariaDB, typically using the MySQL TCP port.

COMMAND	PORTS	NAMES
"/docker-entrypoint.sh"	0.0.0.0: -> /tcp	supervisor_in
"/docker-entrypoint.sh"	0.0.0.0: -> /tcp	storm_ui
"/docker-entrypoint.sh"	0.0.0.0: -> /tcp	nimbus
"/docker-entrypoint.sh"	0.0.0.0: -> /tcp	drpc
"/docker-entrypoint.sh"	/tcp, /tcp, /tcp, /tcp	zookeeper
"sh -c /entrypoint.sh"	/tcp, 0.0.0.0: -> /tcp	xlsiemi_web_sdnmicrosense
"/scripts/entrypoint.sh"	0.0.0.0: -> /tcp, /tcp, /tcp, 0.0.0.0: -> /tcp	xlsiemi-rabbitmq
"docker-entrypoint.sh"	0.0.0.0: -> /tcp	xlsiemi.db_sdnmicrosense

Figure 8. Docker containers deployed for an XL-SIEM installation¹

It is worth noticing that all ports aforementioned are just used internally between docker containers, not being visible outside of the machine that hosts these containers. The exception is the HTTPS port, which is visible for the access to the web dashboard.

Dockerizing the main components of the XL-SIEM provide with huge advantages in term of deployment alternatives. The supervisor_in and the database containers are the ones that demand more resources, the former in terms of computation and the latter in terms of storage and network connectivity (i.e., max number of concurrent connections to the database). Additionally, not all containers require connectivity with all containers. These dependencies, shown in Table 1, allows several deployment alternatives that can be adapted to the characteristics of the infrastructure to monitor.

Table 1. Connectivity dependencies between XL-SIEM containers

	Zookeeper	Nimbus	Supervisor	Drpc	Storm_ui	Database	Dashboard
Zookeeper		YES	YES	NO	NO	NO	NO
Nimbus	YES		YES	YES	YES	NO	NO
Supervisor	YES	YES		YES	YES	YES	NO
Drpc	NO	YES	YES		YES	NO	NO
Storm_ui	NO	YES	YES	YES		NO	NO
Database	NO	NO	YES	NO	NO		YES

¹ Ports hidden for security reasons

Dashboard	NO	NO	NO	NO	NO	YES	
-----------	----	----	----	----	----	-----	--

For example, there can be four main areas of deployment:

Area 1: Dashboard. This is the less critical container in terms of connectivity with other containers. The dashboard will interact just with the Database, which is used by the system administrator to visualize events, alerts or to configure the XL-SIEM. This container needs to be accessible with a Web Browser, which requires to have a TCP port open (which is mapped to the HTTPS port of the container). HTTPS connections are used to interact with the dashboard.

Area 2: Database. This container needs to be visible both to the Dashboard and to the Supervisor (or supervisors if there are more than one). The connection to the dashboard is required because it is its source of information for events, alerts and configuration data. With the supervisor the connectivity is required because it is used by its workers to store events and alerts, to read correlation rules and the configurations to use at the Storm topology. As it contains critical data, it should be deployed in a secure location, ensuring that just the Dashboard and Supervisor containers can access to it.

Area 3: Storm containers (Nimbus, Supervisor, Drpc, Storm_ui). These are the core of the topology, especially the Nimbus, the Supervisor and the Drpc. They are basically considered as processing containers. They don't store persistence information nor need to be exposed to external consumers like the Dashboard does (except when alerts are exported to RabbitMQ queues as will be described later in this document). However, it has higher computational requirements, especially in terms of CPU and RAM (depending on the deployment used it can require a minimum of 4 CPU cores and 6GB RAM). The container Storm_ui is optional and not really required for the Storm topology to work correctly, although it provides with important management information (for example, to check the number of workers running at any moment or the amount of RAM memory used).

Area 4: Zookeeper. This component requires connectivity with the Storm containers basically to check whether they are working properly. This node doesn't store or use information from the database. It also doesn't have major computational resources.

Figure 9 summarizes the requirements required for the container included in each area in terms of security, connectivity with external components, computational resources and network resources.

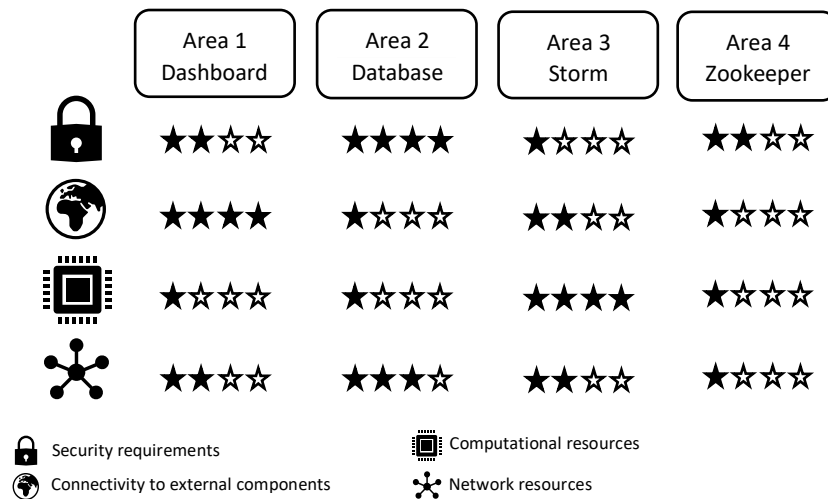


Figure 9. Comparison of requirements for deploying XL-SIEM containers

A different combination of the XL-SIEM containers can be set-up, which depends on the specific needs of the infrastructure where it is to be deployed, such as in the EPES domain considered in SDN-microSENSE.

3.2 Deployment considerations in SDN-microSENSE

The XL-SIEM solution is broken down in several components which are interconnected employing different technologies (Figure 10). These technologies have been chosen having into account a different aspect of the communication of information flows where asynchronous messages play a main role. The following figure is a simplified diagram where only communication technologies involved by the XL-SIEM are depicted.

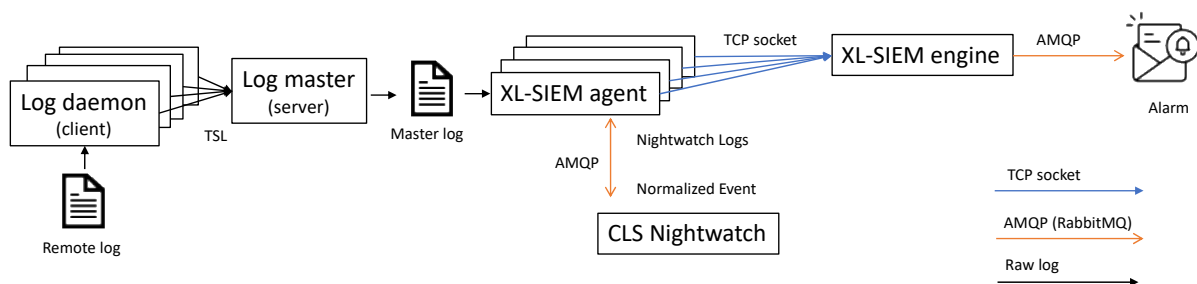


Figure 10. XL-SIEM infrastructure resources

The XL-SIEM solution requires the deployment of XL-SIEM agents into the EPES infrastructure, as well as log daemon. In this deployment, the following issues must be considered:

- Log daemon installation.** To get remote logs is needed running a daemon in the EPES for IDPS detectors, Honeypots, Access control and data request and Machine learning detector components. This daemon acts like a file watcher which detects a log updates and sends it to the Log master which gathers all of those updates in a master log. For each one of these

systems, a daemon has to be installed and configured indicating the log master server address and the local folders to watch out.

- **One XL-SIEM agent for each subnetwork.** Every subnetwork has to contain an instance of the XL-SIEM agent to be monitored. Depending on the type of communication a specific agent has to be deployed and configured.
- **XL-SIEM engine port access.** Due to the communication between the XL-SIEM agent and the XL-SIEM engine it is needed to establish communication connectivity between them.

To deal with a secure deployment environment, the XL-SIEM component must compel not only the security of each component in an isolated way, but also how to achieve secure communications, giving emphasis on the DMZ and not on DMZ environments. The communication security ranges from:

- **Communication path constrains.** The communication path should be constrained to guarantee source and target match with suitable IP address and service port. This is a common rule that could be easily configured of the mean of Firewalls and even the SDN switches themselves. For this, it is recommended to establish the host's IP where the XL-SIEM agents will run. Although, a major control could be by the MAC usage rather than the IP address XL-SIEM agent. For this, it is required to obtain the flows belonging to SDN switches and firewalls allow the message bypass between IP/port sources and IP/port targets.
- **SDN network reinforcement.** The path from the information sources services up to each of intermediate component, such as daemon logs, master log, XL-SIEM agent, XL-SIEM engine, AMQP broker, has to count on at least two paths by the mean of increase the number of SDN switches in order to the SDN self-healing algorithm is able to find out different alternatives when needed.
- **Warranty band width.** The SDN network has to be configured to warranty a continuous information flow of messages. Fortunately, this is straightforward by the mean of the own SDN switches.
- **Secure protocols.** The communication of logs, events and alarms have to use secure protocols and certifications.
- **Operation network avoiding.** Most security recommendations in the industrial network, very similar to ICT networks in Electric grids, suggest minimizing the impact of any solution in the operational network. Having this in mind, the log information should not go through the same network where spring protocols travel, in favour of informational networks devoted to production and business purposes.

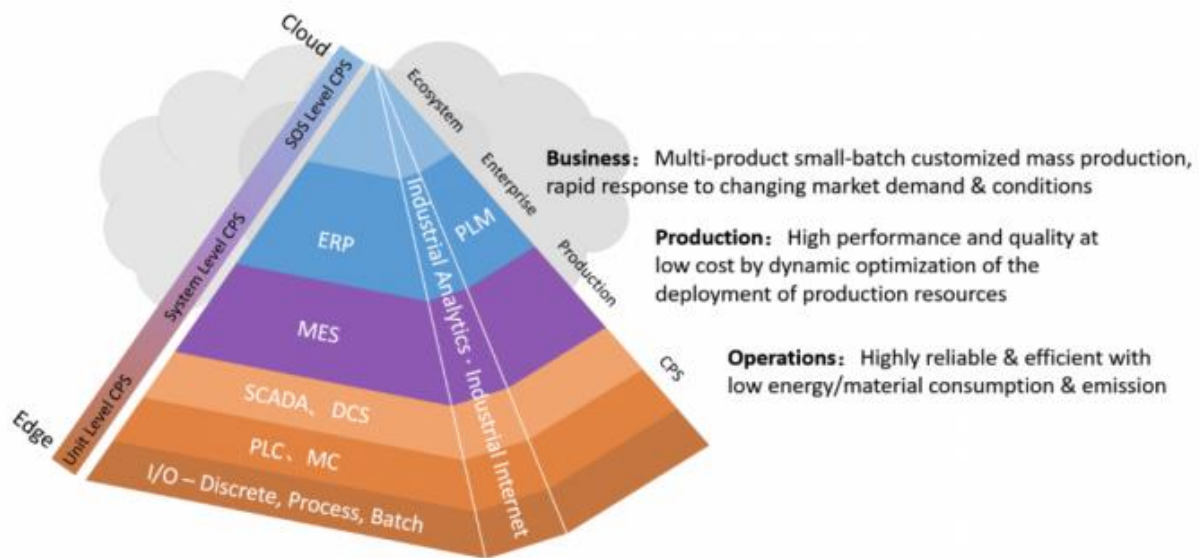


Figure 11. ISA 95/98 standard - Industrial networks

Regarding the communication implementation, the technology employed for the event notification is planned to be by sockets whereas the alarm communication will be by AMQP (RabbitMQ). A technology comparison is given in Table 2. Both technologies are more suitable for the type of communications present in XL-SIEM. In fact, these communications comply with the following properties:

- **Asynchronous paradigm.** Both XL-SIEM agents and the XL-SIEM engine do not need any response after an event or alert notifications have been sent.
- **Streaming flow.** An enormous amount of event and alerts are supposed to be continuously sent. While the connection is alive, multiple messages can be sent.
- **Simple message format.** Notification messages are very simple.

Table 2. Communication technologies comparison

	SOAP	REST	AMQP	TCP Socket
Reliable	Yes	No	Yes	Yes
Message format	XML	JSON, XML, ...	Free	Free
Payload	Based on WSDL schema	Free	Free	Free
Connected mode	Yes	Yes	No (*)	Yes
Communication	Synchronous / Asynchronous	Synchronous	Asynchronous	Synchronous / Asynchronous
Paradigms	RPC / Asynchronous	RPC	Publish-subscribe Queuing RPC	Full duplex
Message Order	No guaranteed	No guaranteed	Guaranteed	Guaranteed
Streaming	No	No	Yes	Yes
Known Client / Server address	Yes	Yes	No	Yes

(*) the publisher establishes a real point to point connection with the AMQP broker regardless the subscriber availability.

To understand the convenience of streaming flow it is relevant to describe its differences from SOAP/REST services communications (Figure 12). The main difference is focussed on the connection time spent by each message sending. In both SOAP and REST cases, it is needed to make a TCP connection to send a single message. As it is known, the connection time is not that negligible. In contrast, in a streaming communication the connection is established only at the beginning, after that you can keep a continuous flow of messages, even in full-duplex way.

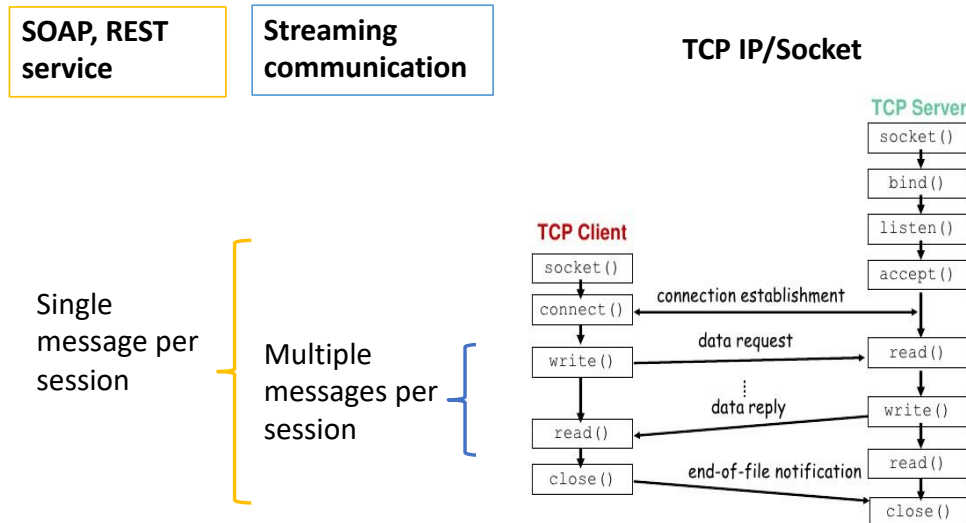


Figure 12. SOAP/REST vs Streaming flow

The AMQP and TCP Socket supply the most features needed for XL-SIEM communications. TCP Socket is the alternative chosen for communication between XL-SIEM agents and the XL-SIEM Engine mainly because the TCP Socket software library is simpler than the AMQP library, which is adequate when the source and target work closely. However, the communication between CLS Nightwatch and the XL-SIEM agent uses AMQP (RabbitMQ) because to the broker usage allows more flexibility of future clients.

4 Monitoring and Incident Detection in SDN-microSENSE

4.1 SDN-microSENSE protocols and threats

The support of incidents associated to EPES specific protocols is one of the main innovations incorporated in the XL-SIEM when integrated in the XL-EPDS component of the SDN-microSENSE architecture. Supporting these protocols opens up the XL-SIEM to a myriad of new possibilities in terms of incident detection, new domains of application and new business opportunities. The following subsections give an initial overview of these protocols, which will be deeply detailed in the rest of WP5 tasks where attack detectors are developed (T5.2, T5.3 and T5.4).

4.1.1 Modbus

Modbus protocol was designed by Modicon in 1979 and since then constitutes the most widely deployed industrial communication protocol. It operates at layer 7 of the Open Systems Interconnection (OSI) model that means it is an application layer messaging protocol. The Modbus protocol is based on the communication of client/server between devices connected on different types of networks. The client sends request to the server and according to the request, the server perform an action and send

a replay, 0. The reserved port of the Modbus protocol is 502. The operation of the Modbus protocol is depicted in the Figure 13.

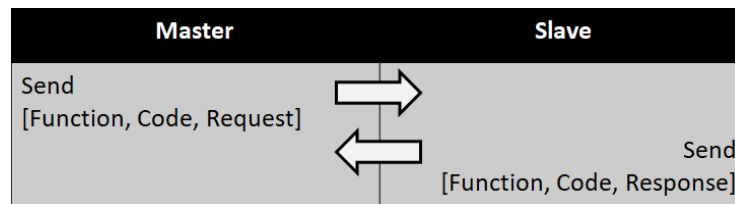


Figure 13. Description of the Modbus protocol operation

The Modbus slave provides to the Modbus master the following objects; the discrete input that constitutes a type of read only Boolean variable, the coil that constitutes a type of read-write Boolean variable, the input register that constitutes a type of read only integer and the holding register that constitutes the type of read and write integer. Table 3 describes all the function types, the function names and the function codes of the Modbus protocols. The most commonly used functions codes of the Modbus protocol are the following. The function code 1 named read coil, the function code 2: read discrete input, the function code 3: read holding registers, the function code 4: read input registers, function code 5: write single coil, function code 6: write single holding register, function code 15: write multiple coils and function code 16: write multiple holding registers. The Modbus protocol is suitable to communicate Programmable Logic Controllers (PLCs) and the remote terminal unit RTUs with a Supervisory Control and Data Acquisition (SCADA) system.

Table 3. Description of the Function types, the Function names and the corresponding Function codes for the Modbus protocol [Knapp+14]

Function Type			Function Name	Function code
Data access	Bit access	Physical Discrete inputs	Read discrete inputs	2
		Physical coils	Read coils	1
			Write single coil	5
			Write multiple coils	15
	16-Bit access	Physical Input registers	Read input register	4
		Physical output registers	Read multiple holding registers	3
			Write single holding register	6
			Write multiple holding registers	16
			Read/write multiple registers	23
			Mask write registers	22
			Read FIFO Queue	24
			File record access	Read file record
	Write file record	21		
Diagnostics		Read exception status	7	
		Diagnostic	8	
		Get com event counter	11	
		Get com event log	12	
		Report slave ID	17	

	Read device identification	43
Other	Encapsulate interface transport	43

According to Drias et al. 0, the most cited attacks of the Modbus TCP protocols affect the mechanisms of the protocol in terms of interception, modification and generation. They assume that the main attacks associated with the Modbus protocol are the following; the Broadcast message spoofing, the Baseline response replay, the Direct slave control, the Modbus network scanning, the Passive reconnaissance, the Response delay and the Man in the Middle. More specifically, the Broadcast message spoofing is the type of attack that sends faked broadcast messages to Modbus server in the device. The Baseline response replay involves recording Modbus messages between a Modbus client and a Modbus server. The Direct Slave Control attack aims to lock out a master (client) and controlling one or more field devices. The Modbus Network Scanning is the type of attack that sends benign messages to all possible addresses on a Modbus network to obtain information about field devices. The Passive Reconnaissance involves passively reading Modbus messages or network traffic. This attack helps the attacks to profile the process details by reconstructing the device application. The Response Delay attack involves delaying response messages so that the master receives out-of-date information from slave devices. This attack intends to sabotage the supervision in case that the command in the Modbus message is a diagnostic message. The Man in the Middle attack (MITM) involves introducing a computer with the appropriate (serial or Ethernet) adaptors to an unprotected communication link. The MITM device is embedding a client and Modbus server; therefore, it can read, modify and fabricate Modbus messages to or from the device. This attack is the most dangerous attack on Modbus protocols as it can take the full control of both parties of protocol; the client and the server in the same time.

Based on the most commonly cited attacks for the Modbus protocol and assuming the threat classification that is provided by ENISA 0 different attack scenarios will be considered to simulate the abnormal traffic of the Modbus protocol and based on the Smod modular framework². The attack scenarios concern the following procedures: Unauthorized access, Failure of devices and systems, Manipulation of information, Information leakage, Network Reconnaissance, Information Gathering and DoS.

More specifically, the attack scenario WriteSingleCoils aims to change the value of a coil. The WriteSingleRegister is the scenario that change the values of a single holding register and the UID brute force attack describe the type of the Brute force attack against PV/Battery inverters. All of the scenarios that have been described above belong to category that contains attacks related to the Unauthorized Access, the Failure of devices and systems and the Manipulation of information.

Taking into consideration the categorization that related to the unauthorized access and information leakage four attack scenarios are going to develop³. The first scenario named ReadCoils, constitutes the attack scenario that reads the value of a specific coil. The ReadDiscreteInput is the scenario that extract the values of the discrete inputs supported by the target. The ReadHoldingRegister is the type of attack that returns the values of the holding registers supported by the target system and the fourth attack in this category is the ReadInputRegister that reads the values of the input registers.

² Smod modular framework homepage, <https://github.com/Joshua1909/smod>

³ <https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter>

Four type of attack tools will be used for the Network Reconnaissance and Information Gathering is a family of threats. More specific, the Getfunc is the attack scenario that discovers the functions codes supported by the target system. The Uidc constitutes the attack scenario that enumerates the UIDs supported by the target system. The Discover identifies whether the Modbus runs in the target system and the Modbusclient describes the scenario that exploits a Modbus vulnerability that allows to an unauthorized actor to read or write against the targeted Modbus slave.

Finally, two scenarios that associated with the DoS attacks will be developed. The WriteSingleCoils constitutes the type of DoS attack scenario, which send continuously malicious Modbus packets (function code 5) to the target system. In a similar way, the WriteSingleRegister constitute also a DoS attack, which sends continuously malicious Modbus packets (function code 06) to the target system.

Table 4 summarizes the type of scenarios that will be developed and the corresponding classification based on ENISA threat taxonomy and on the CAPEC classification.

Further details about this protocol, attacks and detectors are given in D5.2.

Table 4. Description of the attack tools and their corresponding threats for the Modbus protocol.

Threat	Description of the Threat	ENISA Threat taxonomy	CAPEC classification
Function/ writeSingleCoils	Aims to change the value of a coil	Unauthorised Access, Failure of devices and systems, Manipulation of information	180
Function/ writeSingleRegister	Changes the values of a single holding register	Unauthorised Access, Failure of devices and systems, Manipulation of information	180
UID brute force attack against PV/Battery inverters' RPI	Unauthorised Access, Failure of devices and systems, Manipulation of information	Unauthorised Access, Failure of devices and systems, Manipulation of information	112
Function/ readCoils	Reads the value of a specific coil.	Unauthorised Access, Information leakage	180
Function/ readDiscreteInput	Extracts the values of the discrete inputs supported by the target.	Unauthorised Access, Information leakage	180
Function/ readHoldingRegister	Reads the values of the input registers	Unauthorised Access, Information leakage	180

Scanner /getfunc	Discovers the functions codes supported by the target system	Network Reconnaissance and Information Gathering	309
Scanner /uidc	Enumerates the UIDs supported by the target system	Network Reconnaissance and Information Gathering	309
Scanner /discover	Identifies if Modbus runs in the target system	Network Reconnaissance and Information Gathering	309
Scanner /Modbusclient	Exploits a Modbus vulnerability that allows to an unauthorized actor to read or write against the Modbus slave targeted	Network Reconnaissance and Information Gathering	309
Modbus /dos /writeSingleCoils:	Sends continuously malicious Modbus packets (function code 5) to the target system	DoS	125
Modbus /dos /writeSingleRegister:	Sends continuously malicious Modbus packets (function code 06) to the target system	DoS	125

4.1.2 DNP3

DNP3 [UOWM1+20] is a reliable protocol applied largely in the Critical Infrastructures (CIs) in the US. In the Electrical Power and Energy Systems (EPES), DNP3 is adopted to transfer messages between master devices and outstations. It supports several topologies, including a) point-to-point, where an outstation and one master communicate with each other, b) multiple-drop, where several masters and outstations interact each other and c) hierarchical interface, where an entity can operate with both roles. DNP3 includes three layers: a) link layer, b) transport layer and c) application layer. The link-layer offers addressing services, multiplexing, data fragmentation, error checking and link control. On the other side, the transport layer is used as in the case of the Open Systems Interconnection (OSI) model, and it is represented with one byte utilised for fragmenting the DNP3 packets. Finally, the application layer defines a set of functional commands used for managing and controlling the EPES entities, such as Remote Terminal Units (RTUs), Programmable Logic Controllers (PLCs), Intelligent Electronic Devices (IEDs) and smart meters. Apart from the DNP3 serial line communication, DNP3 can be used over TCP/IP where in this case, the aforementioned DNP3 layers are incorporated into the application layer of TCP/IP.

According to O. Igbe et al. [UOWM2+20], various cyberattacks target directly the DNP3 protocol with significant consequences (Table 5).

Table 5. Description of the attack tools and their corresponding threats for the DNP3 protocol.

Threat	Description of the Threat
DNP3 Enumerate	It is a reconnaissance attack aiming to identify whether the DNP3 service operates in the target systems.
DNP3 Info	It is another reconnaissance attack which obtains useful diagnostic information about the utilisation of DNP3.
DNP3 Disable Unsolicited Messages Attack	It is unauthorised access attack, which transmits to an outstation a DNP3 packet with the function code 21, thus disabling all unsolicited messages. Therefore, the outstation will not be able to send any alarm message to the master devices
DNP3 Cold Restart Message Attack	It is another DNP3-related unauthorised access attack, which aims to restart an outstation.
DNP3 Warm Restart Message Attack	Similarly, to the previous case, it intends only to restart the DNP3 service in the target outstation

The aforementioned DNP3 cyberattacks are addressed efficiently by the XL-SIEM sensors. Further details about this protocol, attacks and detectors are given in D5.2.

4.1.3 IEC 60870-5-104

IEC-104 [UOWM3] is a communication protocol provided by the IEC 60870-5 standard for monitoring and controlling automated processes in EPES applications by utilizing the transport capabilities offered by TCP/IP. In particular, it utilizes by default the TCP port 2404. Figure 14 illustrates the payload of this protocol which is named Application Protocol Data Unit (APDU).

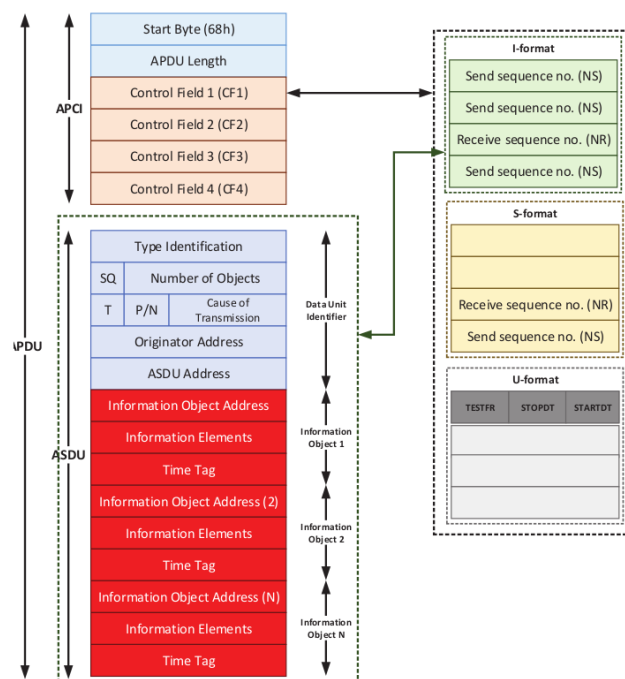


Figure 14. IEC 60870-5-104 Protocol

APDU consists of two parts, namely 1) Application Protocol Control Information (APCI) and 2) Application Service Data Unit (ASDU). APCI includes the start character (68h), the length of APDU and four Control Fields (CFs). On the other side, ASDU is an optional part which is determined by the format of APDU. In particular, APDU can take three formats: 1) I-format, 2) S-format and 3) U-format. The I-format is used to execute numbered information transfers and always includes ASDU. The S-format is used to perform numbered supervisory functions and comprises only APCI. Finally, the U-format is responsible for performing unnumbered control functions and it also includes only APCI. The format of APDU is determined by CF1 and specifically by its two last bits. If the two last bits of CF1 are equal to 00, then the I-format is used. Accordingly, if the last bits of CF1 are equal to 01, then the S-format is applied. Finally, if the aforementioned bits are 11, the U-format is used. Concerning the ASDU, it includes the following fields: 1) Type Identification, 2) Structure Qualifier (SQ), 3) Number of Objects or Elements, 4) T, 5) P/N, 6) Cause of Transmission (CoT), 7) Originator Address (ORG), 8) ASDU Address, or Common Address of ASDU (CoA), 9) Information Object Address (IOA), 10) Information Elements, 11) Time Tag. The Type Identification determines the type of information objects. All information objects of an ASDU must have the same type. SQ specifies how the information objects and elements are structured. The Number of Objects or Elements field denotes the number of information objects or elements depending on the value of SQ. Accordingly, T defines those ASDUs which are dedicated for testing. P/N determines the positive or negative confirmation of an activation command. CoT directs ASDU to specific tasks and simultaneously interprets the data received by the destination side. ORG is an optional field and undertakes to explicitly define the identity of the controlling station (i.e. MTU). CoA defines the address of MTU or RTUs at the application layer. IOA determines the address of an information object. Information Elements provide and transmit specific information and finally, Time Tag provides time information.

The functionality of the IEC-104 protocol relies on the TCP/IP, which itself includes multiple security issues. Moreover, IEC-104 does not include any authentication and authorisation mechanism, thus enabling potential Man-in-The-Middle (MiTM) and unauthorised attacks. In particular, Table 6 describes the attacks can be directly performed against IEC-104.

Table 6. Description of the attack tools and their corresponding threats for the IEC-104 protocol.

Threat	Description of the Threat
M_SP_NA_1_DoS	The specific cyberattack sends continually to the target system M_SP_NA_1 packets
C_SE_NA_1_DoS	This cyberattack floods the target with C_SE_NA_1 packets
C_SC_NA_1_DoS	Similarly, this attack sends continuously to the target system C_SC_NA_1 packets
C_SE_NA_1	This cyberattack constitutes and unauthorised access, transmitting to the target system C_SE_NA_1 packets
C_CI_NA_1	This cyberattack send unauthorised C_CI_NA_1 packets to the target system
C_SC_NA_1	This cyberattack is another unauthorised access attempt related to IEC-104, transmitting C_SC_NA_1 packets to the target
C_CI_NA_1_DoS	This cyberattacks constitutes a DoS related to IEC-104, transmitting continuously C_CI_NA_1 packets to the target system

The aforementioned IEC-104 cyberattacks will be addressed efficiently by the XL-SIEM sensors. Further details about this protocol, attacks and detectors are given in D5.2.

4.1.4 IEC 61850

International Electrotechnical Commission (IEC) 61850 [UOWM1+20] is an abstract international communication standard for EPES systems and particularly for SCADA applications, defining a hierarchical, object-oriented data representation model. In particular, each asset is characterised by a data model composed of naming, diagnostic and configuration information. The purpose of this data model is to facilitate the information exchange among the EPES assets, without referring to their functional and technical details. The IEC 61850 stack consists of four types of messages: a) Manufacturing Messaging Specification (MMS), b) Generic Substation State Events (GSSE), c) Generic Object-Oriented Substation Events (GOOSE) and d) Sampled Measured Values (SMV). IEC 61850 is also used for the control, protection and measurement functions of a substation. These functions are implemented in Intelligent Electronic Devices (IEDs), which can house one or more of these functions. In turn, each of these IEDs must interact with the functions of the IEDs in the rest of the system. Figure 15 shows a typical architecture of a substation where different types of IEDs (SCU⁴ and BCU⁵) interact each other in a communication network.

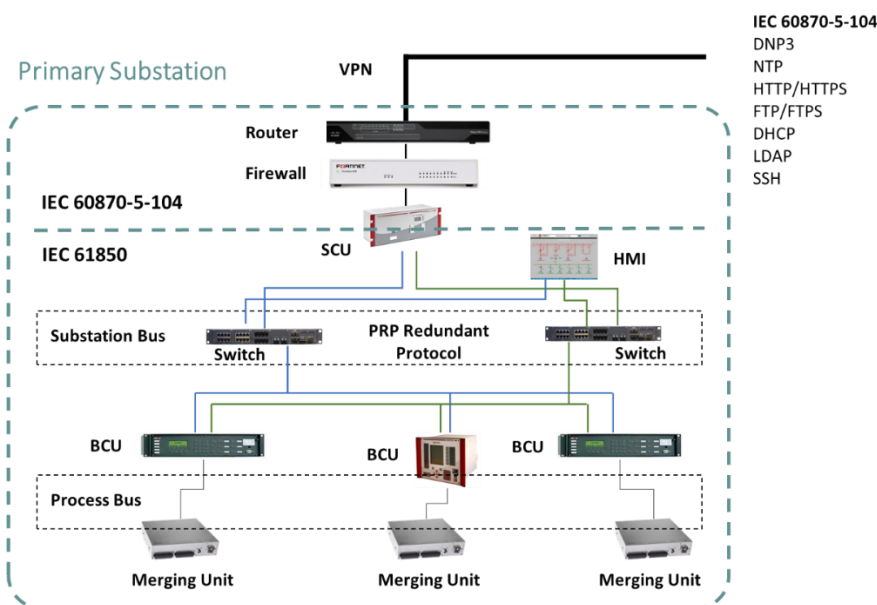


Figure 15. Typical general view of a primary substation

The IEC 61850 standard allows that the control of the substations becomes independent of the manufacturers, being able to interconnect and replace devices belonging to different manufacturers. To achieve this objective, the standard is based on three key principles:

- It defines a unified information model with a hierarchy of names and specific data structures to be used in the different devices.
- It defines a communication protocol and common functionality. This protocol is an agreed language for all equipment in the system. The protocol is designed to be able to send the necessary information to the automated system while maintaining time and availability requirements.

⁴ Substation Control Unit.

⁵ Bay Control Unit.

- It establishes an XML-based configuration file format and a set of formats and tools to facilitate automation and configuration tasks within the engineering process.

Data Modelling

IEC 61850 information model is based on two main levels of modelling:

- The breakdown of a real device (physical device) into logical devices. A logical device represents a group of typical automation, protection or other functions.
- The breakdown of a logical device into logical nodes, data objects and attributes.

The functions executed by the real devices are modelled by the logical nodes. The combination of several of these logical nodes makes up a logical device, and depending on their functionality, logical nodes contain a list of data objects with their corresponding data attributes. Figure 16 gives an example of how each level is included into the upper level. Simple data types (integers, booleans, etc.) are organized into composite data types (quality, scale), and these, in turn, are grouped to form the supported data types CDCs (measurement, controllable data, status information, etc.).

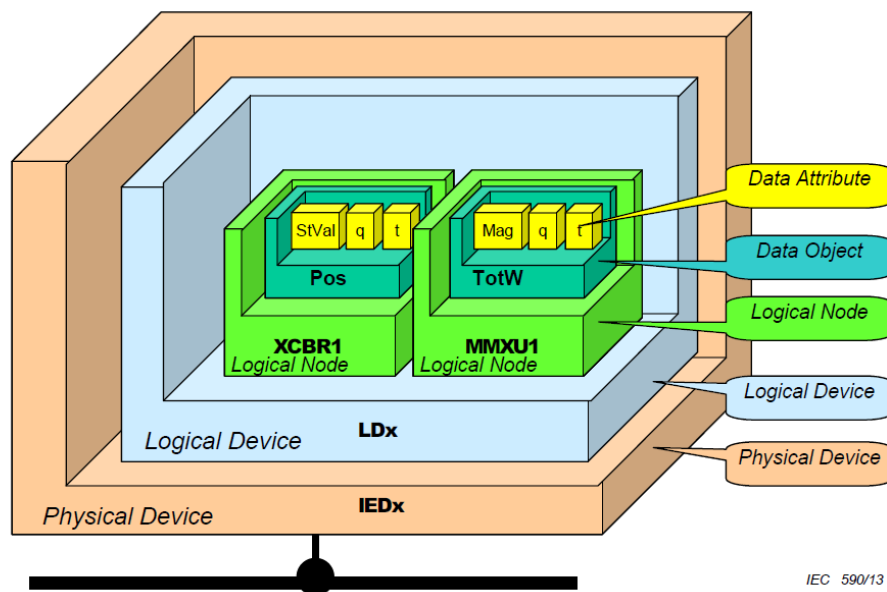


Figure 16. IEC 61850 Data modelling. Source IEC/TR 61850

Communication Protocols

IEC 61850 offers three types of communication models:

- Client/server type communication service model. This model is used for the MMS that sends its messages through TCP connections (Layer 4 OSI). It is used for the exchange of application data, as well as device configuration parameters or monitoring data.
- Sample Values (SMV) model for multicast measurement values. This model is used to provide rapid communication of measurement, protection and control values. It works through Ethernet (Layer 2 OSI) following a publisher-subscriber model.

- Fast and reliable system-wide distribution of data based on a publisher-subscriber model. This model is used for real-time transmission of critical events (GOOSE messages) and like Sampled Measured Values, through multicast Ethernet (Layer 2 OSI).

In the context of the SDN-microSENSE, XL-SIEM focuses on GOOSE and particularly on the attacks listed in Table 7.

Table 7. Description of the attack tools and their corresponding threats for the IEC 61850 protocol.

Threat	Description of the Threat
GOOSE DoS	This refers to a GOOSE-related DoS attack, which floods the target system with GOOSE messages, to block legitimate IEDs from accessing resources
Data Manipulation	This is an unauthorised access attack, which injects malicious GOOSE packets, aiming to impact the grid stability or to cover unauthorized changes
Message Suppression	This attack refers to the hijacking of the GOOSE packets by modifying their header, thus hindering the EPES assets to receive critical GOOSE messages
Disturbance	It refers to electricity-related disturbances and faults that might occur

Further details about this protocol, attacks and detectors are given in D5.2.

4.1.5 IEEE C37.118-2

The IEEE C37.118-2 standard [IEEEC37+11] defines the communication framework for data transmitted between Synchrophasors, which are used for measuring electrical quantities between different parts of the power system. It combines other technologies such as the Global Positioning system (GPS). They also combine PMUs (Phasor Measurement Unit) and PDCs (Phasor Data Concentrator) and additional equipment for visualization and monitoring (Figure 17). At substations, PMUs takes measurements, adding a precise timestamp using a GPS device. These PMUs send their measurements to a PDC towards a control centre.

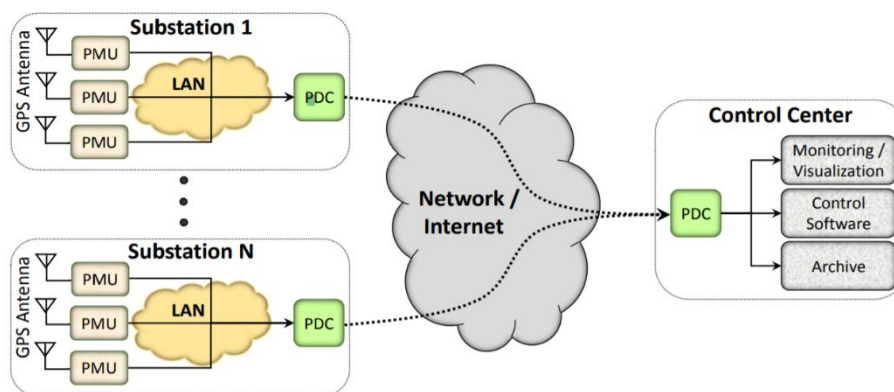


Figure 17. Synchrophasor communication system overview (taken from [Khan+16])

The format of the messages exchanged are specified in the standard, but it doesn't specify the transport protocol, thus missing any security feature that can be implemented at the transport level. Figure 18 depicts the message format for the IEEE C37 protocol. It includes synchronization words, framesize for the number of bytes in the message, the id of the Synchrophasor (ID), the timestamp in the SOC and FRACSEC fields, the DATA fields for the measurement and a CRC in the CHK field.

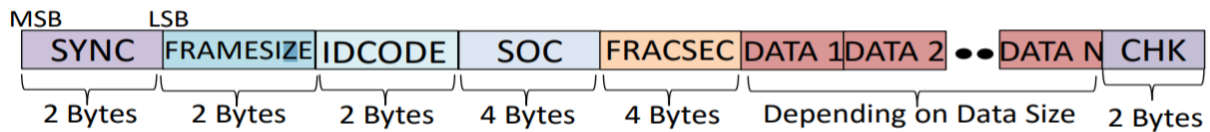


Figure 18. Message format of the IEEE C37.118-2 standard (taken from Khan+16))

There are different types of messages that are specified in the IEEE C37 standard: (1) data messages, for sending measurements, (2) configuration messages for sending information about how to process information, (3) header messages, which include human readable information about different aspects such as the source of the data, filtering, etc, and (4) command messages, that allows to trigger or stop the transmission of information from Synchrophasors. The communication protocol to use is not specified in the standard, although typically it relies on RS232 or IP, with the possibility of using both TCP or UDP.

As it has been mentioned before, the IEEE C37 standard does not specify anything about security. Therefore, it is exposed to several cyber-attacks, as described in Table 8:

Table 8. Description of the attack tools and their corresponding threats for the IEEE C37.118-2 protocol

Threat	Description of the Threat
Reconnaissance Attack	allows to discover vulnerabilities in the Synchrophasors components. These attacks are typically targeting the communication network by exploiting eavesdropping in the network traffic, which allows to capture information such as locations, names of substations, etc
Authentication/Access Attack	allows to get control of devices and resources. In IEEE C37 this is critical because no authentication mechanism is specified. The control of PMUs can be taken just by intercepting packets, injecting malicious ones or replaying them
Man In The Middle Attack (MITM)	Derived from the Authentication/Access Attack is the Man in the Middle Attacks, which allows an attacker to assume the role of one of the communication devices impersonating one of the peers, intercepting messages and altering them for its malicious purposes. In IEEE C37 messages, intercepting and altering configuration messages are a serious threat that can be easily carried out
Replay or Reflection Attack	These attacks rely on MITM attacks to replay communications, altering the content to lead to incorrect decisions
Denial of Service Attack	This type of well-known attacks can be carried out in IEEE C37 by targeting the communication channel between PMUs and PDCs or between substations and control centres. This is done by exhausting any of these components with messages generated by malicious attackers. As there is no security mechanisms specified, there is no way to check if a message is legitimate or not, thus being processed by the Synchrophasor, which becomes stalled due to the need of processing an abnormal amount of messages

4.1.6 Other relevant protocols within the EPES domain

Apart from the domain specific protocols specified before, the XL-EPDS also considers additional protocols that are of interest for the EPES domain. The following subsections details two of them. These protocols are further detailed in D5.2

4.1.6.1 MQTT protocol

The Message Queuing Telemetry Transport (MQTT) protocol is a Client Server publish/subscribe messaging transport protocol. Andy Stanford-Clark and Arlen Nipper originally designed the MQTT but currently is in the OASIS (Organization for the Advancement of Structured Information Standards) and has standard defined in ISO/IEC 20922: 2016 0, 0. It constitutes a standard protocol that has many assets; it is open, simple to use and easy to implement. Moreover, it is used widely for communication in Machine-to-Machine (M2M) and Internet of Things (IoT) system in case that required limited resources because of its lightweight attribute and the small bandwidth requirements [Banks+14].

MQTT has great capabilities for the SCADA systems and in the smart grids, especially if the SCADA and the IoT are required to interact. Since the IoT becomes essential for SCADA systems, the MQTT protocol would be an interesting candidate to replace more common protocols that are not able to adapt to the IoT. An important feature of the MQTT protocol is the Quality of Service (QoS) variable. The QoS variable determines how a message is sent and how the receiver responds to the message. Three levels of the variable can be set to examine the quality. The level 0 denotes that the message has been sent once and no response will be sent. The level 1 guarantee that the message has been reached to the receiver at least once. Moreover, Level 2 means that every message is guaranteed to be sent and received precisely once.

Multiple attack scenarios have been assumed in the literature to examine the vulnerability of the MQTT protocol. More specifically, Andy et al., 0 examined scenarios that associated with the exploitation of data privacy, authentication, data integrity and the existence of Botnets over MQTT protocols. Ozgur et al., 0 examined three different scenarios at the communication level; the Message Integrity Attack (MIA) according to this scenario the bytes of data received by the subscriber of the destination could be changed by a Man In The Middle attack (MITM). The Delay Attack (DA) that describes the situation that data is delayed due to a denial of service (DoS) attack and obtain as a result congestion problems. Finally, the Packet Drop Attack (PDA) that describes the situation that data is entirely lost due to a DoS attack.

Taking into consideration the aforementioned main attacks, four different attack scenarios will be developed to produce the abnormal traffic for the MQTT protocol⁶. The MITM attack scenario that intercepts the communications, filters and dumpers the measurement that are send or received. The second scenario is related to the Unauthorized Access, Failure of devices or systems and Manipulation of information. The Unauthorised publishing to smart devices is the scenario that provides the ability to the attacker to connect to the broker in order to subscribe in all the topics and to publish unauthorized commands. Furthermore two DoS scenarios will be developed; the DoS attacks against MQTT broker that associates with flood connection that sends multiple connection messages to exhaust server resources and the DoS that associates with the large payload attack that publishes spam messages repeatedly to a specific topic in comparison to the legitimate users that cannot publish this

⁶ <https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter>

messages. Table 9 summarizes the type of scenarios that will be developed, and the corresponding classification based on ENISA threat taxonomy and on the CAPEC classification.

Table 9. Description of the attack tools and their corresponding threats for the MQTT protocol.

Threat	Description of the Threat	ENISA Threat taxonomy	CAPEC classification
MITM attack	Attacker intercepts the communications, filters and dumps the measurements send/received by gateway	Man in the middle	94
Unauthorized publishing to smart devices	Attacker connects to the broker, subscribes to all topics and publish unauthorized commands	Unauthorized Access, Failure of devices and Systems, Manipulation of information	180
DoS attacks against MQTT broker: Connect flood	Attacker sends multiple connection messages to exhaust server resources	DoS	125
DoS attacks against MQTT broker: Large payload attack	Attacker publishes spam messages repeatedly to a specific topic, legitimate users cannot publish	DoS	594

4.1.6.2 NTP protocol

The Network Time Protocol (NTP) is one of the mostly used protocols for time synchronization. A client-server model is the type of model that usually describes the NTP protocol. The NTP sends and receives timestamps using the User Datagram Protocol (UDP) and reserves the port number 123. The synchronization of time in smart grids plays a key role, since a common time reference is essential to correlate power quality and to provide the coordination for any distributed actions [0]. Since the smart grids can have many medium and low voltage substations, the time synchronization should be compatible among the devices. When an NTP attack begins, the offset gets significantly higher, it takes a few exchanges before the victim adapts its system time. After the successful procedure of the attack, the victim's system time jumps to the time proposed by the attacker and the offset returns to normal values [0].

Malhotra et al., [0] discussed the main risks that network attackers can exploit to alter the time on client systems that use NTP protocol and categorize the attacks into two categories. The first category is the “on-path attacks” where the attacker occupies a privileged position between the server and the client or hijacks the traffic. The second category of attacks is the “off-path attacks” where the attacker does not observe the traffic between the client and the servers and can be anywhere in the network. The “small-step-big-step attack” constitute an “on-path attacks” that shifts clocks when clients are unlikely to notice. The Kiss o’ Death (KoD) packet attack is the type of the “off-path attack” that can disable NTP at a victim client upon receipt of the spoofed KoD, the client stops querying its servers and stops updating its clock.

Assuming the main attacks for the NTP protocol the main attack scenarios that will be developed in terms of time manipulation process. More specific, will be developed the clock time skimming attack and the KoD packet elimination attack. The Table 10 summarizes the attack scenarios that will produce

the abnormal traffic for the NTP protocol and corresponds them to ENISA threat taxonomy and to the CAPEC classification⁷.

Table 10. Description of the attack tools and their corresponding threats for the NTP protocol.

Threat	Description of the Threat	ENISA Threat taxonomy	CAPEC classification
Time manipulation	Clock time skimming attack	Time manipulation	172
Time manipulation	Kiss o' death packet elicitation attack	Time manipulation	172

4.2 SDN microSENSE security sensors

The protocols described in Section 4.1 are supported by the XL-SIEM thanks to the integration of logs produced by security sensors capable of monitoring such protocols. Within SDN-microSENSE several tools have been developed and integrated with the XL-SIEM as part of the XL-EPDS subsystem of the SDN-microSENSE architecture. The following subsections introduce these security sensors which will be deeply detailed in the tasks where they are developed.

4.2.1 EPES Honeypots

In the context of the SDN-microSENSE project, three honeypots of relevant industrial protocols: IEC 61850, IEC60870-5-104, Modbus have been developed. Each one of these three honeypots are prepared to have a direct connection with XL-SIEM Agent component. The Rsyslog interface exposed by XL-SIEM agent is the communication channel selected for event communication. Rsyslog is run into honeypot machine to recollect inputs to event file `/var/log/honeypot.log` and output the results to XL-SIEM by means of a secure connection.

Each developed honeypot has a different format for the events. Following a description of these format for each honeypot is shown, more information of these can be found in D3.3 [SDN33]

- **IEC61850 Honeypot:**

The format of the log for this honeypot is (Table 11):

- eventId: Identifier of the event
- Name: Name of the event
- Timestamp: Timestamp with the exact time in which the event occurred. The format of the timestamp is yyyy-MM-dd'T'HH:mm:ss*SSSZ
- Parameters: An array of parameter structures with additional information related to the event. The table presents the list of the event registered and the parameters associated

Table 11. Log taxonomy for the IEC61850 honeypot

name	key	value
New connection	ip	Ip of connected client-
Connection closed	ip	Ip of disconnected client
Control operation	ip	Ip of client sending control operation

⁷ Capec classification homepage, <https://capec.mitre.org/>

	ctlNum	ctlNum attribute send by the client
	orCat	Originator category provided by the client
	ln	Logical Node
	dataObject	Requested Control object
	command	SELECT OPERATE
	interlockCheck	Interlock-check requested by client
Read Operation	ip	Ip of client sending control operation
	Ln	Logical node owning requested data object
	dataObject	Data object owning requested data attribute
	dataAttribute	Requested data attribute
Write Operation	ip	Ip of client sending control operation
	Ln	Logical node owning requested data object
	dataObject	Data object owning requested data attribute
	dataAttribute	Requested data attribute

Finally, an example of an event is presented:

```
{"eventId":"0001","name":"New connection","timestamp":"Tue Mar 19 22:04:00 4448757",
"parameters":{"ip":"XX.XX.XX.13","param2":"Another param"}}
```

- **IEC60870-5-104**

The format of the log for this honeypot is (Table 12):

- timestamp: Timestamp with the exact time in which the event occurred. The format of the timestamp is yyyy-MM-dd'T'HH:mm:ss*SSSZ
- sensorid: Sensor identification
- id: Event identification
- src_ip: Source IP address
- src_port: Source TCP/UDP port
- dst_ip: Destination IP address
- dst_port: Destination TCP/UDP port
- data_type: Protocol
- request: The action sent to the honeypot
- response: The response to the request done by Honeypot
- event_type: This field is a code that corresponds to the specific event types cover by this Honeypot. There is more event_types but they are associated with the Conpot honeypot where this honeypot is based on

Table 12. Types of events for the IEC60870-5-104 honeypot

Event	Event_type
Counter interrogation command	C_CI_NA_1
Read command	C_RD_NA_1
Clock synchronization command	C_CS_NA_1
Test command	C_TS_NB_1

Reset process command	C_RP_NC_1
Delay acquisition command	C_CD_NA_1

Finally, an example of an event is presented:

```
{ "timestamp": "2020-06-22T16:44:34.656128", "sensorid": "default", "id": "245cc5ac-ba77-4677-4276-876976583791", "src_ip": "XX.XX.XX.55", "src_port": 6XX39, "dst_ip": "XX.XX.XX.135", "dst_port": 2XX4, "data_type": "iec104", "request": null, "response": null, "event_type": "C_RD_NA_1" }
```

• Modbus

The structure of this Modbus honeypot is the same as the one described for the IEC60870-5-104, due to the fact that both are based on the Conpot honeypot, and extended its functionalities. The extended functionalities in this modbus honeypot are:

- Mask Write Register(FC22 (0x16)): It changes the content of a holding register based on the use of AND and/or OR logical masks and the holding register's current content.
- Read/Write Multiple registers 8FC23 (0x17)): It operates as both read and write operations in a single Modbus command. The write process preceded the read process.
- Read FIFO Queue FC24 (0x18): It reads the content of a register's First-In-First-Out (FIFO) queue

An example of an event is presented:

```
{ "timestamp": "2019-11-25T13:34:53.902098", "sensorid": "default", "id": "167b36dc-af68-4935-8393-490972134866", "src_ip": "XX.XX.XX.80", "src_port": 6XXX4, "dst_ip": "XX.XX.XX.106", "dst_port": 5XXX0, "public_ip": "XX.XX.XX.106", "data_type": "modbus", "request": null, "response": null, "event_type": "NEW_CONNECTION" }
```

4.2.2 Network based incident detection: Enhanced Suricata

According to the Suricata documentation⁸:

“Suricata is a high performance Network IDS, IPS and Network Security Monitoring engine.”

Therefore, Suricata can work both as IDS and IPS. Both modes are based on rules that produces verdicts related to alerts or logs when working as IDS, adding drop, sdrop and reject actions when working as IPS. Suricata sniffs traffic directly from the network interface that is specified in the configuration, allowing to filter traffic related to a specific IP domain. The process that Suricata follows to process network traffic comprises three steps: (1) packet capture, which is the process that sniffs the network traffic from the specified interface, (2) decodes the stream, which is the process that parse the traffic and interpret its content, (3) detection, which, based on the information extracted from the traffic captured, applies the predefined rules and generates logs for those alerts that are triggered, and (4)

⁸ <https://suricata.readthedocs.io/en/suricata-5.0.3/what-is-suricata.html>

output, which generates the corresponding log. The set of rules to use during an execution of Suricata is also configurable.

Suricata already includes with a wide variety of rules capable of detecting a myriad of incidents, from denial of service attacks, to port scanning, suspicious activities, etc. Within SDN-microSENSE additional rules have been created, which allows to detect incidents associated to some of the EPES specific protocols, such as DNP3, IEC 60870-5-104, IEC61850 and Modbus. The following rule is an example that alerts about a potential denial of service over the IEC60870-5-104 protocol:

```
alert tcp $EXTERNAL_NET 2404 -> $HOME_NET any (msg:"PROTOCOL-SCADA IEC 104 force on
denial of service attempt"; flow:to_client,established,no_stream; content:"|68|";
depth:1; content:"|2D|"; within:1; distance:5; content:"|01|"; within:1; distance:8;
detection_filter:track by_src,count 50,seconds 5;
reference:url,dragos.com/blog/crashoverride/CrashOverride-01.pdf; reference:url,us-
cert.gov/ncas/alerts/TA17-163A; reference:url,welivesecurity.com/wp-
content/uploads/2017/06/Win32_Industroyer.pdf; classtype:attempted-dos; sid:43228;
rev:4;)
```

Suricata logs are integrated in the XL-SIEM natively, although new rules added requires to be incorporated to the XL-SIEM data source catalogue. To this end, as it was described in Section 5, the plugin_id that identifies Suricata sensors at the XL-SIEM is 1001, while the plugin_sid used to identify the type of event received from Suricata sensors depends on the rule triggered. In this case, Suricata already identifies rules with a unique numerical identifier, called sid, which is used by the XL-SIEM Agent to map to the plugin_sid variable used to identify the event type at the XL-SIEM side.

Further details about the enhancement carried out in Suricata to incorporate detection capabilities within EPES infrastructures are given in D5.2.

4.2.3 SDN based incident detection: Nightwatch

Nightwatch is an Intrusion Detection and Classification Module (IDCM) for advanced and novel threats to Electrical Power Energy Systems (EPES). It leverages artificial intelligence technologies for accurately and rapidly determining the likelihood that such a system has been compromised. Nightwatch supports low-computational analysis and machine learning techniques for resource constrained devices common in EPES environments. In the SDN-microSENSE, Nightwatch is using information derived from the SDN-controller to determine whether the SDN components are under cyber-attack. Additionally, Nightwatch can determine the type of the attack and likelihood that an SDN component has been compromised.

The communication between Nightwatch and the SDN-controller is made through the SDN-controller's Northbound interfaces. Nightwatch will gather network-related information using Representational State Transfer (REST) based queries. The network information will include the network topology of the SDN switches, available network ports, and statistical information related to the available network ports. The network-based information will enable Nightwatch to elicit the nature of threats targeting SDN components, such as malware, service, or resource disruption.

Nightwatch will be able to consume data from XL-SIEM using the XL-SIEM's RabbitMQ message broker. Nightwatch will use the RabbitMQ to read events from XL-SIEM agents as inputs for its intrusion detection analysis. Security-related events from XL-SIEM will enable Nightwatch to augment its intrusion detection process with additional information on the security posture of the system.

The resulting analysis of Nightwatch based on input from the SDN-controller and the XL-SIEM agents will be made available to the XL-SIEM for a consolidated analysis of the security of the EPES system. The technical specifications and use of Nightwatch with the SDN-controller and the XL-SIEM are further described in D5.2, as part of the SS-IDPS System.

4.2.4 *Modbus Intrusion Detection Sensor*

The Modbus Intrusion Detection Sensor is an ML-based Network Intrusion Detection System (IDS), which is capable of detecting 14 Modbus/TCP related cyberattack types. In particular, the sensor relies on Transmission Control Protocol/Internet Protocol (TCP/IP) network flow statistics and consists of the five following modules: a) Data Collection Module, b) Feature Selection and Pre-processing Module, c) Training Module, d) Detection Module and e) Response Module. The Data Collection Module is responsible for capturing the Modbus/TCP network traffic, using tcpdump. Following this, the Feature Selection and Pre-processing Module undertakes to extract and pre-process the Modbus network statistics that will be used for the detection process. The Training Module constitutes an offline functional unit which is responsible for the training process of the ML models. Next, the Detection Module loads the binary ML model and thus performs the classification process. Finally, based on the outcome of the Detection Module, the Response Module stores the information of the corresponding malicious Modbus/TCP network flow in a log file, with a specific label which indicates the Modbus/TCP cyberattack. The Modbus Intrusion Detection Sensor can detect 14 Modbus/TCP related cyberattacks, including:

- 1) modbus/function/readInputRegister (DoS),
- 2) modbus/function/writeSingleCoils,
- 3) modbus/scanner/getfunc,
- 4) modbus/dos/writeSingleRegister,
- 5) modbus/function/readDiscreteInputs (DoS),
- 6) modbus/function/readHoldingRegister (DoS),
- 7) modbus/function/readCoils (DoS),
- 8) modbus/function/readInputRegister,
- 9) modbus/function/writeSingleRegister,
- 10) modbus/dos/writeSingleCoils,
- 11) modbus/function/readDiscreteInput,
- 12) modbus/scanner/uid,
- 13) modbus/function/readCoils and
- 14) modbus/function/readHoldingRegister.

More details about the Modbus Intrusion Detection Sensor will be provided in D5.3.

4.2.5 *DNP3 Intrusion Detection Sensor*

The DNP3 Intrusion Detection Sensor is also an ML-based IDS which can recognise timely and with high accuracy DNP3-related cyberattacks. The architecture of the DNP3 Intrusion Detection Sensor is composed of six modules: a) DNP3 Traffic Sniffing Module, b) DNP3 Network Flow Statistics Extraction Module, c) Pre-processing Module, d) Training Module, e) Detection Module and f) Notification Module. The first module captures the DNP3 packets, storing them into pcap files. Next, the DNP3 Network Flow Statistics Extraction Module receives the pcap and exports DNP3 flow statistics related explicitly to the DNP3 packets with including information from the previous communication layers (i.e.,

transport layer, network layer, link layer, etc.). Then, the Pre-processing Module undertakes to normalise appropriately the DNP3 flow statistics. Thus, based on this information, the Training Module trains the ML model of the Detection Module, which in turn detects the DNP3-related cyberattacks. Finally, the Response Module stores in a log file the malicious DNP3 flows with a label which specifies the DNP3 cyberattack type. The DNP3 Intrusion Detection Sensor can recognise the following DNP3 cyberattacks: a) DNP3 Disable Unsolicited Messages Attack, b) DNP3 Cold Restart Message Attack and c) DNP3 Warm Restart Message Attack. More information about the DNP3 Intrusion Detection Sensor will be provided in D5.3.

4.2.6 IEC 60870-5-104 Intrusion Detection Sensor

The IEC 60870-5-104 Intrusion Detection Sensor focuses on cyberattacks against IEC-104. It uses an appropriate trained an ML model, which receives as input IEC-104 flow statistics. In particular, IEC 60870-5-104 Intrusion Detection Sensor is composed of six modules: a) Data Capturing Module, b) IEC 60870-5-104 Flow Generator, c) Feature Selection and Pre-processing Module, d) Training Module, e) Detection Module and f) Notification Module. The first module is responsible for monitoring and capturing the IEC-104 network packets, using tcpdump. Next, IEC 60870-5-104 Flow Generator receives the output, i.e., the pcap file of the previous module and generates IEC-104 flow statistics that are related only to the header and the payload of the IEC-104 packets, without including information from the previous TCP/IP layers. Next, the Feature Selection and Pre-processing Module isolates and pre-processes appropriately only those elements that will be given to the ML model, which is trained by the Training Module. The Detection Module incorporates the ML Model, thus recognising IEC-104 cyberattacks. Finally, the Notification Module updates a log file with the malicious IEC-104 flows, comprising a label denoting the IEC-104 cyberattack. D5.3 will detail the IEC 60870-5-104 Intrusion Detection Sensor.

4.2.7 IEC 61850 (GOOSE) Intrusion Detection Sensor

The IEC 61850 (GOOSE) Intrusion Detection Sensor is an ML-based IDS, discovering potential intrusions related to the GOOSE communication model. As in the previous cases, it consists of six modules: a) Data Collection Module, b) GOOSE Flow Generator, c) Feature Selection and Pre-processing Module, d) Training Module, e) Detection Module and f) Response Module. The first module is responsible for sniffing GOOSE packets and storing them into two files: a) pcap and b) JSON. Both files (i.e., pcap and JSON) are processed by the GOOSE Flow Generator, which produces GOOSE flow statistics. The Feature Selection and Pre-processing Module chooses and normalises the features that will be used for the training of the ML model and the training procedure is implemented offline by the Training Module. Following this, the trained ML model is integrated into the Detection Module, which composes the core of the IEC 61850 (GOOSE) Intrusion Detection Sensor, responsible for recognising the various GOOSE cyberattacks. As it was mentioned above, the Detection Module can discriminate four GOOSE-related cyberattacks: a) Data Manipulation, b) DoS, c) Message Suppression and d) Disturbance. Finally, the Response Module updates a log file with the abnormal GOOSE flows, including a label that signifies the corresponding GOOSE cyberattack.

4.2.8 L-ADS

The L-ADS (Live Anomaly Detection System) is a tool developed by Atos which use machine learning based models to detect anomalies in network flows. More specifically, the L-ADS can monitor traffic network directly from the network or read traffic captures to analyse its content. The L-ADS infer anomalous connections to and from devices connected to the infrastructure monitored. The captured

or read traffic is pre-processed and transformed into Netflow, which takes from the packets analysed just relevant information such as source and destination IPs, duration of the flow analysed or the size of the flow in bytes or packets, among others.

The L-ADS [Granadillo+19] works in two modes: supervised and unsupervised. The former can work based on predefined rules that helps to identify anomalous connections, while the latter self-learn and infer in an autonomous way the anomalous connections.

The L-ADS can detect anomalies associated to any protocol working on top of TCP or UDP. In SDN-microSENSE, the L-ADS has been enhanced with support protocols often used in the EPES domain. More details about the L-ADS in SDN-microSENSE are given in 5.3.

4.2.9 CERTH ML detector

The Machine Learning (ML) detector is the tool that identifies abnormal events in the network traffic and provides the corresponding information to the XL-SIEM. The purpose of the ML detector is to expand the capabilities of the L-ADS tool taking into consideration the Modbus, the MQTT and the NTP protocols. The architecture of the ML detector is depicted in the Figure 19 and composed of three main parts. In the first part the network traffic from the Modbus, the MQTT and the NTP protocol is being capture for both normal and abnormal events. The T-shark network protocol analyser⁹ is used to capture packet data from the network and to produce the corresponding pcap files. Then in the second part for the data collection procedure, the CicFlowmeter¹⁰ is used to extract the features of the bidirectional netflows from pcap files and produces the csv files with the records of the features. The third part of the procedure concerns the development of the different ML models per protocol. Based on the methodology that have been proposed for the detection of cyber-attacks [0, 0, 0] the ML detector will be used for the detection of the attacks that associated with the Modbus, MQTT and NTP protocols. The log files with the predictions of each model will be sent to the XL-SIEM. The development of the different ML models focuses on the development of Artificial Neural Network (ANN) models that take into consideration the advantages and the disadvantages of the comparison from different machine learning techniques on SCADA systems. Yasakethu et al., [0] described the comparison of different machine learning techniques for the protection of SCADA systems. The results of the comparison in terms of the ANN prove that the usage of this classifier requires prior knowledge of the anomaly type, needs adequate balanced training data and demands large number of attack training data. There are two main advantages for the ANN models are: they constitute nonlinear data analysis and demand low computational time. The usage of the ANN aims to improve the challenges of big data that associated with difficulties to store, track, analyse, capture, and share the generated data [0]. Moreover, the extension of the ML detectors that have been developed for the multiclass classification, [0, 0, 0] in terms of self-training procedure and using data from the corresponding Use cases that will developed for the SDN-microSENSE project aims to expand the effectiveness of the ML detector for the Smart Grids. The validation of the proposed models will be obtained from the calculation of metrics such as the accuracy, the precision and the recall for each model. The detailed information for the ML detector and its components will be provided in D5.3

⁹ <https://www.wireshark.org/docs/man-pages/tshark.html>

¹⁰ <https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter>

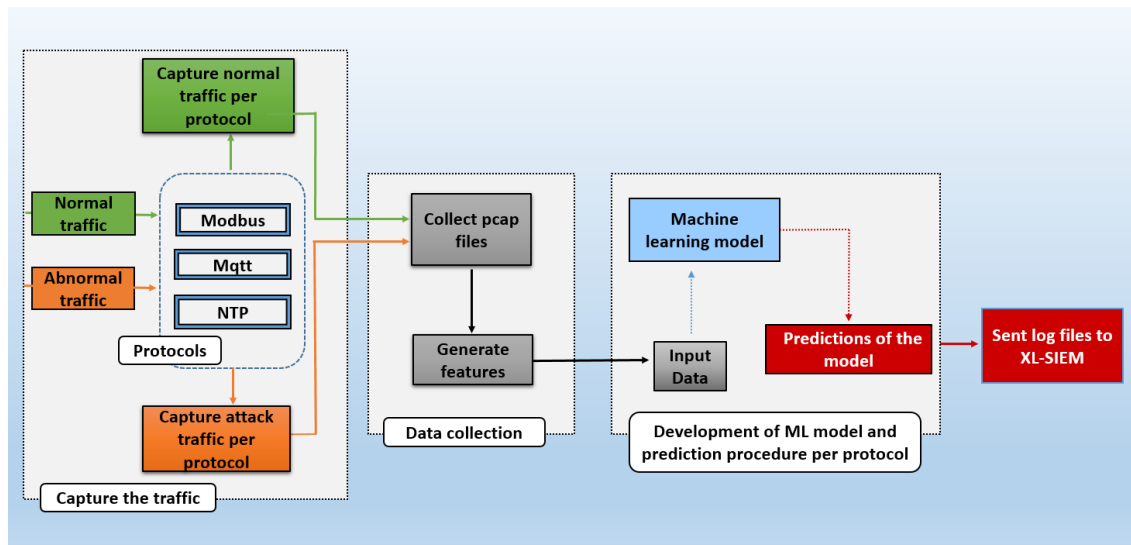


Figure 19. The architecture of the CERTH ML detector

The ML detector will provide the logs to the XL –SIEM tool in Json files. The information that will be provided is the following properties for each event: the Source IP, the Source Port, the Destination IP, the Destination Port, the Timestamp, the protocol and the Label that describes the type of attack. The Json schema that describes the information sent by the ML detector to the XL-SIEM is the following:

```

{
  "type": "array",
  "items": [
    { "type": "object",
      "properties": {
        "Flow ID": {
          "type": "string",
        },
        "Src IP": {
          "type": "string",
        },
        "Src Port": {
          "type": "integer",
        },
        "Dst IP": {
          "type": "string",
        },
        "Dst Port": {
          "type": "integer",
        },
        "Protocol": {
          "type": "integer",
        },
        "Timestamp": {
          "type": "string",
        },
        "Label": {
          "type": "string"
        }
      },
      "required": [
        "Flow ID",
        "Src IP",
        "Src Port",
        "Dst IP",

```

```
"Dst Port",
"Protocol",
"Timestamp",
"Label"
] }
```

4.3 Interfaces and data exchanged

Several mechanisms have been implemented in SDN-microSENSE for the interaction of the XL-SIEM with other components of the platform. The following subsections describe the mechanisms configured in SDN-microSENSE.

4.3.1 XL-SIEM input mechanisms

As depicted in Figure 6, the XL-SIEM receives logs, events and other input data from components, such as detectors or honeypots, through the XL-SIEM agent. The XL-SIEM agent receives raw logs, in different format (i.e., json or plain text) containing different type of information, which are normalized by the XL-SIEM Agent in a common, normalized format. Further details will be given in Section 0. These logs are received by the XL-SIEM Agent using rsyslog. Rsyslog is an open source utility to send registry messages using an IP network. It is natively included in any Unix based distribution so no additional software is required to use it. Although there are several alternatives to send information using rsyslog, one of the most common is the one depicted in Figure 20. In this configuration detectors (or any tool that will send logs to the XL-SIEM Agent), produces logs that are written in a log file. The rsyslog process detects new input to the log file and automatically forwards it to the rsyslog server configured. This is a very flexible approach because minimum configuration is required at the detector (assuming that most of them already write their output to a file) and the rsyslog guarantees a transparent and efficient mechanism to transfer information which can also be secured by using TLS certificated to secure the communication channel.

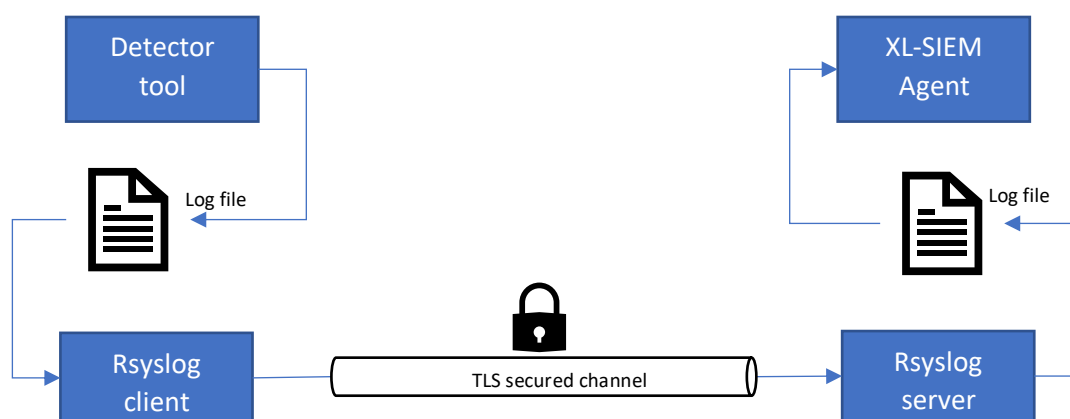


Figure 20. Exchange of information from detectors to the XL-SIEM agent using rsyslog

The following steps were used to create a secure communication channel with the XL-SIEM agent using rsyslog. The only requirement for a tool that wants to communicate to the XL-SIEM agent using rsyslog as described in this document is that logs are written in a log file (i.e., mytool.log). Rsyslog will monitor that file and will forward the new logs included in such file to the Atos XL-SIEM agent

Step 1: Install Rsyslog-gnutls

By default, Linux machines will have rsyslog available. To use it over a secure channel it is required to install the TLS support. For Debian/Ubuntu this is:

```
[root@mymachine ~]# apt-get install rsyslog-gnutls
```

Step 2: Copy the Atos certificate to any folder of your machine

A certificate is required to use this secure channel. A new certificate (atos.cert.pem) was created for this purpose:

```
[root@mymachine ~]# mkdir /etc/rsyslog-keys  
[root@mymachine ~]# cp atos-cert.pem /etc/rsyslog-keys/
```

Step3: Create a configuration file for the secure channel in rsyslog.

At /etc/rsyslog.d create a new file (30-xlsiem.conf). It can be chosen any name as long as it ends with .conf. This content has to be added such file:

```
$ModLoad imfile  
$InputFileName /var/log/mytool.log      <= file which content will be sent to the server  
$InputFileTag mytool                   <= a tag that identifies the program that fills such file  
$InputStateFile mytool.log             <= name of the log file  
$InputFileSeverity alert  
$InputFileFacility local6  
$InputRunFileMonitor  
$InputFilePollInterval 1  
# certificate files  
$DefaultNetStreamDriverCAFile /etc/rsyslog-keys/atos-cert.pem <= path where the Atos  
                                                                certificate is  
  
# make gtls driver the default  
$DefaultNetStreamDriver gtls  
$ActionSendStreamDriverMode 1  # run driver in TLS-only mode  
$ActionSendStreamDriverAuthMode anon  
  
# forward just content of the file associated to such tag to the Atos rsyslog server  
if $programname == 'mytool' then    @@(o)XX.XX.XX.XX:AAAA  
if $programname == 'mytool' then stop
```

Step 4: Restart rsyslog service

```
[root@mymachine ~]# service rsyslog restart
```

Or

```
[root@mymachine ~]# systemctl restart rsyslog
```

Step 5: Check status of the rsyslog service

```
[root@mymachine ~]# systemctl status rsyslog
```

An output without errors should be something like what shown in Figure 21.

```

root@sdnclient:/etc/rsyslog-keys# systemctl status rsyslog
● rsyslog.service - System Logging Service
   Loaded: loaded (/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-04-27 08:19:06 UTC; 46min ago
   TriggeredBy: ● syslog.socket
     Docs: man:rsyslogd(8)
           https://www.rsyslog.com/doc/
   Main PID: 5170 (rsyslogd)
     Tasks: 5 (limit: 1075)
    Memory: 1.7M
   CGroup: /system.slice/rsyslog.service
           └─5170 /usr/sbin/rsyslogd -n -iNONE

Apr 27 08:19:06 sdnclient systemd[1]: rsyslog.service: Succeeded.
Apr 27 08:19:06 sdnclient systemd[1]: Stopped System Logging Service.
Apr 27 08:19:06 sdnclient systemd[1]: Starting System Logging Service...
Apr 27 08:19:06 sdnclient systemd[1]: Started System Logging Service.
Apr 27 08:19:06 sdnclient rsyslogd[5170]: imuxsock: Acquired UNIX socket '/run/systemd/journal/syslog'
Apr 27 08:19:06 sdnclient rsyslogd[5170]: rsyslogd's groupid changed to 110
Apr 27 08:19:06 sdnclient rsyslogd[5170]: rsyslogd's userid changed to 104
Apr 27 08:19:06 sdnclient rsyslogd[5170]: [origin software="rsyslogd" swVersion="8.2001.0" x-pid="5170"

```

Figure 21. Status of a correctly configured rsyslog client based on TLS

Any error will be prompted.

Step 6: Test

The setup can be tested just by adding manually new content to the log file:

```
[root@mymachine ~]# echo "This is a test from file" >> /var/log/mytool.log
```

If everything is correct the Atos agent must have received the message including “This is a test from file” (Figure 22)

```

root@21b344a7bdd8:/etc/rsyslog.d# tail -n 1 /var/log/syslog
Apr 27 08:19:15 sdnclient testlog This is a test from file

```

Figure 22. Proof that the message was received by the client

4.3.2 XL-SIEM output mechanisms

The main output of the XL-SIEM is basically alerts derived from the reasoning carried out by the XL-SIEM engine based on the events received from XL-SIEM agents. To this end, several mechanisms are included in the XL-SIEM engine to export these security alert: to a csv file, to a Kafka broker, etc. The mechanism used in SDN-microSENSE to export security alerts is based on a RabbitMQ server.

On the other hand, XL-SIEM agents send normalized events to the XL-SIEM engine by using a secure TCP socket connection. However, the XL-SIEM provides with an optional mechanism to export these events to third parties through a RabbitMQ server. Using a RabbitMQ broker provides with advantages when integrating other components of the SDN-microSENSE platform:

- Events and alerts are exported in real time.
- Security is guaranteed by using TLS secured channels to communicate with the RabbitMQ server.
- Flexibility to attach any number of consumers just by attaching them to the corresponding queue for events or alerts.

Within SDN-microSENSE, the capability to export events and alerts to a RabbitMQ is used by several components, as depicted in Figure 23.

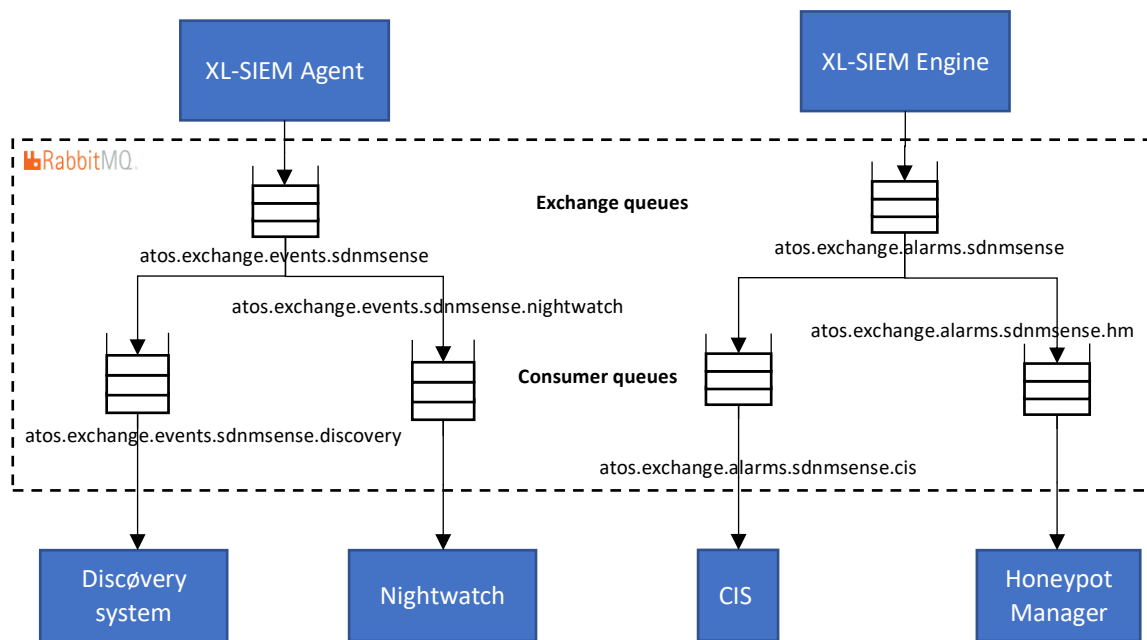


Figure 23. SDN-microSENSE components using the output of the XL-SIEM Agent and Engine through RabbitMQ

- Events exported from the XL-SIEM agent is used by the Nightwatch tool (further described in D3.3 and D5.2). As part of T3.3, Nightwatch uses these events for detecting zero days attacks based on the output from honeypots. Nightwatch also uses these events, along with SDN controller logs, to detect incidents as part of the SDN-IDPS developed in T5.2.
- Events exported from the XL-SIEM agent to the Discøvery system, used to visualize infrastructure topology based on the information retrieved from detectors (further described in D5.3)
- Alerts exported from the XL-SIEM engine that is used by several components of the SDN-microSENSE architecture:
 - By the ATOS CIS, which is used to interface with the ARIEC component developed in T5.5 (described in D5.5).
 - By the Honeypot Manager, part of T3.3 and described in D3.3, which uses them to identity zero days attacks and dynamically deploy additional honeypots in case it is necessary and according to the type of zero-day vulnerability exploited.

The configuration used for setting-up the RabbitMQ queues is of type Fan-out, which leverages on exchange queues where the XL-SIEM Agent and Engine push events and alerts while consumers will create their own consumer queues, attach them to the corresponding exchange queue and read from it. This is the best possible solution when more than one consumer is expected. Otherwise, if all consumers read from the same consumer queue, some data will arrive to some consumers while other consumers will miss some data such as messages pushed to consumer queues are removed when consumed.

The make integration easier a python script has been created to read from the RabbitMQ server set up for the XL-SIEM. The following is the configuration file used for the python connector created when connected to the alarms exchange queues:


```
[SERVER]
SERVER_IP = <IP of the RabbitMQ server>
SSL_PORT = <Port of the RabbitMQ server>

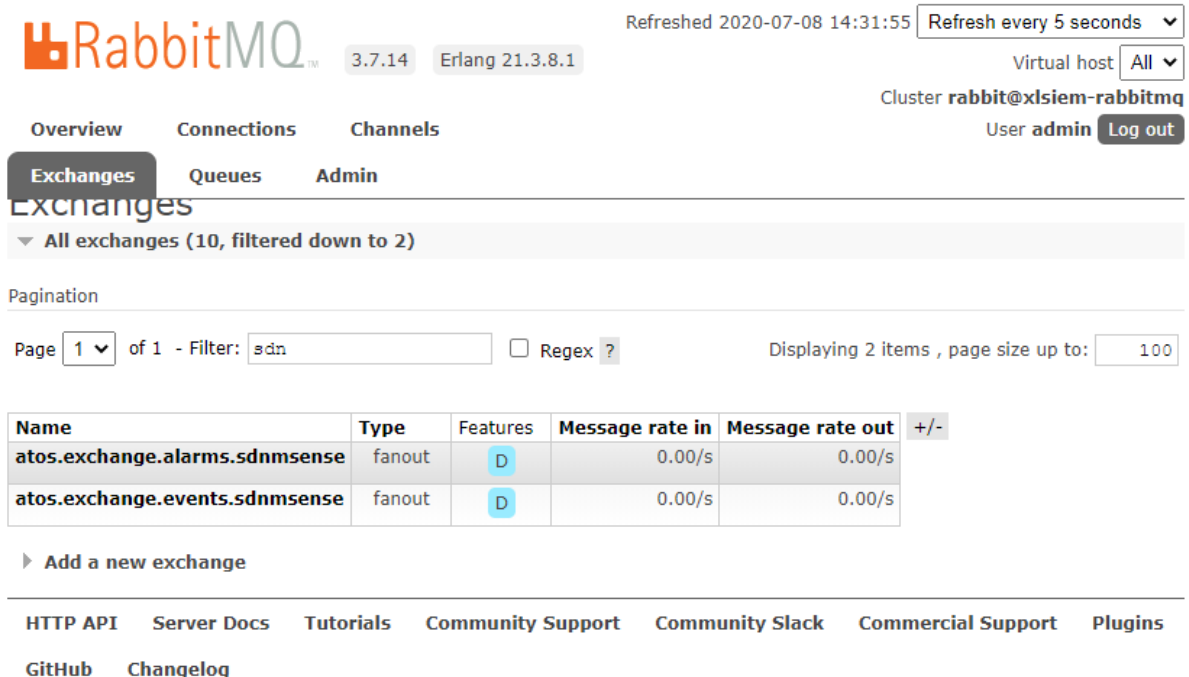
[CLIENT]
CONSUMER_NAME = <component_name>

[QUEUES]
EXCHANGE_QUEUE_NAME = atos.exchange.alarms.sdnmsense
CONSUMER_QUEUE_NAME = atos.queue.alarms.sdnmsense

[CERTIFICATES]
CA_CERT = ./cacert.pem
CLIENT_CERT = ./cert.pem
CLIENT_KEY = ./key.pem
```

The key parts of the configuration are the EXCHANGE_QUEUE_NAME, the CONSUMER_QUEUE_NAME and the CONSUMER_NAME. The EXCHANGE_QUEUE_NAME must match the exchange queue configured at the RabbitMQ server, while the CONSUMER_QUEUE_NAME is used to create the consumer queue at the RabbitMQ server that is bound to the EXCHANGE_QUEUE_NAME. The CONSUMER_NAME is also used to create the name of the consumer queue, added as suffix to the CONSUMER_QUEUE_NAME, meaning that the consumer name will be configured as “CONSUMER_QUEUE_NAME.CONSUMER_NAME”. This is done to better identify at the RabbitMQ server the queues created by components.

Figure 24 shows the RabbitMQ control panel with the exchange queues created by the XL-SIEM Agent and Engine.



The screenshot shows the RabbitMQ management interface. At the top, it displays the RabbitMQ logo, version 3.7.14, and Erlang 21.3.8.1. The interface is refreshed every 5 seconds. The cluster is named 'rabbit@xlsiem-rabbitmq' and the user is 'admin'. The 'Exchanges' tab is selected, showing a list of exchanges. The list is filtered down to 2 items. The table below shows the details of these exchanges.

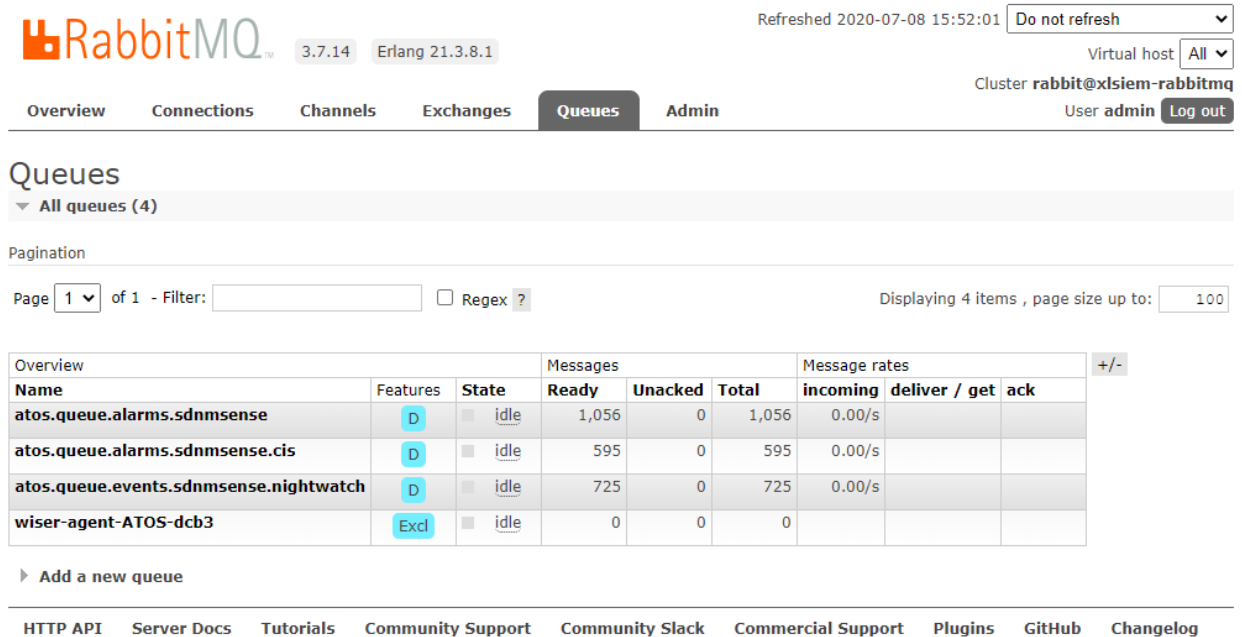
Name	Type	Features	Message rate in	Message rate out	+/-
atos.exchange.alarms.sdnmsense	fanout	D	0.00/s	0.00/s	
atos.exchange.events.sdnmsense	fanout	D	0.00/s	0.00/s	

Below the table, there is a link to 'Add a new exchange'. At the bottom of the interface, there are links for HTTP API, Server Docs, Tutorials, Community Support, Community Slack, Commercial Support, and Plugins.

Figure 24. Exchange queues at the RabbitMQ attached to the XL-SIEM

Figure 25 shows the RabbitMQ panel with the consumer queues attached to any of the exchange queues available. It is worth noticing that the python script provided as generic consumer deletes the consumer queue at exit. This is a normal behaviour and indeed desirable in order to save resources at

the RabbitMQ server. If the queue is created again later the exchange queue will push pending messages to the consumer queue just created, therefore guaranteeing that no messages are lost.



The screenshot shows the RabbitMQ web interface. At the top, the RabbitMQ logo is displayed along with version information (3.7.14, Erlang 21.3.8.1) and a refresh status (Refreshed 2020-07-08 15:52:01). The interface includes navigation tabs for Overview, Connections, Channels, Exchanges, Queues, and Admin. The 'Queues' tab is selected, showing a list of all queues (4). The queues listed are:

Overview				Messages			Message rates			+/-
Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack		
atos.queue.alarms.sdnmsense	D	idle	1,056	0	1,056	0.00/s				
atos.queue.alarms.sdnmsense.cis	D	idle	595	0	595	0.00/s				
atos.queue.events.sdnmsense.nightwatch	D	idle	725	0	725	0.00/s				
wisier-agent-ATOS-dcb3	Excl	idle	0	0	0					

Below the table, there is a link to 'Add a new queue'. The footer of the interface includes links for HTTP API, Server Docs, Tutorials, Community Support, Community Slack, Commercial Support, Plugins, GitHub, and Changelog.

Figure 25. Consumer queues configured at the RabbitMQ server

5 Event connectors

This section will describe the procedure used by the XL-SIEM agent to normalize events, the format of the normalized event and the approach used in SDN-microSENSE to gather information from detectors and honeypots.

The XL-SIEM agent receives logs through a rsyslog server. These logs, generated by detectors, can contain any information and can be specified in any format, such as json or plain text. The XL-SIEM agent receives those logs, filters them (for example, according to the type of log) and extracts the relevant information from them in order to generate a normalized event in a common format.

The XL-SIEM agent is based on plugins which are created to process the logs received from the detectors. In general, every detector has a dedicated plugin capable of processing all types of events that such detector produces. These plugins process logs by using regular expressions, extracting information from logs and matching them to a set of predefined fields included in the normalized event exported to the XL-SIEM engine.

Plugins are composed of three important parts:

- ID definition. Every plugin has a numerical ID that clearly identifies it. It is normally used to identify the detector that sent the logs processed by the plugin. This ID is also used by the XL-SIEM Engine to know the detector associated to the event received and apply the corresponding correlation rules to those events
- Source of logs. Plugins read logs received by the XL-SIEM agent through syslog. As specified in Section 4.3.1, the rsyslog server at the XL-SIEM agent store logs received in a file. Rsyslog can filter then and store these logs in different files depending on some field appearing in the log (for example, the name of the sensor). Every plugin contains the route to the file that contains the logs to process, reading and processing every new entry that is written in such file
- Parsing sections. A plugin section will be able to parse one type of logs. Every plugin will contain one or more parsing section. This is useful to group logs in plugins, because detectors will be able to send more than one type of log. Every section will contain two parts:
 - Regular expressions. This is the regular expression that will parse the log process by the plugin. A perfect matching is required to extract information from the logs. If there is no match, the regular expression of the next section is evaluated. If there is no matching expression in any section, the log is ignored.
 - Normalized fields. This is the mapping between the fields extracted from the logs at the regular expression and the normalized fields used by the XL-SIEM engine. There is a limited set of fields that can be normalized. The data format used for normalized events is based on the OSSIM format¹¹, but adding several changes to simplify its processing. The most important ones are source and destination fields for IP, port and MAC address. It is also included fields for usernames, filenames, and nine free fields that can be mapped to any information extracted from logs. It is important to notice that every plugin section will map a variable called `plugin_sid`, which is a numerical value that identifies the type of event within a plugin. This is also used by the XL-SIEM to identify the type of event and apply the appropriate correlation rules.

¹¹ <https://cybersecurity.att.com/documentation/usm-appliance/events/event-details-fields.htm>

For every new plugin to be developed it is necessary to know exactly the type of events that will be parsed and the exact format. Therefore, it is necessary to compile from detectors this information. In SDN-microSENSE it has been created a template that makes the compilation of information from detectors easier (Table 13).

Table 13. Template to compile log taxonomies from detectors

Log name	A name that identifies the log
Log description	The text describing the log
Important fields	The fields from the log that might be important for the XL-SIEM engine
Field type	The format of the field. This is, whether it is a string, a numerical value, etc
Possible values	If possible, the possible values that every field has (i.e., 0 to 10 for a numeral value)
Field description	A description of the meaning of such field. This is useful to create correlation rules at the XL-SIEM engine
Example	An example of the log that the XL-SIEM agent will receive from this detector

According to the WP5 architecture depicted in Figure 3, the following detectors will be integrated with the XL-SIEM agent:

- IDPS detectors: Enhance Suricata and Data Injection detector. The details of this log taxonomy are given in D5.2
- SDN IDPS: Nightwatch. The details of this log taxonomy are given in D5.2
- Machine Learning based detectors: UOWM, SID and OINF Machine learning based detector, ATOS L-ADS and CERTH machine learning models. The details of this log taxonomy are given in D5.3
- Access control and data request detector. The details of this log taxonomy are given in D5.3
- Honeypots. This logs taxonomy is detailed in WP3.

6 Security alerts

Security alerts are created by the XL-SIEM engine by correlating events received from detectors through the XL-SIEM agent. This correlation is done through a set of correlation rules that evaluate several aspects of associated to the events received from the XL-SIEM agent. This evaluation uses the fields included in the event (for example, to compare source IPs) and include additional variables to compare such as the number of occurrences of certain types of events, a time window where to receive events, etc.

For every new event received from SDN-microSENSE detectors, it is required to evaluate its content, and determine whether it can be inferred an incident, and therefore, to trigger the corresponding alert.


Therefore, new rules have been created to deal and evaluate the events received from SDN-microSENSE detectors. The process to create additional rules comprises several steps, as depicted in Figure 26.



Figure 26. Steps to create new correlation rules

Set up new event definitions

For the XL-SIEM to process new events received from SDN-microSENSE detectors it is required to let the XL-SIEM engine know that new events are available. To do so, for every new detector (this is, for every new plugin developed at the XL-SIEM agent) it is required to configure a new Event type at the XL-SIEM. New event types are defined at the XL-SIEM by using the `plugin_id` defined at the plugin and the `plugin_sid` defined for every event type that a detector is sending. Figure 27 represents the configuration panel at the SDN-microSENSE XL-SIEM that shows two event types for the Dionaeea plugin (`plugin_id=1669`). The Data Source ID field corresponds to the `plugin_id` of the event while the Event Type ID corresponds to the `plugin_sid` of the event. Additionally, the fields category, subcategory, priority and reliability can be fixed. It is worth noticing that the values priority and reliability will be used to determine the severity (shown as risk) of incidents associated to these events.

 SDN-μSense

[Dashboards](#)
[SIEM Analysis](#)
[Configuration](#)
[Reports](#)

Event types (1669, dionaea) [<< Back to Data Source](#) Displaying 1 to 2 of 2 event types


[Insert new event type](#)
[Edit](#)
[Delete selected](#)

Data Source ID	Event type ID	Category	Subcategory	Class	Name	Priority	Reliability
1669	1	Honeypot	Connection_Opened	-	Dionaea: Incoming Connection	2	1
1669	2	Honeypot	Attack_Detected	-	Dionaea: Malware detected	3	1

Figure 27. Example of event definition for one of the honeypots

Classify events

Once the XL-SIEM knows that new types of events will arrive it is required to classify the important ones. For important ones it is understood those that are relevant for the detection of potential incidents, which will be the ones to process when applying incident correlation rules. At the XL-SIEM this classification is done through EPL statements, which basically filter events given by a specific plugin_id and plugin_sid. Figure 28 shows the statement used to classify events with plugin_id=1001 and plugin_sid=2009286 as events related to the detection of scans using the Modbus protocol. Events matching these two values are classified into Scada_Modbus_scanning_detected events.

 SDN-μSense

[Dashboards](#)
[SIEM Analysis](#)
[Configuration](#)
[Reports](#)

[Correlation Bolts](#)
[EPL Directives](#)
[EPL Statements](#)
[EPL Variables](#)
[Schemas](#)
[Storm Topology Configuration](#)

Statement Name *

Scada_Modbus_scanning_detected

EPL

```
insert into Scada_Modbus_scanning_detected select
* from ossimSchema_default where (plugin_id=1001)
and (plugin_sid=2009286)
```

Update Clear Cancel

Figure 28. Example of an event classification for Modbus related events

Create directives

Next step is the definition of the directive that represents the correlation rule to apply when events classified in the previous step arrive. Here is where the logic used to trigger security alerts is applied. The events classified in previous steps are used to apply rules that uses variables such as number of occurrences, events appearing within a certain period of time, or values contained in such event. The

syntax used to define these rules is EPL¹². Figure 29 represents an example of a correlation rule that triggers a security alert about a scanning against an IP. This alert is triggered when three events (classified as Scada_Modbus_Scanning_detected) occurs within 100 seconds and against the same IP (a.dst_ip). In addition, and similar to the definition of events, categories, subcategories, reliability and priority are defined for this alert.



Directive SID *	27003_2
Name *	directive_event: SCADA attack, Modbus scanning or fingerprinting against DST_IP
EPL Pattern	pattern [every-distinct(a.dst_ip,100 seconds) a=Scada_Modbus_scanning_detected -> (b=Scada_Modbus_scanning_detected((b.src_ip=a.src_ip) and (b.dst_ip=a.dst_ip)) -> c=Scada_Modbus_scanning_detected((c.src_ip=a.src_ip) and (c.dst_ip=a.dst_ip)))]
Category	Alarm
Subcategory	Scada
Reliability *	8
Priority *	4
<input type="button" value="Update"/> <input type="button" value="Clear"/> <input type="button" value="Cancel"/>	

Figure 29. Example of directive (correlation rule) for a Modbus related incident

Add directives to correlation

Finally, the new correlation rules are added to the correlation engine (called Correlation Bolt in Storm terminology). Every correlation bolt contains the list of rules that are active, which depends on the domain or other requirements where the XL-SIEM is used. Figure 30 represents the panel where the new rule is added in the section “Directive”.

¹² https://docs.oracle.com/cd/E13157_01/wlevs/docs30/epl_guide/overview.html



The image shows the SDN-μSense Configuration Panel. At the top, there is a navigation bar with links: Dashboards, SIEM Analysis, Configuration, and Reports. Below this, there is a sub-navigation bar with tabs: Correlation Bolts, EPL Directives, EPL Statements, EPL Variables, Schemas, and Storm Topology Configuration. The main content area is a form for configuring a correlation bolt. The form has several sections:

- CorrBolt name ***: A text input field containing "test".
- Parallelism ***: A dropdown menu set to "1".
- Data Schema ***: A dropdown menu set to "ossimSchema". Below it is a link "Insert new Schema?".
- Description**: A large text area for a description.
- Categories ***: A section with a "Scan Directives" dropdown menu (showing "Scan Directives", "Malware Directives", "DoS Directives", "BruteForce Directives", "Network Directives") and an "Add" button. Below the menu is a list of selected categories with an "[X]" button to remove them.
- Directives ***: A section with a dropdown menu set to "2" and an "Add" button. Below the menu is a list of selected directives with an "[X]" button to remove them.
- Policies**: A section with a dropdown menu set to "ATOS - AVAPI filter" and an "Add" button. Below the menu is a list of selected policies with an "[X]" button to remove them.

Figure 30. Configuration panel where new directives are added to the correlation engine

This process for generating new correlation rules has been followed for every new detector and every new event received from SDN-microSENSE. The effort was required not just to create the new rules but to apply specific threat intelligence to explicitly know how to build the appropriate correlation rule. To this end it was required to understand the information contained in the events and the nature of the attacks detected, identifying patterns that need to be translated to rules and clearly identify alerts and avoid false positives.

Security alert format

The security alerts exported by the XL-SIEM uses a data format very similar to the format of the normalized events, also based on OSSIM and using json, but with several differences.

Table 14 describes the list of fields included in a security alarm. Apart from the fields that are common to events, it is worth noticing the following aspects:

- BACKLOG_ID: A unique ID used to identify the alert
- EVENT_ID : The XL-SIEM allows to use alerts as events and apply cross correlation rules using alerts as events. This field represents the ID associated to the alert when it acts as an event.

- **RELATED_EVENTS**: As it was mentioned before, security alerts are triggered by correlating events (one or more than one like in the example above). This field represents the IDs of the events that has been used to generate the alert.
- **RELATED_EVENTS_INFO**: This field contains an json array, every element containing the complete events used to correlate and generate the alert.

Table 14. Fields of the security alert message exported by the XL-SIEM

XL-SIEM	Description
DATE	Date and time of the alert.
SENSOR	Agent that sent the events used to create the alert.
PLUGIN_SID	ID assigned to identify the alert type.
EVENT_ID	Unique ID number assigned to the alert by the XL-SIEM. Used to use alerts as events and perform cross correlation rules
PROTOCOL	Protocol used for the source/destination associated to the alert, for example, TCP IP.
CATEGORY	Event taxonomy for the alert, for example, Authentication or Exploit.
SUBCATEGORY	Subcategory of the alert taxonomy type listed under Category. For example, this would be Denial of Service, if the category were Exploit.
SID_NAME	Name of the external application or device that produced the alert.
PLUGIN_ID	ID associated with the external application or device that produced the alert.
PRIORITY	Priority ranking, based on value of the alert. Each alert type has a priority value, used in risk calculation.
RELIABILITY	Indicates the level of trust on the detector that has provided the events used to generate the alert
RISK	Risk level of the event, being 0 the minimum and 10 the maximum
SRC_IP	IP address for the source of the events associated to the alert
DST_IP	IP address for the destination of the event associated to the alert
SRC_IP_HOSTNAME	Hostname of the event source.
DST_IP_HOSTNAME	Hostname of the event destination.
SRC_PORT	External or internal asset source port for the event associated to the alert
DST_PORT	External or internal asset destination port for the event associated to the alert
FILENAME	Only applicable to certain alerts.
BACKLOG_ID	Internal ID used by the XL-SIEM to identify the alert
USERNAME	Only applicable to certain alerts.
PASSWORD	Only applicable to certain alerts

ORGANIZATION	The organization where the incident has been discovered. Useful with multitenancy based deployments (one XL-SIEM instance, several organizations).
USERDATA1-9	Custom data variable depending on the alert and the sensor.
RELATED_EVENTS	Event_id of the events that have triggered the alert.
PLUGIN_NAME	The name of the agent sending the event associated to the alert
RELATED_EVENTS_INFO	json with the complete information about events that triggered the alarm (one or more events)

These alerts are stored at the XL-SIEM database and exported to the RabbitMQ defined in Section 4.3.2, and consumed by several components of the SDN-microSENSE architecture as defined in Figure 23.

7 Unit Testing

7.1 Unit Testing Environment

A testbed has been created to verify and validate the interconnection of the different detectors used in SDN-microSENSE with the XL-SIEM deployed at the XL-EPDS. This testbed has been deployed at ATOS premises and is publicly available for partners to test interconnection with XL-SIEM inputs and outputs.

The testbed architecture is shown in the following figure:

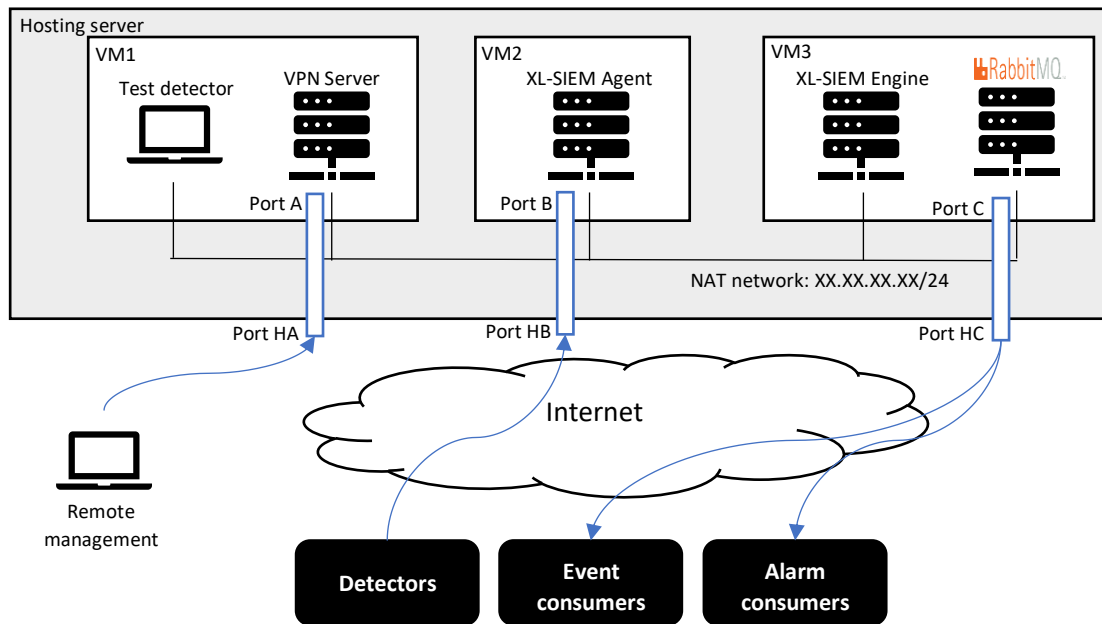


Figure 31. XL-SIEM testbed in SDN-microSENSE

The testbed has been deployed in a hosting server at Atos premises. It has been mounted by using several virtual machines that corresponds to several components of the XL-SIEM interconnected through a NAT network. The details of these virtual machines are the following:

- VM1: It contains a VPN server, that is used for the remote management of the complete infrastructure. It also contains a testing client that allows to send test logs to the XL-SIEM agent. The Port A of the VM1 is forwarded to the Port HA of the hosting server, which allows to establish a VPN connection from the public internet using Port HA.
- VM2: It contains an instance of the XL-SIEM Agent. This agent contains the rsyslog server that listens in Port B for logs coming from detectors. To do this, the Port B at the VM2 is forwarded to the Port HB, which is publicly available for detectors to send logs.
- VM3: It contains an instance of the XL-SIEM Engine containers, which receives normalized events from the XL-SIEM Agent directly through the NAT network by using a secure TCP connection directly to the VM3. This VM3 also contains a dockerized instance of a RabbitMQ server that receives events from the XL-SIEM at VM2 through the NAT network and security alerts from the XL-SIEM Engine directly within the local loopback at VM3. Consumers of events

and alerts can connect to the RabbitMQ server through the Port HC of the hosting server, which is forwarding the Port C of the RabbitMQ server at VM3.

Therefore, just ports HA, HB and HC of the hosting server are exposed to the public internet. The access to the rest of services available at the VMs of the testbed (such as SSH connections or access to web servers to manage dashboards) is just possible through the VPN server.

It is worth noticing that these ports are just used at the testbed, not being used or exposed in any deployment on the real pilots.

As shown in Figure 31, several detectors and consumers of events and alarms has been interconnected to the testbed for checking connectivity. These tests are part of the Unit Testing phase defined in T2.4 as part of the evaluation strategy depicted in Figure 32

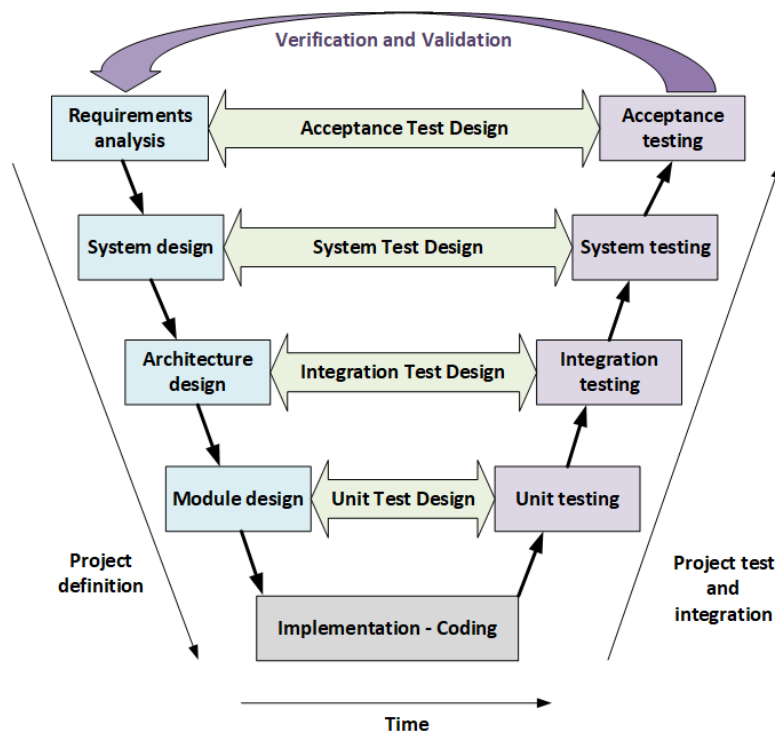


Figure 32. Evaluation strategy in SDN-microSENSE (extracted from T2.4)

7.2 Unit tests

The following tests have been carried out. It is worth noticing that all IPs has been hidden. Ports appearing in the tests are fictitious and does not correspond to any active or available service.

Test Case ID	XL-EPDS-01	Component	XL-SIEM Agent
Description	Basic connectivity between a test detector and the XL-SIEM Agent and normalization of the event		
SPEC ID	SPEC-F1, SPEC-F2	Priority	High

Prepared by	ATOS	Tested by	ATOS
Pre-condition(s)	A test detector and an XL-SIEM Agent needs to be deployed		
Test steps			
1	At the test detector a log is simulated by adding a new testing log in the file monitored by the rsyslog client: echo '[testSensor] TEST_EVENT: {src_ip=XX.XX.XX.11, port=111, dst_ip=XX.XX.XX.99, dst_port=397, desc="This is an event"}' >> /var/log/test.log		
2	The rsyslog client automatically dispatch the log to the XL-SIEM agent, which is received and stored in /var/log/test_logEvent.log root@ec0438170ba7:/var/log# tail -n 1 test_logEvent.log Jul 22 14:27:50 sdnclient testlog [testSensor] TEST_EVENT: {src_ip=XX.XX.XX.11, port=CC, dst_ip=XX.XX.XX.99, dst_port=BB, desc=" This is an event "}		
3	The log is normalized by the XL-SIEM agent: 2020-07-22 14:27:50,733 Output [INFO]: { "event": { "type": "detector", "date": "1595428070", "device": "XX.XX.XX.9", "interface": "eth0", "plugin_id": "30000", "plugin_sid": "1", "src_ip": "XX.XX.XX.11", "src_port": "BB", "dst_ip": "XX.XX.XX.99", "dst_port": "CC", "userdata1": "VEVTVF9FVkVOVA==", "userdata2": "VGhpCyBpcyBhIGFzZHMgZWZw50", "log": "SnVsIDIyIDE0OjI3OjUwIHNkbmNsaWVudCB0ZXN0bG9nIFt0ZXN0U2Vuc29yXSBUrVNUX0VRU5U0iB7c3JjX2lwPTExLjExLjExLCBwb3J0PTExMSwgZHN0X2lwPTk5Ljk5Ljk5LCBkc3RfcG9ydD0zOTcsIGRlc2M9IlRoXMgaXMGYSBhc2RzIGVhdmlvudCjIA==" }, "fdate": "2020-07-22 14:27:50", "event_id": "cc2711ea-936b-0242-ac11-000285b6525a" }		
Input data	Test log: [testSensor] TEST_EVENT: {src_ip=XX.XX.XX.11, port=BB, dst_ip=XX.XX.XX.99, dst_port=CC, desc="This is an event"}		
Result	Log correctly normalized by the XL-SIEM Agent: 2020-07-22 14:27:50,733 Output [INFO]: { "event": { "type": "detector", "date": "1595428070", "device": "XX.XX.XX.9", "interface": "eth0", "plugin_id": "30000", "plugin_sid": "1", "src_ip": "XX.XX.XX.11", "src_port": "BB", "dst_ip": "XX.XX.XX.99", "dst_port": "CC", "userdata1": "VEVTVF9FVkVOVA==", "userdata2": "VGhpCyBpcyBhIGFzZHMgZWZw50", "log": "SnVsIDIyIDE0OjI3OjUwIHNkbmNsaWVudCB0ZXN0bG9nIFt0ZXN0U2Vuc29yXSBUrVNUX0VRU5U0iB7c3JjX2lwPTExLjExLjExLCBwb3J0PTExMSwgZHN0X2lwPTk5Ljk5Ljk5LCBkc3RfcG9ydD0zOTcsIGRlc2M9IlRoXMgaXMGYSBhc2RzIGVhdmlvudCjIA==" }, "fdate": "2020-07-22 14:27:50", "event_id": "cc2711ea-936b-0242-ac11-000285b6525a" }		
Test Case Result	Achieved		

Test Case ID	XL-EPDS-02	Component	Honeypots
Description	Communication between honeypots logs and XL-SIEM		
SPEC ID	SPEC-F1, SPEC-F2	Priority	High
Prepared by	ATOS	Tested by	TECN
Pre-condition(s)	Honeypot and XL-SIEM Agent needs to be deployed		

Test steps	
1	A testing honeypot sends an example log to the XL-SIEM Agent using rsyslog at port 5200
2	The log is transmitted through the rsyslog secured channel
3	The logs is received by the XL-SIEM Agent
Input data	Log from the testing honeypot honeypot {"eventID":"0004","name":"Read operation","timestamp": "Tue Jul 21 09:38:22 2020","parameters":{"key":"ip","value":"XX.XX.XX.19:55157"}, {"key":"ln","value":"LPHD1"}, {"key":"dataObject","value":"PhyHealth"}, {"key":"fc","value":"ST"}]}
Result	Data received by the XL-SIEM agent at /var/log/syslog Jul 21 09:38:30 digitalenergypi honeypot {"eventID":"0001","name":"Connection closed","timestamp": "Tue Jul 21 09:38:30 2020","parameters":{"key":"ip","value":"XX.XX.XX.19:CC"}]}
Test Case Result	Achieved

Test Case ID	XL-EPDS-03	Component	Nightwatch
Description	Communication between Nightwatch logs and XL-SIEM		
SPEC ID	SPEC-F1, SPEC-F2	Priority	High
Prepared by	ATOS	Tested by	CLS
Pre-condition(s)	Testbed deployed		
Test steps			
1	Nightwatch sends an example log to the XL-SIEM Agent using rsyslog at port 5200		
2	The log is transmitted through the rsyslog secured channel		
3	The logs is received by the XL-SIEM Agent		
Input data	Message sent by CLS with a simple plain text message		
Result	Log received at the XL-SIEM Agent: Jun 26 12:17:46 preetika-VirtualBox mytool This is a test file from CLS.		
Test Case Result	Achieved		

Test Case ID	XL-EPDS-04	Component	UOWM detectors
--------------	------------	-----------	----------------

Description	Communication between Modbus Intrusion Detection Sensor logs and XL-SIEM		
SPEC ID	SPEC-F1, SPEC-F2	Priority	High
Prepared by	ATOS	Tested by	UOWM
Pre-condition(s)	Testbed deployed		
Test steps			
1	Modbus Intrusion Detection Sensor sends an example log to the XL-SIEM Agent using rsyslog at port 5200		
2	The log is transmitted through the rsyslog secured channel		
3	The logs is received by the XL-SIEM Agent		
Input data	Message sent by UOWM with a simple plain text message		
Result	Log received at the XL-SIEM Agent: Jun 30 16:34:06 snf-7871 uowm_test This is a test from file		
Test Case Result	Achieved		

Test Case ID	XL-EPDS-04	Component	XL-SIEM Engine
Description	Simple generation of test alert		
SPEC ID	SPEC-F3, SPEC-F4, SPEC-F5	Priority	High
Prepared by	ATOS	Tested by	ATOS
Pre-condition(s))	Log sent to the XL-SIEM agent. Correlation rules prepared at the XL-SIEM Engine		
Test steps			
1	Test XL-EPDS-01 is carried out to send an event to the XL-SIEM agent		
2	Normalized event is sent to the XL-SIEM engine from the XL-SIEM Agent		
3	Correlation rules are applied, and a security alert is triggered		
Input data	Test logs, as defined in XL-EPDS-01		
Result	Security alert issued by the XL-SIEM agent.		

	 <p>Test Case Result</p> <p>Achieved</p>
--	-----------------------------------------------------------------------------------------------------------------------------------

Test Case ID	XL-EPDS-05	Component	XL-SIEM engine (RabbitMQ)
Description	Test to check that consumers can read alerts properly from the RabbitMQ where the XL-SIEM exports alerts		
SPEC ID	SPEC-F3	Priority	High
Prepared by	ATOS	Tested by	ATOS
Pre-condition(s)	Test XL-EPDS-04 RabbitMQ deployed and configured		
Test steps			
1	Test XL-EPDS-04 is carried out to generate a simple alert		
2	Check the RabbitMQ panel that the alert is correctly pushed into the alerts queue		
Input data	Test log		
Result	Alert available at the RabbitMQ queue		

	 <p>Evidences of alert received by the RabbitMQ server from the XL-SIEM</p> <p>Message 1</p> <p>The server</p> <p>Exchange atos.exchange.alarms.sdnmsense</p> <p>Routing Key</p> <p>Redelivered</p> <p>Properties</p> <p>delivery_mode: 2</p> <p>headers</p> <p>content_encoding: UTF-8</p> <p>content_type: application/json</p> <p>Payload</p> <p>1539 bytes</p> <p>Encoding: string</p> <p>{ "AlarmEvent": { "DST_IP_HOSTNAME": "00000000", "RELATED_EVENTS": "[a1c011ea80fb0242ac11002f4a50bc0]", "DST_IP": " " } }</p>
Test Case Result	Achieved

8 Innovation Summary

The main innovations derived from the work carried out in T5.1 are related to the support of protocols commonly used by EPES, such as Modbus, DNP3, IEC 60870-5-104, IEC61850 and IEEE C37.118. With the support of these protocols the XL-SIEM spans its usage to additional domains not possible before SDN-microSENSE, such as EPES and also other domains where those protocols are commonly used (i.e., Modbus, largely used in the industry domain). More specifically, adding support of these protocols has entailed several activities such as understanding the logs generated as a result of the activities carried out using them, processing them, parsing them and interpreting the information contained. Additionally, an effort of security intelligence has been required to

- 1) identify patterns associated to the type of logs received, evaluating different criteria such as their occurrence,
- 2) identify anomalies associated to those patterns,
- 3) create correlation rules to automate the generation of security alerts upon the reception of logs matching those patterns identified as incidents
- 4) export those alerts for their usage by other components of the SDN-microSENSE platform, namely to
 - a. export anonymous threat intelligence by the ARIEC,
 - b. evaluate impact and design self-healing actions through the S-RAF, and
 - c. automatically deploy honeypots by the Honeypot Manager upon reception of zero-day attack alerts.

As it has been highlighted before, this deliverable is the first document that reports outcomes of WP5 tasks and many of the aspects described here will be extended in more detail in the following deliverables of WP5:

- Detection of attacks associated to EPES protocols is described in D5.2
- Machine learning based tools to detect attacks associated to EPES protocols are described in D5.3.
- Detection and reporting of access control activities and privacy aspects are described in D5.4.

Threat information sharing capabilities and anonymization are described in D5.5 as part of the development of the ARIEC component.

9 Conclusions

This deliverable has presented the details of the XL-EPDS module, in charge of managing the detection of cyber incidents within EPES infrastructure. Within the XL-EPDS module, this deliverable describes the core of this module, represented by the XL-SIEM which bridges the detectors deployed in the EPES infrastructure monitoring different protocols commonly used by the EPES domain.

This deliverable has also described the mechanisms developed to exporting security alerts and normalized events, which interface with several components of the SDN-microSENSE infrastructure. These mechanisms are based on messaging queues using RabbitMQ.

This is the first document reporting activities carried out in WP5, including details of the protocols being monitored by the XL-EPDS and initial highlights of the detectors developed to monitor those protocols, which are connected to the XL-SIEM to send logs for its correlation, applying new correlation rules created specifically for the detection of incidents within EPES infrastructures.

Finally, this document has also detailed the testbed deployed to test the initial connections of external components, not just WP5 components but also from other WPs, with the XL-SIEM. A set of unit tests has also been carried out and described in this deliverable.

10 References

- [Andy+17] Andy, S., Rahardjo, B., & Hanindhito, B. (2017, September). Attack scenarios and security analysis of MQTT communication protocol in IoT system. In 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI) (pp. 1-6). IEEE.
- [Banks+14] Banks, A. and Gupta, R, "MQTT version 3.1.1," OASIS Standard, 2014
- [Bedi+18] Bedi, G., Venayagamoorthy, G. K., Singh, R., Brooks, R. R., & Wang, K. C. (2018). Review of Internet of Things (IoT) in electric power and energy systems. *IEEE Internet of Things Journal*, 5(2), 847-870.
- [Cejka+16] Cejka, T., & Robledo, A. (2016). Detecting Spoofed Time in NTP Traffic. In The 4th Prague Embedded Systems Workshop (PESW2016).
- [Chamou+19] Chamou, D., Toupas, P., Ketzaki, E., Papadopoulos, S., Giannoutakis, K. M., [A., & Tzovaras, D. (2019, September). Intrusion Detection System Based on Network Traffic Using Deep Neural Networks. In 2019 IEEE 24th International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD) (pp. 1-6). IEEE.
- [Drias+2015] Drias, Z., Serhrouchni, A., & Vogel, O. (2015, July). Taxonomy of attacks on industrial control protocols. In 2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS) (pp. 1-6). IEEE
- [Errata+15] Errata, O. S. I. A. (2015). MQTT Version 3.1. 1 Plus Errata 01
- [Gartner2020] Kelly Kavanagh, Toby Bussa, Gorka Sadowski. Magic Quadrant for Security Information and Event Management. Published 18 February 2020. Available online at <https://www.gartner.com/en/documents/3981040>
- [Granadillo+19] Granadillo, Gustavo & Diaz, Rodrigo & Medeiros, Ibéria & Gonzalez-Zarzosa, Susana & Machnicki, Dawid. (2019). LADS: A Live Anomaly Detection System based on Machine Learning Methods. 10.5220/0007948904640469.
- [IEEEC37+11] IEEE Standard for Synchrophasor Data Transfer for Power Systems," in IEEE Std C37.118.2-2011 (Revision of IEEE Std C37.118-2005), vol., no., pp.1-53, 28 Dec. 2011, doi: 10.1109/IEEESTD.2011.6111222.
- [Ketzaki+19] Ketzaki, E., Drosou, A., Papadopoulos, S., & Tzovaras, D. (2019, October). A light-weighted ANN architecture for the classification of cyber-threats in modern communication networks. In 2019 10th International Conference on Networks of the Future (NoF) (pp. 17-24). IEEE.
- [Khan+16] Khan, Rafiullah, et al. "IEEE C37. 118-2 Synchrophasor Communication Framework-Overview, Cyber Vulnerabilities Analysis and Performance Evaluation." *IC/SSP*. 2016.
- [Knapp+14] Knapp, E. D., & Langill, J. T. (2014). Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems. Syngress.
- [Malhotra+16] Malhotra, A., Cohen, I. E., Brakke, E., & Goldberg, S. (2016, February). Attacking the Network Time Protocol. In NDSS.
- [Marinos+13] Marinos, L. (2013). Smart Grid threat landscape and good practice guide. White Paper, European Network and Information Security Agency (ENISA).

- [Ozgur+17] Ozgur, U., Nair, H. T., Sundararajan, A., Akkaya, K., & Sarwat, A. I. (2017, October). An efficient MQTT framework for control and protection of networked cyber-physical systems. In 2017 IEEE Conference on Communications and Network Security (CNS) (pp. 421-426). IEEE.
- [Rinaldi+16] Rinaldi, S., Della Giustina, D., Ferrari, P., Flammini, A., & Sisinni, E. (2016). Time synchronization over heterogeneous network for smart grid application: Design and characterization of a real case. *Ad Hoc Networks*, 50, 41-57.
- [SDN22] SDNmicroSENSE Deliverable D2.2. User & Stakeholder, Security and Privacy Requirements 2020
- [SDN23] SDNmicroSENSE Deliverable D2.3 Platform Specifications and Architecture. 2020
- [SDN24] SDNmicroSENSE Deliverable D2.4 Pilot, Demonstration & Evaluation Strategy. 2020
- [SDN33] SDNmicroSENSE Deliverable D3.3 EPES Honeypots. 2020
- [SDN51] SDNmicroSENSE Deliverable D5.1 XL-SIEM System. 2020
- [SDN53] SDNmicroSENSE Deliverable D5.3 ADS and CLS Discøvery Systems. 2020
- [SDN54] SDNmicroSENSE Deliverable D5.4. Overlay Privacy Framework. 2020
- [SDN55] SDNmicroSENSE Deliverable D5.4. Cloud-based Anonymous Repository of Incidents. 2020
- [Toupas+19] Toupas, P., Chamou, D., Giannoutakis, K. M., Drosou, A., & Tzovaras, D. (2019, December). An Intrusion Detection System for Multi-class Classification Based on Deep Neural Networks. In 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA) (pp. 1253-1258). IEEE.
- [UOWM1+20] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, "A survey on scada systems: Secure protocols, incidents, threats and tactics," *IEEE Communications Surveys & Tutorials*, 2020.
- [UOWM2+20] O. Igbe, I. Darwish and T. Saadawi, "Deterministic Dendritic Cell Algorithm Application to Smart Grid Cyber-Attack Detection", 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), 2017. Available: 10.1109/csccloud.2017.12 [Accessed 15 July 2020].
- [Yasakethu +13] Yasakethu, S. L. P., & Jiang, J. (2013, September). Intrusion detection via machine learning for SCADA system protection. In 1st International Symposium for ICS & SCADA Cyber Security Research 2013 (ICS-CSR 2013) 1 (pp. 101-105).