

RESISTO:

D2.7 RESISTO platform and tools reference architecture - final



RESISTO

D2.7 – RESISTO PLATFORM AND TOOLS REFERENCE ARCHITECTURE - FINAL

Document Manager:	Alberto Neri	LDO	Editor
--------------------------	--------------	-----	--------

Project Title:	RESilience enhancement and risk control platform for communication infraSTructure Operators
Project Acronym:	RESISTO
Contract Number:	786409
Project Coordinator:	LEONARDO
WP Leader:	BTC

Document ID N°:	RESISTO_D2.7_191218_01	Version:	1.0
Deliverable:	D2.7	Date:	18/12/2019
		Status:	APPROVED

Document classification	Public
--------------------------------	---------------

Approval Status	
Prepared by:	Alberto Neri (LDO)
Approved by: (WP Leader)	Zhan Cui (BTC)
Approved by: (Coordinator)	Bruno SACCOMANNO (LDO)
Advisory Board Validation (Advisory Board Coordinator)	N.A.
Security Approval (Security Advisory Board Leader)	Paolo DI MICHELE (LDO)

CONTRIBUTING PARTNERS

Name	Company / Organization	Role / Title
Alberto Neri	LDO	RESISTO Task 2.4 leader, LDO RESISTO Technical Coordinator
Emanuele Aonzo	LDO	Contributor
Lucio Brighella	LDO	Contributor
Fabio Cirinnà	LDO	Contributor
Marco Carli	RM3	RESISTO WP5 leader / Associate Professor
Stefano Panzieri	RM3	RESISTO Innovation Manager / Full Professor
Giuseppe Celozzi	TEI	RESISTO WP4 leader
Panagiotis Karaivazoglou	ICCS	Contributor / Scientific Researcher
Michael A. Skitsas	ADI	Contributor
Javier Valera	INT	Contributor
Gael Haab	EMI	Contributor / Scientific Researcher
Risto Laanoja	GT	Contributor
Andrei Avădănei	BSS	Contributor
Ioan Constantin	ORO	Contributor
Evangelos Sfakianakis	OTE	Contributor
Jorge Carapinha	ALB	Contributor
Sylvia Bach	BUW	Contributor / Scientific Researcher
Zhan Cui	BTC	RESISTO WP2 leader

DISTRIBUTION LIST

Name	Company / Organization	Role / Title
PMT	RESISTO CONSORTIUM	NA
Markus MULLER	EC DG REA	EC Programme Officer
General Public	NA	NA

REVISION TABLE

Version	Date	Modified Pages	Modified Sections	Comments
0.9	08/11/2019	All	All	Submitted to SAB Review
1.0	18/12/2019	All	All	Final release

COPYRIGHT STATEMENT



© 2018-2021 This document and its content are the property of the RESISTO Consortium. All rights relevant to this document are determined by the applicable laws. Access to this document does not grant any right or license on the document or its contents. This document or its contents are not to be used or treated in any manner inconsistent with the rights or interests of the RESISTO Consortium or the Partners detriment and are not to be disclosed externally without prior written consent from the RESISTO Partners. Each RESISTO Partner may use this document in conformity with the RESISTO Consortium Grant Agreement provisions.

The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Program, under the Grant Agreement No 786409.

The views and opinions in this document are solely those of the authors and contributors, not those of the European Commission.

PROJECT CONTACT



LEONARDO

Via Puccini 2 – Genova (GE) – 16154 – Italy

Tel.: +39 348 6505565

E-Mail: bruno.saccomanno@leonardocompany.com

RESISTO PROJECT – PUBLISHABLE EXTENDED ABSTRACT

Communications play a fundamental role in the economic and social well-being of the citizens and on operations of most of the CIs. Thus they are a primary target for criminals having a multiplier effect on the power of attacks and providing enormous resonance and gains. Also extreme weather events and natural disasters represents a challenge due to their increase in frequency and intensity requiring smarter resilience of the Communication CIs, which are extremely vulnerable due to the ever-increasing complexity of the architecture also in light of the evolution towards 5G, the extensive use of programmable platforms and exponential growth of connected devices. The fact that most enterprises still manage physical and cyber security independently represents a further challenge. RESISTO platform is an innovative solution for Communication CIs holistic situation awareness and enhanced resilience (aligned with ECSO objectives). Based on an Integrated Risk and Resilience analysis management and improvement process availing all resilience cycle phases (prepare, prevent, detect, absorb, etc.) and technical resilience capabilities (sense, model, infer, act, adopt), RESISTO implements an innovative Decision Support System to protect communication infrastructures from combined cyber-physical threats exploiting the Software Defined Security model on a suite of state of the art cyber/physical security components (Blockchain, Machine Learning, IoT security, Airborne threat detection, holistic audio-video analytics) and services (Responsible Disclosure Framework) for detection and reaction in presence of attacks or natural disasters. Through RESISTO Communications Operators, will be able to implement a set of mitigation actions and countermeasures that significantly reduce the impact of negative events in terms of performance losses, social consequences, and cascading effects in particular by bouncing efficiently back to original and forward to operational states of operation.

EXECUTIVE SUMMARY

This document, D2.7 “RESISTO platform and tools reference architecture - final”, relevant to Task 2.4, illustrates the reference architecture of the RESISTO platform.

The RESISTO reference architecture has been defined according to a two-step process, object of Task 2.4.

The aim of the first step has been to fix the main system functional flows and to take a snapshot of the system components, so to identify their role and the services/interfaces they mean to provide. Deliverable 2.6 ([4]) represents exactly this first step.

Task 6.1, Software architecture definition ([12]), has been an intermediate step with the aim to analyze the RESISTO platform from a software point of view by specifying the overall static and behavioural structure in terms of software components: their functions, internal and external interfaces intended as means to exchange data as well as data contents definition.

A final step of architecture definition is the present document, D2.7. The relationship among the system components is defined to properly drive component development and adaption actions as well as the platform integration phase. The complete definition has been performed in parallel with Task 2.5, “Operational Use Case and Validation Plan”, in order to have a complete definition of operators testbeds (Critical Infrastructure to be protected).

From a methodological point of view, as well as D2.6, the document first provides an overview of the RESISTO platform, next describes and refines the RESISTO architecture detailing the sub-systems and components. Functional modules (e.g., the Risk Predictor, the Workflow Manager, and the Orchestration Controller) and infrastructure elements (e.g., the Data Integration Layer and the Cockpit) are both presented and assessed. Section “Concept of execution” aims to capture the dynamic behavior of the system. Finally services provided and requested by each component are described.

CONTENTS

ABBREVIATIONS	12
1. INTRODUCTION	16
1.1. Document Identification	16
1.2. Document Overview	16
2. SYSTEM ARCHITECTURE	17
2.1. Long Term Control Loop Functional Architecture	18
2.2. Short Term Control Loop Functional Architecture	21
2.3. Long and Short Term Control Loops interaction	23
2.4. Architecture Components	25
3. DETAILED DESCRIPTION OF ARCHITECTURE COMPONENTS	27
3.1. Long Term Control Loop components	27
3.2. Short Term Control Loop software components	28
3.2.1. Data Integration Layer	28
3.2.2. Cyber/Physical Events Correlator	28
3.2.2.1. Correlator Engine	31
3.2.2.2. Machine Learning Module	33
3.2.2.3. Interconnection Layer	34
3.2.3. Risk (Impact) Predictor	34
3.2.3.1. CISIApro 2.0 – Dynamic Risk Analysis Tool	35
3.2.3.2. Layers & Resources Module	38
3.2.3.3. Entity maker Module	39
3.2.3.4. Modeler Module	40
3.2.3.5. State variables Module	41
3.2.3.6. Link State Module	42
3.2.3.7. CISIAmat – the CISIApro 2.0 on-line engine	42
3.2.4. Workflow Manager	44
3.2.5. Orchestration Controller	44
3.2.6. Emergency Warning Communication Function	45
3.3. Cockpit	46
3.3.1. RESISTO HMI layout	46
3.3.2. HMI features – Panel management	47
3.3.3. HMI features – Reporting	50
3.4. Platform Common Services	52
3.4.1. IAM	52
3.4.2. KSI Blockchain Infrastructure	52
3.4.3. Knowledge Base	53
3.4.4. Guardtime Machine Integrity	54
3.4.4.1. XDAL and Dockets	54
3.4.4.2. Guardtime MIDA	54
3.4.4.3. Why KSI Blockchain	55
3.4.5. Responsible Disclosure Framework	56
3.5. RESISTO detectors	62
3.5.1. Airborne Threats Detector	62
3.5.2. Audio/Video analysis	63
3.5.3. Smart Spectrum Surveillance	66
3.5.3.1. RADIOFILTER	66

3.5.3.2. RANMONITOR	67
3.5.4. IoT Based Sensors	68
3.5.5. OSINT and the Machine Learning based Threat Detector	70
3.6. Telecommunication Operators Infrastructures	71
3.6.1. Altice Labs test bed	71
3.6.1.1. Altice Labs testbed interfaces	73
3.6.2. Orange Romania testbed	73
3.6.2.1. Orange Romania testbed interfaces	76
3.6.3. OTE testbed	77
3.6.4. British Telecom testbed	79
4. CONCEPT OF EXECUTION	82
4.1. Short Term Control Loop operation flow	82
4.2. Continuous improvement of the overall resilience	84
5. COMMUNICATION MODES, MESSAGE PATTERNS AND FORMATS	86
5.1. Communication modes	86
5.1.1. Data Integration Layer: Message broker communication modes	86
5.1.2. Message broker technologies	88
5.1.3. Design Choice of Message Broker Communication Mode and Technology	88
5.1.4. REST services paradigm	90
5.2. Message formats for external interfaces	90
5.2.1. Design Choice of Message Serialization Format	90
5.2.2. Message Exchange Formats	90
5.2.3. Design Choice of Message Exchange Format	93
5.3. Secure RESISTO communication	95
6. RESISTO PLATFORM INTERFACES IDENTIFICATION	96
7. System Requirements Traceability	98
8. Conclusions	102
REFERENCES	103

INDEX OF FIGURES

Figure 1 - RESISTO logical architecture	17
Figure 2 - Long Term Control Loop Functional Architecture	18
Figure 3 - Risk and resilience management process	20
Figure 4 - Storage format of RIs in the Knowledge Base of the RESISTO platform	20
Figure 5 - Short Term Control Loop Functional Architecture	21
Figure 6 - RIs flow, step 1: RIs estimation	23
Figure 7 - RIs flow, step 2: RIs measurement	24
Figure 8 - RIs flow, step 3: Estimated vs. Measured RIs comparison	24
Figure 9 - RESISTO architecture components	25
Figure 10 - Correlator components	29
Figure 11 - Raw data from a SCADA source	29
Figure 12 - Correlator main data flow	30
Figure 13 - Example of a correlation rule	30
Figure 14 - Example of correlation	31
Figure 15 - Logical Architecture of the Cyber/Physical Events Correlator	31
Figure 16 - Apache Storm: Topology Nodes	32
Figure 17 - Apache Flume architecture	34
Figure 18 - Risk Predictor Architecture	35
Figure 19 - CISIApro DB structure	36
Figure 20 - CISIApro SQL output data structure	37
Figure 21 - CISIApro user interface	37
Figure 22 - CISIApro Module: Layers & Resources	38
Figure 23 - CISIApro Module: Entity Maker	39
Figure 24 - CISIApro Module: Modeler	40
Figure 25 - CISIApro Module: State Variables	41
Figure 26 - CISIApro Module: Link State	42
Figure 27 - URANIUM platform civil protection panel	43
Figure 28 - ATENA platform	44
Figure 29 - Cockpit HMI layout with three RESISTO monitors	46
Figure 30 - Cockpit HMI layout Screen 2	47
Figure 31 - Alarm details list	48
Figure 32 - Panels	48
Figure 33 - Types of Panels	48
Figure 34 - Three-screen HMI	49
Figure 35 - Example: Two panels, GIS and CISIApro for one screen	50
Figure 36 - KSI Infrastructure layers	53
Figure 37 - Deployment model for Knowledge Base	54
Figure 38 - Guardtime MIDA	55
Figure 39 - Responsible Disclosure Framework block diagram	57
Figure 40 - Responsible Disclosure Framework: Company Manager workflow	58
Figure 41 - Responsible Disclosure Framework: Company Teams workflow	59
Figure 42 - Responsible Disclosure Framework: Security Researcher workflow	60
Figure 43 - Responsible Disclosure Framework: Programs and Disclosures	61
Figure 44 - Audio Analytics System	64
Figure 45 - Video Analytics System	65
Figure 46 - Intelligence Audio/Video Surveillance System	65
Figure 47 - RADIOFILTER Architecture	67
Figure 48 - RANMONITOR Architecture	68
Figure 49 - IoT Based Secure Sensors Architecture	69
Figure 50 - OSINT based threat detector architecture	70
Figure 51 - ALB testbed physical configuration for 5G use case	71
Figure 52 - 5G testbed functional architecture and building blocks	72
Figure 53 - Physical layout of ORO testbed	74
Figure 54 - H2020 Virtual Infrastructure	75

Figure 55 - RESISTO Components Deployment in ORO's Testbed	75
Figure 56 - syslog adapte	77
Figure 57 - OTE Core Lab description	78
Figure 58 - Generic architecture of OTE's Openstack testbed	79
Figure 59 - BT testbed architecture for multicast and unicast video delivery	79
Figure 60 - UK network infrastructure for unicast and multicast services	81
Figure 61 - Short Term Control Loop - Decision Support phase - Event	82
Figure 62 - Short Term Control Loop - Decision Support phase - Alarm	83
Figure 63 - Short Term Control Loop - Reaction phase	84
Figure 64 - High Level Control Loop	85
Figure 65 - Publish/Subscribe interaction mode	87
Figure 66 - Point-to-Point interaction mode	87
Figure 67 - Topic-based Publish/Subscribe mode	89
Figure 68 - MQTT Technology integrated with Kafka [18]	90
Figure 69 - IDMEF Message Structure	91
Figure 70 - STIX Relationship Example	92
Figure 71 - IDEA message event categories	94
Figure 72 - RESISTO component diagram	96

INDEX OF TABLES

Table 1 - Deliverables of WP3, defining the long term control loop	19
Table 2 - RESISTO components and corresponding paragraphs	26
Table 3 - Input and output interfaces for the audio/video analytics component	66
Table 4 - SONATA orchestrator Northbound API	73
Table 5 - ORO testbed components	76
Table 6 - syslog messages specification	77
Table 7 - System Requirements vs. architecture components traceability	101
Table 8 - Reference Table	104

ABBREVIATIONS

2G, 3G, 4G, 5G	Second, third, fourth and fifth generation of mobile phone systems
AAC	Audio Analytics Component
AI	Artificial Intelligence
AP	Access Point
API	Application Programming Interface
APT	Advanced Persistent Threat
APN	Access Point Name
BLE	Bluetooth Low Energy
BT	British Telecom
CA	Certification Authority
CDN	Content Distribution Network
CEP	Complex Event Processing
CESNET	Association of universities of the Czech Republic and the Czech Academy of Sciences
CI	Critical Infrastructure
CIRCL	Computer Incident Response Center Luxembourg
CISIA	Critical Infrastructure Simulation by Interdependent Agents
CPN	Central Processing Node
CVE	Common Vulnerabilities and Exposures
DLP	Data Loss Prevention
DoS	Denial of Service
DSLAM	Digital Subscriber Line Access Multiplexers
EPL	Event Processing Language
ETSI	European Telecommunications Standard Institute
EU	European Union
EWCF	Emergency Warning Communication Function
GIS	Geographical Information System
GPU	Graphics Processing Unit

GUI	Graphical User Interface
HMI	Human Machine Interface
HTTP	HyperText Transfer Protocol
HTTPS	HTTP over SSL
HW	HardWare
IAM	Identification and Authentication Management
IDEA	Intrusion Detection Extensible Alert
IDS	Intrusion Detection System
IGMP	Internet Group Management Protocol
IP	Internet Protocol
IPS	Intrusion Prevention System
IMSI	International Mobile Subscriber Identity
IoT	Internet of Things
IPTV	Internet Protocol Television
KB	Knowledge Base
KPI	Key Performance Indicator
KSI	Keyless Signature Infrastructure
IVRE	Instrument de veille sur les réseaux extérieurs (Dynamic Recon of Unknown Networks)
L2S	Layer 2 Switch
LAN	Local Area Network
LDP	Label Distribution Protocol
LTE	Long Term Evolution (= 4G)
LTCL	Long Term Control Loop
MAC	Medium Access Control
MANO	Management and Orchestration
MAR	Multicast Access Router
MHR	Mixed-Holistic-Reductionist
MIDA	Machine Integrity, Defense and Awareness
MISP	Malware Information Sharing Platform

ML	Machine Learning
MNO	Mobile Network Operator
MOM	Message Oriented Middleware
MPLS	Multiprotocol Label Switching
MQTT	Message Queue Telemetry Transport
MSC	Multi Service Core
MSE	Multicast Service Edge
NFV	Network Function Virtualization
OAI	Open Air Interface
OLT	Optical Line Terminal
ORO	Orange Romania SA
OSINT	Open Source Intelligence
OWASP	Open Web Application Security Project
PC	Personal Computer
PDN	Public Data Network
PHP	Hypertext Preprocessor
PON	Passive Optical Network
PoP	Point of Presence
PSIM	Physical Security Information Management
PTZ	Pan Tilt Zoom
QoS	Quality of Service
RAN	Radio Access Network
REST	Representational State Transfer
RI	Resilience Indicator
RSA	Rivest, Shamir, Adleman cryptography algorithm
SC2	Smart City & Security Center
SDN	Software Defined Network
SDR	Software Defined Radio
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol

SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
STB	Set Top Boxes
STCL	Short Term Control Loop
SW	SoftWare
UE	User Equipment
UAV	Unmanned Aerial Vehicles
USB	Universal Serial Bus
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TV	Television
VAC	Video Analytics Component
VLAN	Virtual LAN
VM	Virtual Machine
VPN	Virtual Private Network
WP	Work Package
WSN	Wireless Sensor Network
XDAL	eXtensible Data Attribution Language
XML	eXtensible Markup Language

1. INTRODUCTION

1.1. Document Identification

The present deliverable, D2.7 - “RESISTO platform and tools reference architecture – final”, aims to present the outcomes of Task 2.4 “RESISTO reference architecture for long term preparation and short term disruptions”.

1.2. Document Overview

The deliverable proceeds as follows:

- Section 2 presents an overview of the RESISTO platform architecture;
- Section 3 illustrates in detail the components of the RESISTO platform;
- Section 4 describes some representative use cases through a sequence diagram representation to explain the concept of execution among the components composing the RESISTO platform;
- Section 5 presents the communication infrastructure that provides the integration between all the entities, internal and external, of the RESISTO platform;
- Section 6 reports the interfaces identified in the RESISTO system;
- Section 7 reports the traceability of system requirements vs. RESISTO components;
- Section 8 presents the conclusion of the deliverable;
- Section 0 completes the document reporting the references.

2. SYSTEM ARCHITECTURE

The logical architecture of RESISTO is depicted in Figure 1. The platform integrates two control loops both running on top of the Communication Infrastructure and interlinked with each other.

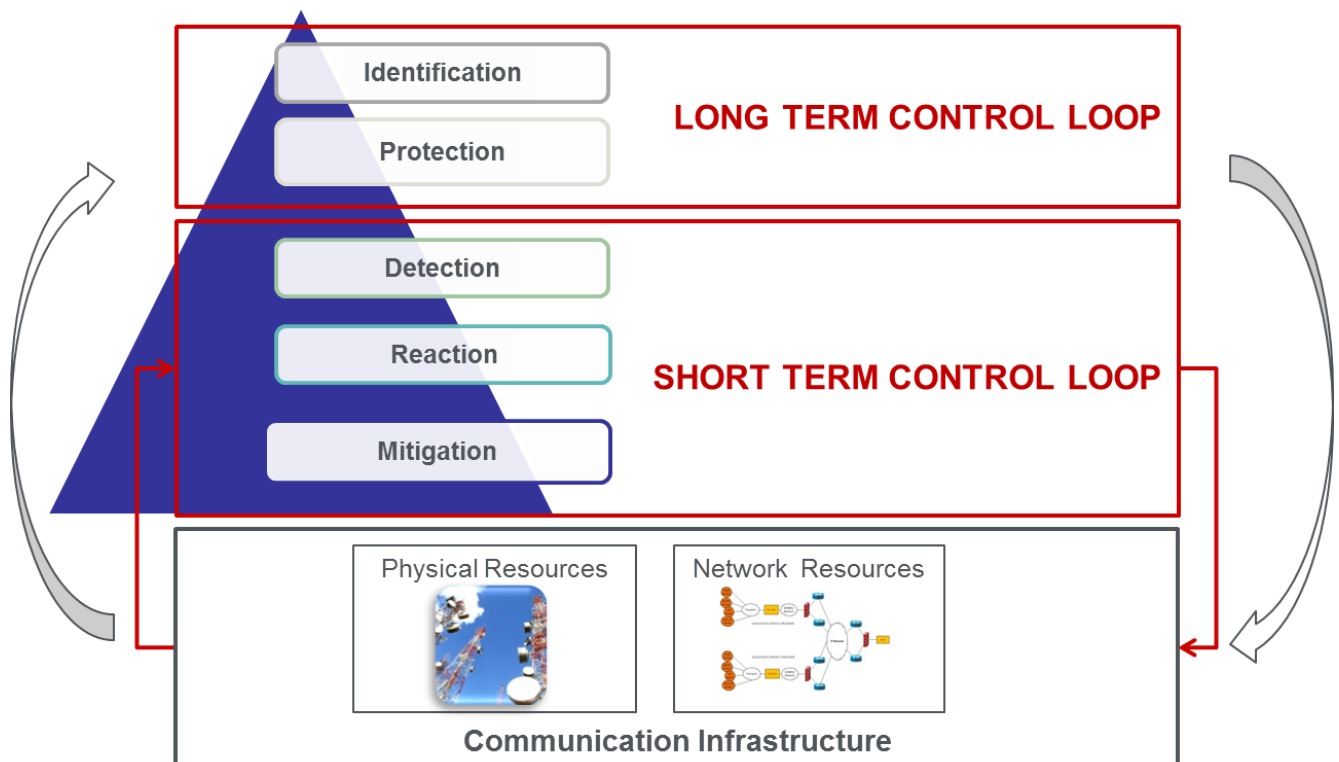


Figure 1 - RESISTO logical architecture

The **Long Term Control Loop** (LTCL) is an offline activity, following a well-defined methodology and supported by advanced tools, aimed to assess infrastructure vulnerabilities and cyber and physical security threats and consequently to define assets configuration and interventions in order to improve Critical Infrastructure (CI) resilience and robustness. For each loop cycle a set of Resilience Indicators (RIs), relevant to critical threat event typologies, are estimated and stored in a Knowledge Base (KB).

A LTCL cycle is performed on a periodic basis or when particular events take place (new threats or discovery of previously undetected vulnerabilities). It is typically conducted annually, quarterly or even monthly,

The **Short Term Control Loop** (STCL) is the runtime component of the platform. It promptly reacts to detected cyber/physical attacks and events that may impact the operational life of the system. It enhances situation awareness and provides operators with a Decision Support System cockpit able to implement the best reactions to an identified adverse event with the aim of mitigating the event's effects and restoring standard operating conditions. Facing adverse cyber/physical events, some actual RIs values are measured and stored in the KB.

Moreover, during a LTCL cycle, the comparison between RIs estimated at previous cycles with actual RIs measured by the STCL facing run-time threat events, establishes a higher level global control loop able to continuously improve infrastructure resilience and methods (see, for example, Figure 64).

2.1. Long Term Control Loop Functional Architecture

The RESISTO Long Term Control loop (LTCL) is in charge of:

- defining the configuration of the Communication Critical Infrastructure (CI) according to the security assessment, and
- updating it on a periodic basis, in the order of months, or when particular events take place (new threats or discovery of previously undetected vulnerabilities).

The functional architecture of the LTCL is depicted in Figure 2. It substantially performs a “Risk and resilience assessment analysis”.

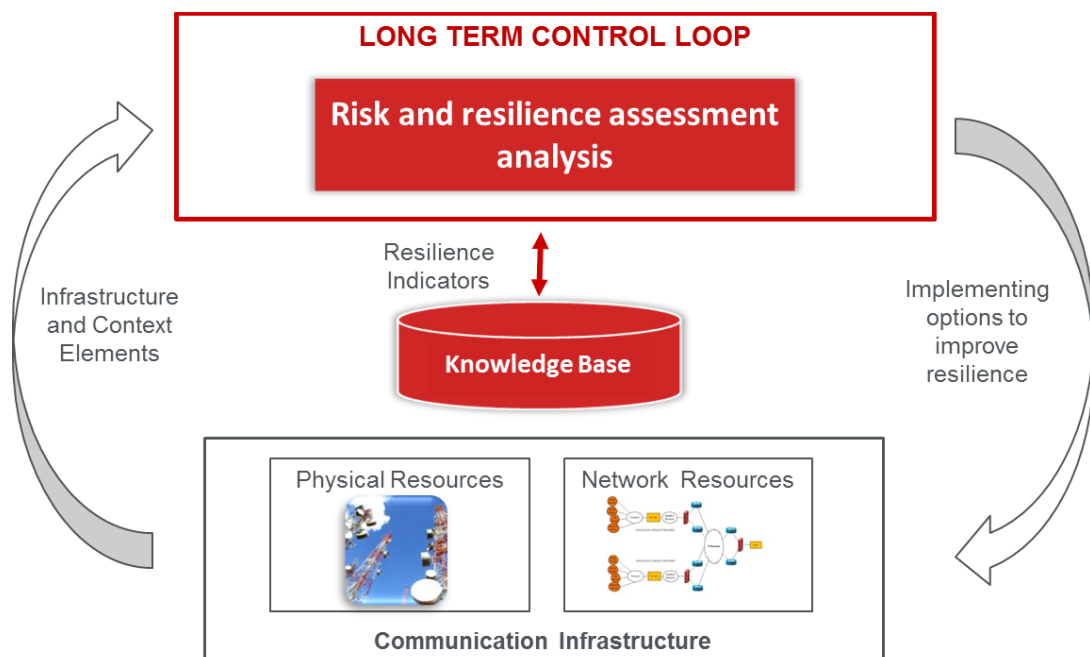


Figure 2 - Long Term Control Loop Functional Architecture

The definition of the LTCL is the main objective of WP3. Therefore, more detailed descriptions of the LTCL and its tools will be provided in the deliverables of WP3, summarized in Table 1.

Task	Deliverable	Description
Task 3.1 Long term learning cyber-physical risk and resilience management	D3.1	Risk and resilience management process for cyber-physical threats of telecom CI - first
	D3.2	Risk and resilience management process for cyber-physical threats of telecom CI - final
Task 3.2 Methods/Plans for joint cyber-physical security management process	D3.3	Methods for cyberphysical security management for telecom CI - first
	D3.4	Methods for cyberphysical security management for telecom CI- final
Task 3.3 Physical protection and prevention methods: assessment and cyber-physical interaction	D3.5	Damage/ Vulnerability models for physical and cyber threats of telecom CI - first
	D3.6	Damage/ Vulnerability models for physical and cyber threats of telecom CI - final
Task 3.4 Risk and resilience quantities and related KPIs for telecommunications infrastructure	D3.7	KPIs, quantities and metrics for cyberphysical risk and resilience of telecom CI - first
	D3.8	KPIs, quantities and metrics for cyberphysical risk and resilience of telecom CI - final
Task 3.5 Desktop application to use case scenarios for second use cases refinement	D3.9	Analytical security assessment application to use cases and their refinement

Table 1 - Deliverables of WP3, defining the long term control loop

The LTCL is based on the risk and resilience analysis management process, which shall help to identify and evaluate risks and suggest treatment and mitigation strategies.

While the short term loop provides tools for direct reaction against attacks in real-time, the long term loop should be conducted on a periodic basis or in case of specific events, e.g. after changes in the setup of the infrastructure or detection of a new type of threats. As a result the long term loop leads to the identification of criticalities and definition of long term strategies.

The implementation of the long term control loop is based on a sophisticated risk and resilience management process extending the ISO 31000 standard (see [5] section 2). The process is structured into nine sequential steps to be followed in a closed loop, as shown in Figure 3.

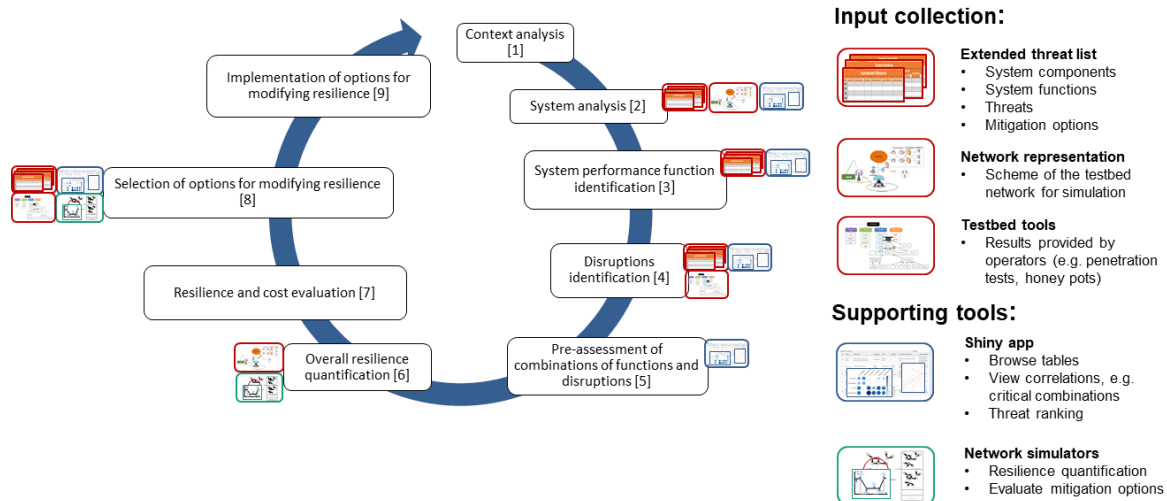


Figure 3 - Risk and resilience management process

Specific input about the system, and possible threats and counter-measures are needed at different execution steps of the loop. A tabular Excel template collects this information. A complete description from the collected information can be found in [7] section 2.3. A web-application is developed to allow a fast and facile browsing through the tables and to infer further information, such as critical combinations of system functions and threats and a ranking of the threats, which serve as additional input to other steps of the resilience management process. This method allows a semi-quantitative assessment of critical risks. An in-depth description of this tool is provided in [6] section 5.

The resilience Matrix quantification of the critical risks is further supported by infrastructure simulation approaches, which can be performed by two simulation tools: the platform-integrated simulator CISIApro of the short-term control loop (see [7] section 5.2 and par. 3.2.3 of this document) and the offline simulator CaESAR (see [7] section 5.1). In addition, the simulators allow to assess and rank possible mitigation strategies. It should be noted, that these simulations are not performed directly at the response to an attack but rather on a periodic basis to identify weak points of the current setup of the infrastructure. The results of the simulation processes will be summarized tabularly (see Figure 4) and integrated as output in the web-application.

	Event<1>	...	Event<i>	...	Event<M>
Funct<1>					
...					
Funct<j>			Estimated RI(1;i;j) RI(2;i;j) RI(3;i;j)		
...					
Funct<N>					

Figure 4 - Storage format of RIs in the Knowledge Base of the RESISTO platform

2.2. Short Term Control Loop Functional Architecture

The RESISTO Short Term Control Loop (STCL) is a typical run-time control loop. It is in charge of detecting potential physical, cyber and physical/cyber combined threat events that may impact the operational life of the system and react promptly.

The Short Term Control Loop:

- monitors the physical and cyber security status of the infrastructures, correlating the physical and cyber domain events and monitoring communication infrastructure data in order to collect and/or detect anomalies and provide early warnings on security attacks or events adversely impacting security;
- evaluates the event impact with respect to performance degradation of detected anomalies and security attacks on the communication CI and interlinked CIs, if known, based on the cascading effect;
- supports decision making providing a qualitative and quantitative What-If analysis tool in order to evaluate the best communication CI reconfiguration;
- drives reaction and mitigation by means of action workflows (composed of directives to intervention teams, physical protection devices activation) and, mainly, of orchestrated Communication Network reconfiguration and protection function activation.

The STCL functional control flow is reported in Figure 5.

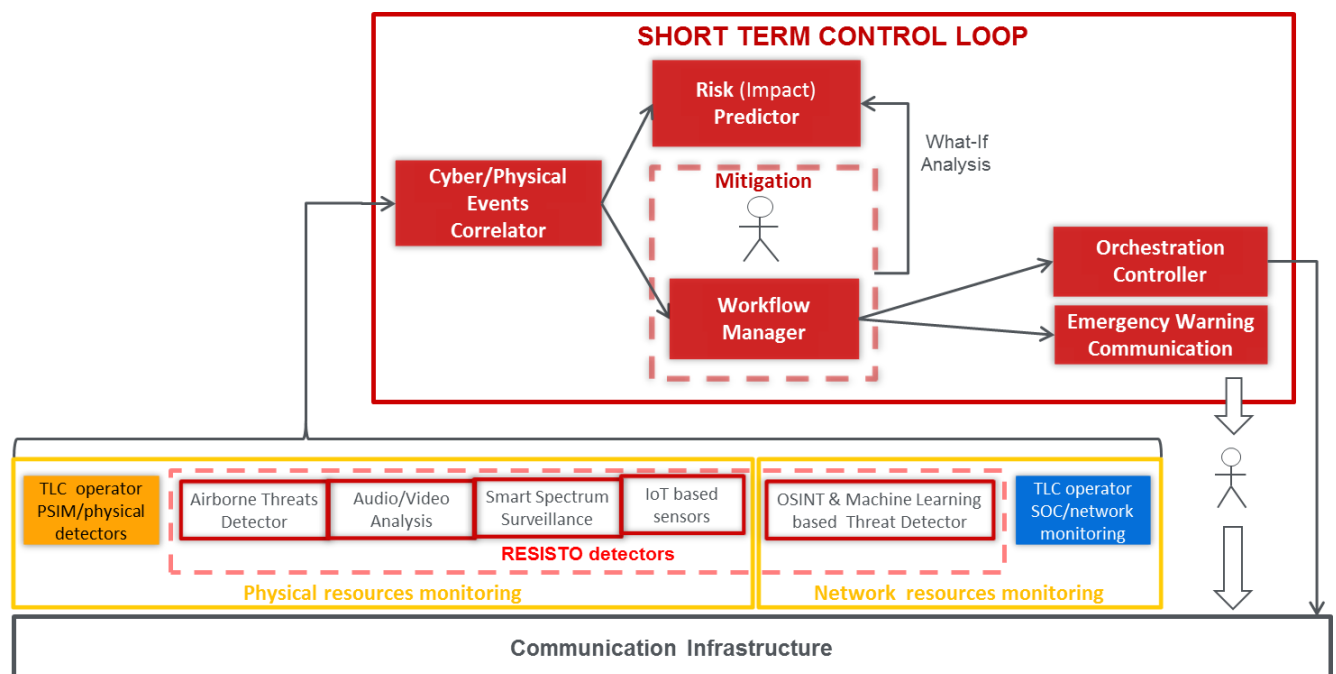


Figure 5 - Short Term Control Loop Functional Architecture

Input data to the STCL can be grouped into the following categories:

- 1) physical events (e.g., intrusions, damage) or potentially dangerous events (i.e. unauthorized UAVs);
- 2) cyber-attacks;
- 3) physical communication infrastructure monitoring data (e.g., power and energy usage information and faults);
- 4) communication network monitoring data (e.g., traffic, alarms, faults).

The sources of such data and information could be:

- legacy PSIMs (Physical Security Information Management) system(s) or other detectors made available by the TLC operator,
- legacy SOCs (Security Operating Centers) or other cyber-attack detectors made available by the TLC operator,
- RESISTO additional physical threat detectors (e.g., airborne threats detection systems, smart spectrum surveillance),
- RESISTO additional cyber threat detectors such as OSINT (Open-Source Intelligence)-based detectors.

From a functional point of view, input data are collected by the **Cyber/Physical Events Correlator**, a rule-based engine applying customized rules. The Correlator not only propagates, as alarms, externally detected and collected attack/anomaly events but it also generates alarms on its own from apparently harmless events and monitoring data. This latter action is performed by using several event correlation techniques, such as logical, causal and temporal correlation based on event time. The Correlator is also equipped with a Machine Learning (ML) based module. ML algorithms can be applied to communication monitoring data with the aim of providing parameters and thresholds to the rule based engine. For a more complete description of the Cyber/Physical Events Correlator see par. 3.2.2.

Anomalies detected by the Correlator trigger the **Risk (Impact) Predictor**. The Risk Predictor evaluates and highlights the impacts of the detected anomaly on the communication infrastructure and, mainly, on the services provided by the infrastructure. The Risk Predictor Engine acts at run-time on a model of the Communication Infrastructure.

The Communication Infrastructure is modelled according to different off-line interlacing points of view:

- Under a reductionist perspective, each infrastructure is decomposed into a web of interconnected physical elementary entities and their behaviour depends on the (mutual or not) interactions with the other reductionist elements;
- Applying a holistic approach, each infrastructure is modelled as a (logical) reality with its own identity, functional properties and recognizable boundaries. It interacts with other similar entities according to reduced identifiable set of relationships. With such a perspective it is easy to identify the roles that each infrastructure plays in a specific context.
- From a Service point of view, a Service Entity represents a logical element, organizational or real, that provides an aggregate resource such as a QoS (Quality of Service) level.

Moreover the Risk Predictor supports the decision making process allowing a “What-If analysis” and thus simulating the application of countermeasures and reconfiguration and their impact on system resilience. For a more complete description of the Risk (Impact) Predictor see par. 3.2.3.

In parallel with the Risk (Impact) Predictor, the Correlator also triggers the **Workflow Management** software engine in charge to guide the operator during the reaction phase. On the basis of the alarm type, the most appropriate workflow is selected and executed. A workflow is a conditional sequence of steps. Each step can specify a procedural action (e.g., alert a reaction team with an emergency message sent through the EWCF, drive a physical actuator (e.g., lock a physical gate), carry out a complex action on the Communication Network (e.g., activate a Virtual Network Security Function, isolate a faulty or attacked component, reconfigure a part of the network, disable a 5G slice, etc.). For a more complete description of the Workflow Management see par. 3.2.4.

Complex actions on the Communication Infrastructure are performed by the **Orchestration Controller**. The Orchestration Controller is built around the concept of Software Defined Security

(SDS) taking advantage of the Network Function Virtualization (NFV) and Software Defined Networking (SDN) paradigms of the underlying communication network. The Orchestrator Controller implements complex security functions and services composing less complex/primitive security mechanisms/functions acting on physical resources (i.e. network physical equipment) as well as on Virtual Network Functions in a NFV/5G perspective. The Orchestration Controller operates on a communication infrastructure already controlled by a telecommunication operator, so it works on top of a simple SDN Controller, on the northbound side, or on top of a more complex network Operational Support System (OSS). For a more complete description of the Orchestration Controller see par. 3.2.5.

The **Emergency Warning Communication** (EWC) function is activated when there is a need for sending instant messages, targeted alerts and operating instructions to specific categories of users that are present in a certain area where events like natural disasters, physical or cyber-attacks are occurring. In particular, rescue teams called to execute actions on the infrastructure can leverage on the received information. For a more complete description of the Emergency Warning Communication see par. 3.2.6.

2.3. Long and Short Term Control Loops interaction

As already explained in section 5.3 of [7], Long and Short Term Control Loops interact with each other by means of Resilience Indicators (RIs). The interaction are explained in 3 steps.

Step 1: RIs estimation

In the last LTCL steps:

- characterize (quantify) CI «as is» resilience (step 6);
- identify the most critical couples (function; (threatening)event) showing RIs not in line with required SLA (step 7);
- select interventions on CI in order to improve resilience for most critical couples (function; event) estimating RIs in the new «to be» configuration (step 8);
- implement interventions (step 9).

So, at the end of each LTCL cycle Estimated RIs are stored in KB as shown in Figure 6.

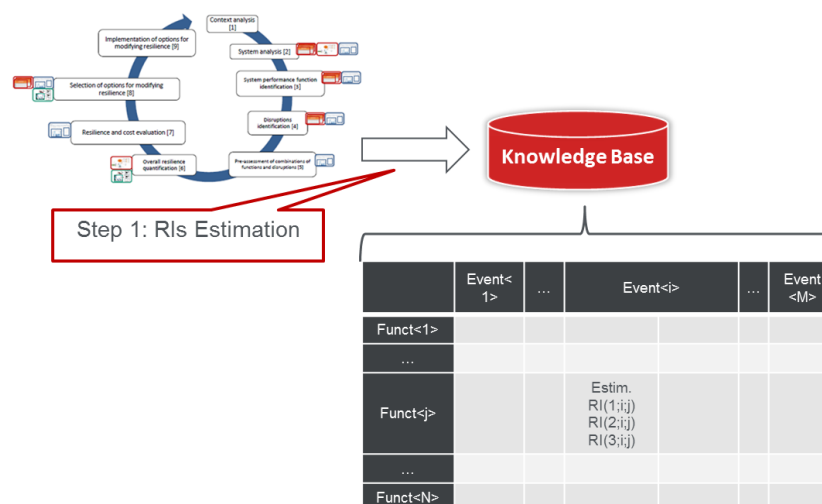


Figure 6 - RIs flow, step 1: RIs estimation

Step 2: RIs measurement

STCL operators, facing Event<i> type, measure actual RIs and store them in the Knowledge Base as shown in Figure 7.

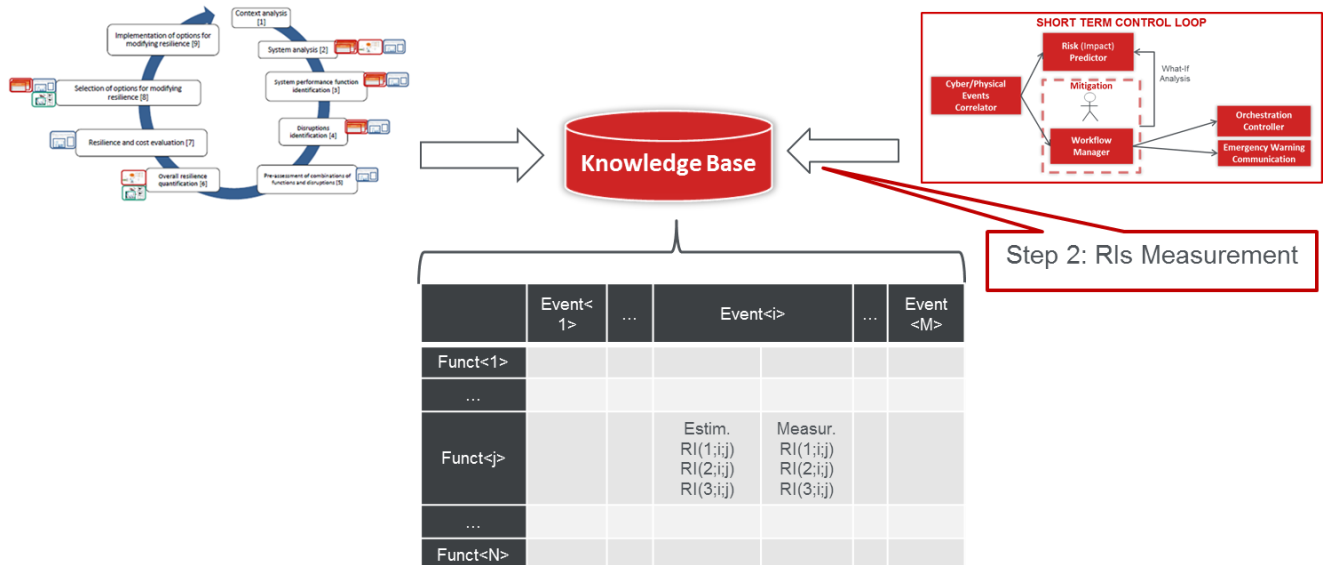


Figure 7 - RIs flow, step 2: RIs measurement

Step 3: Estimated vs. Measured RIs comparison

As shown in Figure 8 comparison between Estimated and Measured RIs are taken into account in the next LTCL cycle to improve Critical Infrastructure resilience or estimation methods if needed.

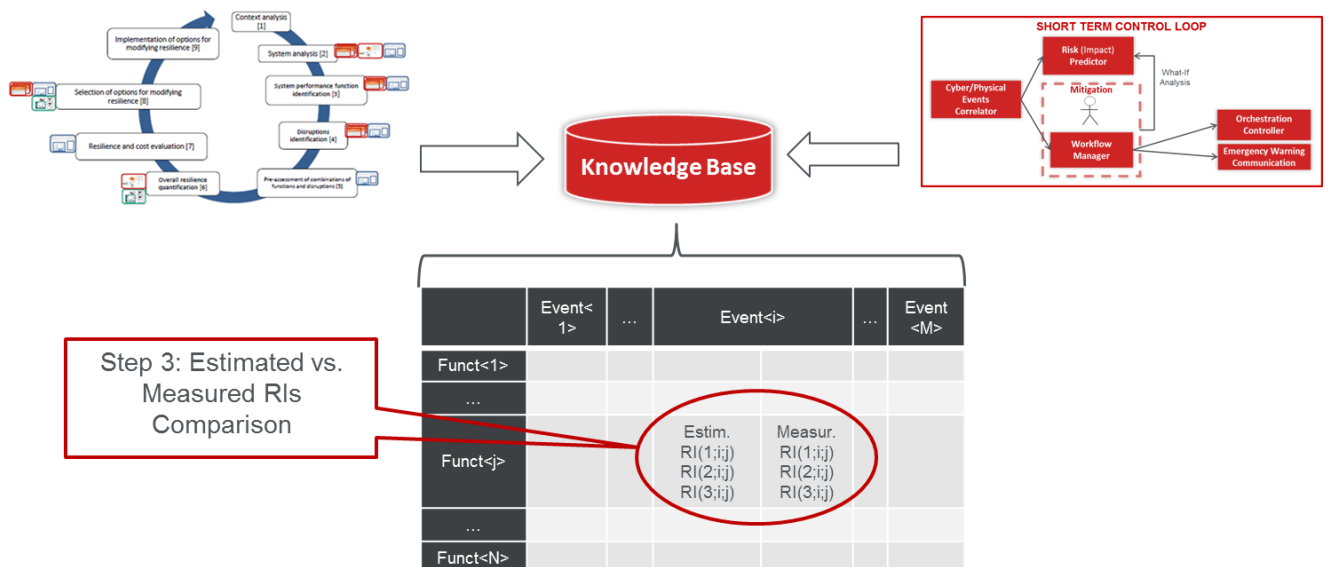


Figure 8 - RIs flow, step 3: Estimated vs. Measured RIs comparison

A sequence diagram explaining the RIs flow is described in par. 4.2.

2.4. Architecture Components

The RESISTO architecture consists of different interacting components, as depicted in Figure 9 below.

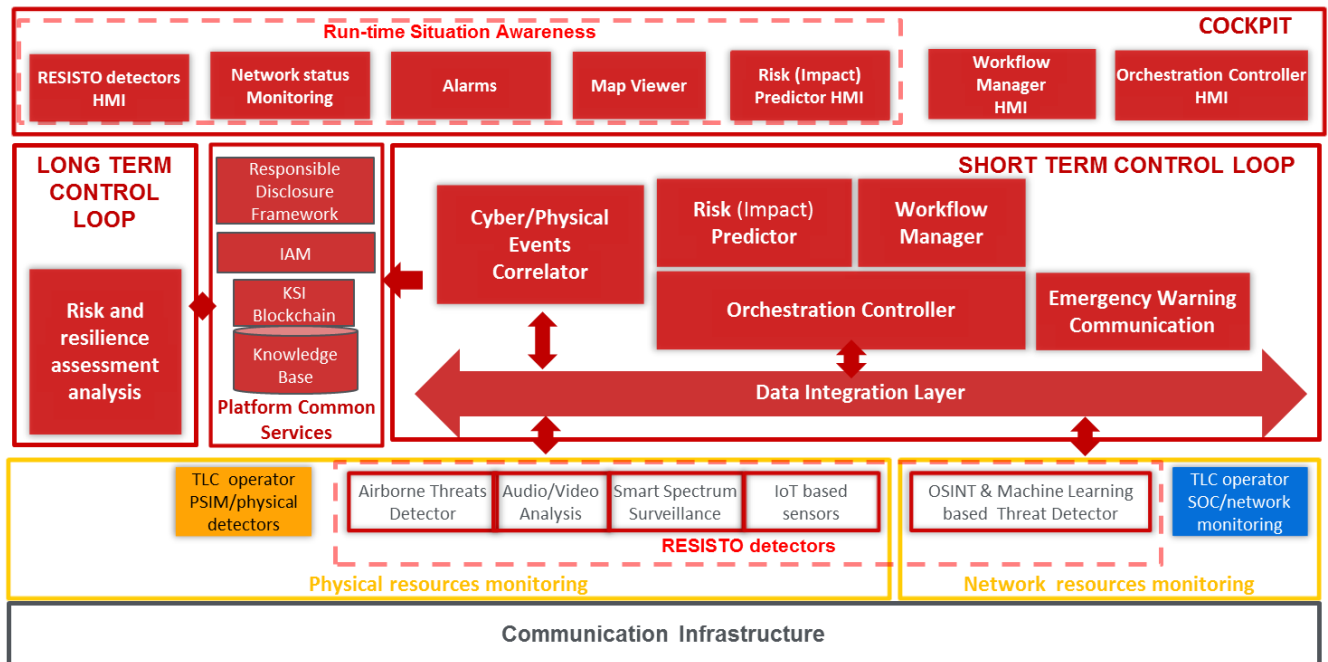


Figure 9 - RESISTO architecture components

A detailed description of each component is provided in chapter 3. The correspondence between each component and the specific paragraph devoted to its description is reported in Table 2.

Component	Sub-component	Paragraph
Long Term Control Loop		3.1
Short Term Control Loop		3.2
	Data Integration Layer	3.2.1
	Cyber/Physical Events Correlator	3.2.2
	Risk (Impact) Predictor	3.2.3
	Workflow Manager	3.2.4
	Orchestration Controller	3.2.5
	Emergency Warning Communication	3.2.6
Cockpit		3.3
Platform Common Services		3.4
	IAM	3.4.1
	KSI Blockchain Infrastructure	3.4.2
	Knowledge Base	3.4.3
	Responsible Disclosure Framework	3.4.5
RESISTO Detectors		3.5
	Airborne Threats Detector	3.5.1
	Audio/Video analysis	3.5.2
	Smart Spectrum Surveillance	3.5.3
	IoT Based Sensors	3.5.4
	OSINT and Machine Learning Threat Detector	3.5.5

Table 2 - RESISTO components and corresponding paragraphs

3. DETAILED DESCRIPTION OF ARCHITECTURE COMPONENTS

3.1. Long Term Control Loop components

The LTCL process is structured into nine sequential steps to be followed in a closed loop, as described in par. 2.1.

In Figure 3 the support of the tools to the different steps of the risk and resilience management is visualized. The usage of the tabular inputs, graphical network representations and further input tools is indicated by orange boxes and the support by the Shiny app and simulation tools by blue or green boxes, respectively.

In summary, the following main tools to support the nine-steps of the risk and resilience management process are identified:

1. A web-application to browse the tabular inputs and inferred quantities thereof. The application is based on the Shiny package of the free programming language for statistical computing R ([6] section 5).
2. Infrastructure simulators to quantify the resilience Matrix of the infrastructure for various setups, i.e. different classes of threats considering various mitigation options. The CISIApro simulator is described in detail in [7] section 5.2 and in par. 3.2.3 of this document. A description of the C++ based simulator CaESAR is given in [7] section 5.1. The simulations do not need to run in real-time on the server of the platform but may be outsourced on a periodic basis to a computing cluster or cloud to allow for the intensive computations.

In [6] other methods and strategies have been evaluated and will be used in RESISTO. These methods are helpful to improve our knowledge about possible threats and the consequences of these (Step 4 from the Risk and resilience management process, Figure 3):

1. Attack Tree: Attack trees are a way to describe an attack, in the LTCL they can be used to identify situations that the organization has no defense process in place. Each attack should have countermeasures in place (see [6] section 4.1).
2. Honeypots: A honeypot is a set of physical, HW and/or SW modules simulating legitimate interactions with external users while they are instead separate entities not performing any real operation and/or handling real data. The honeypot is a long time used technique to detect and analyse sophisticated intrusions while keeping the real system safe. Moreover, it provides the possibility to observe an attack over time enabling the system both to learn about new threats and to assess known ones. Since it collects detailed historical information, the technique is particularly useful for the long-term loop purposes. For more details, see [6] section 4.2.
3. Penetration test: A penetration testing (pentest) is a combination of techniques that considers various issues of the systems and tests, analyses, and gives solutions. It is based on a structured procedure that performs penetration testing step-by-step. Undertaking a series of penetration tests will help test security arrangements and identify improvements. When carried out and reported properly, a penetration test can give knowledge of nearly all technical security weaknesses and provide the information and support required to remove or reduce those vulnerabilities (see [6] section 4.3).
4. "Mitre ATT&CK": Telecom providers should constantly monitor the information in the "MITRE ATT&CK" knowledge base on potential attacks in order to develop and update threat models for risk assessment of their networks (see [6] section 4.4).

In order to enable an analysis of the consequences of the different threats in the telecommunication infrastructure and to make it possible to choose mitigation strategies, to increase the various resilience measures of the infrastructure, metrics and KPIs are needed. The deliverable [8] describes suggested metrics correspond to the steps six and seven of the Risk and resilience management

process (see Figure 3). For an overview of the different KPIs see Table 5 in [8]. Currently, [8] is under review and it will be updated for the final report D3.8 about RESISTOs KPIs.

In the LTCL, simulations are performed on a periodic basis, to identify the weakness of the current system. In the lap of time between one simulation and the next simulation, many changes can occur. The telecommunication infrastructure could change, others threats could be identified (e.g. with the help from methods as pentest or honeypots), additional performance metrics for the analysis from the resilience of the infrastructure could be identified.

To ensure that all changes are taken into account in the next simulation, all new relevant information for the LTCL will be stored in the knowledge Base of the RESISTO platform (see Figure 4).

If a new threat or an additional performance metric is identified, a new column, respectively a new row will be added on Figure 4. In addition, changes in the system (i.e. components of the testbed) would imply a potential change of all the calculated values in the table. After any significant changes in the system and/or the identification of a new threat or performance metric, the simulation and calculation of the resilience indicators needs to be redone.

For RESISTO, this procedure is feasible, since the testbeds are of manageable size. We assume, that for a real telecommunication infrastructure it would be challenging to rerun the simulation for any change in the system and new automated procedures would be needed (maybe separating the infrastructure in subparts to be analyzed).

3.2. Short Term Control Loop software components

The detailed description of the architecture and functionality of the individual STCL components is amply reported in deliverable [4]. However, since the definition of the interfaces of the RESISTO platform components is one of the main objective of this deliverable, it was considered appropriate for greater comprehension to add here a brief summary of the main task of each component highlighting the technological aspects adopted for their implementation.

3.2.1. Data Integration Layer

The role of the Data Integration Layer is to act as the middleware component of the RESISTO platform. It receives and dispatches all information among RESISTO components and between the RESISTO platform and all external systems.

It also provides infrastructural and communications services for functional components and for this reason a detailed description is provided in chapter 5, dedicated to explaining the communication methods that have been chosen both for the interaction between the internal components of RESISTO and for communications with external sources.

3.2.2. Cyber/Physical Events Correlator

The Cyber/Physical Events Correlator is composed by the following main components, as depicted in Figure 10:

- Correlator Engine: component correlating data source events and identifying potential threats based on a list of rules set by skilled operators and the rules originating from a machine-learning expert system: the Machine Learning Module.
- Machine Learning (ML) Module: component based on the application of machine-learning algorithms allowing identification of standard behavioral models for the traffic originating from network data sources.
- Interconnection Layer: dedicated to data exchange between data originating from network data sources and the results produced by the Correlator modules performing data analysis. The component is implemented using a distributed system for message management based on a producer/consumer model.

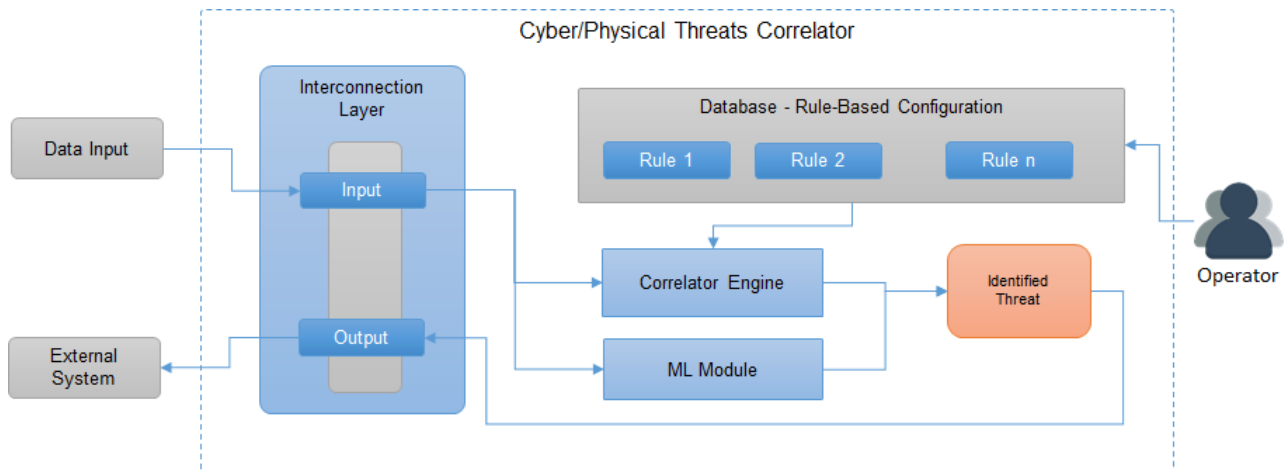


Figure 10 - Correlator components

The main objective of the correlator is to correlate events and to identify potential threats. Events are acquired through an interconnection layer that deals with managing input/output messages. All of the sources will send log data into the messaging system in raw format, to be analyzed and correlated. An example of RAW data of a SCADA source is reported in Figure 11.

```

May  4 15:18:33 192.168.1.1 n2osevents[0]: CEF:0| Networks|OS|17.5.2-1746F|VI:NEW-
NODE|New node appeared|3|app=other dvc=192.168.1.1 dvchost=nozomi-sg.local cs1=2.5
cs2=true cs1Label=Risk cs2Label=IsSecurity dmac=54:1f:15:1a:11:3c dpt=902 msg=New
other node 192.168.10.10 smac=01:0c:19:1e:1d:14 spt=60162 proto=TCP
start=1525447113163

```

Figure 11 - Raw data from a SCADA source

All data acquired from the data sources will be subject to normalization, operation necessary to make the data comparable to each other. This operation allows the correlation of heterogeneous sources.

The module that deals with identifying threats are:

- the Correlator Engine (see Figure 12). To identify such threats the module must be provided with a list of well-defined rules, which are set manually by skilled operators
- the ML Module, through traffic analysis and the use of expert algorithms.

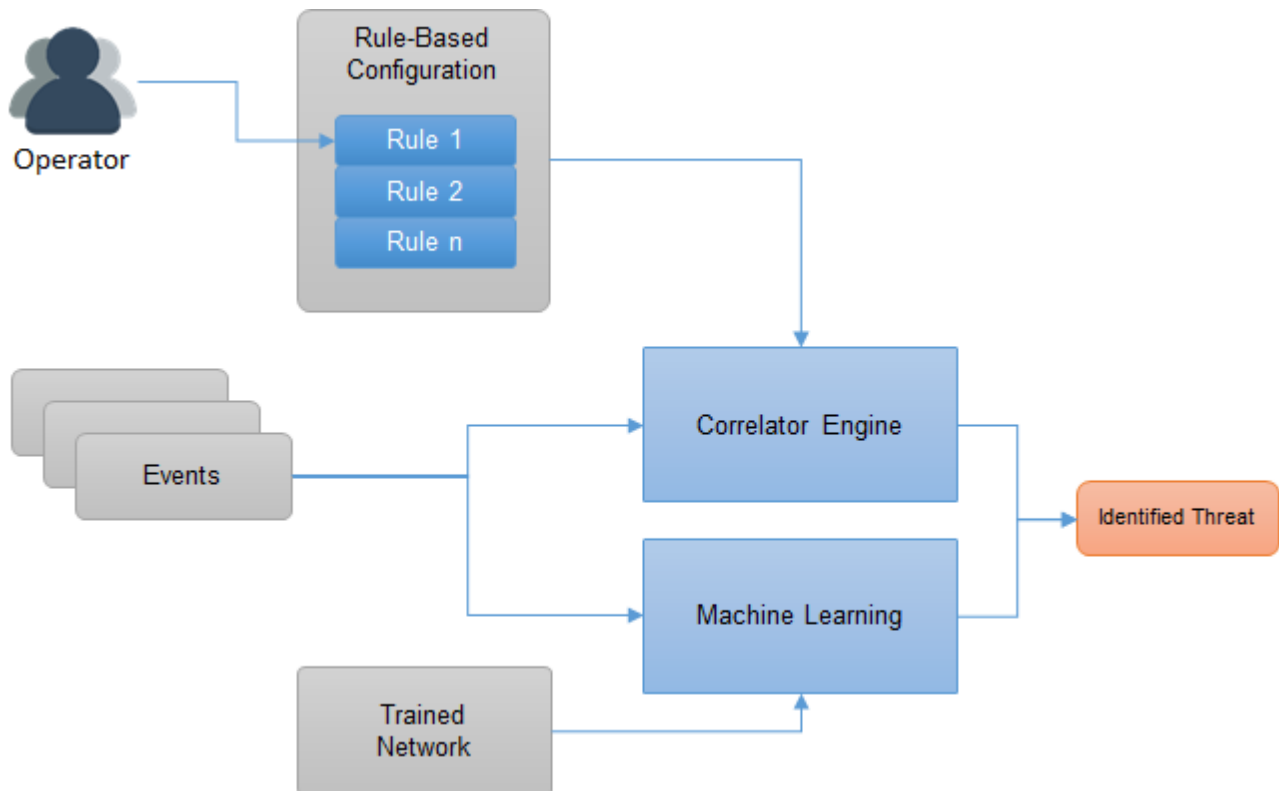


Figure 12 - Correlator main data flow

The Correlator Engine, realized with the below described technologies, will always be running to identify when a rule matches. When the rules are updated, either manually by the operators or automatically by the expert module, the engine adapts itself to the changes in real time.

An example of a correlation rule is depicted in Figure 13 (details in par. 3.2.2.1).

```

SELECT scadaEV1.msg AS scadaEvent1Msg, scadaEV2.msg AS scadaEvent2Msg
FROM      PATTERN[every      scadaEV1=SCADAEvent(signature='VI:NEW-ARP')      ->
scadaEV2=SCADAEvent(signature='SIGN:DHCP-OPERATION')]
WHERE timer:within(10 sec)]
  
```

Figure 13 - Example of a correlation rule

The list of rules and real-time events are the inputs for the correlation engine. The output of the engine is a potential threatening event detected by rules evaluation. In the example in Figure 14, a motor generates an alarm when two SCADA events, with particular characteristics, occur within a defined time window.

```
{
  "ts": 1544540170540,
  "ruleName": "Intrusion attempt",
  "ruleSeverity": "MEDIUM",
  "newEvents": [
    {
      "scadaEV1": {
        "signature ": "VI:NEW-ARP",
      },
      "scadaEV2": {
        "signature ": "SIGN:DHCP-OPERATION",
      }
    }
  ]
}
```

Figure 14 - Example of correlation

The correlation shows:

- rule-related information (i.e., name, severity and timestamp of the match);
- information related to the correlation original events.

The alarm is normalized in a standard format and pushed into the message manager of the interconnection layer, so that it can be retrieved by all external systems. The detailed architecture is shown in Figure 15.

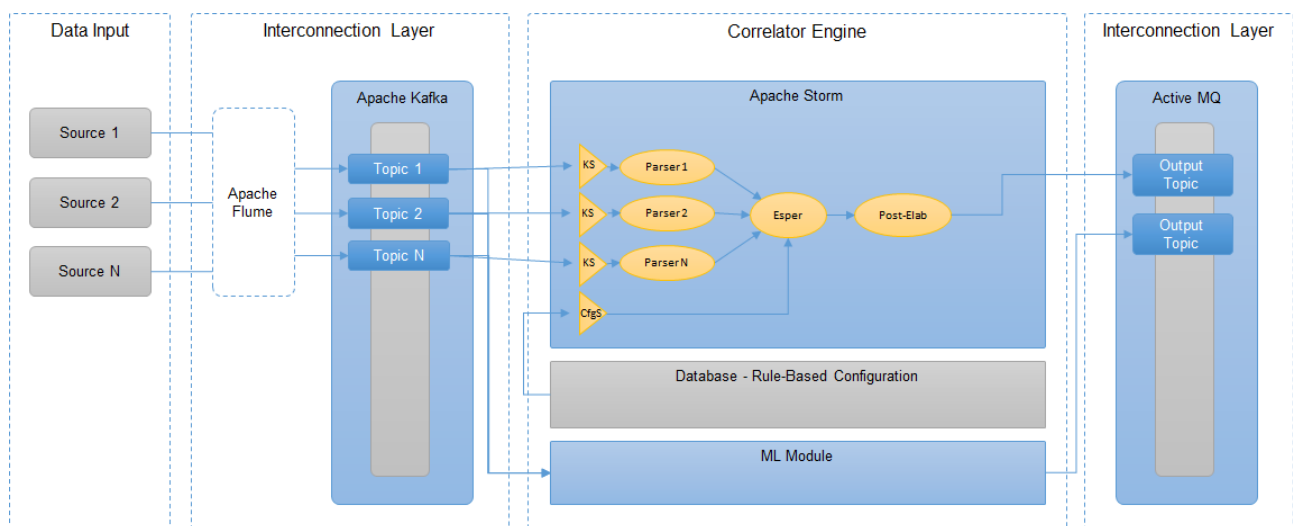


Figure 15 - Logical Architecture of the Cyber/Physical Events Correlator

3.2.2.1. Correlator Engine

The Correlator Engine is mainly based on Apache Storm and Esper technologies.

Apache Storm is a free open source software for distributed computing of real-time processes. The Storm architecture is of type Master-Slave:

- Nimbus, the master node, is tasked with distributing software code on the cluster, assigning tasks to the different machines and monitoring potential failures on the part of the tasks;
- the Supervisor slaves wait for the tasks to be assigned by Nimbus and can instruct workers to perform the assigned tasks;
- the work calculated by a single topology is divided and allotted to many workers distributed on different machines belonging to the same cluster.

Zookeeper supports coordination of the tasks among the various demons.

A topology is a process graph (Figure 16). Each node of the topology contains a piece of processing logic and the links between the nodes specify how data must be transferred from one node to another. A topology is always active as Storm is able to manage hardware failures and re-assign failed tasks to other available machines belonging to the cluster with no loss of data.

Storm provides the elementary commands required for construction of topologies:

- spout: a topology node capable of capturing an input stream or, in other words, a data source of tuples;
- bolt: a processing step (similar to a map of the map-reduce job) that:
 - receives input one tuple at a time;
 - processes it;
 - possibly emits one or more tuples which could generate new streams.

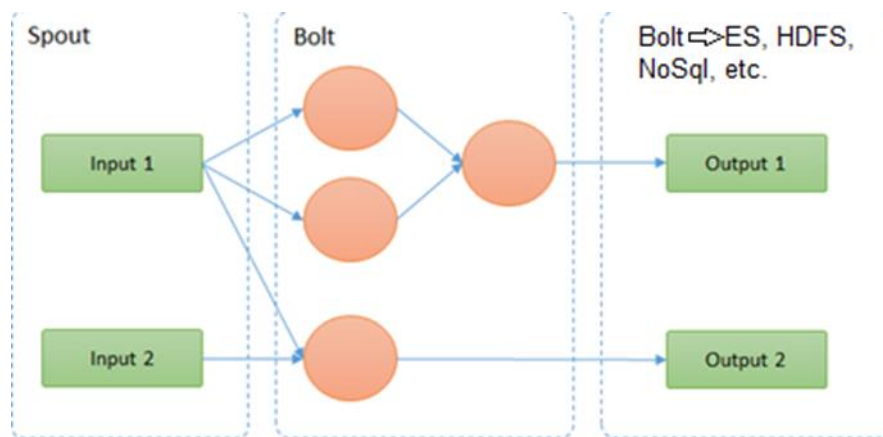


Figure 16 - Apache Storm: Topology Nodes

When defining a topology, it's possible to associate a different degree of parallelism to each node so as to best adjust available resources based on the type of processing planned.

Data exchanged within a topology are called "tuples". The lifecycle of a tuple begins with a spout, the source of a tuple.

Whenever Storm requests a tuple from Spout:

- the tuple remains in the queue but assumes the "pending" state thus blocking elaboration by other processes;
- the tuple is assigned an "id" allowing the tracking of all messages generated within the topology during processing;
- Storm sends the tuple to all bolts specified in the topology and keeps track of all generated messages; based on the processing results, Storm sends to the spout that generated the tuple a message of type:
 - ack: so that the tuple will ultimately be eliminated from the queue,
 - fail: so that the tuple will be re-inserted into the queue to undergo re-processing.

Esper is a Complex Event Processing (CEP) component able to perform Event Stream Processing. This feature allows real-time or quasi-real-time processing. The Esper engine operates in a different manner compared to a database management system. Instead of storing data and performing queries on the data stored, it allows applications to store their own queries and directly launch them on the data. The processing mode is continuous and a reply is in real-time whenever the conditions contained within the query are met.

Esper provides two principal methods for processing events and they are:

- Event pattern,
- Event stream query.

The first method is based on a language allowing specification of expression-based patterns for event matching. This event processing method, at the base of which is the implementation of a state machine, expects to receive event sequences or a combination of event sequences and allows their correlation based on time factors. On the contrary, the second method offers the possibility to define queries allowing filtering, aggregation and correlation (through join operators) as well as to analyze event streams. These queries follow the EPL (Event Processing Language) syntax. EPL is a declarative language similar to SQL and by which it is possible to define CEP statements for the correct processing of an event stream.

EPL differs from SQL in that it uses the view concept as opposed to the table concept. These views represent the various operations required to structure and extract data from event streams. Note: once the queries are expressed (in EPL), they must be registered (at runtime) in the engine so that Esper can verify the results of the queries applied to the incoming events and possibly forward them in real-time to one or more subscribed listeners. This mode of operation allows, if necessary, a simple and swift modification to the already registered queries and offers the possibility re-process the events.

3.2.2.2. Machine Learning Module

The Machine Learning module allows the detection of anomalous communication traffic situations if compared with the statistically detected traffic density. Control flow historical data retrieved from the past are used as first baseline for the elaboration of the statistical procedure. Historical data repository can be increased continuously in order to tune the machine learning based detector with respect to evolving data traffic curves.

The operation principle of the machine learning module is based on a periodical schedule of a statistical procedure which will analyze information relevant to control flow packets. Statistical data analysis will take into account trends and data distribution in order to compute metrics for anomalies evaluation. These metrics will also be related to the outputs of the descriptive statistics. Descriptive statistics provides great insight into the data in terms of count, mean, standard deviation, minimum value, percentiles. The usage of descriptive statistics will highlight usual traffic patterns. The machine learning module will consider information related to data traffic and will evaluate specific parameters which will be linked to well-known cyber attacks. For example, considering a scenario related to a Denial of Service (DoS) attack.

The statistical procedure will be implemented as software modules developed in R or Python languages.

A more complete description of the Machine Learning Engine will be included in RESISTO deliverable D4.5 - Description and definition of the Correlator intelligence.

3.2.2.3. Interconnection Layer

The correlation module requires scalable and flexible interconnection instruments for the exchange of data between data sources and the modules collecting the results of the analyzed data. The technology used to implement this layer is Apache Kafka (see par. 5.1) and, where necessary, Apache Flume as a supporting component.

The use of Apache Flume originates from the need to optimize data acquisition management and centralization. The architecture is based on the Agent concept representing an implementation of the producer/consumer model (Figure 17).

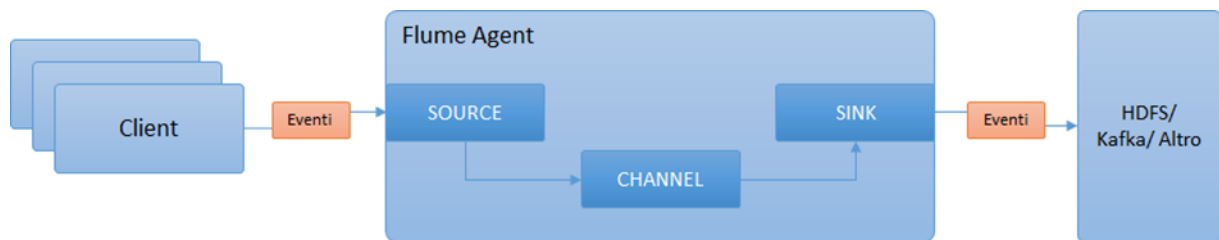


Figure 17 - Apache Flume architecture

There are three fundamental concepts underlying this technology:

- **Events:** an event is data produced by an external source and forwarded to a final destination. From a network point of view, it is a byte-array composed of a header and a payload. The “external data” is inserted in the payload and is ignored by Flume. All information required for management and processing of events is accumulated in the header under the form of a key-value couple.
- **Client:** a Client is an entity that collects data, encapsulates it in events and sends it to one or more Agents. It could be a Flume client that collects data produced by external applications according to standard formats (for example, Apache, Log4j) or, if no Flume client is available, it could be developed ad hoc.
- **Agent:** Agent is the central Flume component: it receives events generated by an external client, deposits them in an internal queue and sends them to one or more final destinations.

Like Kafka, Apache Flume is based on the producer-consumer model and possesses a similar architecture.

3.2.3. Risk (Impact) Predictor

The Risk (Impact) Predictor will simulate the impact of anomalies and security attacks on the Communication Infrastructure and on the interlinked CIs executing at run-time on a model of the Communication Infrastructure. It will also support the decision making process allowing a “What-If analysis” by simulating the application of countermeasures and reconfiguration and their impact on system resilience.

This is based on CISIPro 2.0 (Critical Infrastructure Simulation by Interdependent Agents) [13], a software engine able to calculate complex cascading effects, taking into account (inter)dependencies and faults propagation among the involved complex systems.

CISIPro 2.0 is an Agent-Base simulation software and is mainly composed of two modules. The first one is the off-line tool known as CISIPro in which it is possible to design and implement complex and highly interdependent scenarios. While the second one is the on-line tool called CISIPro which implements in Simulink (Mathworks) the real engine at the base of the Risk Predictor module.

CISIPro is a software platform based on a database-centric architecture in which the database plays a crucial role. This means a centralized asynchronous design that allows good modularity and scalability where each element of the informatics infrastructure interfaces, independently, with the centralized database (DB) in order to get the last actualized data from the field. For the

implementation of the engine simulator, the Matlab language was used to develop a redistributable Matlab App.

3.2.3.1. CISIapro 2.0 – Dynamic Risk Analysis Tool

This section describes the Risk Predictor architecture, commonly adopted in Critical Infrastructure Protection projects, in order to provide a new type of Dynamic Risk Assessment tool. Such a tool is based on CISIapro software engine, which is able to calculate complex cascading effects, taking into account (inter)dependencies and faults propagation among the involved complex systems. CISIapro has been developed inside the H2020 Project ATENA and in RESISTO will be update to version 2.0 adding some important functionalities related with the modeling of telecommunication infrastructures. As mentioned in section 5.2 of deliverable [3], modelling complex interdependent systems, using the Mixed-Holistic-Reductionist (MHR) approach, is a prerequisite to produce an effective Risk Predictor tool. Once modelled the involved scenario, with CISIapro 2.0 it is possible to implement the MHR methodology.

CISIapro 2.0 is an Agent-Base simulation software and it is mainly composed by two modules. The first one is the off-line tool known as CISIapro 2.0 that allows the design and implementation of complex and highly interdependent scenarios. While the second one is the on-line tool called CISIamat which represents the real engine at the base of the Risk Predictor module. This engine will be integrated in the RESISTO architecture as in Figure 18.

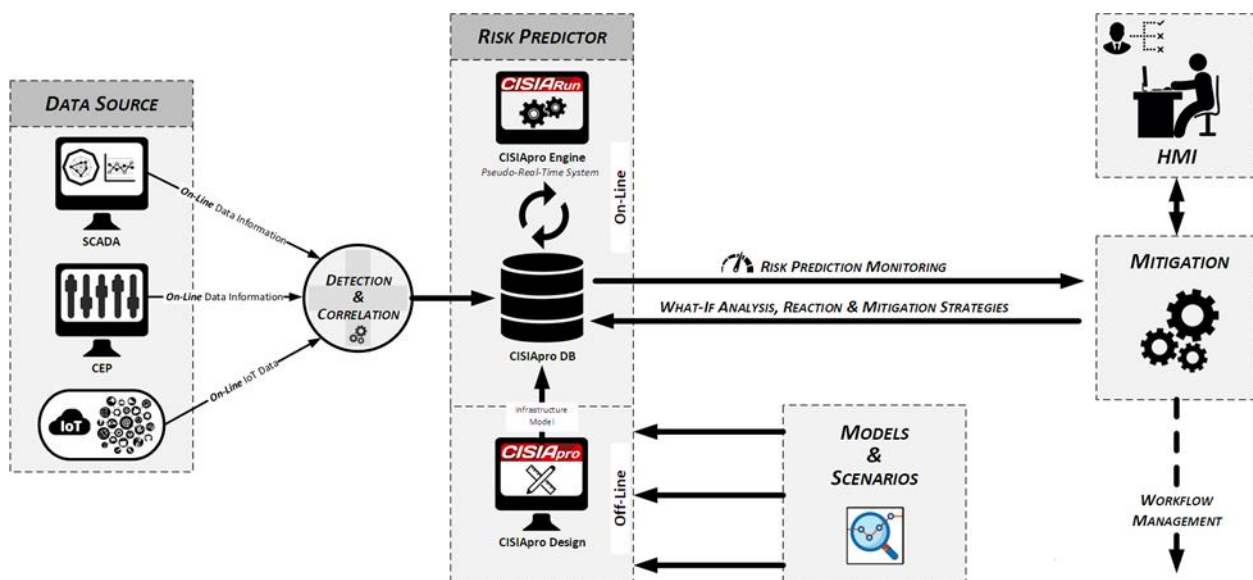


Figure 18 - Risk Predictor Architecture

CISIapro 2.0 is a software platform based on a database-centric architecture in which the database plays a crucial role. This means a centralized asynchronous design that allows a good modularity and scalability where each element of the informatics infrastructure interfaces, independently, with the centralized database (DB) in order to get the last actualized data from the field (e.g. SCADA Systems), Complex Event Processing and generic IoT data systems.

From this point of view, CISIAmat engine does not only analyze actual situation and calculate the risk projected in the possible near future but, first, it plays the important role of Hybrid Risk Evaluation Tool. Hybrid because it is able to get information of different natures (sensor and data acquisition and complex event processing systems) and translating them in operational levels of resources, faults or services for the entities introduced in the critical infrastructure model.

With the proposed architecture, through CISIApro 2.0 modelling software, it is possible to dynamically change the interdependencies model and plugin other modules in order to have a pseudo-real-time scalable and flexible system, which can be changed at any time.

Figure 19 shows the database structure. The DB stores the information needed for the representation of several Critical Infrastructures, such as:

- Each entity is a specific instance of an entity type;
- Each entity has a status made of variables with values;
- Each entity has ports for exchanging resources;
- Each resource is associated with a MHR layer/net;
- Each layer has proper interdependencies;
- Each interconnection is made of a couple of ports, associated to two entities.

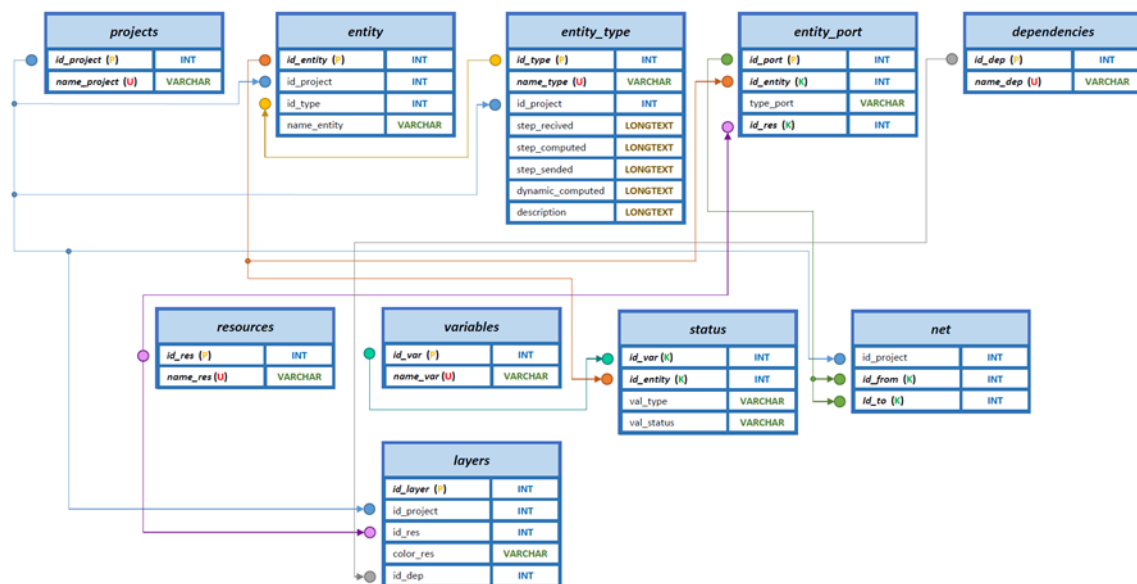


Figure 19 - CISIApro DB structure

Outputs of CISIApro 2.0 are stored in a different database with specific features, see Figure 20, such as the record time-stamp in terms of date, time and milliseconds. In this way, any downstream module can retrieve data regarding the latest actualized critical situation in the modelled scenario. Adjacency matrices which represent interdependencies existing between entities are generated during the design phase. During the simulation, the matrices are represented as queue data structures to speed up computations.

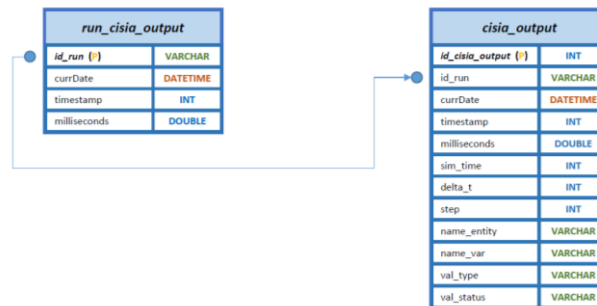


Figure 20 - CISIApro SQL output data structure.

It should be noted that CISIApro 2.0 has introduced efficient ways to model, execute and debug simulations and cascading effects. In particular, an intuitive Graphical User Interface (Figure 21) is provided to create entities and connect them in easy way.



Figure 21 - CISIApro user interface

In the following sections some brief descriptions explain the main modules provided by CISIApro design tool to exploit the potential of proposed modelling techniques.

3.2.3.2. Layers & Resources Module

Thanks to the Layers & Resources module (Figure 22) it is possible to instantiate all the required layers in a critical infrastructures scenario model. It is the first step for the simulation implementation. Assigning a specific resource to a corresponding dimension also gives us a deeper awareness with respect to the nature of the managed information.

The screenshot displays the CISIApro 2.0 'Layers & Resources' module. The interface includes a sidebar with icons for Database, Entity Maker, Layers & Resources (active), State Variables, Links State, and a color picker. The main workspace shows a table of resources for project 'ATENA_2016'.

id_res	name_res	dim_res	name_dep	color	id_project
1	COMMAND	1	Logical	#000000	1
2	CONTROL	1	Logical	#000000	1
3	CYBER_FAULTS	4	Cyber	#000000	1
4	FISR_LEVEL	1	QoS	#000000	1
5	LOAD_LEVEL	1	QoS	#000000	1
6	OPERATIVE_LEVEL	1	QoS	#000000	1
7	POWER	1	Physical_Power	#000000	1
8	POWER_REQUEST	1	Logical_Power	#000000	1
9	DATA_FLOW	1	Logical	#000000	1
10	MEASURES	1	Logical	#000000	1
11	NP_WATER	1	Physical_Water	#000000	1
12	WATER	1	Physical	#000000	1
13	LOGICAL_POWER	1	Logical_Power	#000000	1
17	PIPPORES	1	Pippodep	#000000	1

resources to be deleted

id_res	name_res	id_project
--------	----------	------------

color picker

Figure 22 - CISIApro Module: Layers & Resources

3.2.3.3. Entity maker Module

In the Entity Maker module (Figure 23), using the integrated PHP code editor, it is possible to instantiate and program behaviors for each considered entity class. Once this step is completed, the introduction and duplication in the design phase will be allowed. Each entity class is composed of four modules that are executed, several times, during the simulation run:

- 1) RECEIVED, which evaluates the received resources and faults;
- 2) DYNAMIC COMPUTED, which implements dynamic evolution;
- 3) INSTANT COMPUTED, which implements instantaneous evolution;
- 4) SENT, which evaluates the resources that are sent to the downstream entities.

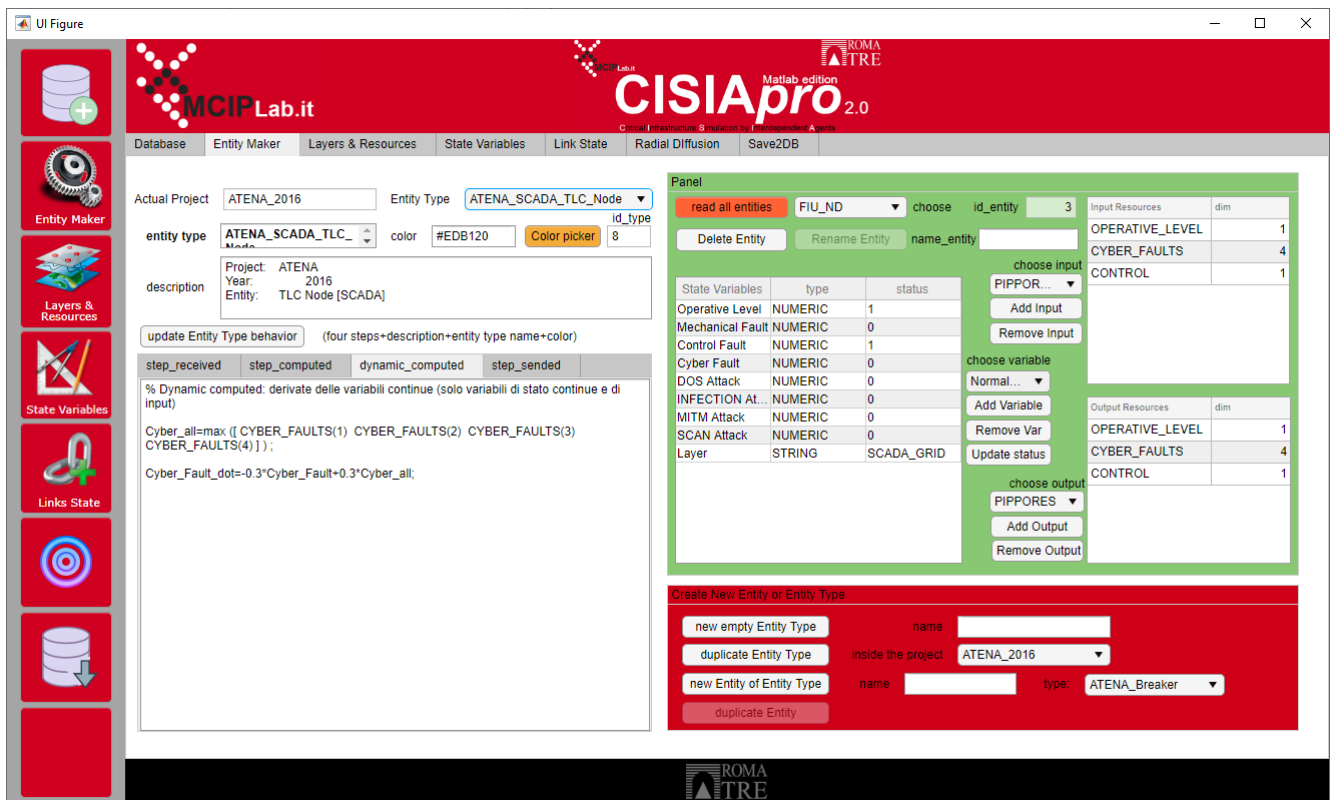


Figure 23 - CISIapro Module: Entity Maker

3.2.3.4. Modeler Module

The Modeler, in the new version of CISIApro 2.0, has been developed using SIMULINK (MATLAB) modeling approach (Figure 24). The choice has been motivated by considering the rapid prototyping of such environment and its ability of mixing block of CISIApro with dynamic systems allowing the design of innovative applications.

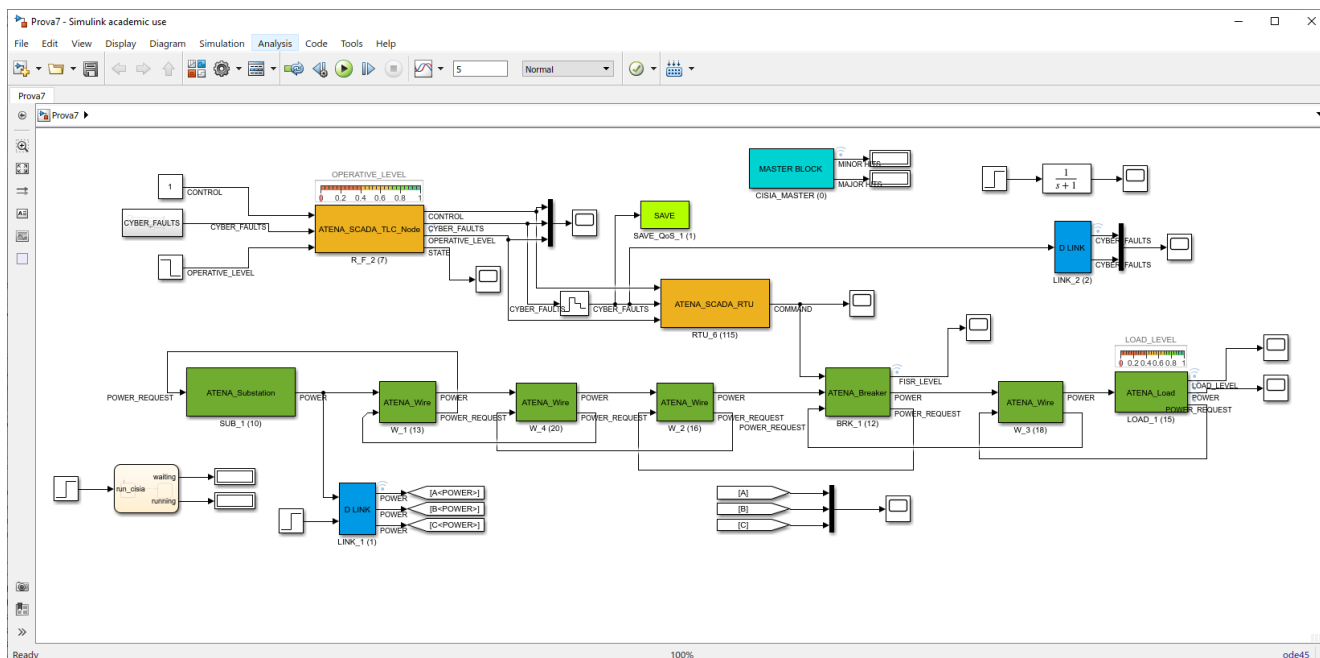


Figure 24 - CISIApro Module: Modeler

3.2.3.5. State variables Module

As previously mentioned, with CISIapro 2.0 simulation, defining Input and Output is not required. This is possible because they are calculated, instant by instant, during the simulation time, with respect to entity state variables and especially evaluating operational levels related to each modelled element. In State Variables module (Figure 25) indeed it is possible to set the initial state for every variable that is part of the simulation.

The screenshot displays the CISIapro 2.0 software interface, specifically the 'State Variables' module. The interface includes a sidebar with navigation icons for Database, Entity Maker, Layers & Resources, State Variables (selected), Link State, Radial Diffusion, and Save2DB. The main workspace shows a project named 'ATENA_2016'. Below the project name, there are buttons for 'Load Variables', 'Add a row', 'Delete Selected', and 'Update Table'. A table lists 19 state variables with columns for 'id_var', 'name_var', 'val_type', and 'id_project'. To the right of the table, there is a section titled 'variables to be deleted' with a table for 'id_var', 'name_var', and 'id_project'.

id_var	name_var	val_type	id_project
1	Operative Level	NUMERIC	1
2	AC Fault	NUMERIC	1
3	Average consumption_kWh	NUMERIC	1
4	Command Fault	NUMERIC	1
5	Connected Loads	STRING	1
6	Control Fault	NUMERIC	1
7	Cyber Fault	NUMERIC	1
8	Door Open	NUMERIC	1
9	DOS Attack	NUMERIC	1
10	FISR Level	NUMERIC	1
11	Gas Fault	NUMERIC	1
12	INFECTION Attack	NUMERIC	1
13	Load Power Request	NUMERIC	1
14	Mechanical Fault	NUMERIC	1
15	MITM Attack	NUMERIC	1
16	Normal Users	NUMERIC	1
17	Power Fault	NUMERIC	1
18	Power Request Fault	NUMERIC	1
19	Remote Fault	NUMERIC	1

Figure 25 - CISIapro Module: State Variables

3.2.3.6. Link State Module

Links State is the most recent module tool introduced in CISIApro 2.0 software for RESISTO scenario model needs (Figure 26). Such module was designed in order to be compliant to model in which dynamic links are required. A dynamic link is defined as a link that connects two entities to each other, which could change its state during different simulation. For instance, it allows to model scenarios, like transportation infrastructures or telecommunication networks, where an entity may represent an element of the system and links represent multiple available paths (e.g. SDN dynamic routing). Through this mechanism it will be possible to instantiate all the multiple connection, among the involved entities, activating only one of them at a time.



Figure 26 - CISIApro Module: Link State

3.2.3.7. CISIAMat – the CISIApro 2.0 on-line engine

Usually Risk Analysis and Business Continuity are mainly focused on prevention and consequences mitigation. In RESISTO case study a near real-time performances monitoring is also considered in order to have an ongoing decision support system.

In RESISTO project CISIAMat engine is part of the STCL. The STCL continuously monitors the CI systems and its cyber-physical state, in order to detect the presence of failures/attacks in a real-time environment. To defend against such attacks, the STCL shall compute domain specific mitigation actions and propose them to the RESISTO operator, in order to enforce them into the underlying plant and preventively mitigate consequences on the service provision. The main aim of the on-line loop is to increase the awareness of the operator, displaying possible impact/consequences and exposure to risk with respect to the actual adverse events on the physical infrastructure and on the telecommunication network.

It is important to understand the primary role played by Risk Predictor (CISIApro On-Line engine) together with the Mitigation Module. First of all, CISIApro On-Line engine provides an impact evaluation of detected anomaly. In order to mitigate the effects, the Decision Maker, also supported

by a Workflow Manager, can choose among different sequences of possible reaction strategies to send to the Risk Predictor module. Risk Predictor, in turn, starting from actualized scenarios and QoS (Quality of Service) levels of involved devices, simulates What-If scenarios to provide useful information for the Decision Maker with respect to 'forthcoming' critical situations.

Thanks to CISIamat on-line engine is also possible to create, as already demonstrated in URANIUM and H2020 ATENA [14] European projects, web-based synoptic platforms. Such projects made evident that it is important to provide an intuitive user experience in order to simplify and speed up decision processes in emergency situations. For instance, in URANIUM project (Figure 27) it was developed a GIS (Geographic Information System) demonstrating how it is possible to carry out a What-If analysis taking into account different emergencies procedures and considering available civil protection resources at a specific point in time.



Figure 27 - URANIUM platform civil protection panel

Another important example is represented by H2020 ATENA project synoptic platform (Figure 28). In this case, a What-If analysis procedure was exploited in order to test possible consequences of adopted Electrical Grid reconfiguration with respect to all interconnected critical infrastructure (Water distribution, Gas network and involved SCADA systems).

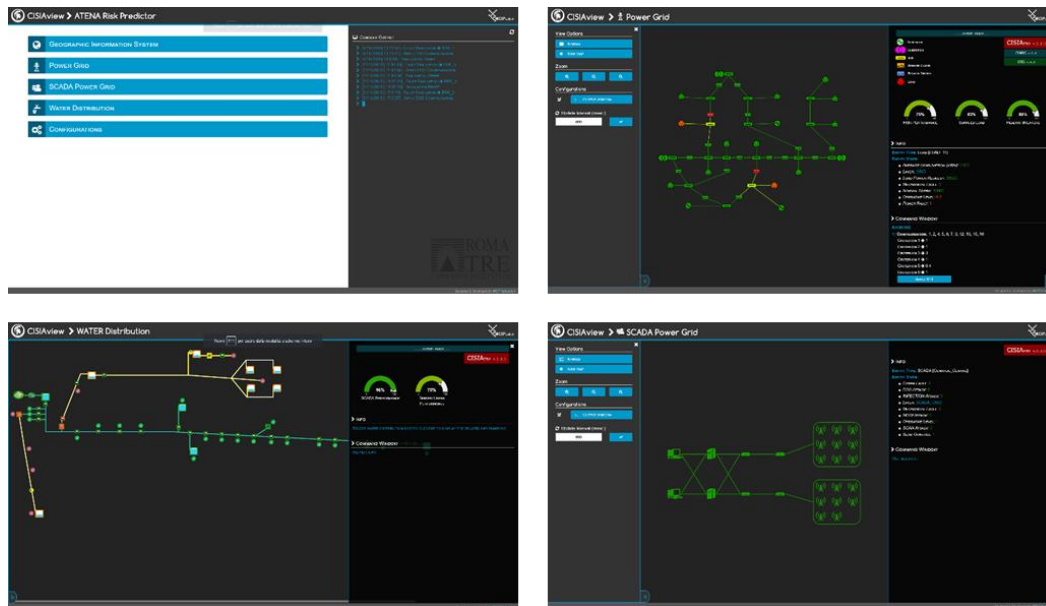


Figure 28 - ATENA platform

3.2.4. Workflow Manager

This component will deal with response and mitigation. Through a software module, the Workflow engine based on Leonardo's SC2 application, the decisions and mitigation actions of TLC security operators are supported.

On the basis of the target value and quantitative analysis performed by the Risk Predictor, a human decision maker will select the best reaction/mitigation strategy to mitigate the risks or to recover from a damaged situation. After the operator's choice, the Workflow engine will define the specific actions to be enforced.

The Workflow engine included in the RESISTO platform is an extremely effective tool for managing critical infrastructure security. It is a Business Process Model (BPM) engine for the configuration and execution of automatic or semi-automatic processes, consisting of sequences of actions and reactions, which can be triggered by a defined event. Given a certain alarm/event, it permits selecting and executing the most appropriate workflow, i.e. a conditional sequence of tasks.

The workflow execution is carried out via Activiti, an open-source workflow engine written in Java that can execute business processes described in standard BPMN (Business Process Model and Notation) 2.0. The workflow is represented by an xml file, which is managed by the Activiti engine through its deployment in the H2 database.

At run time, the Workflow Manager receives the alarm trigger for activating a specific sequence of tasks on a certain ActiveMQ queue. The workflow is executed by the Activiti engine. The alarm lifecycle (New, In progress, Closed) is instead managed through the publication of its state updates on the corresponding ActiveMQ topic.

A complete description of Workflow Manager can be found in [10] and [11].

3.2.5. Orchestration Controller

The RESISTO Orchestrator provides an abstraction layer for the countermeasure application process on network elements. Countermeasures commands are sent to the Orchestrator as a set of high-level instructions involving network functionalities. The Orchestrator translates the countermeasure into a set of instructions, optimized based on the current status of the network.

The Orchestrator functionalities can be piloted either through the API or the Human Machine Interface (HMI).

While the API allow automated control of the Orchestrator functionalities, the HMI allows operators to issue orders manually, exercising a finer control on operations. The HMI will be populated with actions based on the network stack. The Orchestrator can be updated and upgraded leveraging Node-RED workflows [25].

The dynamic deployment of countermeasures leverages Software Defined Networking (SDN)/Network Function Virtualization (NFV) technologies in order to have awareness of the underlying networking stack. Based on the networking technologies that are available, more actions will be available as countermeasures and thus in the Orchestrator. A comprehensive analysis of the capabilities enabled by the various technologies can be found in [10].

To communicate with the underlying network elements, the Orchestrator uses standard northbound APIs, depending on the network technology. More specifically, three configurations can be distinguished:

- **Networks with NFV capabilities.** In this configuration, the Orchestrator supports the ETSI Open Source MANO (OSM) northbound API. The OSM Northbound API is based on the NFV SOL005 ETSI standard [16].
- **SDN Network.** In this configuration, the Orchestrator supports the ONOS SDN controller northbound API [27].
- **Legacy Network.** In this case, the network offers limited possibility of dynamic reconfiguration. To leverage the RESISTO framework, the insertion of a limited number of SDN devices in critical interconnection points will be suggested. In this way, it will be possible to leverage the advantages of an SDS approach with a minimal investment.

RESISTO Orchestrator extends the capability of available systems, by integrating an optimization step. This process exploits several Key Performance Indicators (KPI), including

- Latency,
- Packet loss,
- Routing resiliency,
- Isolation constraints.

The optimized deployment of countermeasure is particularly useful when traffic rerouting must be performed. Multiple scenarios can be reduced to this problem, such as sudden unavailability of a network device (e.g., natural disaster or cyber-attack) or the deployment of an additional security measure (e.g., IDS through which the traffic must be routed). In this case, by leveraging multi-objective optimization, the Orchestrator can rapidly restore the service on the network while keeping Service Level Agreements (SLA) satisfied and improving the overall distribution of traffic. Such an approach would be impossible with traditional load balancing. A more complete coverage of multi-objective re-routing is available in [10].

A detailed description of Orchestration Controller can be found in [10].

3.2.6. Emergency Warning Communication Function

The Decision Maker policy implementation will resort to the Emergency Warning Communication Function (EWCF) service when events like natural disasters, physical or cyber-attacks occur. The EWCF is designed to send targeted alerts and/or informational instant messages to specific categories of users such as rescue teams or security officers that will be physically present at specific target areas (e.g. group of mobile cells, geographical perimeter, etc.).

The implementation will include a server and an Android application. The server will expose an interface towards the other modules of the RESISTO framework that need to communicate information concerning a physical-cyber attack to the intervention team that operates where the telecom infrastructure is located. The rescue team will leverage on the application information, both textual and visual. In particular, the position of points of interests or of the other team members will be

collected and visualized. The app will be available on Android smartphones, or on any other Android device.

The EWCF service will be implemented in a microservice architecture using Docker containers. The service will have its own persistent storage and database where information regarding teams, the users, and events are registered. The server will be connected with an Android app that will be installed on the user terminals of the team members. The same app will be connected to an IoT platform and will relay sensor values to the platform. In particular the GPS position of the terminal will be collected in the IoT platform and in this way will be shared among the team members. A messaging platform will be used instead to connect the terminals among themselves and with the main service that in turn will receive specific messages from the Workflow Manager.

The software architecture chosen is based on a node.js REST server, Mongodb for the local service database. Messaging will be based on Google Firebase cloud service and thinger.io has been used as IoT platform. Nevertheless, the architecture is modular and can be adapted to use other external platforms.

A more detailed description of EWCF can be found in [11].

3.3. Cockpit

The Cockpit represents the set of HMI applications that RESISTO operators can use to command & control run-time operations, i. e., the Short Term Control Loop run-time HMI.

RESISTO will have a main HMI that will allow access by the operators to all the features of the system.

The interface is of the multi-screen Web type and will be mainly based on the SC2 HMI application which makes use of panels and cells in order to allow display of multiple data on the screen. Furthermore, the interface layout can be customized at any given moment on-the-fly according to the needs of the operator connected to the system.

3.3.1. RESISTO HMI layout

The RESISTO HMI will be based on 3 monitors as illustrated in Figure 29 below. The cockpit configuration can be modified at any time according to the space requirements of the hosted applications.

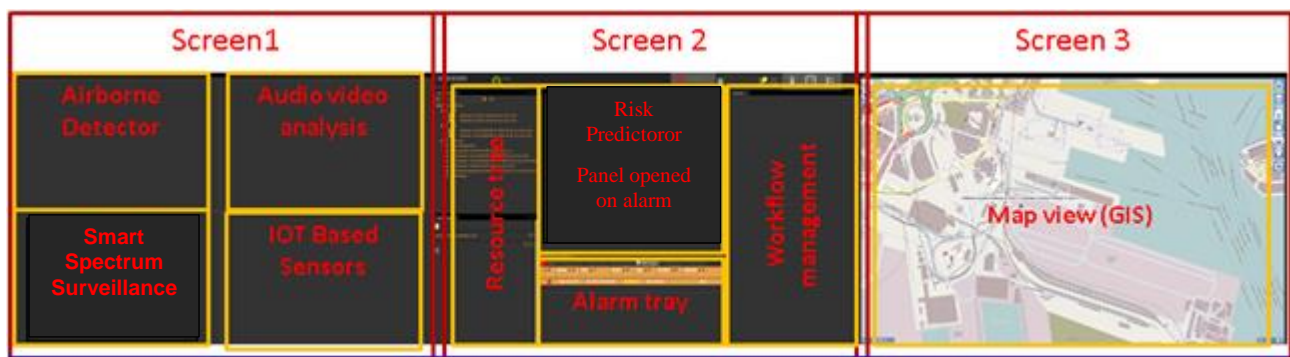


Figure 29 - Cockpit HMI layout with three RESISTO monitors

The RESISTO HMI will be based on 3 monitors with the following characteristics:

- **Screen 1:** will host vertical RESISTO web applications such as the RESISTO detectors' HMIs. It will be possible to host more applications using a «multipanel» view.
- **Screen 2:** will host the basic SC2 resource and alarm management modules:

- *Resource tree view*: a list of resources managed by RESISTO: network devices, sensors, applications, alarm sources et al.
 - *Alarm tray*: shows the currently opened alarms.
 - *Workflow management of the alarms*: workflow diagram, action tasks and user interactive window.
 - A view of a web app related to a RESISTO subsystem such as CISIApro at the center of page showing the infrastructure status. In this manner it will be possible to provide the operator with an immediate view of an alarm situation and the impact on the CI.
- **Screen 3**: will host a GIS-based application displaying georeferenced resources and alarms. For every resource, sensor or alarm source that needs a geo-reference it will be possible to view the related position on a map. The map can show a base cartography and a set of layers related to the information needed.

3.3.2. HMI features – Panel management

The RESISTO Cockpit is built using the SC2 HMI application which makes use of panels and cells in order to allow display of multiple data on the screen. Screen 2 in of the RESISTO layout. Figure 30 below is an example of such an SC2 feature.

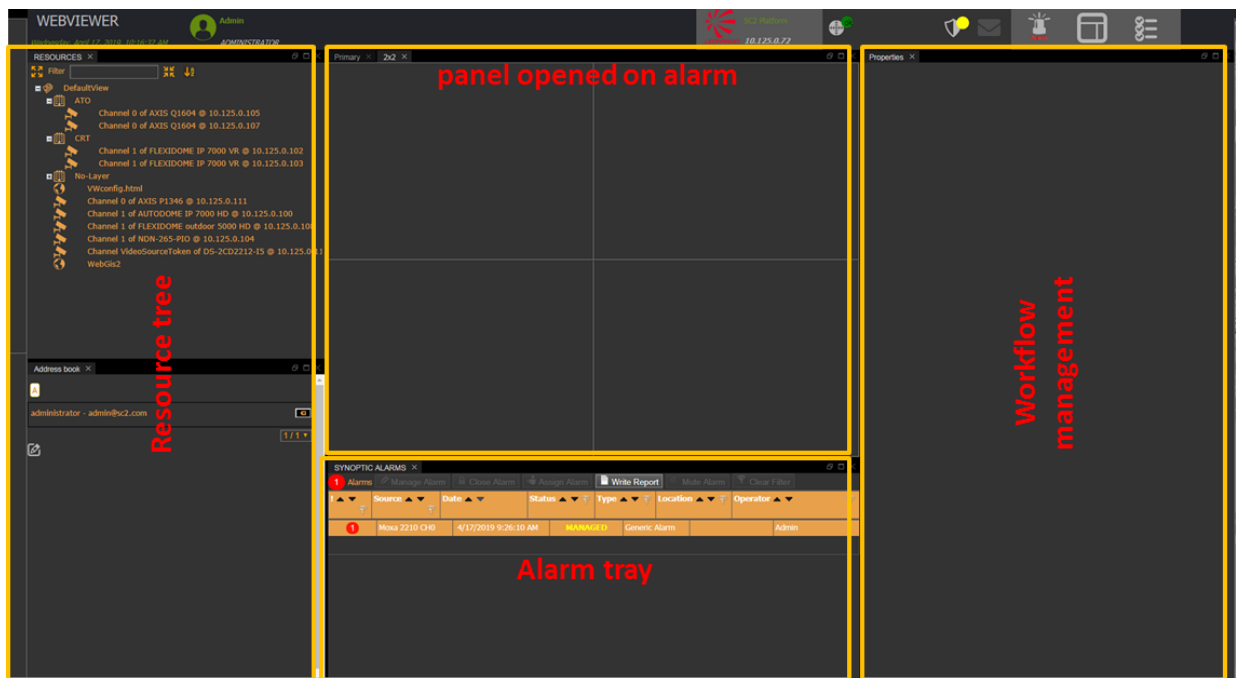


Figure 30 - Cockpit HMI layout Screen 2

As previously described in par. 3.3.1, Screen 2 displays three SC2 cockpit components:

- Resource tree,
- Alarm tray,
- Workflow management.

An additional panel is also displayed in Screen 2. This panel is opened on an alarm click and it can display, for example:

- alarm details (see Figure 31 below),
- video streaming of pictures or data,
- information for the operator in order to mitigate the incident.

8 Alarms		Manage Alarm	Close Alarm	Assign Alarm	Write Report	Mute Alarm	Clear Filter
▼ ▲ 🔊		Source ▼ ▲ 🔊	Date ▼ ▲	Status ▼ ▲ 🔊	Type ▼ ▲ 🔊	Location ▼ ▲ 🔊	Operator ▼ ▲ 🔊
3	Moxa 2214 CH0	10/23/2018 3:45:45 PM	NEW	Moxa Alarm	Zona2		
3	Moxa 2214 CH0	10/23/2018 3:10:26 PM	NEW	Moxa Alarm	Zona2		
3	Moxa 2214 CH0	10/23/2018 3:05:32 PM	NEW	Moxa Alarm	Zona2		
3	Moxa 2214 CH0	10/23/2018 3:03:34 PM	NEW	Moxa Alarm	Zona2		
3	Moxa 2214 CH0	10/23/2018 2:53:30 PM	NEW	Moxa Alarm	Zona2		
3	Moxa 2210 CH0	10/23/2018 2:52:06 PM	NEW	Moxa Alarm	Zona1		
1		10/23/2018 2:16:21 PM	NEW	Helio Alarm			

Figure 31 - Alarm details list

Details of the opened alarm are source, timestamp, alarm status (NEW, MANAGED), type of alarm and location. It is also possible to perform operations on the alarms, for example, to manage the alarm and perform the workflow, to assign the alarm to another operator and to close the alarm only if the workflow of management is completed.

Figure 32 below shows a panel within an HMI screen. It is possible to select a panel by clicking on it, while all other panels will be hidden by the selected panel.

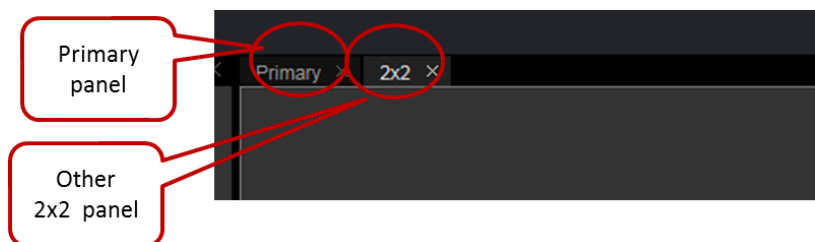


Figure 32 - Panels

A panel can be configured single (1x1) or multiple as in Figure 33 below.

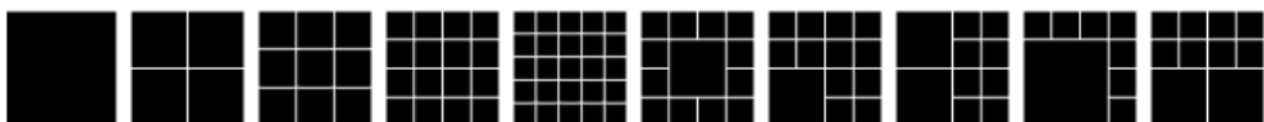


Figure 33 - Types of Panels

For example, a three screen HMI could be configured as in Figure 34.

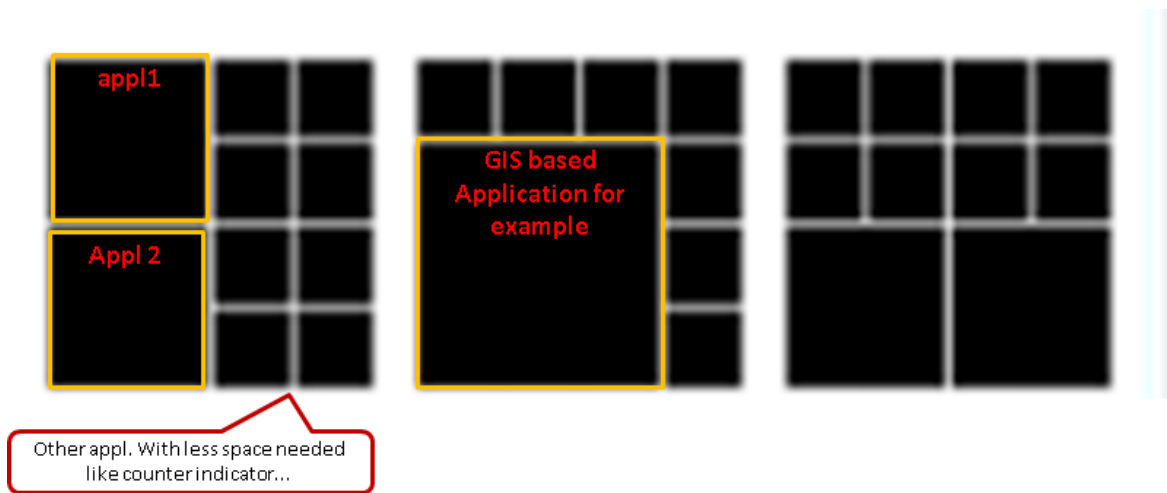


Figure 34 - Three-screen HMI

where:

- SC2 can personalize panels and can use multiple panels to show information that can't be seen in a single monitor. If needed, more panels can be added.
- Visualization Layouts are fully configurable and can be saved and restored.
- The Layouts can be customized for each user.
- SC2 is fully dockable so every component can be shown/hidden and moved into every screen and every position of the screen.

Each panel is configured as a set of tiles that can be 1x1, 2x2, ..., 5x5, etc. If it's needed to display more than one application simultaneously, SC2 offers the possibility to divide a screen into more than one zone. In addition, it allows configuration of different user layouts so that each user need view only a subset of applications. For example, a set of users could manage a class of alarm types and another set of users could manage a different set. Another option is to add superimposed panels on the same screen so as to allow display of more than one application in the same screen by alternately hiding and displaying the panels as desired.

For example, in Figure 35 one screen could host two panels: GIS and CISI Apro.

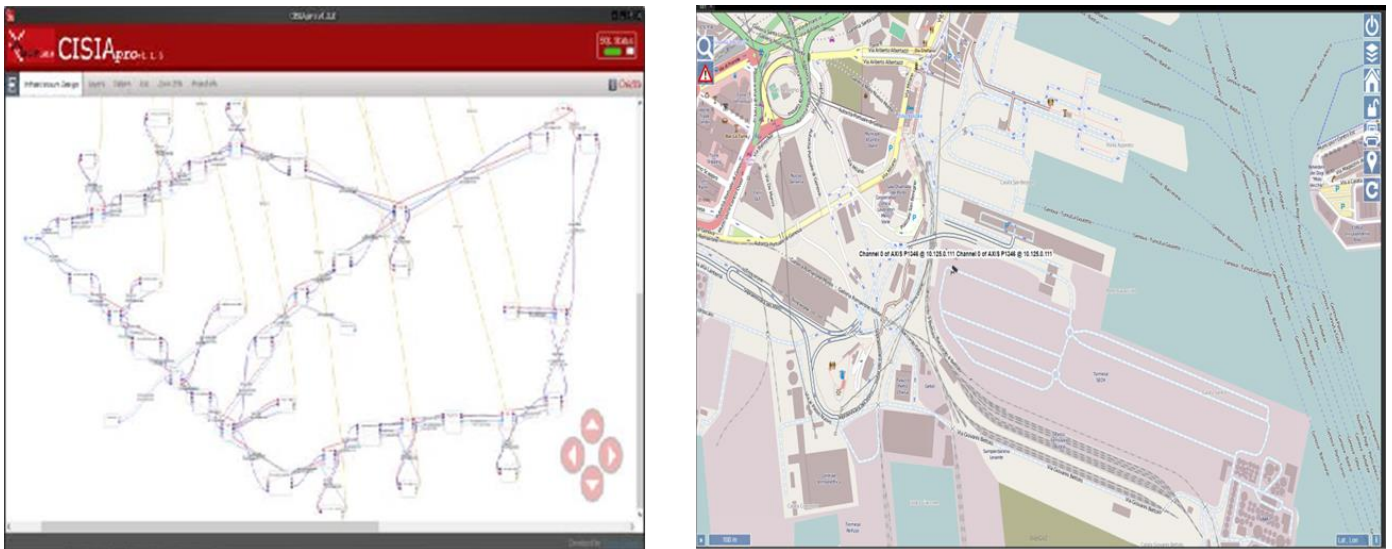


Figure 35 - Example: Two panels, GIS and CISIApro for one screen

By clicking on an icon on the screen a user will be able to view alternately the two applications on the same screen.

Another configuration would be to host the two applications in a 2-tiles panel.

In order to configure any cockpit layout a SC2 HMI administrator will first need to define the tiles and panels and their configuration on each screen. Then s/he will be required to insert applications in the desired configuration panels. Applications are seen by SC2 as web pages defined by their URLs.

The cockpit manages web pages as a type of resource that can then be viewed in the resource tree view. By dragging a resource of type web page from the tree view to the desired panel, the application end user will be able to view the application in the panel.

Each panel has an instance of a Chrome embedded browser inside which allows page rendering.

Once the layout design has been completed according to specific needs, it can be saved and assigned to an application end user so that when the user logs in, the layout of his or her competence appears.

3.3.3. HMI features – Reporting

The RESISTO platform will use the reporting features of SC2 based on queries performed on the system's data. SC2 query results can be exported and saved on Excel files.

SC2 search functions allow the users to query the system regarding the following items:

1. Events,
2. Alarms,
3. Actions,
4. Alarm Statistics,
5. User logs.

thus producing the following reports:

➤ Event Report

This feature generates event reports stored in the database, filtered for time range, event type, source that generated the event and zone.

For each event, the following fields are displayed in tabular format:

- Type: the type of event stored in the archive,
- Name: the name of the device that generated the event,
- Description: the description of the device that generated the event,
- Date: the date the event was generated,
- Zone: area associated with the event,
- Info: additional information, if available,
- Altitude / Latitude / Longitude: coordinates for GIS display, if available.

➤ **Alarms**

This feature generates the alarm report stored in the database and can be filtered for time range, alarm type, priority, and zone.

For each alarm, the following fields are displayed in tabular format:

- Type: the type of alarm stored in the archive,
- Status: alarm state stored in the archive,
- Date: the date the alarm was generated,
- Zone: zone associated with alarm,
- Source: resource that generated the event,
- Priority: priority associated,
- Operator Name: the name of the operator who ran the alarm,
- Info: additional information, if available.

Moreover, for each alarm, the user can get the list of the specific actions that were executed during alarm management (see the Action report below for details).

➤ **Actions**

This feature generates a report of alarm actions, filtered for time interval, action operator, alarm type, priority, and zone.

For each action, the following fields are displayed in tabular format:

- Type: the type of alarm associated with the action performed,
- Status: alarm status stored in the archive,
- Date: the date the alarm was generated,
- Zone: zone associated with alarm,
- Source: the type of resource that generated the alarm,
- Priority: the priority associated with the alarm Schema,
- Operator: operator name that has taken charge of the alarm,
- Action Type: Action Type executed,
- Report: report that was written after the end of an alarm,
- Operator: the name of the operator or application that executed the action,
- Action Date: the date the action was taken.

➤ **Alarm Statistics**

This feature generates statistics on:

- aggregated status (NEW, MANAGED, CLOSED),
- type (Intrusion, Diagnostic, Video analysis, etc.).

in a user-definable time period.

Each statistical quantity is displayed both in pie chart and tabular format.

➤ **Users Logs**

This function generates a report of actions performed by system users. The report can be filtered for time, operator name, and role.

For each user, the following fields are displayed in tabular format:

- User Name: the name of the operator who executed the action,
- Operation: operation performed,
- Date: date of execution of the operation.

Moreover, some RESISTO components may also have specific reporting functions for the domain to which they belong.

3.4. Platform Common Services

3.4.1. IAM

The RESISTO system uses an Identification and Authentication Management (IAM) component that allows users to be authenticated and to be granted with appropriate permissions.

The system is based on what is offered by the SC2 product. SC2's IAM system can use native authentication managed by the same system or interface with external market IAM systems, or it can use a Directory Server based on the LDAP communication protocol. For the needs of the RESISTO system the use of the internal SC2 IAM module is considered adequate.

All components of RESISTO that are integrated into the main HMI will use the RESISTO IAM. Operators will be registered and profiled by means of the administrative HMI and associated with a RESISTO user profile. It is in charge of the IAM component to take care of logging into the subsystems hosted in the main HMI. Therefore the HMI of each subsystem will be displayed at login on the main HMI, depending on the role of the operator. The operator will not have to repeat the login operation as it will be performed automatically by the RESISTO system with a Single Sign-on mechanism.

The authentication mechanism is based on username and password.

SC2 is equipped with an administration interface that allows the complete configuration of the system, in particular it is possible to configure users with their access credentials: username and password. An appropriate password policy can be managed.

The RESISTO users will be managed with a User web page that will allow the RESISTO IAM Administrator to create, delete, and edit user profiles. User privileges are defined by associating a user name with a role.

System users will have access to various features based on IAM profiling that assigns them rights. Each user is assigned a specific role in the system, rights are assigned to roles, and these rights allow operations (functionalities) to be performed on the system.

3.4.2. KSI Blockchain Infrastructure

KSI Blockchain is built and accessed through the KSI Infrastructure.

KSI Infrastructure is layered and hierarchical. KSI Blockchain is created and maintained by the KSI Core cluster; requests are accepted, and responses are distributed using a hierarchy of Aggregator nodes. The blockchain is distributed using a hierarchy of Extender nodes.

Lowest layer, customer-facing Aggregator and Extender are packaged into so-called Gateway server.

The core and aggregation, extender networks are operated as a permissioned blockchain, by Guardtime. Branches of aggregation and distribution hierarchies can be operated by third parties. In RESISTO project Guardtime operates all KSI Infrastructure components, providing access to pre-configured endpoints.

The gateway layer should be hosted on-premises for the best possible security and service quality.

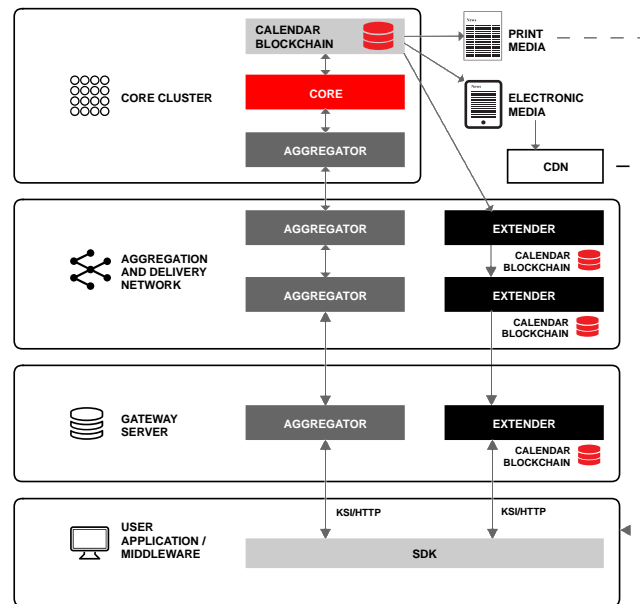


Figure 36 - KSI Infrastructure layers

KSI signatures are server-based, meaning that signing is only possible with online access to the KSI service. The verification of the signatures can be done both offline and online.

3.4.3. Knowledge Base

The Knowledge Base is the repository storing set of information about Communication CI configuration, security procedures, protection-reaction/mitigation strategies, recovery procedures. Data stored on the DB are secured and their integrity monitored and enforced via blockchain technology.

In order to provide guaranteed data integrity and accountability the data stored in repository is signed using KSI Blockchain (Figure 37). Each version of a 'document' is signed using KSI Blockchain and cryptographically linked with previous versions, and with source data if available, and with attribution information of source and responsible person or system, thus establishing a provenance chain of documents.

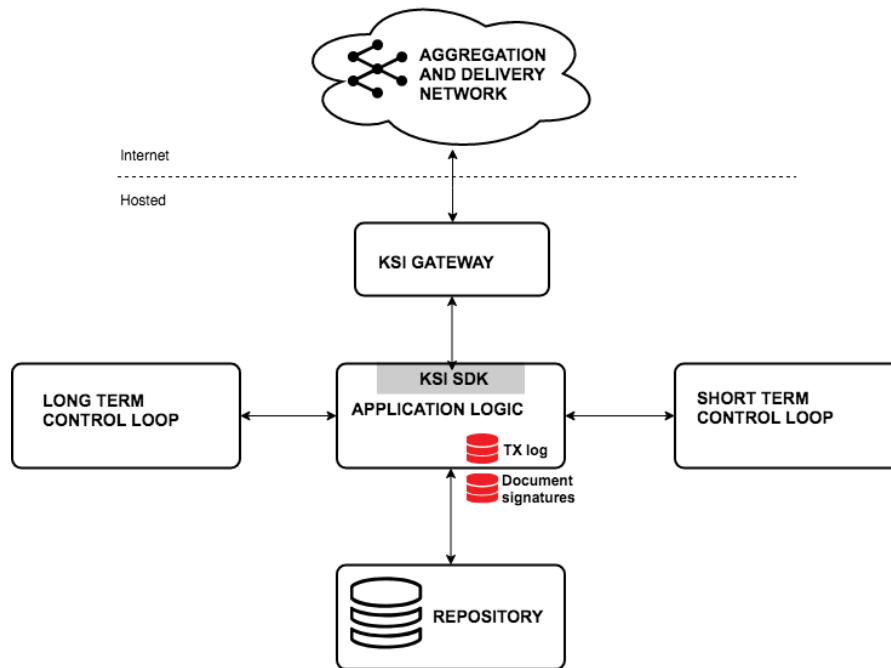


Figure 37 - Deployment model for Knowledge Base

In order to provide lower level transparency, accountability and insider threat detection all transactions are recorded in trusted transaction log where all entries are linked and signed using KSI Blockchain. In order to provide privacy of extracted proofs, the signed log is redactable.

3.4.4. Guardtime Machine Integrity

IoT device cybersecurity monitoring solution is based on Guardtime MIDA framework, further developed and extended with RESISTO specific message formats and IoT device security policies (see par. 3.5.4 for a description of RESISTO IoT devices). First, main underlying technologies are described, followed by a brief description of the MIDA platform, its adaptation for IoT cybersecurity use-case and integration with the RESISTO project.

3.4.4.1. XDAL and Dockets

XDAL, or the eXtensible Data Attribution Language, provides the basic data structure for the MIDA State Captures. XDAL defines the syntax and semantics of the MIDA construct called “Dockets”. These “Dockets” provide an interoperable and self-contained construct to cryptographically link data authenticity, identities, and contextual based information using the KSI Signature. The KSI Signature is the output of participating in the KSI Blockchain, and cryptographically proves the data has not been altered, provide the identity of the participant, and includes system agnostic time.

Once sealed with a KSI Signature, these Dockets enable the state captures from the MIDA Agents and Services to be portable, allowing them to be verified across boundaries and in perpetuity. In MIDA, the dockets are used to encapsulate and protect the identities of the agents or services, data authenticity of the state capture and time associated with the capture. These can then be cryptographically verified by the State Management Services, owners, and auditors.

3.4.4.2. Guardtime MIDA

Guardtime MIDA (Machine Integrity, Defense and Awareness) provides capabilities for machine and environmental integrity state capture, defense, and remediation for distributed, cloud, and IoT environments. The main capabilities of MIDA are providing awareness and reporting, remediation techniques, malicious or accidental state change detection, and enhanced analytics.

The primary components of the MIDA are:

- Guardtime KSI Blockchain – provides the distributed trust anchor for Docket Verification.
- MIDA Services – provide distributed data capture for AWS/Azure events such as instance creation or security group changes and KSI Blockchain integration (not used in the RESISTO project).
- MIDA Agents – provide distributed data capture for system state and configuration changes. These reside within the actual IoT sensors and devices and monitor events such as file changes, directory changes, and processes. Data is pushed to MIDA State Management Services, usually through the Sentry/
- MIDA State Management Services – provides backend event analysis, correlation, exchange and storage management. Consists of the Sentry (pre-validating ‘firewall’), Broker (initial message routing, storage), and Venture (workflow, alerting).
- MIDA Dashboard – Web UI which provides visualization, event correlation and alerting; and is interface for auditing and forensic analysis.

A logical overview of the MIDA is depicted at Figure 38. Here, alerts are forwarded to RESISTO Cyber/Physical Threat Correlator for further processing. More details will be given in Deliverable 4.3 - “Techniques and Procedures for cyber/physical threats Detection”.

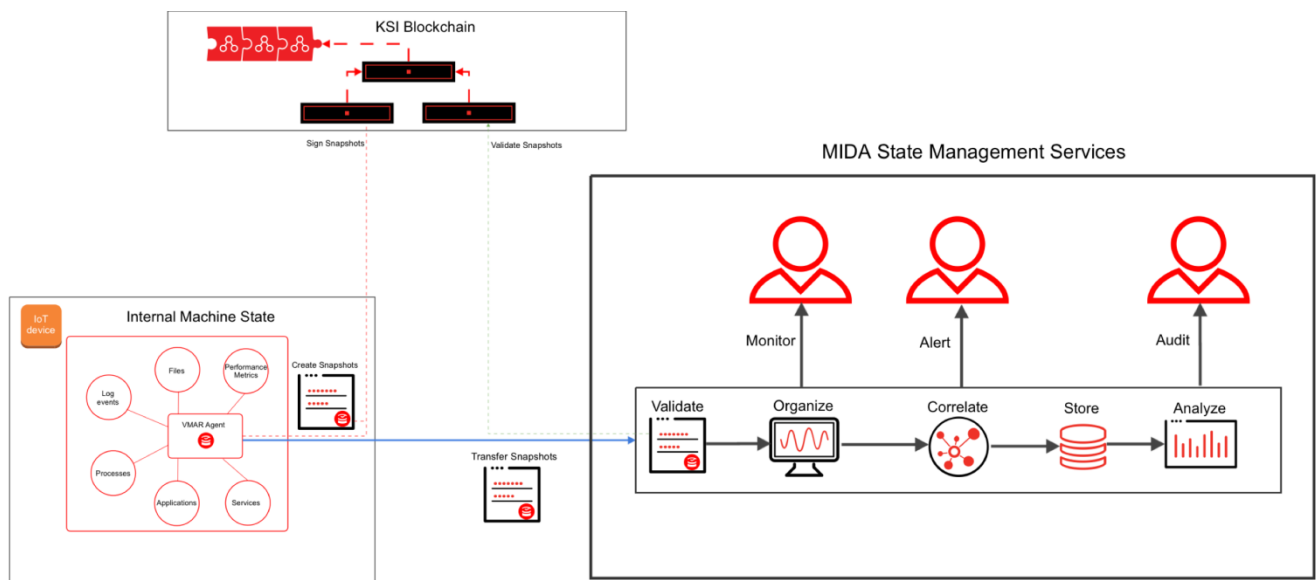


Figure 38 - Guardtime MIDA

3.4.4.3. Why KSI Blockchain

In providing protection against cyber threats to IoT sensors in the RESISTO project the KSI Blockchain provides auxiliary function. By capturing monitored system's state changes in docket and anchoring docket in blockchain we achieve following benefits, most of which are not efficiently achievable using state-of-the-art technologies:

- Data is immutable and associated with producing system's identifier,
- Data is cryptographically time-stamped,
- Integrity is protected for long term, measured in tens of years; even if efficient quantum computer materializes,
- Signed containers are exportable and transferrable without key management efforts,

- Signed containers are suitable for audit and forensics; even privileged users cannot fabricate data,
- Process Integrity of data capture, transfer, processing and storage processes can be demonstrated,
- Key leaks do not have catastrophic effects.

3.4.5. Responsible Disclosure Framework

Responsible Disclosure Framework (RDF) provides tools and a feature-rich system that allows security researcher, independent contractors and other 3rd party security providers to report discovered vulnerabilities on the end user's infrastructure while they stay in contact with stakeholders responsible for the system in scope throughout the internal Software Development Life Cycle & Patch Management procedures.

The framework has several critical modules:

- Configuration and administration
 - **Scope Definition & Management of Assets** allows end users to define what is the scope of testing for security teams, vulnerabilities to be ignored, responsible disclosure & bug bounty policies;
 - **Vulnerability Triage** allows end users to define teams and pre-verify any reported vulnerability against duplicates, false positive before opening a fix procedure;
- Vulnerability life cycle
 - **Threat Classification engine** allows both security research and end users team to classify or re-classify findings during the patch management process using Risk Rating Methodologies, identified in the context of for RESISTO project, such as OWASP ([26]) Risk Rating Methodology or Common Vulnerability Scoring System;
 - **Vulnerability lifecycle GUI & Rewarding module**, Vulnerability lifecycle is managed via a user friendly GUI to register and track the progress of the vulnerability, this in turn will trigger the Rewarding module to reward security contractors (which includes the Hall of Fame module update) depending on the finding type and impact which would allow teams to prioritize the findings and track the patch process progress.

RDF main functionalities are reported in Figure 39.

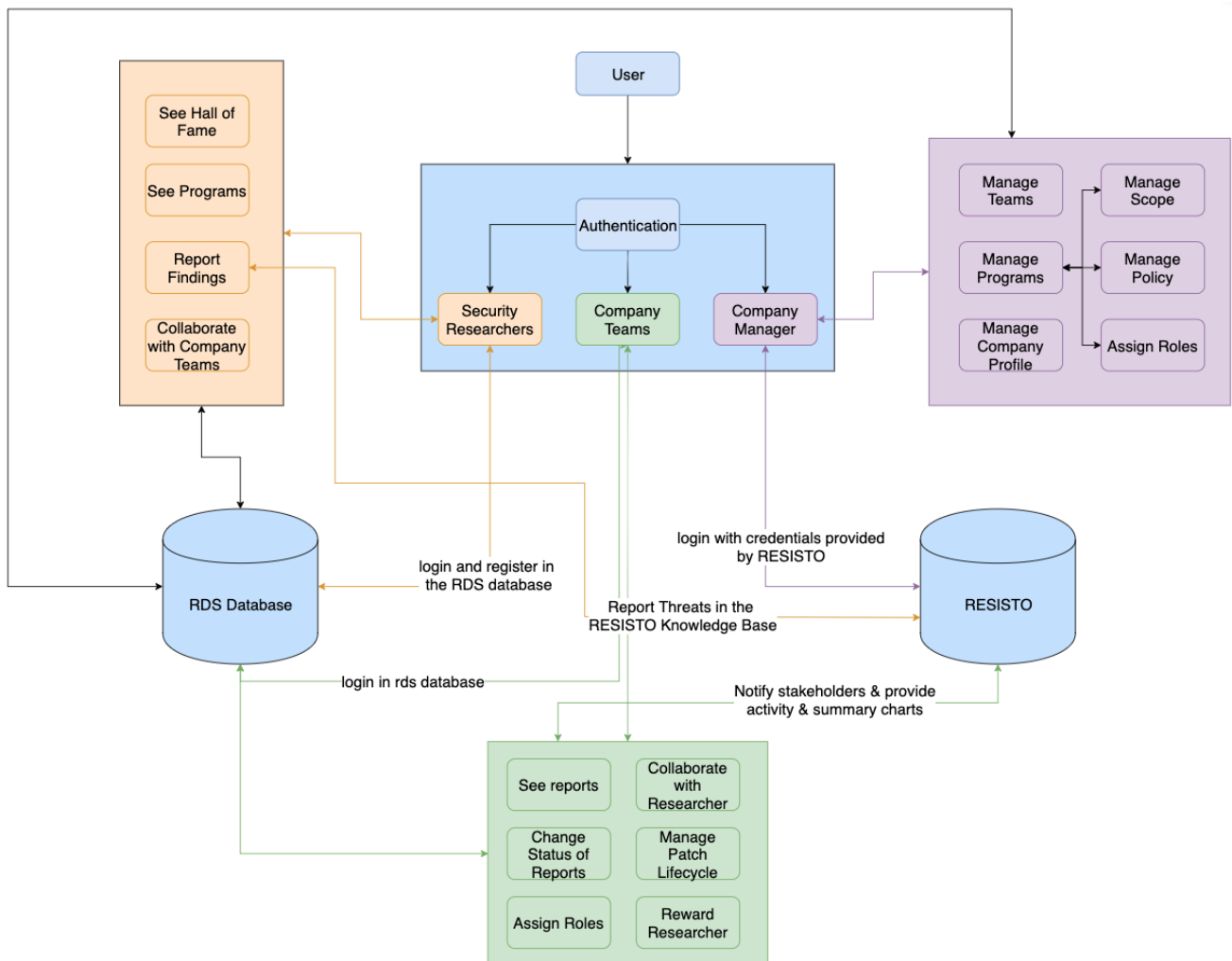


Figure 39 - Responsible Disclosure Framework block diagram

The workflows in the following figures describe the main functionalities, based on user type as follows:

- **Company Manager:** users authorized & defined in the RESISTO platform that can manage their companies and teams (Figure 40);
- **Company Teams Members:** users of the end-users authorized to manage reported bugs to certain systems defined the scope (Figure 41);
- **Security Researcher:** any security researcher, independent contractors and other 3rd party security providers that want to report bugs to the end-user systems which are defined in the scope of testing (Figure 42).

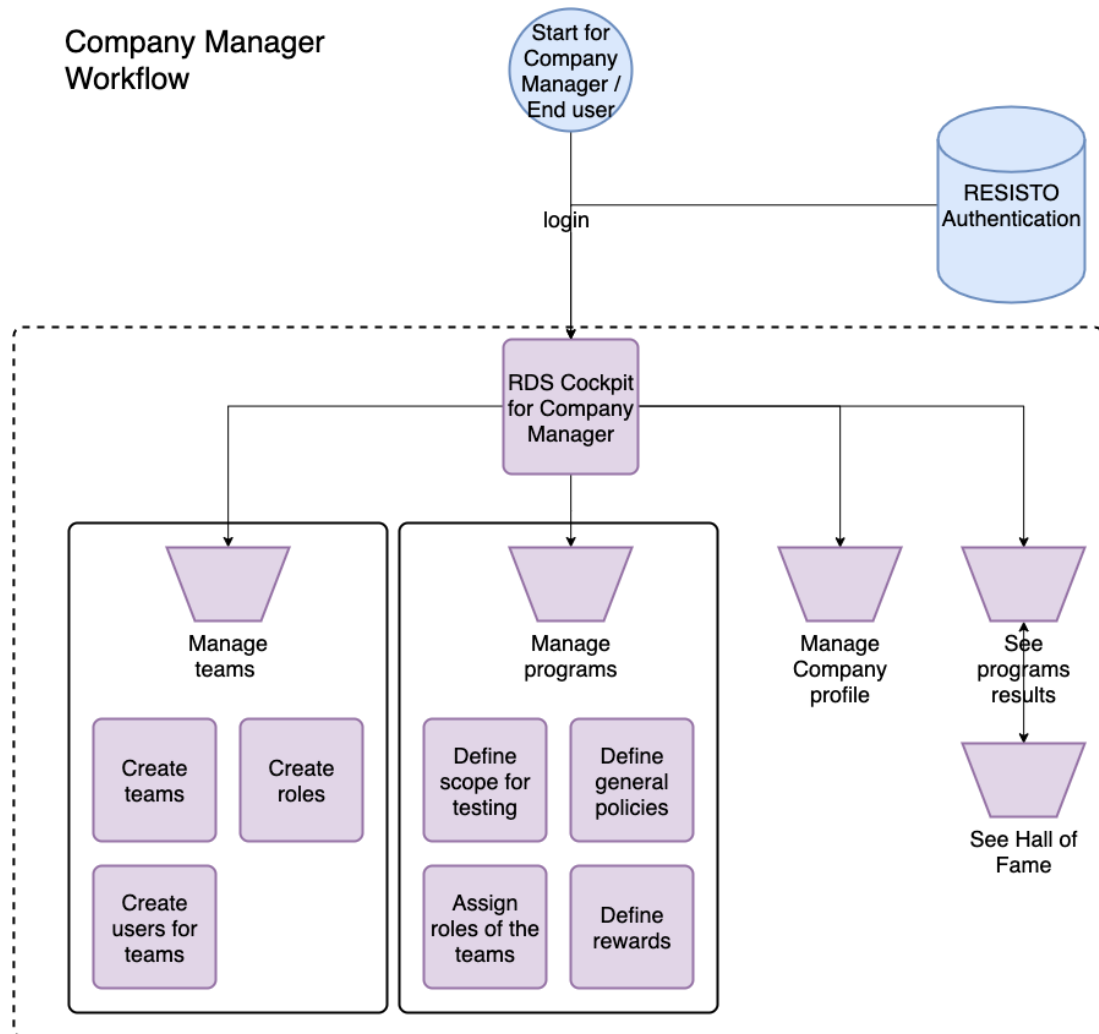


Figure 40 - Responsible Disclosure Framework: Company Manager workflow

Company teams Workflow

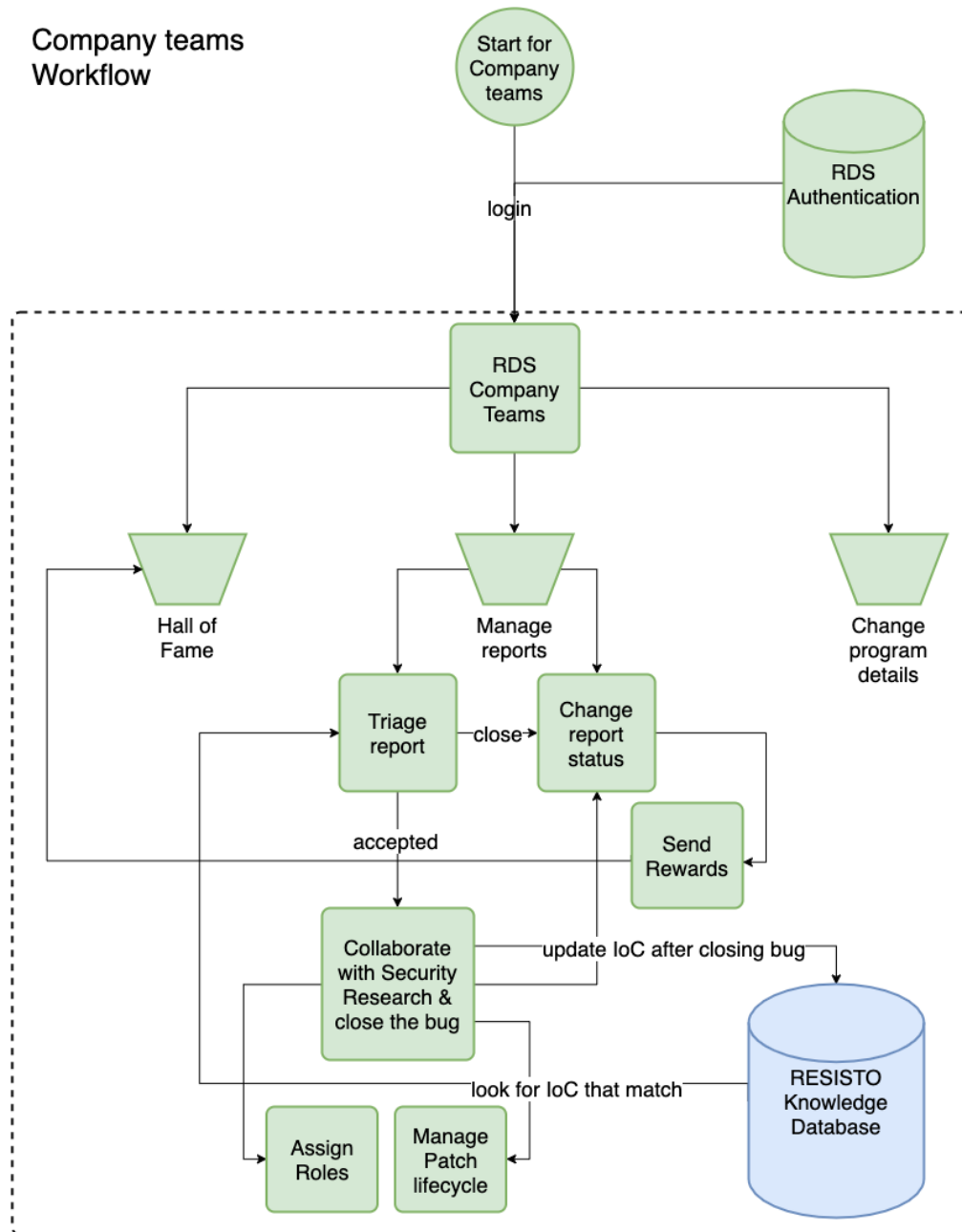


Figure 41 - Responsible Disclosure Framework: Company Teams workflow

Security Researcher Workflow

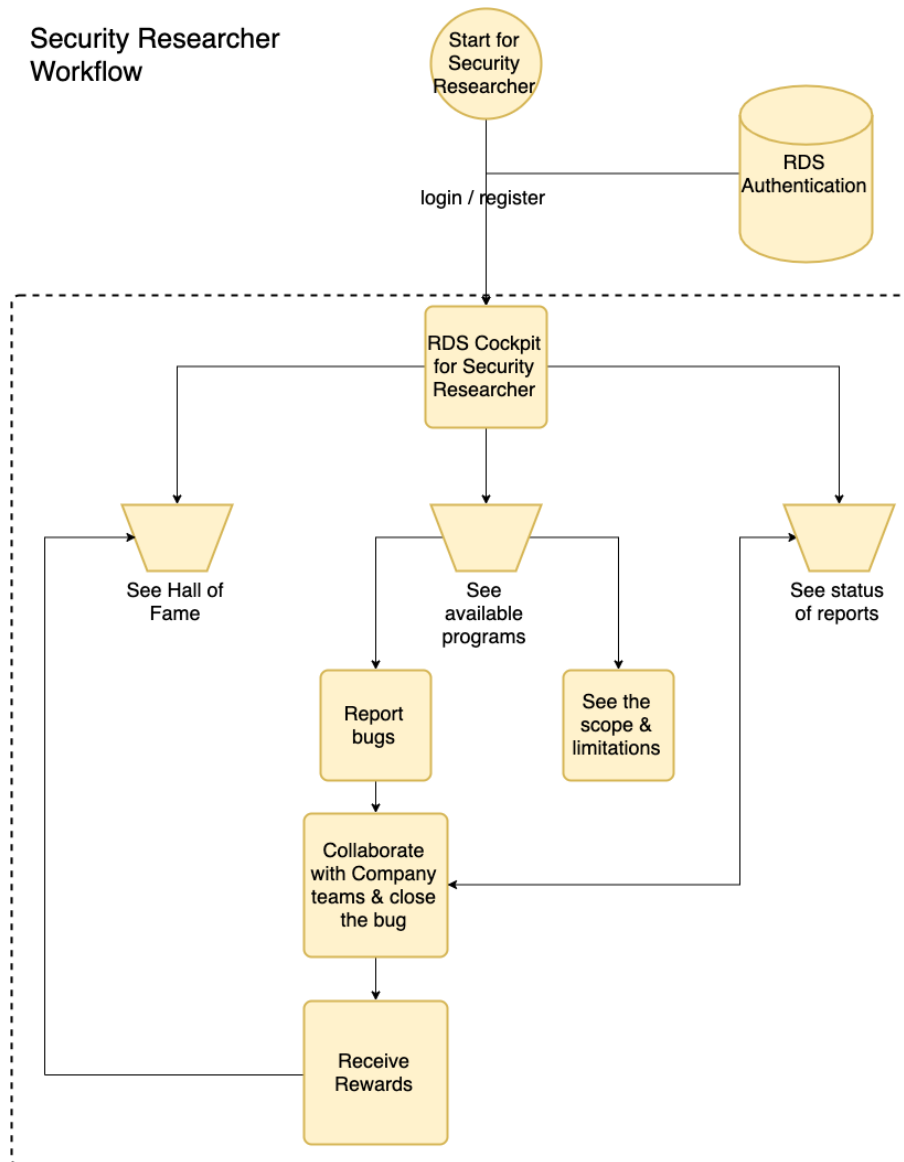


Figure 42 - Responsible Disclosure Framework: Security Researcher workflow

The Responsible Disclosure Framework proposed by Bit Sentinel Security provides a series of benefits for parties involved in this process:

- For security researchers, independent contractors and other cyber security services:
 - A disclosure guideline and a program policy designed to guide security research into a particular service, product or IoT device;
 - a simple interface to report and follow the fixing process during the whole lifecycle.
- For end users:
 - Establish a compliant process for receiving and acting on vulnerabilities discovered by third-parties during any penetration testing engagement;
 - Align with NIST best practices for accepting and managing security feedback;
 - Allow teams to prioritise the findings and initiate, supervise and close patching process for individual vulnerabilities while staying in contact with security researchers;
 - Functionalities to integrate internal or third-parties triage team do the heavy-lifting;
 - Promote a positive relationship with the security researcher community.

The framework will have to integrate with external sources to achieve the following

- a) Functionalities to integrate internal or third-parties triage team do the heavy-lifting;
- b) Promote a positive relationship with the security researcher community.

RDF is integrated in the flow of RESISTO as a tool/system to collect discovered vulnerabilities in the telecom provider development organization. The vulnerabilities once collected will need to be corrected and the tool allows tracking of the corrections throughout the internal Software Development Life Cycle & Patch Management procedures (Note: for SW that the telecom provider outsources the framework can be used to handle only the registration/patch management).

For instance one of the events from the OSINT flow can report a new vulnerability to the knowledge base, if there is not already a patch for the vulnerability the patch lifecycle must start, if a patch is already available a plan must be made to deploy it.

The company can define both Public and Private programs in order to have a balance between the quality of reports and the risks they are exposing the networks. The company can decide if a vulnerability will be publicly disclosed or if they will keep it private, only as an internal future reference (Figure 43).

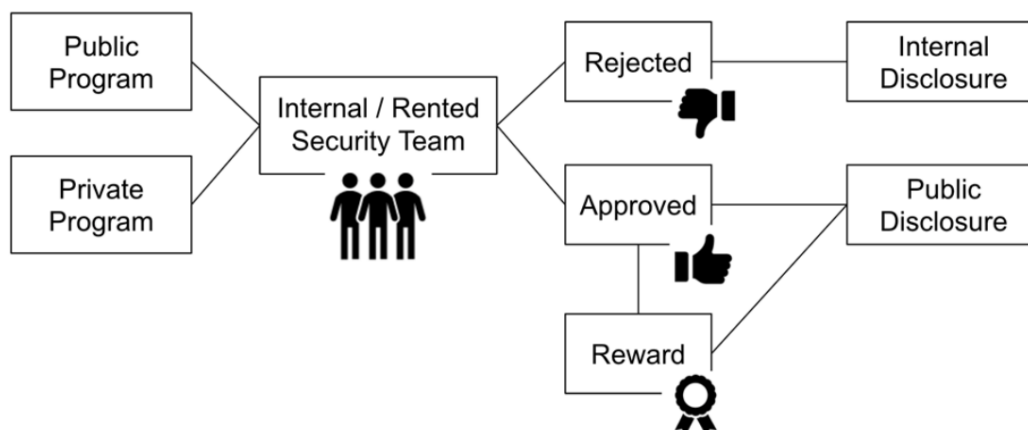


Figure 43 - Responsible Disclosure Framework: Programs and Disclosures

The Responsible Disclosure Framework aims at integrating vulnerability management functionalities within RESISTO framework by:

- fetching required information from the Knowledge Database, in order to define the potential vulnerabilities;
- providing all the required functionalities for the internal teams in order to:
 - define the scope and rules for testing, for example:
 - IP range 192.168.0.0/24 is out of scope,
 - *.site.com are in the scope of testing,
 - CSRF (Common Source Route File) vulnerabilities are out of scope;
 - define the rewards for vulnerabilities based on their potential impact score (according to Risk Methodology):
 - after the stakeholders of the system in scope for the reported vulnerability validates the issue and fix it, they can choose to credit their profile and then reward the

researcher in cryptocurrency or cash or simply give a certain number of points which that are considered for the Hall of Fame (a ranking system which keep track of the results of the most active researchers);

- report potential vulnerabilities:
 - security researcher, independent contractors and other 3rd party security providers can report vulnerabilities by selecting any of the assets within scope defined by the end-users and then provide technical information about the vulnerability;
- validate the potential vulnerability:
 - after the vulnerability is reported, a unique ticket id is opened and the stakeholder is notified; stakeholders, which are defined by the end-user, can triage the issues and assign staff to patch the vulnerability while they can stay in contact with the user who reported the bug for updates or clarifications;
- monitor the vulnerability through the entire patch cycle, functionalities available in different form for researchers but also for the stakeholders:
 - report the finding,
 - confirm/reject/request additional information from the security researcher,
 - notify the stakeholders,
 - patch the finding,
 - confirm from the security researcher that the issue was fixed,
 - reward the security researcher, if appropriate;
- Hall of Fame module that enables the company to motivate both internal and external security specialists to find bugs in order to both receive rewards but also remain on the Hall of Fame page, where the best security researchers are displayed;
- Rewarding module, that enables the company to reward the security researchers by swags, points for Hall of Fame or other forms or remuneration.

3.5. RESISTO detectors

3.5.1. Airborne Threats Detector

Considering the emerging use of unmanned devices, UAVs (Unmanned Aerial Vehicles) or drones are nowadays more and more regarded as potential human-driven, physical threats. Within RESISTO, mixed techniques involving low cost radars combined with acoustic sensors are implemented to detect small airborne objects and moving targets. The relevant airborne threats detection system is a lab prototype offered by ICCS and can act complementary to other relevant, more sophisticated radar sensors provided within the project.

The specific airborne threat detection system is a set of tools designed and developed to detect the presence of small UAVs and drones as airborne threats and provide alarm signals. The system can be also deployed in small unprotected areas such as antenna telecom parks providing additional situational awareness and perimeter defence against low-flying threat aircrafts.

The system consists of active and passive sensors namely radar and acoustic sensors respectively.

The radar sensor is of a Doppler type, able to detect and track fast moving targets providing good visibility in harsh conditions (dusk, rain or snow). In contrast to optical or infrared systems, Doppler radars detect more accurately small sized objects in the (optimal) 3 GHz - 30 GHz frequency band with a sensing range of several kilometers. ICCS exploited and tested its existing Doppler CW (Continuous Wave) radar lab prototypes at microwave frequencies (X-band / 10-12GHz, K-band / around 24GHz, 2.5GHz and 875MHz) with the 24GHz band yielding optimal results. However, radar's

detection capability depends on the target's RCS (Radar Cross Section) and its ability to distinguish between small objects and clutter that the sensor will also pick up especially for objects with low RCS as the drones are. This is tackled with advanced signal processing and the combined use of the acoustic sensors.

The acoustic sensors used in this system are a set (array) of high sensitivity dynamic microphones for a low cost, low power combination. Acoustic sensors have many advantages that include non-line-of-sight, omni-directionality, passiveness, low-cost and low-power, playing a potential key role in situational awareness; since they do not depend on the target's size, but rather on its acoustic signature i.e. sound of the engine. The acoustic microphone arrays are used as a second sensor modality to detect broadband acoustic emissions from approaching targets. By exploiting the target's strong emitted sound harmonics, moving targets can be detected by acoustic sensors, through tracking of the strong sound harmonic lines they emit (in the 20Hz–2kHz range).

Advanced signal processing and machine learning techniques are applied to the radar and acoustic data, both in the time-domain and the frequency-domain to achieve detection and to estimate the target's angle of arrival. The above system's tools and components (radars and acoustic sensors) can be either used separately or in combination, through a multiplexing console and a windows-based computer (laptop or desktop) that performs the sensing-data processing and is physically connected to the sensors. Target detection visualization is enabled through the Labview software environment.

In the framework of RESISTO, the technical assessment through the relevant use cases and scenarios will be held with commercial drone platforms (i.e. DJI Phantom 3 Advanced Drone) along with the UAV platforms of ADI (ADITESS). To comply with newest technology trends, implementation of mixed techniques will be pursued; potentially in conjunction with visual methods (i.e. cameras), depending on the ADI platforms payloads. More detailed description of the ICCS airborne threats detection system and tools is already given in [9].

Concerning the integration of the above system within the overall RESISTO architecture, it should be noted that the sensing tools may act as plug-in modules providing alerts to the PSIM part, affecting mainly the short-term control loop. The overall detection system is a standalone one and thus an interface to the overall RESISTO platform with Data Integration Layer, based on Mule Enterprise Service Bus, will be defined in order to make easy data exchange between the two systems.

Concerning the physical connection to the RESISTO platform, the detection system can be either connected to the same local network (LAN) with the RESISTO platform or, at least, can be IP visible from the RESISTO platform, depending on the final configuration that will be defined towards the project evolution and integration related aspects. The RESISTO correlator will provide a pub-sub Message Queue on which the potential threat detectors, will be able to publish potential threat events.

Consequently, and since the aim of the airborne threats detection system is to extract and provide potential intrusion events corresponding to the presence of potential moving airborne threats (UAVs and drones), a threat event with relevant attributes will be provided.

The data exchange between the ICCS detection system and the RESISTO platform will use JSON-formatted data objects corresponding to events.

3.5.2. Audio/Video analysis

Video and Audio sensors are widely used in surveillance operations and protection of critical infrastructures. Intelligence algorithms are applied in audio and video streams for the real-time detection of events for the early identification of illicit activity. Pattern recognition and machine learning techniques are used to extract acoustic events (i.e. gunshot, screaming, glass breaking) or to classify persons, vehicles and other objects that are moved within the controlled by the infrastructure area. Both the audio and video analytics modules form an intelligence surveillance system where the security operator is notified with an alert about the suspicious activity accompanied with important

information such as location (source) and type of the event, detected objects, etc. This intelligent process reduces the effort of the operator by monitoring in a 24/7 base a huge number of sensors.

The Intelligent Audio Analytics Component (AAC), allows the detection of abnormal behaviour regardless of the field of view, while also allowing the triggering of the system with the occurrence of pre-defined keywords. The solution implements well established methods from the fields of audio coding, machine learning and speech recognition and allows efficient operation on low cost power limited devices (or embedded systems) for the detection of screaming, glass breaking and gunshots within the environment. The Video Analytics Component (VAC) provides the necessary functionalities for visual surveillance analytics, aiming to identify and provide methods that abstract the information of interest contained in video surveillance streams.

The design of Intelligence Surveillance System including both the AAC and VAC is based on two different processing levels, one with reduce processing capability (infrastructure based on embedded system, i.e. Raspberry PI 3) and one with enhanced processing capability (i.e. GPU enabled computers or servers). Based on the application, audio or video, several components will be deployed on each level of processing. AAC is composed from two main components: the audio feature extractor and the classifier which are deployed on the first level and second level respectively (Figure 44).

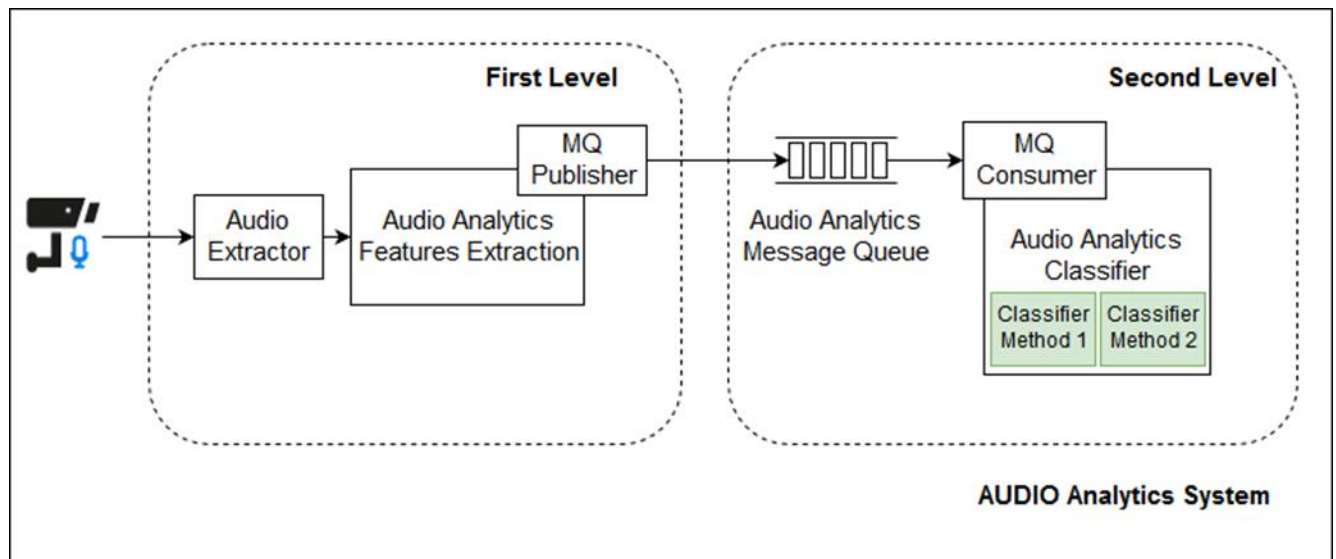


Figure 44 - Audio Analytics System

Similarly, VAC component is divided in several sub-components according the functionalities. Lightweight components such as motion detection are deployed on embedded systems (first level) while more demanding components such classification and tracking are run on the second processing level (Figure 45).

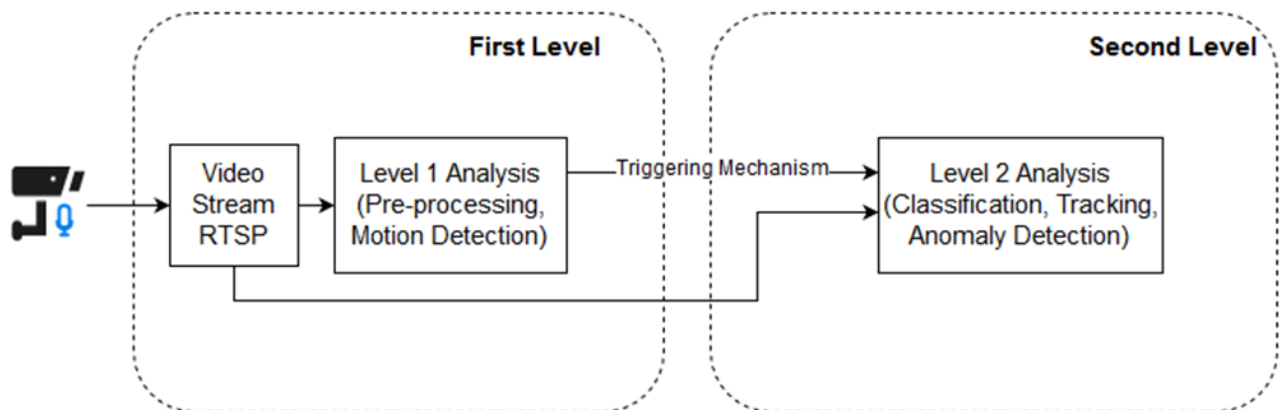


Figure 45 - Video Analytics System

Additionally, within RESISTO project, audio and video analytics modules will be enhanced with some other components in order to support the smooth operation, logging and correlation of the events. In particular, a local database will be used to store the generated alarms, fusion component in order to correlate alarms that are generated from more than one sensor, cross-cue component which is responsible to send a PTZ order to a camera in case of and acoustic event. Furthermore, Mini-UAV System components that support the deployment of Mini-UAV platforms as part of the surveillance operation (aerial image/video) will be deployed on the second level resources.

Figure 46 presents a higher level of the intelligence surveillance system.

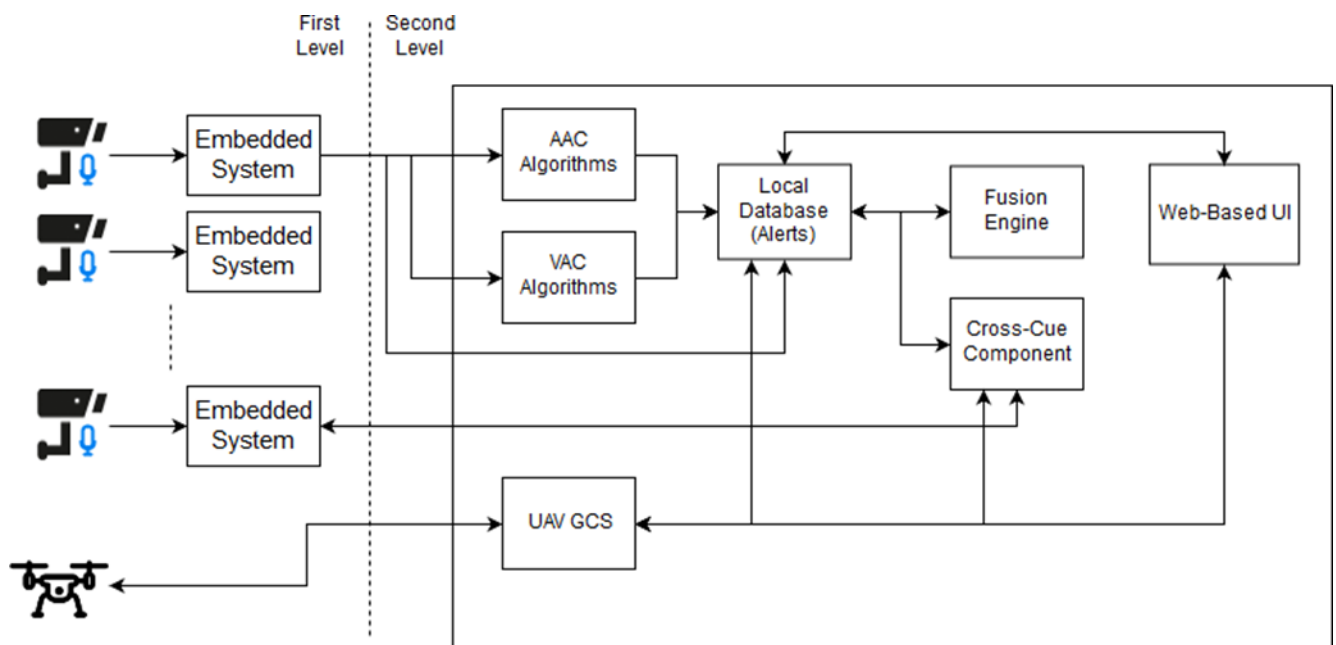


Figure 46 - Intelligence Audio/Video Surveillance System

The interfaces and internal communication of these components are based on well-known protocols such as REST API, Message Buses (ActiveMQ), RTSP, GStreamer, etc. Table 3 summarizes the input and output interfaces of each component.

Component	Input	Output
Audio Extractor	Audio Stream (from sound card, gstreamer)	Audio Chunks (predefined window)
Feature Extractor	Audio Chunks (predefined window)	JSON with features
Classifier	Features (JSON)	Event (JSON) with analysis information (i.e confidence)
Video Stream (Video Proxy)	RTSP (or other) from camera	RTSP
Level 1 VAC	RTSP	Event (JSON), Trigger to VAC 2
Level 2 VAC	RTSP, Trigger (from VAC 1)	Event (JSON)
Local Database	REST API	
Fusion Engine	Message Bus Consumer (Events)	Message Bus Consumer (Events)
Cross-cue	Event (Message Bus or REST API), JSON	Order to PTZ or UAV (JSON)
UAV GCS	MAV Link, Video Stream	Events, Metadata (JSON)

Table 3 - Input and output interfaces for the audio/video analytics component

3.5.3. Smart Spectrum Surveillance

Smart Spectrum Surveillance is a set of tools, RADIOFILTER and RANMONITOR, being developed by INTEGRASYS in the context of RESISTO project which makes use of IoT sensors for the detection of physical events/threats in telecom critical infrastructures. In the next subsections RADIOFILTER and RANMONITOR architectures are described.

3.5.3.1. RADIOFILTER

RADIOFILTER is a standalone system used for the detection of non-authorized Access Points (AP), Bluetooth and Wi-Fi devices as well as connections in manned facilities as well as intrusion detection in unmanned facilities.

The architecture (Figure 47) is made up of the two following components: Monitoring Wireless Sensor Network (WSN) and Central Processing Node (CPN) / Server in a wired star topology where the CPN/Server is the central node. Below, a description of each of these components along with their respective inputs and outputs follows:

- **Monitoring WSN.** Set of monitoring sensors distributed across the facility to be protected. Each of the sensors include a HW platform and a pair of Bluetooth and Wi-Fi transceivers which can capture, identify (MAC addresses) and analyze the packets sent by any device using these technologies (Wi-Fi AP/client and Bluetooth devices). This information is relayed by the sensors to the CPN / Server.
- **CPN / Server.** This central node aggregates the information gathered by each of the monitoring sensors and processes it, then checking against a whitelist of allowed devices/AP that should be an input from the end user (telecom operator). This way, the presence of a person using a device which is not whitelisted would prompt an event which is reported to the RESISTO platform STCL as an IDEA/JSON message. The same goes for any rogue Wi-Fi AP and also unauthorized device connections. On top of that, the system will add a location feature that would approximately compute the position of the unauthorized device/AP. This central node also acts as a web server for the cockpit in the platform where the end user can visualize the user interface with the most relevant parameters and also configure the input parameters (whitelists, update rate, bands monitored, etc.).

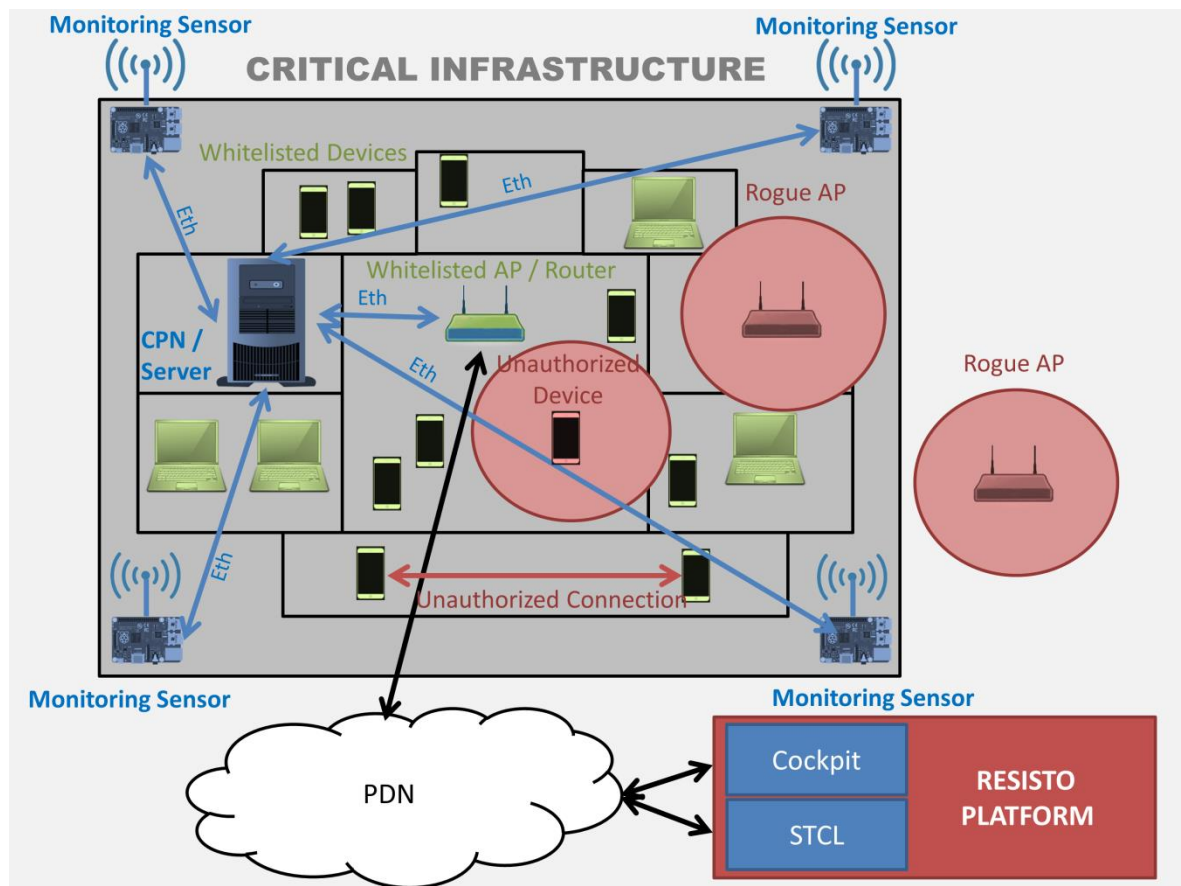


Figure 47 - RADIOFILTER Architecture

3.5.3.2. RANMONITOR

RANMONITOR is a standalone system used for the detection of IMSI-catchers/rogue base stations, misconfigured small cells and interferences to the cellular network (intentional and non-intentional).

The architecture (Figure 48) is made up of the three following components: Monitoring cellular modem, a Software Defined Radio (SDR) Spectrum Analyzer and a Central Processing node (CPN) / Server. Below, a description of each of these components follows:

- Monitoring cellular modem. A multi-band, multi-RAN modem able to detect and identify the surrounding cells in a specific area.
- SDR Spectrum Analyzer. SDR platform able to monitor the radio spectrum in a specific area.
- Central Processing Node (CPN) / Server. This central node receives monitoring and ID parameters from the cellular modem as well as radio spectrum data from the SDR Analyzer and process it, then checking against a whitelist with the cells that are expected/authorized to be in a specific area (based on a central location and a specific coverage radius). This way, the presence of any rogue base station/ cell which is not whitelisted would prompt an event which is reported to the RESISTO platform STCL as an IDEA/JSON message. Radio interference events (intentional or not) are also detected and reported to the STCL. This central node also acts as a web server for the cockpit in the platform where the end user can visualize the user interface with the most relevant parameters and also configure the input parameters (whitelists, update rate, monitored bands, monitored technologies, etc.).

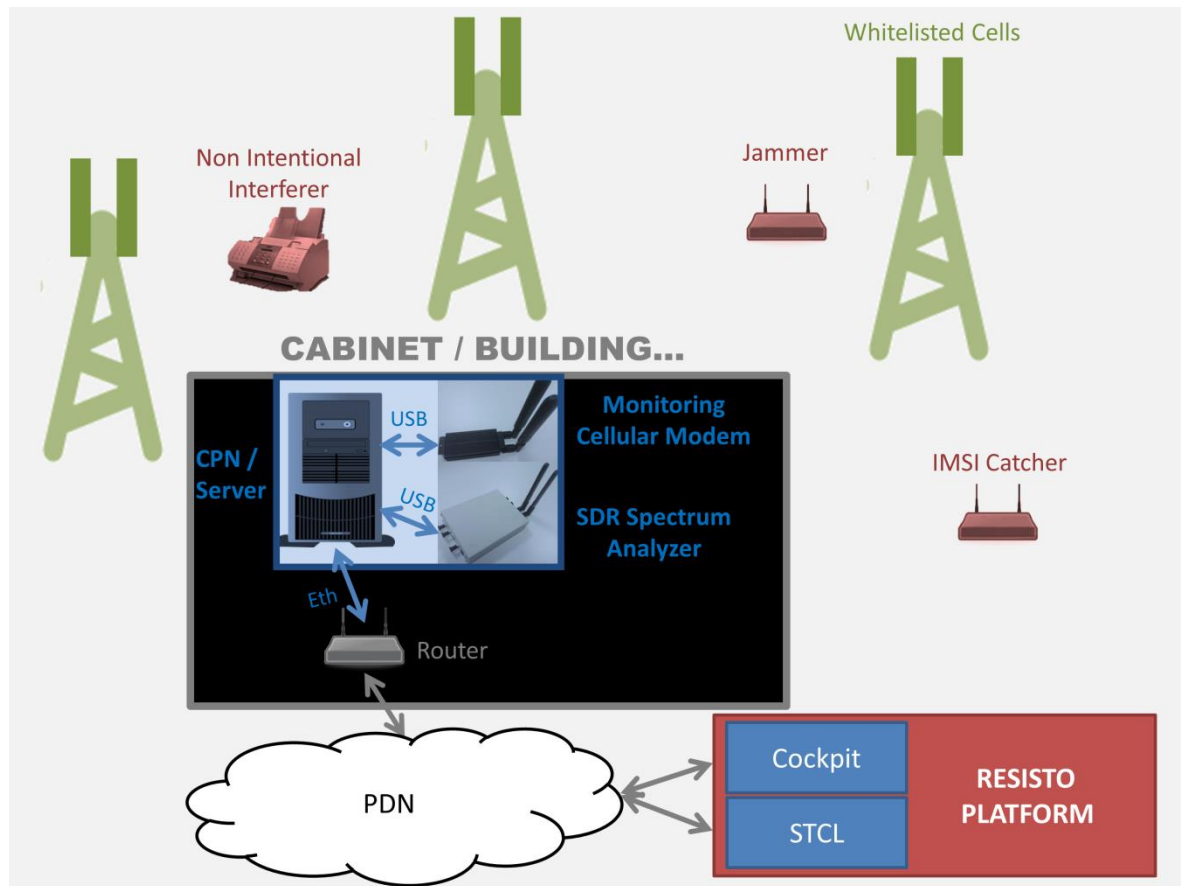


Figure 48 - RANMONITOR Architecture

3.5.4. IoT Based Sensors

IoT Sensor networks may gather sensitive data or be used by a malicious adversary to conduct attacks. Therefore, security is a key concern for such networks and for that reason, particular attention is paid to secure the sensors themselves.

In RESISTO, the solution researched for securing the sensors will be based on the premise of having a secure boot and up-to-date software in the hardware platform of each used sensor through secure periodic firmware updates.

The architecture (Figure 49) for this setup is made up of the three following components: A Secure IoT Sensor Network, a Firmware Update Server and an Auxiliary Authentication Server. Below, a description of each of these components follows:

- **Secure IoT Sensor Network.** This is the set of sensors to be secured. In order to achieve this, the sensors periodically poll the Firmware Update Server to query if there are secure firmware updates available. In case there are, after authentication both from the sensor and server side (a process in which both the Firmware Update and the Auxiliary Authentication Server exchange messages with the sensors), a digitally signed firmware image available from the Firmware Update Server is downloaded and verified (integrity and authenticity) by the sensors. If everything is correct, the sensors install the new firmware version. The connection from the sensors to the server is further secured through the use of two-factor authentication (Firmware Update Server over Ethernet and Auxiliary Authentication Server over a BLE backchannel).

- **Firmware Update Server.** This is the main server and is in charge of detecting new available patches which fix security vulnerabilities as well as generating a signed image that is available to the sensors. This server should be certified as a trusted entity and located in secure premises.
- **Auxiliary Authentication Server.** Auxiliary on-site server used in the process of two-factor authentication to increase security.

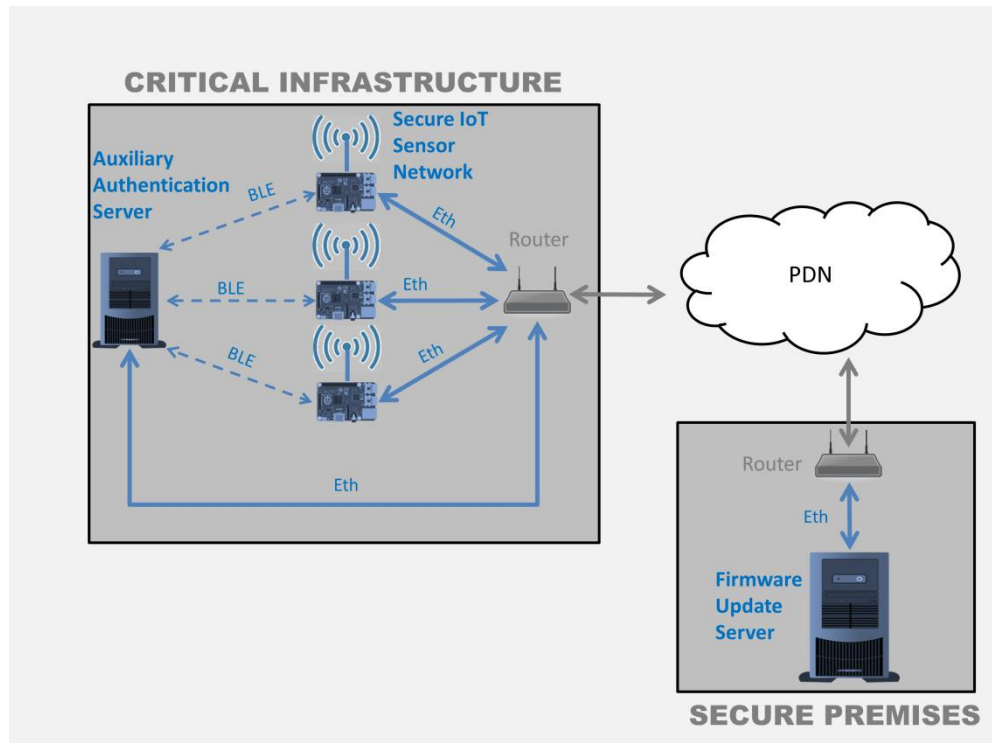


Figure 49 - IoT Based Secure Sensors Architecture

3.5.5. OSINT and the Machine Learning based Threat Detector

Threat intelligence includes in-depth analysis of both internal and external threats. Threat Intelligence is the analysis of internal and external threats to an organization in a systematic way. The threats that Threat Intelligence attempts to defend against include zero-day threats, exploits and Advanced Persistent Threats (APTs). Threat intelligence includes in-depth analysis of both internal and external threats.

The use of open source intelligence (OSINT) techniques will help better understand potential threats surrounding a telecom organization or a specific sector, crawling and learning from publicly available sources. In this task, different threat intelligence sources will be considered and crawled.

The main features of the crawler will be:

- To identify CVE that could be found on devices that can be exposed on the operator network;
- To detect potential misconfigurations and known vulnerabilities.

The crawler will store the type of device, the software running on it, the potential misconfigurations/vulnerabilities based on the knowledge of the operator network.

RESISTO approach will be based on integration of different tools (Figure 50) and OSINT platforms:

- A crawler to access CIRCL, a MISP framework-based OSINT that collect threat intelligence events;
- Another source of data about vulnerabilities can be found on Twitter and a crawler based on its API will also be considered to collect threat intelligence events;
- IVRE framework will be used to perform network recon to correlate the data collected on the internal operator network with the events found by the CIRCL crawler;
- A machine learning platform will be then used to process the events collected by the crawlers.

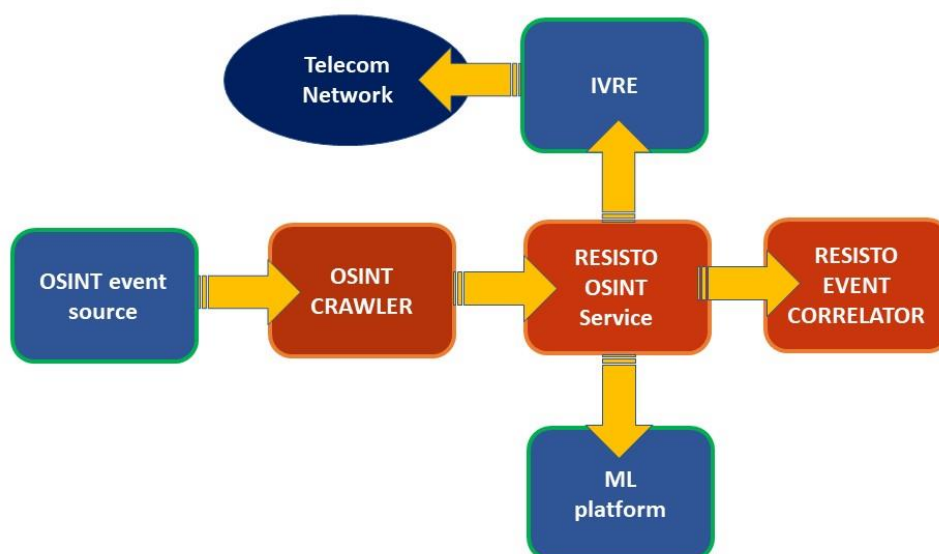


Figure 50 - OSINT based threat detector architecture

3.6. Telecommunication Operators Infrastructures

Telecommunication Operators Infrastructures are not part of RESISTO platform. This section contains, as example, a short description of some testbeds to be used during the validation phase. The complete set of testbeds are described in [3].

3.6.1. Altice Labs test bed

Figure 51 illustrates the topology of the testbed that will support the 5G use case, as well as the main components.

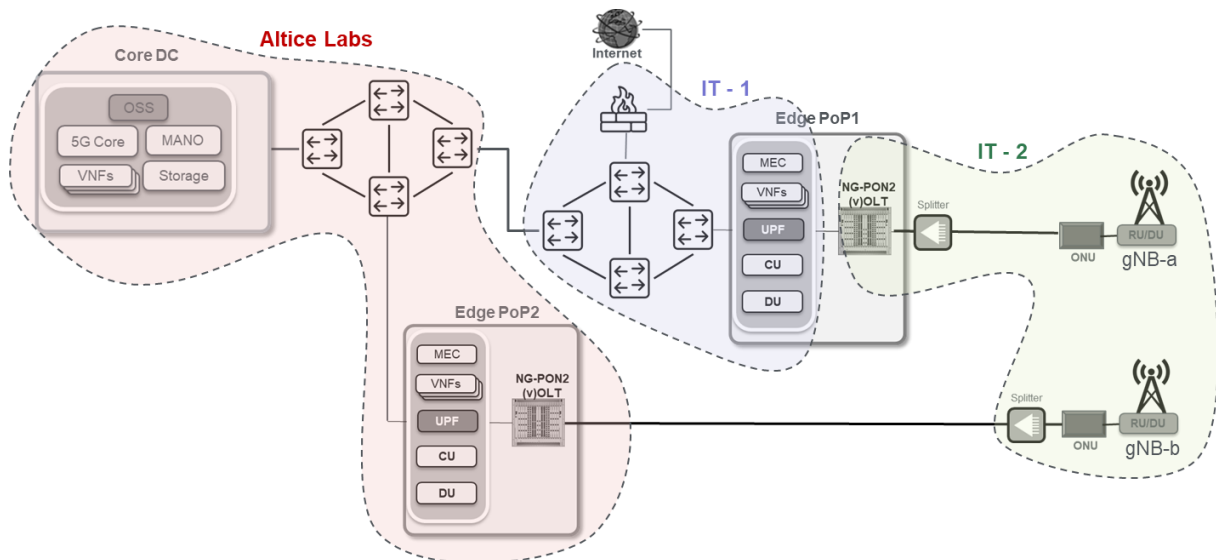


Figure 51 - ALB testbed physical configuration for 5G use case

The physical resources are deployed in two sites, Altice Labs (ALB) headquarters and the Institute of Telecommunications (IT), both of which are located in the city of Aveiro, separated by roughly 2 km. The infrastructure in IT is located in two adjacent buildings, IT-1 and IT-2. The transport network infrastructure providing connectivity between the two physical sites is based on NGPON-2 technology.

The main infrastructure components are briefly described below:

- Radio Access Network (RAN) – two gNBs based on Open Air Interface (OAI); currently supports 4G; availability of 5G NR expected soon; utilization of 3.6 GHz frequency band has been approved for experimental activities in the scope of RESISTO and other 5G-related projects.
- Edge Points of Presence (PoPs) – host the edge components of the 5G testbed, including edge computing functions and RAN components (DU/CU); the two PoPs are physically located in IT-1 and Altice Labs.
- Transport network – includes the Backhaul/Midhaul/Fronthaul component; it is based on Altice Labs' NG-PON2 technology.
- Core DC – hosts the core components, especially the 5G core (Fraunhofer's Open 5G Core) in stand-alone mode and the MANO (SONATA 5.0) components.

From a functional point of view, the structure of the 5G testbed is represented in Figure 52.

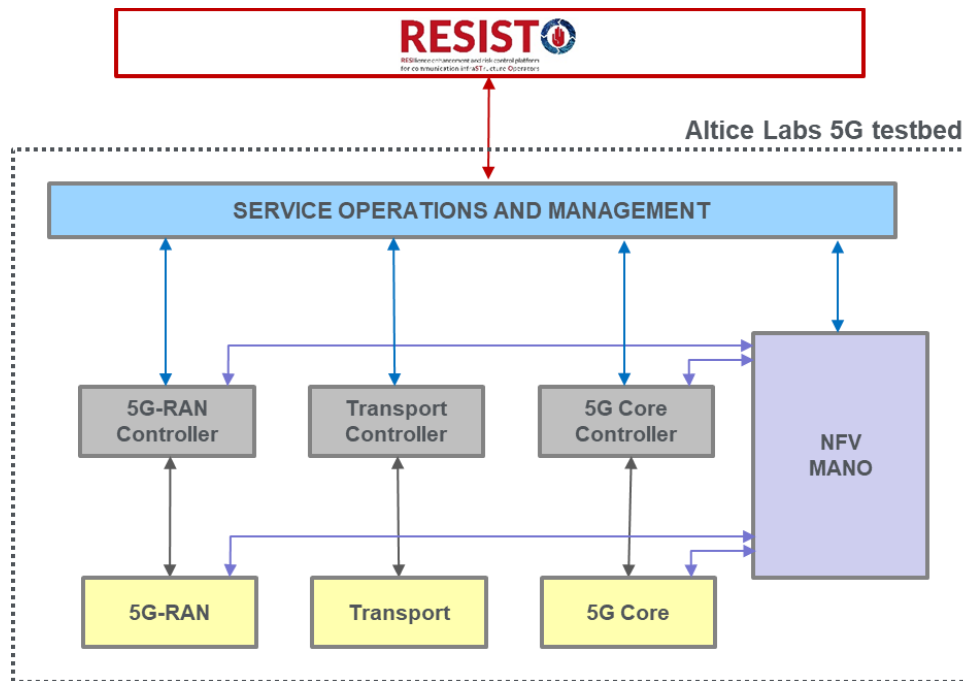


Figure 52 - 5G testbed functional architecture and building blocks

The architecture separates network domain management (Network Domain Controllers and NFV MANO) from E2E service management (E2E Service Operations).

The Service Operations and Management block includes the top-level orchestration functions, to enable zero-touch network and end-to-end service management. It translates service definition into configuration of physical and virtualized resources needed for service establishment, using orchestration to coordinate between domain controllers, the NFV-MANO. Additionally, the E2E Service Operations and Management triggers the components of the domain management systems and NFV-MANO to apply the configuration of the required resources.

NFV MANO focuses on the management and orchestration tasks that have an impact on the resources on top of which 5G-VINNI services are deployed and executed. NFV MANO also includes the management and orchestration of the network services running on the resources.

Domain-specific controllers are supposed to actuate directly on the respective network resources and abstract the characteristics of the infrastructure to the service operations and management layer above:

- 5G RAN controller,
- Transport controller,
- 5G Core controller.

The RESISTO platform (namely the STCL) is supposed to interact with the testbed through the interface that is represented by the red arrow in Figure 52.

3.6.1.1. Altice Labs testbed interfaces

As represented in Figure 52, the interoperation between the RESISTO platform and the 5G testbed is accomplished through the service operations and management northbound interface, which is provided by the SONATA orchestrator. The specification of this interface is provided in <https://github.com/sonata-nfv/tng-api-gtw/tree/master/tng-router#service-platform>. A selection of supported API endpoints is presented in Table 4.

Endpoints	Description
/	The root of the API.
/api/v3	The v3 root of the API.
/api/v3/functions	Lists available functions (VNFs) in the Catalogue
/api/v3/packages	Manages packages (uploading, downloading, etc.)
/api/v3/pings	The Gatekeeper's readiness and liveness endpoint
/api/v3/policies	Manages placement and run-time policies
/api/v3/records/functions	Lists function records available in the Repository
/api/v3/records/services	Lists service records available in the Repository
/api/v3/requests	Manages services' and slices' lifecycle events (creation, scaling, deletion, etc.)
/api/v3/services	Lists available services (NSs) in the Catalogue
/api/v3/settings	Manages different kinds of settings
/api/v3/slas	Manages SLA agreements, configurations, licenses, templates and violations
/api/v3/slices	Manages slice templates
/api/v3/slice-instances	Lists slice instances (creation and deletion are handled through requests)
/api/v3/users	Manages users
/api/v3/users/roles	List user's roles (user's roles are defined together with routes)
/api/v3/users/sessions	Manages user's sessions (logging in)

Table 4 - SONATA orchestrator Northbound API

3.6.2. Orange Romania testbed

The Orange Romania (ORO) testbed is built in a redundant architecture closely simulating the production networks of Orange Romania. Most resources are doubled, both physical equipment and virtual machines. The ORO testbed closely simulates the real-life usage of OROs production networks, albeit in a scale-down manner and includes most of the monitoring capability and capacity of the production networks. During the testing of OROs Use Case, the testbed will behave and output data as close as a real-life production network as possible.

The testbed comprises Active Network Equipment, Security Equipment, Computing and Storage Components and Passive Network Equipment closely simulating the RAN and CORE parts of the production 4G and 5G networks.

ORO's test bed is designed to include as many elements from the real network infrastructure as possible – thus replicating most of the core network functionalities and services. The lab is equipped with various network elements ranging from high speed backbone routers to mobile and B2B access routers. The physical layout is shown in Figure 53.

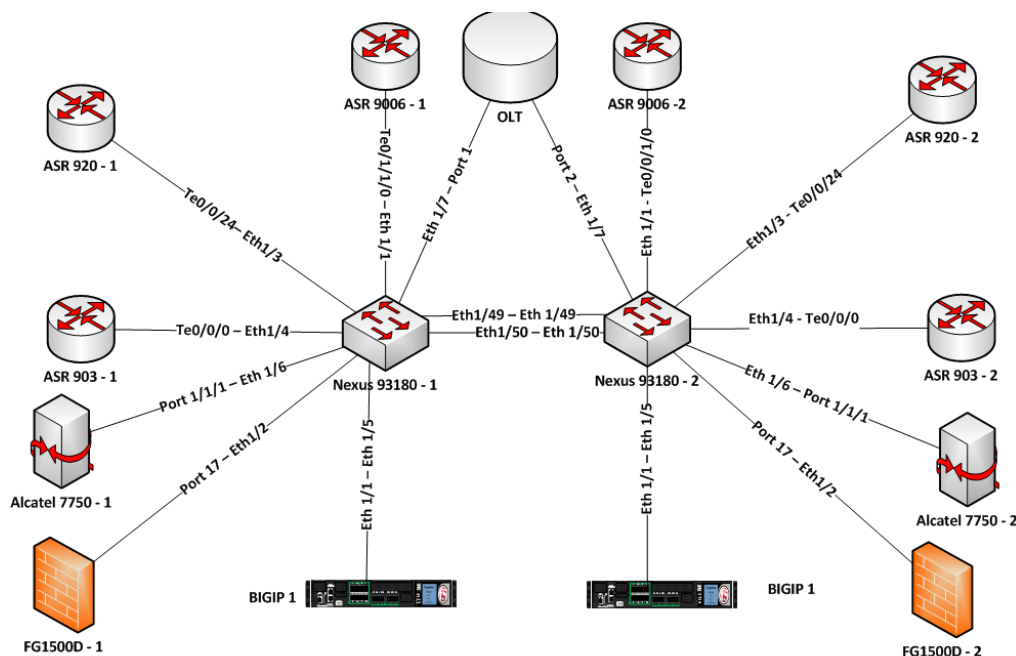


Figure 53 - Physical layout of ORO testbed

The equipment shown in the above diagram form a fully functional MPLS network. All the equipment are aggregated using high speed links on the pair of Nexus switches – thus giving the testing engineers very high flexibility and deploying new designs in a very short amount of time with no additional physical cabling – all with the usage of 802.1Q (VLAN) tagging.

The MPLS Layer contains PE routers from a wide range that are used to deliver backbone functionality (P Routers), mobile PE Router and B2B PE Routers. MPLS network will run LDP and IS-IS as the underlay routing protocol and use various QoS mechanisms to emulate a real deployment. Services delivered across the MPLS network include MPLS Layer 3 VPNs, MPLS Layer 2 VPNs (both VPLS and EoMPLS). MPLS network will also include a pair of Route Reflectors for L3 and L2 VPNs autodiscovery. MPLS Network will also integrate an OLT to simulate an attack/outage on the part of subscriber network.

The security fabric and datacenter layer is achieved using a few next-generation security equipment and application delivery controllers like:

- Fortinet FortiGate (URL Filtering, Centralised Antivirus, IDS and IPS, DLP, E-mail filtering, Layer 4 Firewall)
- F5 BIGIP (Web Application Firewall, Enhanced application layer enrichment and protection)

Equipment which are not shown in the diagram also include EPG for mobile device gateway.

The test bed will also include various servers which run VMWare and Openstack hypervisors for virtualized solutions and datacenter services emulation.

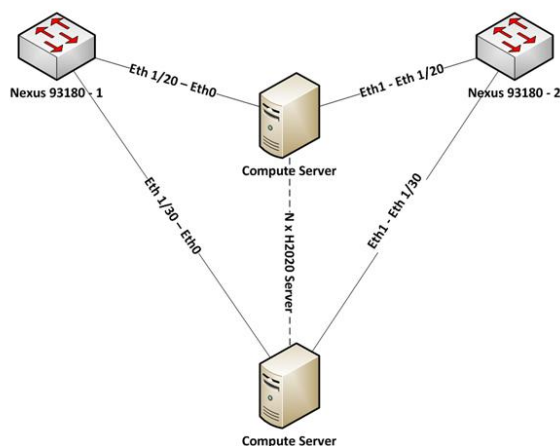


Figure 54 - H2020 Virtual Infrastructure

The purpose of the servers is also to deploy the necessary infrastructure components for RESISTO platform. Overall the logical design will look like in Figure 55.

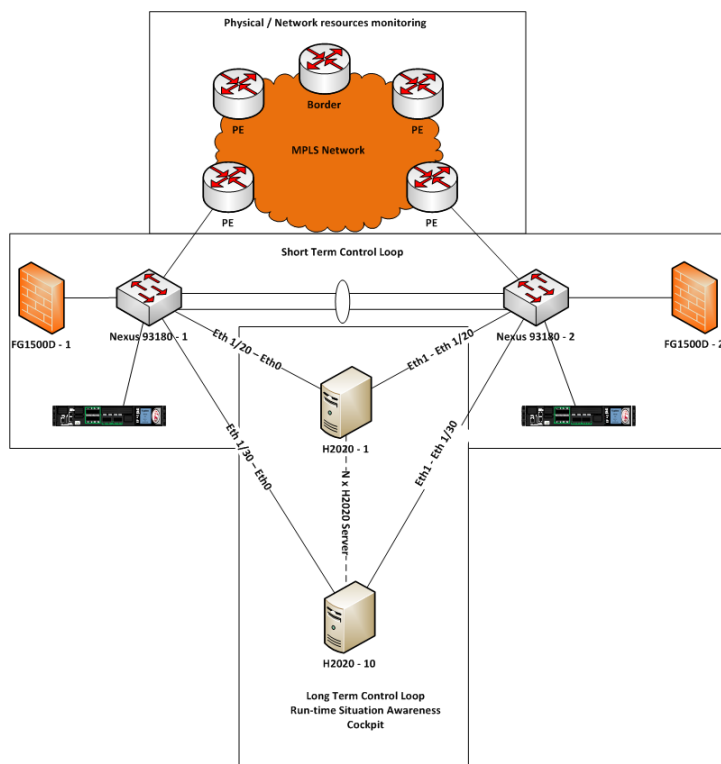


Figure 55 - RESISTO Components Deployment in ORO's Testbed

RESISTO components will be deployed in a virtualized environment over Openstack or VMWare – giving the test engineers the possibility to scale-out different components both vertically (grow a single VMs resources) and horizontally (deploy more instances of the same type of VM).

A detailed description of ORO testbed is reported in [3] section 4.3.1.

3.6.2.1. Orange Romania testbed interfaces

The main part if the components in ORO testbed will send syslog messages to be consumed by RESISTO Short Term Control Loop. Table 5 lists all the components in the testbed that will output syslog messages relevant to OROs use-cases.

Component	Name (Link to datasheet)	Description (Function in testbed)	Message Format
Alcatel 7750-1 Alcatel 7750-2	Alcatel-Lucent 7750 Service Router	Core Router/Provider Edge	Syslog
ASR 903-1 ASR 903-2	Cisco ASR900 Series Aggregation Service Router	Distribution Router	Syslog
ASR 920-1 ASR 920-2	Cisco ASR920 Series Aggregation Service Router	Access Router	Syslog
C7606	Cisco 7606 Router	Core Router/Provider Edge	Syslog
OLT	Optical Line Terminal	N/A	N/A
Nexus 93810-1 Nexus 93810-2	Cisco Nexus 9000 Series Switch	Switch - Used for cabling in testbed	N/A
ASR 9006-1 ASR 9006-2	Cisco ASR9000 Series Aggregation Service Router	Border Router/Core Router	Syslog
H2020 Servers	HP Proliant DL300 Series Servers	Servers hosting virtual infrastructure	Syslog
FG 1500D-1 FG 1500D-2	Fortinet FortiGate 1500D Firewalls / UTM	Firewall, IPS/IDS, Malware Detection&Protection	Syslog
BIGIP1 BIGIP2	F5 BigIp Load Balancers	Load Balancers, Web App Firewalls	Syslog

Table 5 - ORO testbed components

All the equipment in the testbed, including active network equipment, active security equipment, servers and virtual infrastructure and hardware/physical security devices (such as presence sensors, motion sensors, etc.) will forward syslog messages to the interface.

There will be no need for a polling protocol, in such. For ease of development and integration, all equipment will use syslog as close to the RFC5424 requirements as possible. Syslog messages details are reported in Table 6.

Protocol used	Syslog, RFC5424
Transport Used	UDP 514
Source IP Address	Source IP address SHALL be interpreted as the originator of the syslog message (i.e. – there will be no syslog relays between testbed equipment and the Interface)
Message Encoding	ASCII (UTF-8)
Message Size	A minimum of 480 Bytes and a Maximum of 65536 Bytes
	Each UDP frame contains one syslog message
Message Content	As Per RFC5424. The MSG part of the messages will be further described from the syslog samples embedded as it is specific to each equipment and technology. A relevant correlation between the content of different MSG parts from syslog messages and the corresponding cyber security or physical security event will be made.
Severity Level	All testbed equipment will forward syslog messages with Severity <= 4. (Values 5 thru 7 will be not forwarded)

Table 6 - syslog messages specification

As shown in

Figure 56 a specific syslog adapter will publish UDP syslog received messages on the relevant Correlator kafka topic (see par. 6).

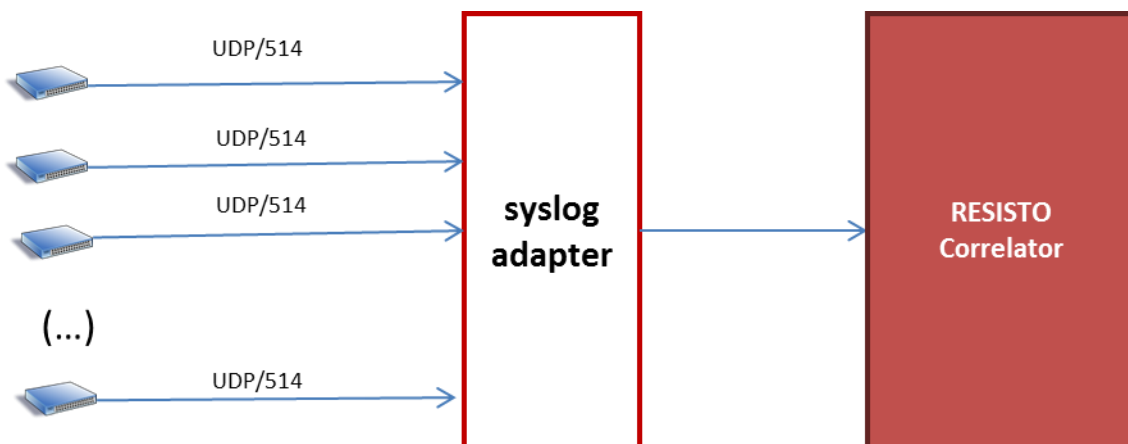


Figure 56 - syslog adapter

r

3.6.3. OTE testbed

OTE's core lab infrastructure (Figure 57) for the needs of RESISTO pilot will be interconnected with OTE's openstack cloud (Figure 58).

Core lab's interfaces are mainly 10G and 1G both fiber and copper. There is a traffic generator which is Spirent test Center and has direct/physical connection to other components with cable (fiber for data and copper for management).

All metro Ethernet Switches (distributors) are Huawei connected with Layer 2 and L3 VPN connections to distant sites while locally are connected cable (both fiber and copper). All BNG Routers are CISCO and are considered core network. They have connections to other components

with cable (fiber and copper) and indirect connections to other components located in other sites through Layer 2 and L3 VPN connections.

Regarding security they lab has firewalls enabled through ACLs (Access Control Lists).

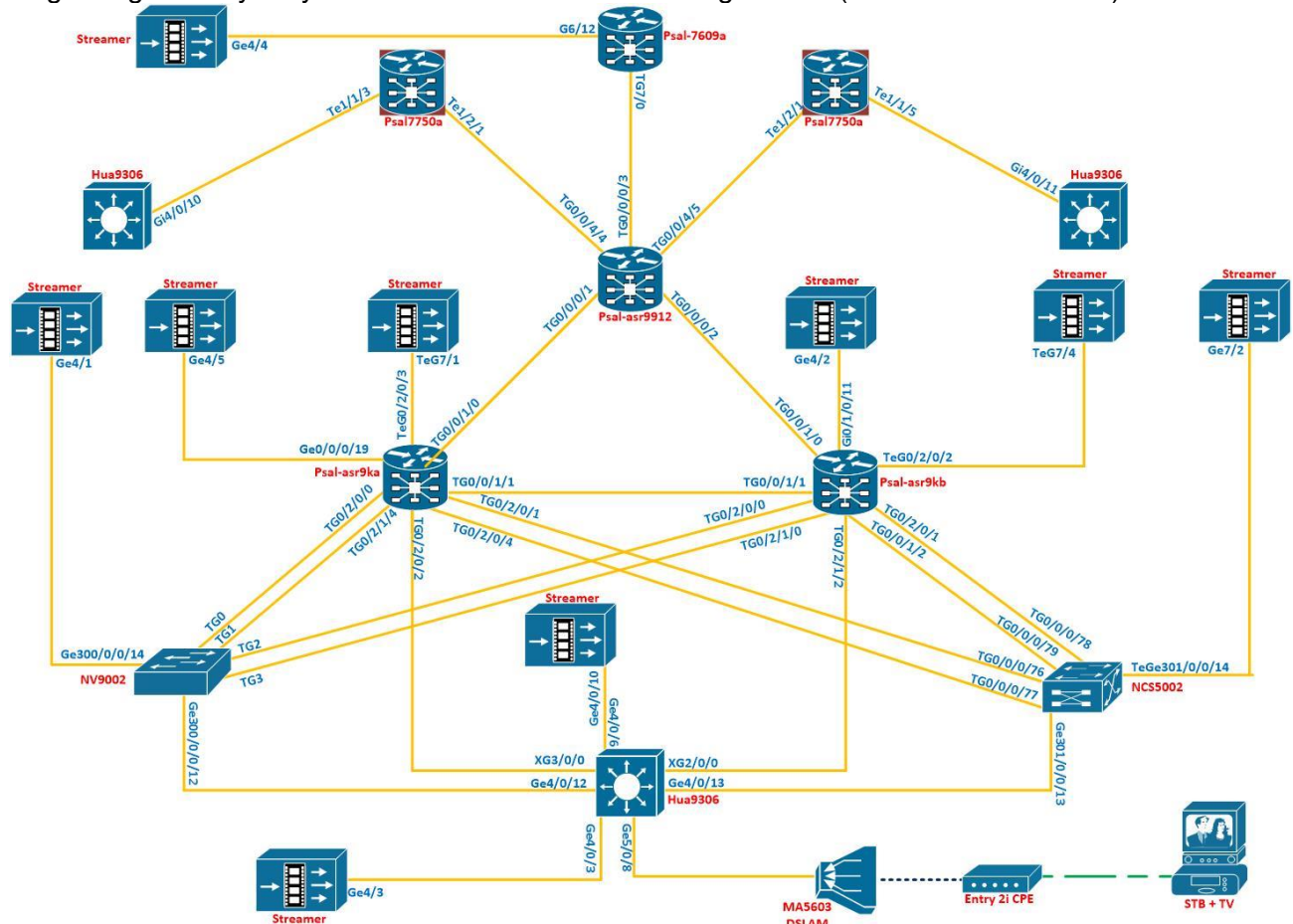


Figure 57 - OTE Core Lab description

The second part of this testbed is the openstack testbed an indicative architecture of which is presented as follows.

The Openstack testbed OS is Ubuntu Server 16.04 LTS, while the current Openstack version is Queens; it can be upgraded based on project's specific requirements. Ubuntu Server 16.04 LTS, as one of the most active providing Long Term Support and ensuring that any issues that come up within the next five years will be dealt with. The testbed can be utilized for new technologies experimentation, either for Proof of Concept or for field trials.

The testbed is composed of by gateways, various controllers and compute nodes (interconnected via switches/routers which can be formulated in any required architecture. The testbed generic architecture/layout is depicted in Figure 58.

A more detailed description of OTE testbed is contained in [3].

OTE testbed equipment will be monitored by RESISTO through a SNMP adapter.

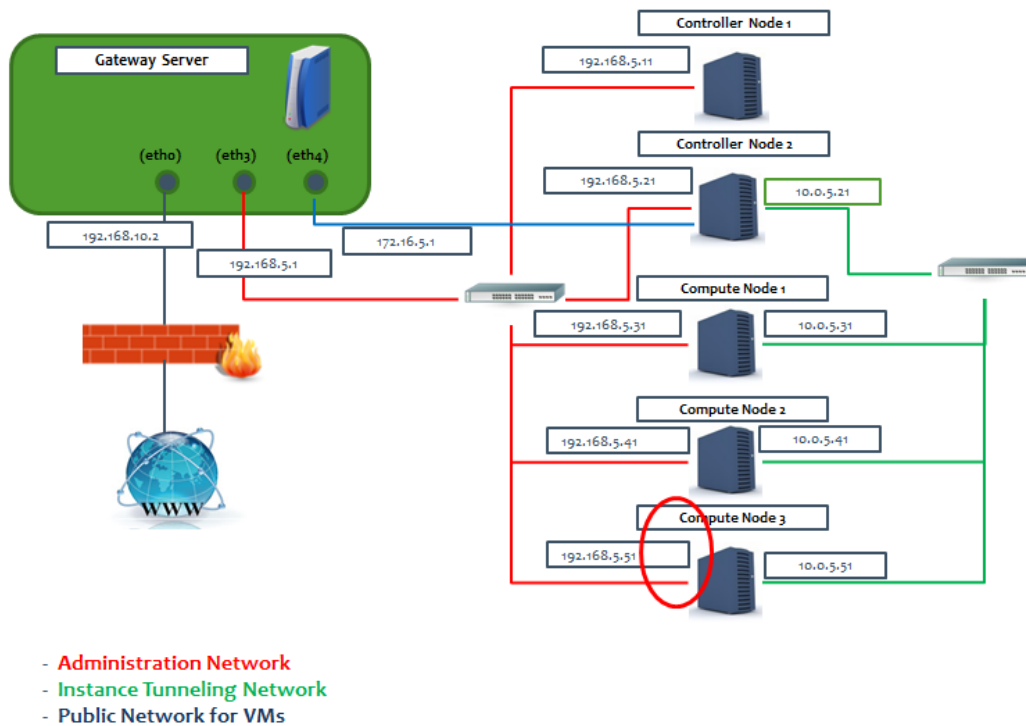


Figure 58 - Generic architecture of OTE's Openstack testbed

3.6.4. British Telecom testbed

British Telecom (BT) testbed is a virtual testbed for IPTV implementing the typical network infrastructure for delivering video streams using multicast and unicast transmissions to different types of video clients or end-devices. Figure 59 shows the initial architecture of the testbed.

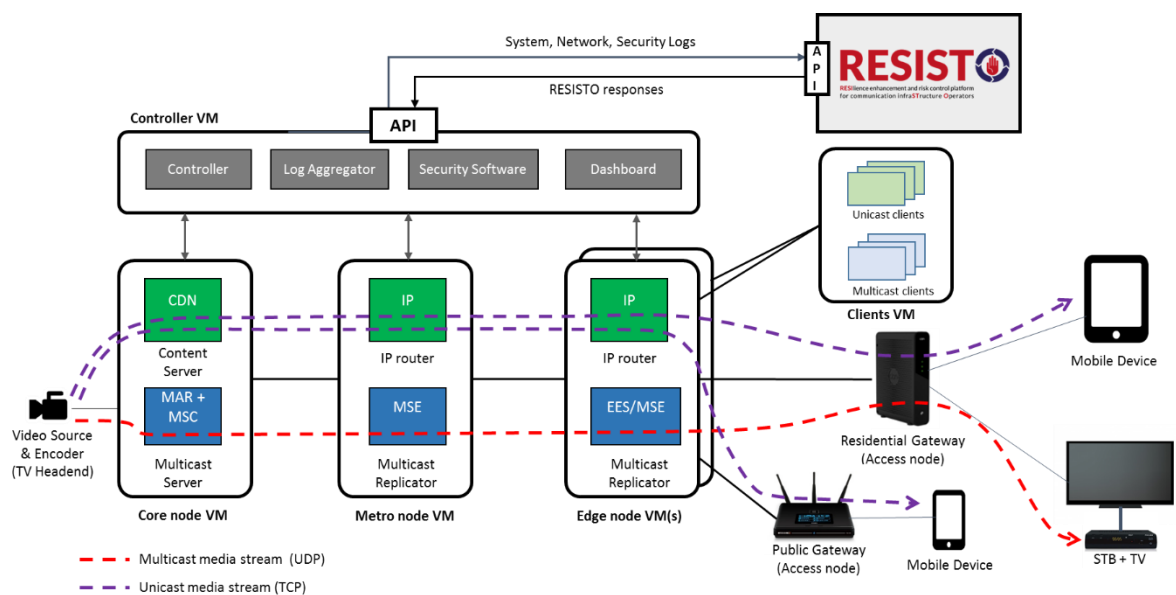


Figure 59 - BT testbed architecture for multicast and unicast video delivery

The majority of the testbed components will be deployed in a virtualized environment, i.e. BT Research Platform, which could be connected to some physical components such as WiFi router or

mobile phones depending on the test scenarios. We strive to integrate the TV headend as physical device into the testbed in order to provide live video stream into the delivery network. In particular the following Virtual Machines (VMs) should be deployed to represent the network infrastructure:

- **Core node VM:** This VM will contain components that replicate the main functions of a multicast server (i.e. MARs - Multicast Access Routers and MSC - Multi Service Core) as well as of a content server that holds a copy of the video stream that can be transmitted to client devices using unicast route. The content server also acts as a retransmission server for handling the requests to retransmit lost multicast packets. Content Distribution Network (CDN) responsible for authenticating users/devices and verify clients have the rights to access the contents
- **Metro node VM:** This VM will contain components that replicate the main functions of multicast replicators and unicast IP routers as usually deployed on metro nodes.
- **Edge node VM:** This VM will represent the nodes at the edge of the network infrastructure and contain components that replicate the main functions of associated multicast replicators and IP routers. Depending on the test scenarios there may be more than one edge node VMs deployed in the testbed. During the design and implementation phase we will examine further whether or not to co-locate the edge and metro node functions together in order to simplify the testbed architecture.
- **Clients VM:** This VM will contain (software) components that simulate a number of unicast and multicast clients in the network consuming the video streams (simultaneously) via unicast and multicast transmission respectively.

In addition to the virtual machines described above we envisage the deployment of a Controller VM which (among others) should comprise the following components

- **Controller:** This component acts as a hub to control the main operation of multicast and unicast transmission in the testbed. It is the one that will handle the requests and responses of the RESISTO platform.
- **Log Aggregator:** This component collects the relevant system, network and security logs produced within the testbed and provides them to the RESISTO platform. The RESISTO platform will continuously monitor the provided logs and takes actions based on the outcome of its internal correlation engine.
- **Security Software:** This component represents the threat monitoring and detection service of the network infrastructure. Essentially its main function is to produce security logs and alerts in the case of suspected cyber-attacks on the network infrastructure. Such cyber-attacks may either be simulated or carried out using external penetration testing tool.
- **Dashboard:** This component provides a graphical interface showing the current state of the network, e.g. network health, traffic statistics, security alerts, etc.

The controller VM is exchanging information with the RESISTO platform via their respective APIs, e.g. to provide network and security logs, or to receive RESISTO responses in case of network or security incidents.

During normal operation a live or recorded video stream is transmitted from the TV Headend; the video content is encoded according to service requirements and sent to both the content and multicast servers. In order to reduce network loads the video is streamed to multicast group addresses and replicated by the multicast-capable (software) routers deployed at core, metro and edge node VMs. Any multicast-capable end-devices such as Set Top Boxes (STBs) or multicast clients that have joined or subscribed to the multicast group via IGMP will receive the video stream as multicast packets. Consumers on mobile devices are using apps to select the video event and stream

the video using unicast transmission. In some cases video delivery via home or public access nodes (gateway) can be done using multicast transmission too. In case of unicast transmission the video quality (i.e. encoding scheme) can be chosen dynamically by consumers based on their personal preferences, device capability (e.g. processing power) and channel quality. If multicast packets get lost during transmission, e.g. due to noises in the channel or network failures, the multicast device will request for packet retransmission from the content server. Such packet retransmission happens at the application layer and uses unicast delivery method.

There is a possibility to link up with operational BT IPTV networks either to use its streamed videos as the testbed's TV head-end, and/or to use sanitized events from operational networks as shown in Figure 60. Descriptions of the components could be found from this site <https://www.btplc.com/SINet/sins/pdf/511v2p0.pdf> and they will be included in RESISTO deliverable D2.8.

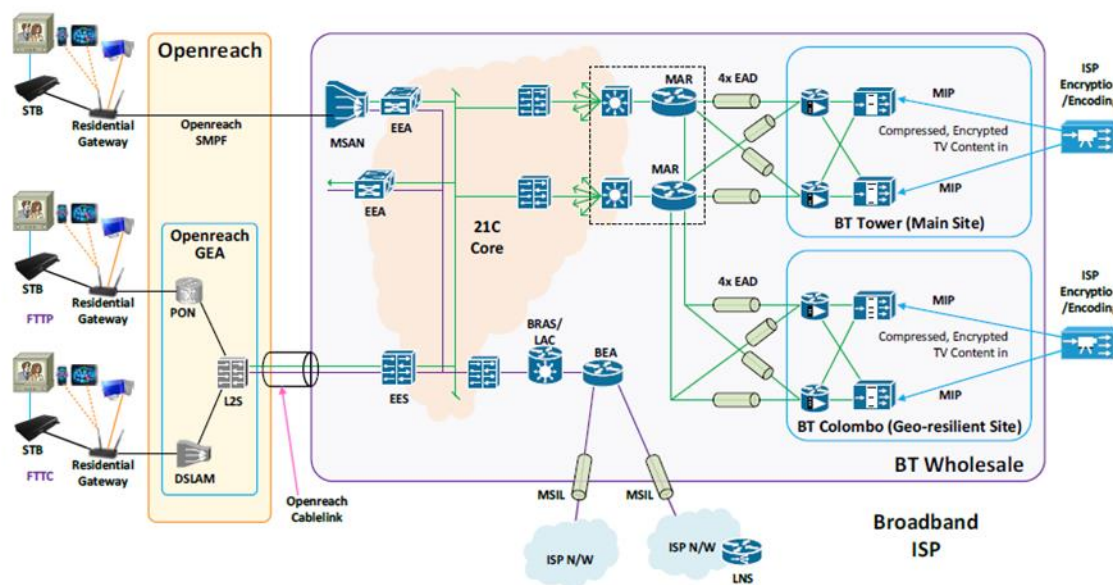


Figure 60 - UK network infrastructure for unicast and multicast services

BT IPTV delivery at core network consists of tens of Core Routers. They join through fully meshed VPN tunnels. There are two MARs. There are hundreds of Multicast Service Edge (MSE) nodes and MSC nodes, thousands of Layer 2 Switches (L2S) on the access networks delivering multicast services to homes by DSLAMs (Digital Subscriber Line Access Multiplexers) and PONs (Passive Optical Networks).

Alerts and log data from the platform will be provided by columns and values formats, JSON or syslogs as appropriate. There are also past physical events affecting the network, which will be provided as appropriate, these physical events include cable cuts, and power outage events.

4. CONCEPT OF EXECUTION

This chapter illustrates some sequence diagrams aimed at describing the interactions among RESISTO components for main RESISTO use cases.

4.1. Short Term Control Loop operation flow

For clarity of exposition, the interactions between RESISTO detectors, Short-Term Control Loop components and human operators are illustrated in three successive diagrams.

The diagram in Figure 61 shows what happens when a detector detects an event as a status change, not to be considered as an alarm.

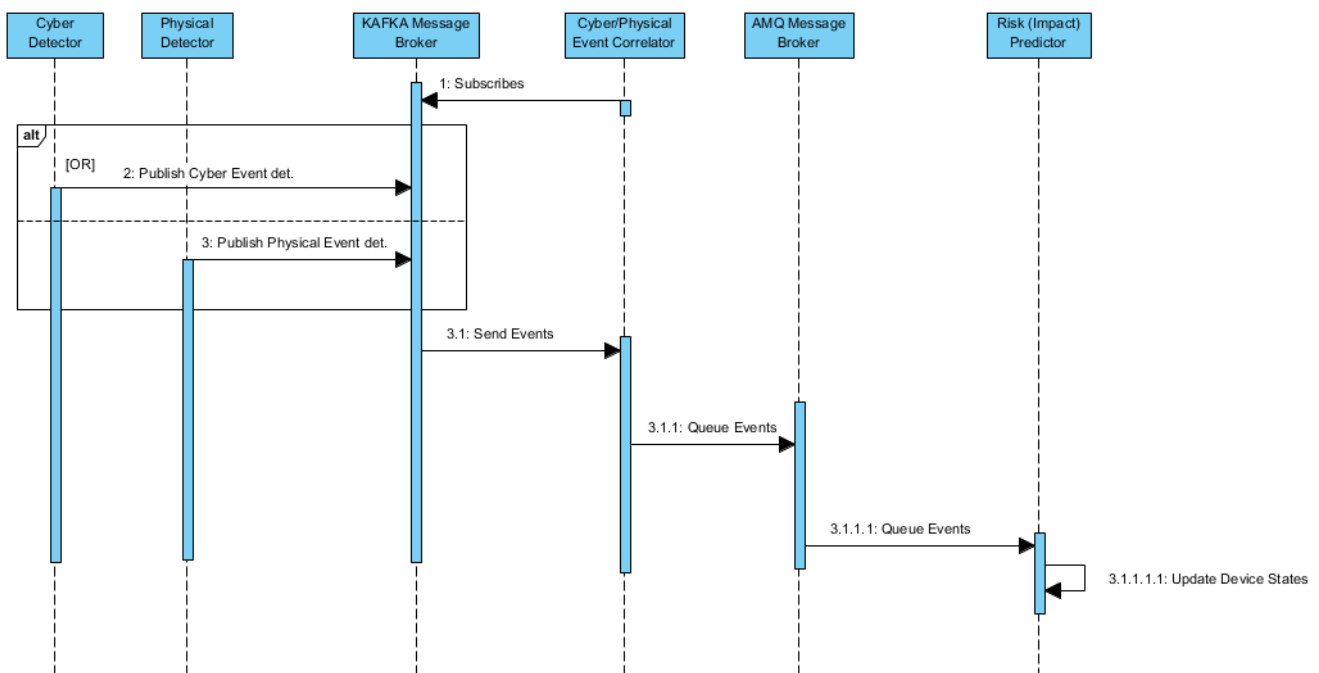


Figure 61 - Short Term Control Loop - Decision Support phase - Event

At the beginning, the «Cyber-physical Events Correlator» component subscribes to the KAFKA Message Broker.

Whenever a physical or cyber event is detected, it is published by the detector, via the KAFKA Message Broker, to the «Cyber-Physical Events Correlator» component. The received information is sent to the Correlator's internal repository in order to update the field model, and at the same time, it is analyzed by the engine in order to identify any potential alarm.

In addition, the event is also queued, via the AMQ Message Broker, to the «Risk Predictor» component and displayed on the HMI.

Whenever the event is a simple change of status (closed door, faulty sensor, etc.), the «Risk Predictor» simply updates its field model coherently with the one present in the «Cyber-physical Events Correlator» component.

Instead, the following two diagrams show what happens when a detector detects an event which constitutes a risk.

In Figure 62, interactions between RESISTO components and the Short-Term Operator are showed up to when the operator has collected enough information to decide how to mitigate the incoming risk.

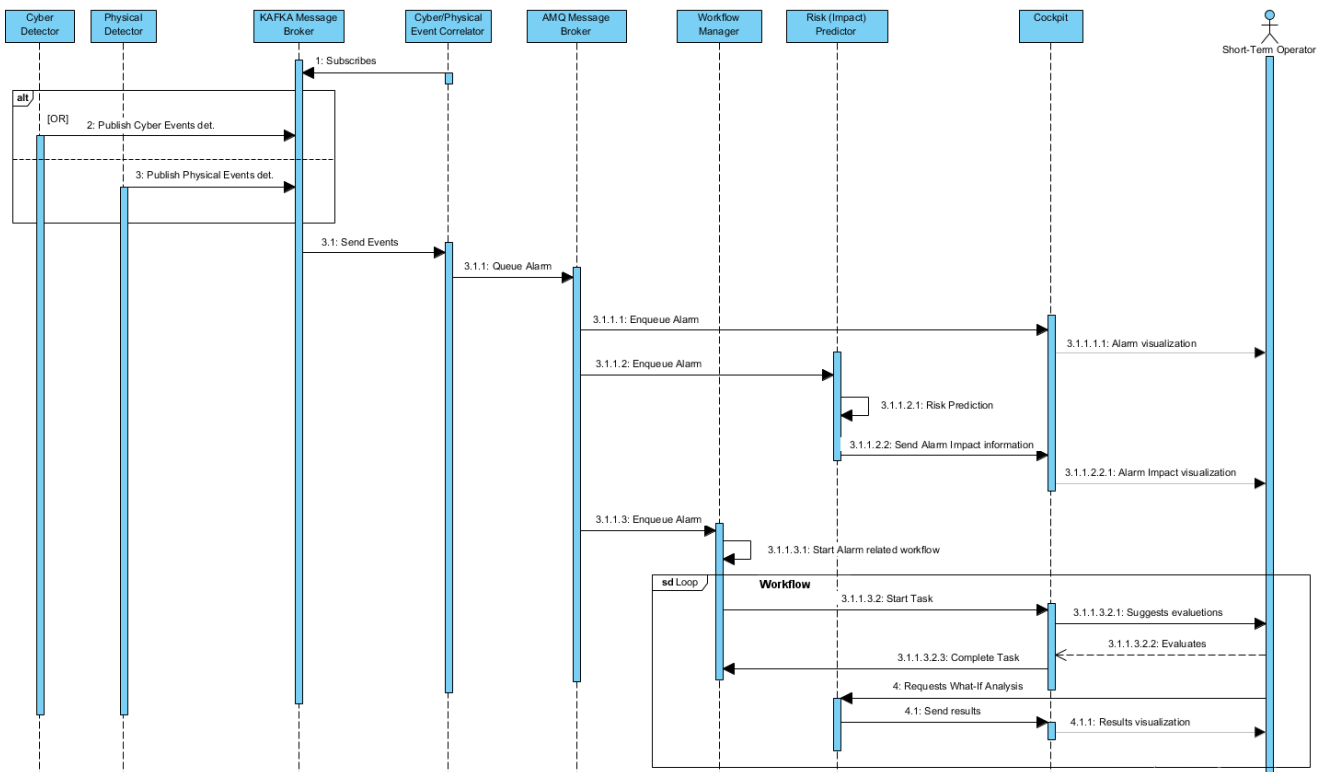


Figure 62 - Short Term Control Loop - Decision Support phase - Alarm

When the Correlator recognizes that the event itself, or the correlation with other events, represents an actual alarm, the alarm is sent to the Risk Predictor in order to allow impact analysis and to the Workflow Manager in order to allow both display on the HMI as well as launch the alarm-related workflow. The workflow provides an evaluation that, combined with the risk impact calculated by the Risk (Impact) Predictor, is presented to the operator and which the operator may choose to apply the most appropriate reactions. To reach an informed decision, the Short Term Operator will be provided with the display of both the queued alarm as well as of the expected impact generated by the Risk Predictor.

The operator can require a number of *What-If analyses* on the «Risk Predictor» component, in order to evaluate what the best risk mitigation strategy could be.

The following diagram in Figure 63 shows possible actions taken by the operator to mitigate the encountered alarm.

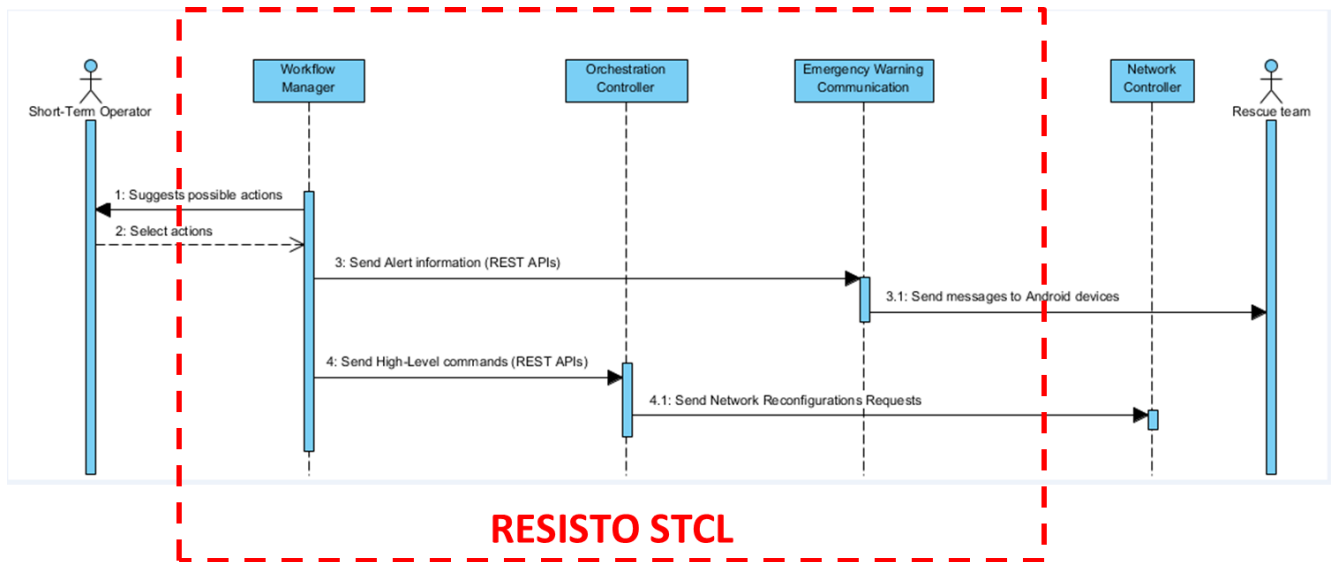


Figure 63 - Short Term Control Loop - Reaction phase

A Short-Term operator, progressing in the workflow and based on the information obtained from previous what-if analyses, requests the «Workflow Manager» component to perform one (or both) of the following actions:

- Communicate, via REST API messages, to the «Emergency Warning Communication» component, in order to send information about the detected risk (type, location and description of the event generating the risk, etc.).
The EWC component will then forward the information, via Android messages, to single rescue operators or teams involved in mitigating the event.
- Communicate, via high-level REST API messages, to the «Orchestration Controller» component, so that it can send appropriate reconfiguration commands to the «Network Controller» component. «Network Controller» means equipment provided by the telecommunication operator, outside the RESISTO STCL border, able to control the network equipment, as example it could be a SDN controller, a simple SDN controllable equipment as well as a complex Operations Support System.

4.2. Continuous improvement of the overall resilience

In order to constantly improve overall resilience, the RESISTO platform will integrate the Short Term Control Loop and Long Term Control Loop into a higher level loop, based on appropriate Resilience Indicators.

Resilience indicators, specific for each type of event, are estimated by the Risk and Resilience Management Methodology and stored on the Knowledge base.

The diagram in Figure 64 shows how this continuous loop works.

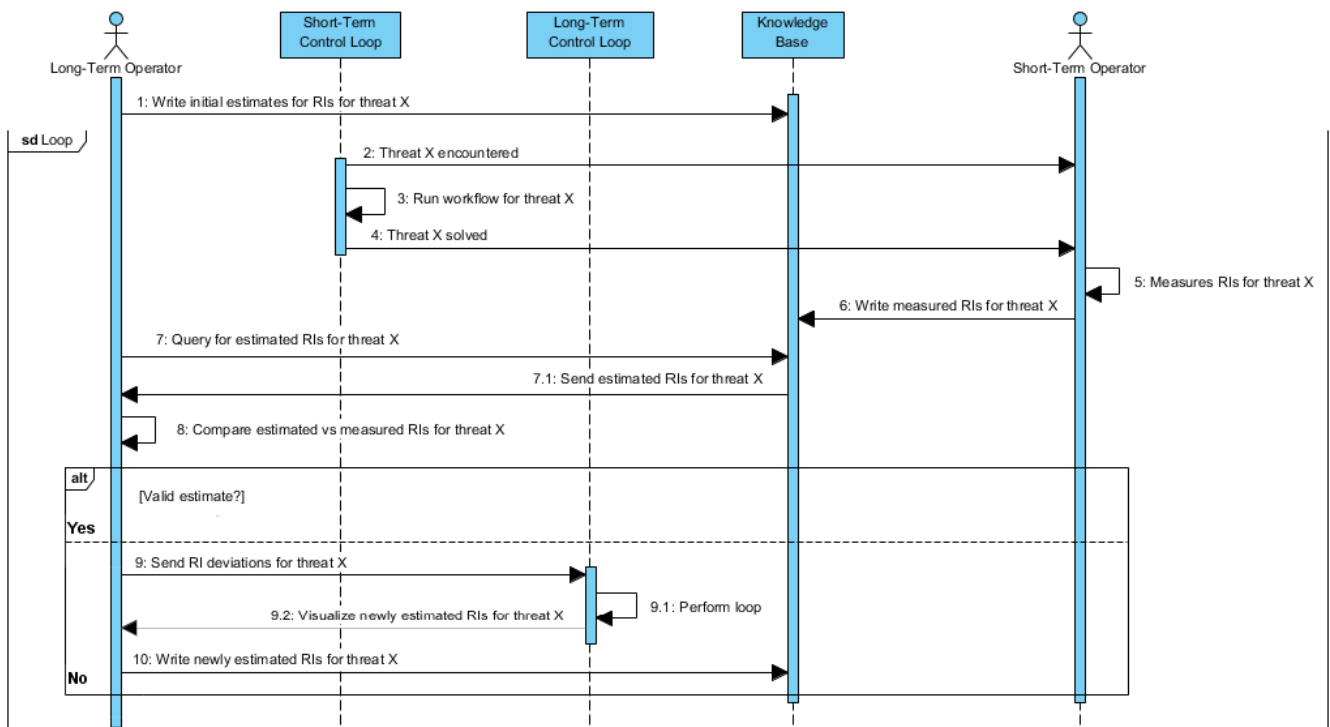


Figure 64 - High Level Control Loop

At each iteration, the Long Term Control Loop estimates the Resilience Indicator values for specific events. Whenever the Short Term Control Loop recognizes the occurrence of a given “X” event and after mitigation of the impact resulting from the event, the Short Term Control operator measures the defined RI for the encountered event and writes current measurements of the RI to the Knowledge Base.

During the following LTCL cycle, the LTCL operator performs a comparison between estimated and measured RIs. In the event that value deviations are significant, the deviation will be provided as input to the Long Term Control Loop in order to improve estimation methods and/or select new actions on the CI with the aim of improving the RIs.

5. COMMUNICATION MODES, MESSAGE PATTERNS AND FORMATS

The RESISTO platform is a distributed system formed by heterogeneous components; for this reason, a communication layer is necessary in order to guarantee the integration between the entities in an agnostic manner, i.e., independent of the type of language and platform used to build the individual components.

5.1. Communication modes

Since there are needs for both asynchronous and synchronous communications within the RESISTO components, the RESISTO software architecture provides two complementary communication mechanisms:

- Message broker (asynchronous mode),
- REST services paradigm (synchronous mode).

The first mechanism is represented by the Data Integration Layer component that realizes the communication infrastructure and allows the RESISTO system to exchange information internally and with the various external entities.

REST services are provided by RESISTO internal components to guarantee direct interaction between components.

5.1.1. Data Integration Layer: Message broker communication modes

A message broker is a software module that allows asynchronous integration via message exchange.

The architecture of a messaging system is typically a star or hub with a messaging server, the broker, used by various components or applications to communicate with each other. One of the advantages of this model is that applications send messages only to the broker's address, which then routes the messages to the right application based on certain criteria (queue name, message properties). In this manner, the applications need to know only the address of the broker. Any given application within the messaging system is not required to know the physical addresses of the other applications with which it must communicate.

This type of interaction is strongly decoupled because the message is sent over a channel where it is the message broker's responsibility to deliver the message to the interested parties. However, the task of the message broker is not only to pass data, since it also deals with aspects related to security, priority of messages and orderly forwarding. Middleware focused on providing message-based integration is called Message Oriented Middleware (MOM). A message broker usually supports different interaction modes:

Publish / Subscribe: In this mode, a publisher sends messages to the channel and the message broker sends them to different receivers based on subscriptions. This type of interaction supports different scenarios including one-to-many or many-to-many;

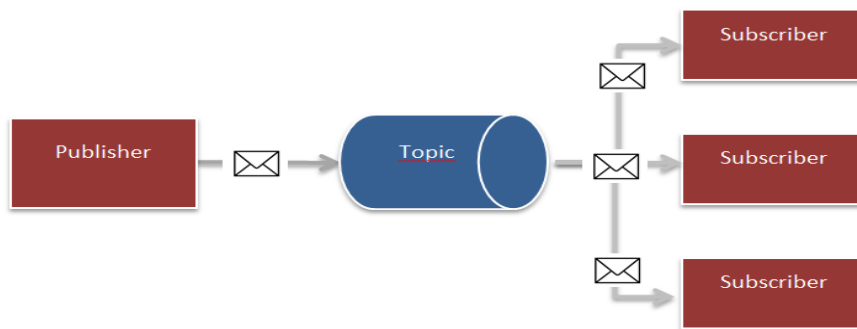


Figure 65 - Publish/Subscribe interaction mode

Point-to-Point. In this mode, a user sends a request on a specific queue (corresponding to the consumer) and the consumer sends the reply on the same queue; in fact it is an asynchronous realization of the request / reply mode;

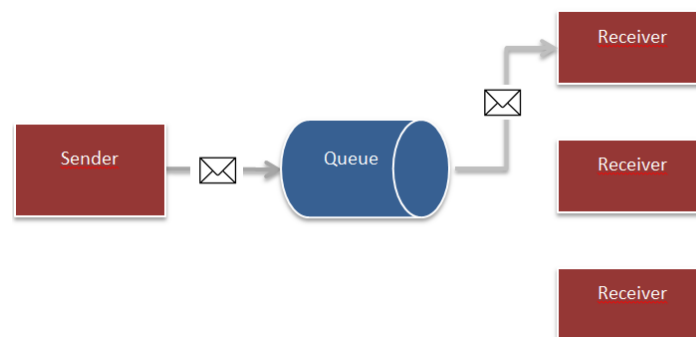


Figure 66 - Point-to-Point interaction mode

The advantages of using MOM systems to manage the exchange of messages between the entities of a distributed architecture are mostly the following:

- **Loose coupling:** this aspect, as already outlined in the previous sections, makes the publisher / subscriber paradigm particularly suitable to be chosen as a communication pattern for the RESISTO platform since it simplifies the interoperability between RESISTO and external systems. Decoupling represents the degree of independence of the modules within an architecture. In a communication scenario, decoupling allows the actors involved in the exchange to be mutually independent from one another. The interacting parts do not need to know each other and the participants do not need to be active at the same time in order to communicate. The Publisher can publish even when the Subscriber is disconnected and in the same way the Subscriber can receive notifications even after the Publisher is no longer reachable
- **Scalability:** due to its own architecture which allows to parallelize the forwarding of information between the producer endpoint and more message consumers, the publish / subscribe paradigm favors scalability compared to classic client / server systems and optimizes performance. Furthermore, in a publish / subscribe system it is possible to add additional subscribers in the future without having to change the sender's service.

In addition, message brokers not only ensure message routing, they also ensure data persistence.

5.1.2. Message broker technologies

Recently, various messaging systems are imposing themselves on the market, each characterized by its own peculiarities and with different pro / cons. As part of the RESISTO project, two extremely popular open-source systems have been considered and can be considered complementary:

- Apache Kafka,
- MQTT.

Apache Kafka

Apache Kafka is an open source instant messaging system, characterized by low latency and high speed that allows the management of a large number of operations in real time from thousands of clients, both in reading and writing.

Since Kafka is a platform developed to speed up and maximize the security of data flows produced by messaging, it represents one of the most performing and scalable solutions and is preferred to traditional message brokers. The principal strengths of the Kafka platform are increased productivity and greater reliability when compared to other messaging systems. Apache Kafka uses a distributed stream processing engine to build real-time data pipelines and streaming applications. It allows you to receive data from different types of sources (called producers), processing them within its architecture and making them available to recipients (consumers). The high performance offered by Apache Kafka compared to other instant messaging systems is due to the fact that it is based on the concept of writing "logs". The Kafka broker makes the messages persistent, through writings whose actual execution is delegated to the management of the OS. Adding a message to the queue corresponds to adding a log line. Logs can be compacted or eliminated usually based on a lazy approach that prefers to delay such operations at times when the broker is not busy managing large flows, or when this operation becomes undeliverable.

A core concept is that Kafka stores streams of records in categories called topics. A topic is a category or feed name to which records are published. Topics in Kafka are always multi-subscriber; that is, a topic can have zero, one, or many consumers that subscribe to the data written to it. In addition to this it offers redundancy through a system of clusters so that you can keep the contents always available even after failures.

MQTT

MQTT (MQ Telemetry Transport) is a machine-to-machine (M2M) connectivity protocol and it is considered the de facto standard communication protocol for the Internet of Things. It is an extremely lightweight publish/subscribe messaging transport. MQTT has been designed for limited devices and networks with low bandwidth, high latency or that are substantially unreliable. The principles on which it is based are those of minimizing the requirements in terms of bandwidth and resources while maintaining a certain reliability and degree of certainty of sending and receiving data.

Other highlighted features of MQTT are a reduced transport overhead, thanks to a 2-byte fixed-length header, and minimized exchanges to reduce network traffic. Finally, the protocol presents a mechanism for notifying the interested parties of an abnormal disconnection of a client, using functions called Last Will and Testament. There are many popular implementations such as Mosquitto (open source) and HiveMQ (commercial).

5.1.3. Design Choice of Message Broker Communication Mode and Technology

For the communication scenarios of the RESISTO system, characterized by signaling of events/alarms coming from heterogeneous data sources, it was decided to make use of the

Publish/Subscribe messaging model. In this model the sending and receiving of the message is decoupled, i.e., is executed asynchronously. The sender of the message (publisher) does not send the message directly to the receiver but forwards it to the central MOM system which takes care of receiving and persisting the message and routing it to the client systems that have subscribed to receive the messages. The publisher is not interested in knowing who has subscribed and at the same time the subscriber can decide to receive only a subset of published messages.

The messages are grouped according to categories and it is the aim of the message broker to filter the messages by selecting them according to the category they belong to.

Typically Publish/Subscribe systems work in two main modes:

- **Topic-Based:** in this mode, messages are published on "topics" or logical names. That is, a given topic becomes the keyword through which consumers subscribe to a certain pattern of events. Publishers notify such events. Subscribers receive messages for which they have subscribed to the topic. Each subscriber receives the same messages that arrive at the "topic".

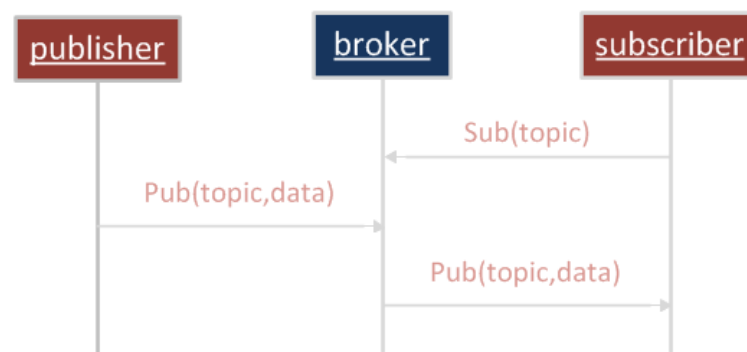


Figure 67 - Topic-based Publish/Subscribe mode

- **Context-based:** in this case the classification of the messages to be forwarded to the consumer does not take place according to the topic, but according to the metadata, i.e., attributes that characterize the messages, or according to the content of the messages. It is the subscriber's responsibility to define the rules to be applied to the incoming messages. The message broker's task is to activate filters to route messages to subscribers when a pattern matches the rule.

Within the RESISTO project, the **topic-based** typology will be adopted. Identified topics to be used as communication channels will be illustrated in par. 6.

As described above, Kafka appears to be the most suitable messaging system for the RESISTO platform and will be used as a message broker to manage communication with external sources.

In comparison to other messaging systems, Kafka has a better throughput, built-in partitioning, replication and inherent fault-tolerance, which makes it a good fit for large-scale message processing applications.

However, since the RESISTO platform must also be interconnected with IoT sensors characterized by devices with poor hardware resources, it was considered appropriate to also make available the

MQTT technology, using the appropriate connectors or through a MQTT proxy to integrate it into Kafka (see Figure 68).

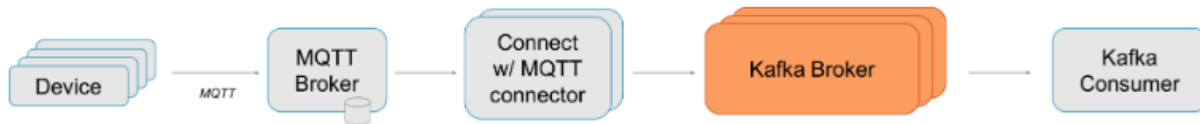


Figure 68 - MQTT Technology integrated with Kafka [18]

5.1.4. REST services paradigm

The REST (Representational State Transfer) architecture is a lightweight way of designing distributed applications. This paradigm is used in systems that expose Web Services and therefore use the Request / Response mechanism as a message exchange pattern. Recently, in order to create an Application Service Interface (API), the REST mechanism is preferred to the implementation solutions that use the SOAP protocol. Unlike SOAP technology, which uses HTTP as a transport protocol (SOAP over HTTP), the REST architecture uses HTTP as an application protocol, fully exploiting its advantages. An important element that distinguishes the REST architecture is the fact that it uses the standard HTTP methods, such as POST, in a way that is consistent with the HTTP protocol definition. An undoubted benefit of the systems that implement REST APIs is the possibility of using a variety of data formats (JSON, XML, Yaml) , whereas SOAP only allows XML For this reason systems with REST architecture that use JSON are simpler to realize and are characterized by greater performance (less overhead and less use of bandwidth).

5.2. Message formats for external interfaces

5.2.1. Design Choice of Message Serialization Format

Data exchanges between all components shall take place in the form of JSON objects (JavaScript Object Notation). Its syntax is a subset of the standard ECMA and its structure consists of collections of name/value pairs. The choice to use the JSON format to serialize the messages exchanged within the RESISTO infrastructure derives from the following considerations:

- JSON has a simple structure and a human-readable data representation;
- There are libraries managing this type of data for each language;
- The overhead added by its structure is almost negligible and the parsing and the construction of messages has a minimal computational impact even for devices with low performance;
- It represents a known standard that makes it easy to integrate it with external systems by eliminating the complication of having to interface with a proprietary format.

Furthermore, this format easily allows a possible expansion of the structure: if needed, it will be possible to add fields without compromising the existing package structure in order to expand the services offered.

5.2.2. Message Exchange Formats

A crucial and fundamental aspect for the integration of the different components concerns the format of the information shared between the STCL and external sources. The main difficulty consists in making different parts communicate using a single language as it is not easy to define a unique message format coming from different systems that form a very diverse ecosystem, characterized by sensors and detectors that use different interfaces to exchange information.

Moreover, the complexity of structuring this information also depends on the fact that the types of information concern both physical security and cyber security, two completely separate fields.

In the cybersecurity area there have been several attempts to define standards. The main objective is to identify a dictionary to indicate a generic event / attack coming from different types of sensors and then develop a common language based on common enumerations able to facilitate the sharing of information on computer security.

The following sections describe the main standards that have emerged in the cybersecurity field and that we propose to use in the RESISTO platform:

1. *IDMEF* - Intrusion detection message exchange format,
2. *STIX 2.0* - Structured Threat Information eXpression,
3. *IDEA* - Intrusion Detection Extensible Alert.

IDMEF

IDMEF defines a data and transport model for sharing security event data exported by intrusion detection systems and by event correlation engines [19]. The purpose of IDMEF is to define formats for the exchange of intrusion detection information and to define a structure for the storage of this information. It is designed to foster interchange of data between commercial and open source intrusion detection equipment and incident management stations. Data exchanges are done using XML

IDMEF descriptions include information about the analyzer itself, timing (analyze/create/detect time), network data about the source and target and an event classification that can include a CVE reference. An IDMEF message can be either an alert or a heartbeat. It is a well-structured object-oriented format (Figure 69) documented in the RFC 4765.

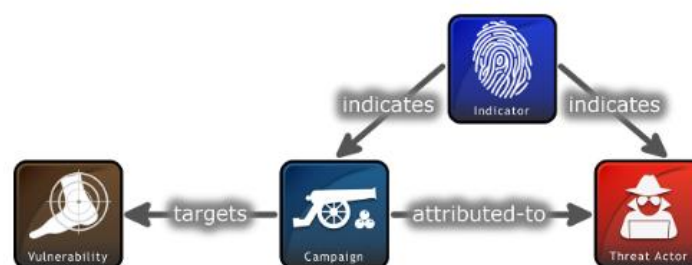


Figure 69 - IDMEF Message Structure

STIX 2.0

STIX is a language for describing a wide range of security-related information.

STIX enables organizations to share Cyber Threat Intelligence (CTI) with one another in a consistent and machine readable manner, allowing security communities to better understand what computer-based attacks they are most likely to see and to anticipate and/or respond to those attacks faster and more effectively. STIX is designed to improve many different capabilities, such as collaborative threat analysis, automated threat exchange, automated detection and response, and more.



STIX Relationship Example

Figure 70 - STIX Relationship Example

STIX Objects categorize each piece of information with specific attributes to be populated. Chaining multiple objects together through relationships allow for easy or complex representations of CTI. With STIX, all aspects of suspicion, compromise and attribution can be represented with objects and

descriptive relationships. STIX information can be visually represented for an analyst or stored as JSON to be machine readable [20].

IDEA

IDEA is a very flexible format for security event exchange developed by CESNET an association of universities of the Czech Republic and the Czech Academy of Sciences that is responsible for operating and developing the Czech national [e-infrastructure](#) for science, research and education.

IDEA format arises from the need to include most of the information exchanged between honeypots, agents detection probes, system logs and network traffic in order to define the security event model [21].

The solution proposed by CESENT is to implement a structure of the event message considering the existing standards and evaluating the benefits and disadvantages. IDEA is based on the JSON format where serialization of messages is simple and where it was decided to use a two level deep tree of keys and values. Descriptive data formats, and especially security event formats, are based on key:value models, where a key can be a simple token or a path in a directory tree.

In order to better clarify the IDEA format structure, an example taken from CSENET documentation is shown below [22].

```
{
  "Format": "IDEA0",
  "ID": "4390fc3f-c753-4a3e-bc83-1b44f24baf75",
  "CreateTime": "2012-11-03T10:00:02Z",
  "DetectTime": "2012-11-03T10:00:07Z",
  "WinStartTime": "2012-11-03T05:00:00Z",
  "WinEndTime": "2012-11-03T10:00:00Z",
  "EventTime": "2012-11-03T07:36:00Z",
  "CeaseTime": "2012-11-03T09:55:22Z",
  "Category": ["Fraud.Phishing"],
  "Ref": ["cve:CVE-1234-5678"],
  "Confidence": 1,
  "Note": "Synthetic example",
  "ConnCount": 20,
  "Source": [
    {
      "Type": ["Phishing"],
      "IP4": ["192.168.0.2-192.168.0.5", "192.168.0.10/25"],
      "IP6": ["2001:0db8:0000:0000:0000:ff00:0042::/112"],
      "Hostname": ["example.com"],
      "URL": ["http://example.com/cgi-bin/killemall"],
      "Proto": ["tcp", "http"],
      "AttachHand": ["att1"],
      "Netname": ["ripe:IANA-CBLK-RESERVED1"]
    }
  ],
  "Target": [
    {
      "Type": ["Backscatter", "OriginSpam"],
      "Email": ["innocent@example.com"],
      "Spoofed": true
    },
    {
      "IP4": ["10.2.2.0/24"],
      "Anonymised": true
    }
  ],
  "Attach": [
    {
      "Handle": "att1",
      "FileName": ["killemall"],
      "Type": ["Malware"],
      "ContentType": "application/octet-stream",
      "Hash": ["sha1:0c4a38c3569f0cc632e74f4c"],
      "Size": 46,
      "Ref": ["Trojan-Spy:W32/FinSpy.A"],
      "ContentEncoding": "base64",
      "Content": "TVpqdXN0a2lkZGluZwo="
    }
  ],
  "Node": [
    {
      "Name": "cz.cesnet.kippo-honey",
      "Type": ["Protocol", "Honeypot"],
      "SW": ["Kippo"],
      "AggrWin": "00:05:00"
    }
  ]
}
```

5.2.3. Design Choice of Message Exchange Format

In the RESISTO project the choice is to adopt the IDEA standard as a message exchange format. The reason for choosing the IDEA standard is mainly due to the following considerations:

- simple data structure not too verbose or complex,
- data model based on the JSON standard,
- easily extendable data scheme to manage different types of events,
- incident classifications based on eCSIRT.net taxonomy [23].

The schema in Figure 71 shows the list of categories/subcategories for the classification of events for the IDEA message [24].

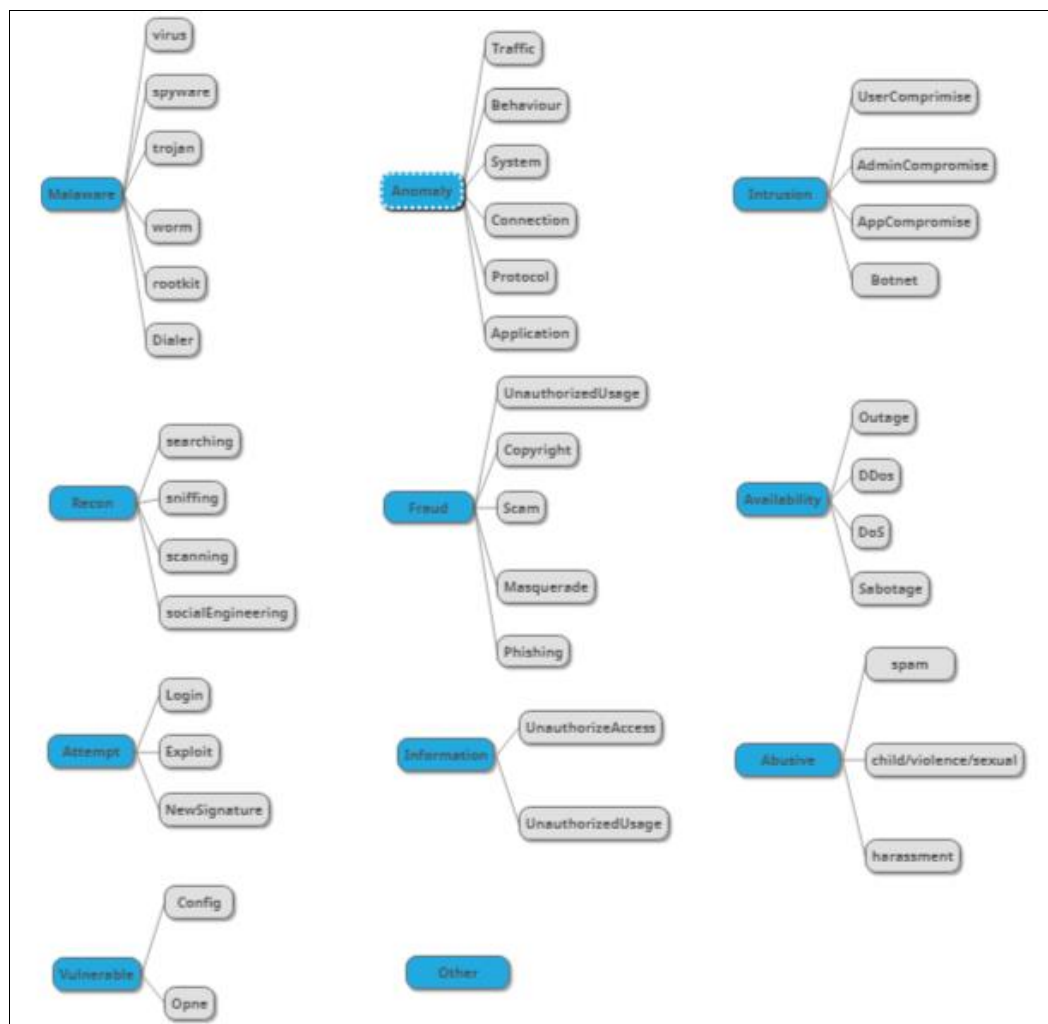


Figure 71 - IDEA message event categories

However one must keep in mind that this format was developed to describe alarms in a cyber environment. For this reason the IDEA scheme will be expanded with information on events related to the physical domain. Moreover, the technologies used in RESISTO for detection of physical threats are heterogeneous (UAVs, drones, IoT devices, audio/video and radio spectrum sensors) and each characterized by specific properties.

An aspect that is still in the evaluation phase and that will be analyzed at a later stage, when the Use Cases and relevant testbeds definition will be completed, is represented by the information collected from the PSIM systems. For the type of events / alarms coming from these sources the possibility of adopting a different standard must be considered.

The following chapter will describe the peculiar information of the various detectors and briefly indicate the fields and properties that must be added to the "IDEA" scheme.

5.3. Secure RESISTO communication

A primary requirement of the RESISTO platform is that it should reduce the level of risk of a telecommunication infrastructure without introducing additional vulnerabilities within the system itself. So, RESISTO must guarantee efficiency of communications among the various components and at the same time ensure confidentiality and integrity of data exchanged among the modules. For these reasons, the proposed architectural solution will be implemented according to the guidelines and best practices indicated by the OSWAP foundation [26] and following the concept of "Security by Design" to build a robust and secure system. The OWASP guidelines and principles shall be adopted, for accurate understanding of the data flows, interaction among the different components, potential threats and data handling.

RESISTO needs to communicate on a network with the various components, so the network infrastructure on which RESISTO will be installed shall be intrinsically safe, secure and reliable. This kind of connectivity is a prerequisite for system installation.

All communications foreseen by the system shall be made safe, secure and reliable using standardized and widely used methods.

The information security framework shall provide:

- Authentication: the act of verifying a claim of identity;
- Confidentiality: information cannot be made available or disclosed to unauthorized individuals, entities, or processes;
- Integrity: data integrity means maintaining and assuring the accuracy and completeness of data over its entire lifecycle. This means that data cannot be modified in an unauthorized or undetected manner.

Communications security is guaranteed through specific protocols that use authentication and encryption mechanisms as symmetric and asymmetric keys associated with digital certificates in standard X.509 format.

To protect web-type http based communications between client and server, the HTTPS protocol can be used. Standard HTTPS protocol is implemented natively by Internet browsers and Web servers, so this communication protocol can be used by properly configuring these components.

Where the communication uses different protocols not based on http(s), there are similar extensions that introduce security based essentially on SSL/TLS protocol. In some cases it is more convenient to use a VPN that guarantees point to point security at the IP protocol level. There are open source implementations of this mechanism like OpenVPN that is widely used in a long time.

To use these mechanisms that involve the use of cryptography, it is necessary to have the appropriate cryptographic credentials consisting of key pairs, usually RSA type ones, and the relative standard digital certificates X.509. These credentials can be purchased by Certification Authority (CA) or generated independently with specific software. In the context of RESISTO project validation phase, aiming to a TRL7 prototype, certificates and keys needed for the correct functioning of the solution could be generated internally with appropriate software such as OpenSSL.

6. RESISTO PLATFORM INTERFACES IDENTIFICATION

One of the first aspects that must be addressed during the definition of a software architecture are the interfaces used to exchange information between the various entities that constitute the system. Figure 72 presents the component diagram of the RESISTO platform modules.

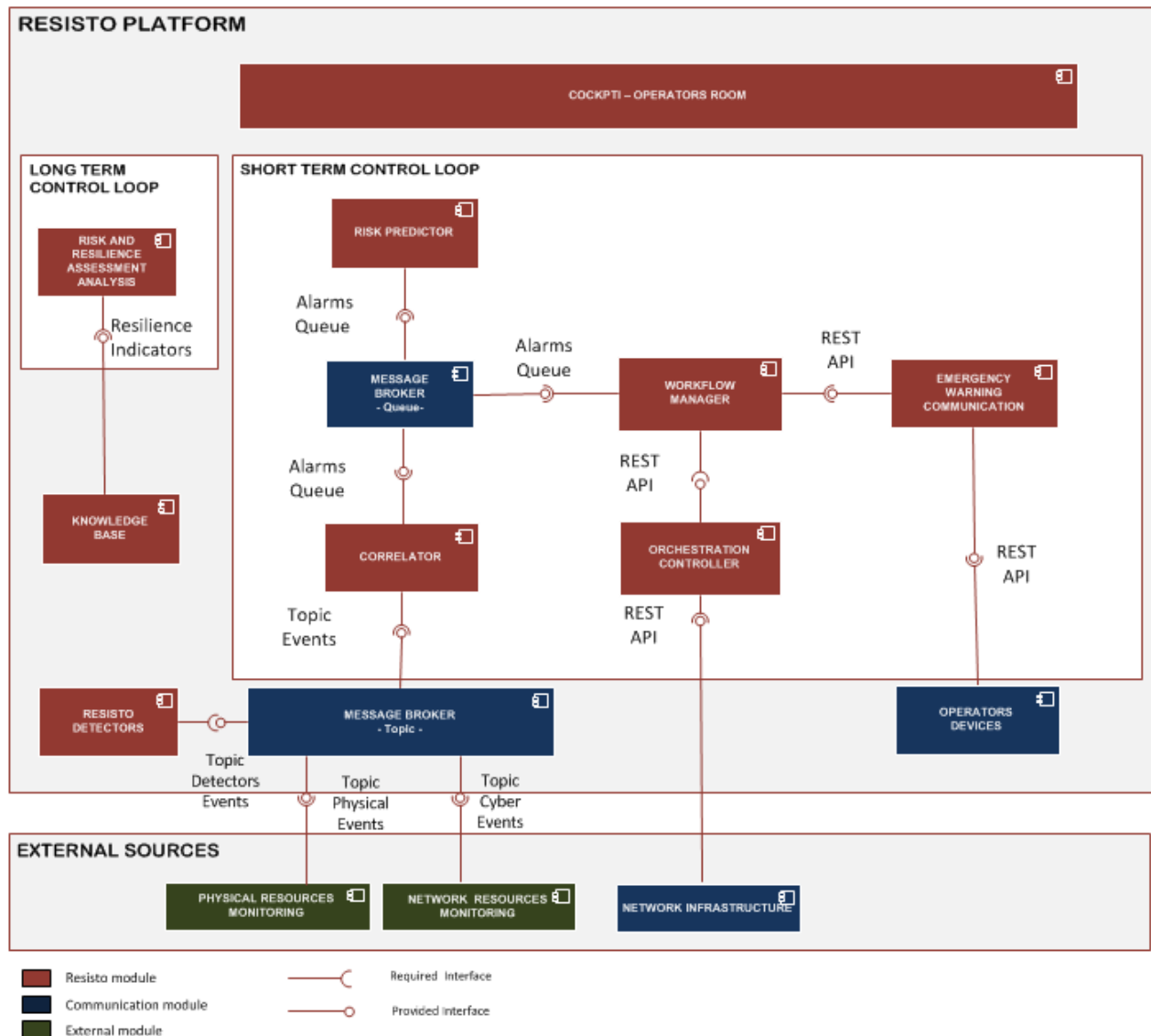


Figure 72 - RESISTO component diagram

The diagram illustrates STCL and the LTCL component data flows. The aim is to report the interfaces provided and required by each component and understand how each component interacts with the others.

It should be noted that the LTCL does not interact directly with the external sources. Its role is to identify the resilience indicators for specific events to better evaluate and estimate prevention mechanisms. The results of every LTCL elaboration cycle will be written on the RESISTO platform Knowledge Base.

The STCL is the runtime component of the RESISTO platform. It is involved immediately when a critical situation occurs in the detection/reaction/mitigation phase to promptly detect adverse events. It

must immediately collect all the necessary information from the physical/network resources monitoring and form the RESISTO detectors.

The message exchange between these components is based on asynchronous mechanisms using a message broker to increase efficiency and performance.

The Risk Predictor and the Workflow Manager receive the alarms from the Correlator and this interface will be realized through a messaging queue system.

The Workflow Manager needs to send commands to the Orchestrator, to define the most appropriate network configuration, and to the Emergency Warning Communication component in order to drive on-field operators' interventions. In these cases a REST API mechanism is preferred.

The interfaces to the Cockpit component are not explicitly shown in the figure, as this element is dedicated to containing the HMIs of the STCL components. The method of integration with the HMI applications used by the operators is specified in chapter 3.3.

STCL external and internal interfaces are described in detail in D6.1 ([12]) and not reported in this document for confidentiality reasons.

7. SYSTEM REQUIREMENTS TRACEABILITY

The following Table 7 contains the traceability between the RESISTO system requirements collected in [2] and the RESISTO components described in this document in charge to cover the requirement.

Req. Id	Requirement Description	Component
RES_FUN_0005	RESISTO shall exploit the outcomes of the cyber security and the physical security systems of the TLC infrastructures (if existing).	Cyber/Physical Events Correlator
RES_FUN_0006	RESISTO shall provide physical intrusion detection based on a variety of sensors, such as audio/video/radar and other passive and active sensors.	RESISTO Detectors
RES_FUN_0030	The RESISTO system shall be able to receive, collect and process alert events relevant to physical detection.	Cyber/Physical Events Correlator
RES_FUN_0070	RESISTO shall suggest to the operator the necessary steps to mitigate the effect of a cyber/physical attack.	Workflow Manager
RES_FUN_0100	The RESISTO system shall collect non-authorized personnel access inside the telecom facility if provided by the operator.	Cyber/Physical Events Correlator
RES_FUN_0390	The RESISTO system shall have a whitelist with all authorized radio devices inside the telecom facility.	Small Spectrum Surveillance
RES_FUN_0400	The RESISTO system shall have a list of all authorized cells or base stations and their operating frequency range in a target area.	Small Spectrum Surveillance
RES_FUN_0550	The <i>Mitigation Module</i> shall provide automated or semi-automated mitigation responses based on pre-defined templates.	Workflow Manager
RES_FUN_0560	The <i>Risk (Impact) Predictor</i> shall also include a network impact as well.	Risk Predictor
RES_FUN_0570	The <i>Risk and resilience assessment analysis</i> shall also take into consideration network single point of failure nodes, using network metrics such as: <ul style="list-style-type: none"> • Link state protocol databases for alternative IGP routes, • BGP secondary paths for EGP routes, • HSRP/VRRP/GLBP statuses for gateway redundancy. 	LTCL
RES_FUN_0585	The RESISTO platform shall provide a mitigation action at run-time when any backup resource will not be available anymore.	Workflow Manager

Req. Id	Requirement Description	Component
RES_FUN_0660	The <i>Vulnerability Disclosure Framework</i> shall be able to authenticate users and security researchers.	Responsible Disclosure Framework
RES_FUN_0670	The <i>Vulnerability Disclosure Framework</i> shall be able to provide users with functionalities to define the scope for testing, rewards for different types of threats.	Responsible Disclosure Framework
RES_FUN_0680	The <i>Vulnerability Disclosure Framework</i> shall be able to allow Security Researchers to submit findings.	Responsible Disclosure Framework
RES_FUN_0690	The <i>Vulnerability Disclosure Framework</i> shall be able to reward Security Researchers based on a matrix of rewards defined by users.	Responsible Disclosure Framework
RES_FUN_0700	The <i>Vulnerability Disclosure Framework</i> shall be able to help Security Researchers and users to monitor vulnerabilities reported through the whole cycle: <ul style="list-style-type: none"> • report the finding, • confirm/reject/request additional information from the security researcher, • notify the stakeholders, • patch the finding, • confirm from the security researcher that the issue was fixed, • reward the security researcher, if appropriate. 	Responsible Disclosure Framework
RES_FUN_0870	The RESISTO platform shall be able to produce a report for each attack/mitigation action set containing relevant elements such as duration of the attack, types of traffic or sensors that triggered the attack for security insight, etc.	Cockpit
RES_FUN_1040	The RESISTO system shall support the distribution of human-readable reports in industry-standard formats such as .PDF or .HTML.	Cockpit
RES_FUN_1105	The RESISTO platform shall ensure user access authentication acc. to the security requirements.	IAM
RES_FUN_1106	All users of the RESISTO platform shall be authenticated.	IAM
RES_FUN_1107	The RESISTO system shall be able to order the seamless relocation and restoration of virtualized network resources in the event of failure or cyber/physical attack if provided by the operator control system, such that service continuity can be guaranteed.	Orchestrator Controller
RES_FUN_1108	The RESISTO system shall maintain updated information of compute, storage and network resources within the relevant infrastructure domain.	Risk Predictor

Req. Id	Requirement Description	Component
RES_IMP_0010	The <i>Smart Spectrum Surveillance</i> shall provide an interface in the Cockpit in order to change settings for the tools developed.	Cockpit
RES_INT_0230	Network interfaces of the Virtual Machines that host RESISTO components shall offer full support for both IPv6 and IPv4 TCP stacks.	Design Constraint for all component
RES_INT_0240	Network interfaces shall support standardized IEEE 802.3 Ethernet technology for interoperability.	Design Constraint for all component
RES_SEC_0105	The integrity of the relevant information sent by the security sensors in the system shall be protected by the RESISTO system.	KSI Blockchain
RES_SEC_0110	Access to the RESISTO system shall be granted using the principle of "Least Privilege", meaning that any program, any interface, any debugging and testing console and every user of RESISTO should operate using the least set of privileges necessary to complete the job.	IAM
RES_SEC_0120	Each user shall be identified by a unique user identity so that users can be linked to and take responsibility for their actions.	IAM
RES_SEC_0160	RESISTO shall support the User's user access / segregation of duty requirements i.e. it supports set up of standard and group profiles.	IAM
RES_SEC_0245	Platform shall support remote login using encrypted protocols, such as HTTPS and SSH with only TLSv1.2 or above algorithms.	IAM
RES_SEC_0280	All staff and third parties who access the RESISTO network remotely shall only be authenticated using the approved remote access authentication mechanism	IAM
RES_SEC_0435	Resisto shall use services for protecting integrity and confidentiality of the data	KSI Blockchain, IAM
RES_SEC_0440	The RESISTO platform shall transmit all passwords over a secure connection.	Data Integration Layer
RES_SEC_0640	Access to Confidential, Personal and security data shall be logged.	IAM
RES_SEC_0650	RESISTO shall provide the user with the ability to import and export data from other systems in standard formats such as CSV, XML, XLS (e.g. "physical security alerts of a selected time interval").	Cockpit
RES_SEC_0720	User and system data shall be stored in a data store with adequate access control measures/policies.	IAM, Knowledge base

Req. Id	Requirement Description	Component
RES_SEC_0730	Direct access to the platform's data store shall only be allowed to users with privileged access rights (such as system administrators).	IAM, Knowledge base
RES_OPR_0125	The system shall be able to run on OSs and/or Virtualization environments offering "snapshot" mechanism in order to provide immediate reverse in case of major fault.	All

Table 7 - System Requirements vs. architecture components traceability

8. CONCLUSIONS

The present deliverable D2.7 - “RESISTO platform and tools reference architecture - final” documents the final results and outputs of Task 2.4 - “RESISTO reference architecture for long term preparation and short term disruptions” included in Work Package WP2 - “Use Cases and holistic systems modelling” as defined in [1].

Deliverable D2.7 overrides previous D2.6 ([4]), it also includes and refines the main elements already contained in D6.1 ([12]). Some interface details contained in D6.1 are not reported in this document for confidentiality reasons.

In D2.7 the High-level loop composed by the interaction between the Long-term and Short-Term Control Loops is fully described. This interaction triggers a Resilience Continuous Improvement process.

The Long-Term Control Loop is only briefly illustrated in this deliverable in relation to the Short-Term Control Loop the design of which is the main concern of this task. A detailed description of the Long-Term Control Loop can be found in [5] and it will be further described and refined in D3.2 – “Risk and resilience management process for cyber-physical threats of telecom CI – final”.

In the course of the task RESISTO platform architectural components have been identified and described from a functional and also technological point of view. For each component the most suitable technologies based on the partners’ current portfolios have been chosen. Thus the RESISTO components will make use of open source products such as *Apache Storm*, *Esper*, *Apache Kafka* and *Apache Fume* as well as of partners’ proprietary solutions such as, for example, *CISIApro* developed by Roma Tre and *SC2* developed by Leonardo.

REFERENCES

ID	REFERENCE
[1]	RESISTO - Grant Agreement. Project Starting Date: May, 1st 2018, Innovation action Number 786409 RESISTO, and following amendments, Ref. Ares(2019)3472075-28/05/2019
[2]	RESISTO D2.1 - End user requirements for integrated cyber-physical risk and resilience management, platform and tool
[3]	RESISTO D2.5 - Telecommunication system model and interfaces - final
[4]	RESISTO D2.6 - RESISTO platform and tools reference architecture –first
[5]	RESISTO D3.1 - Risk and resilience management process for cyber-physical threats of telecom CI
[6]	RESISTO D3.4 - Methods for cyber-physical security management for telecom CI
[7]	RESISTO D3.6 - Damage/Vulnerability models for physical and cyber threats of telecom CI
[8]	RESISTO D3.7 - KPIs, quantities and metrics for cyber-physical risk and resilience of telecom CI – first
[9]	RESISTO D4.1 - “Active and Passive Sensor Definition”
[10]	RESISTO D5.2 - Interim software Defined Security System and Decision Making Module
[11]	RESISTO D5.3 - Interim workflow Definition and Emergency Warning Communication Function
[12]	RESISTO D6.1 – SW architecture definition
[13]	CISI Apro Webpage http://cisiapro.dia.uniroma3.it/
[14]	www.atena-h2020.eu
[15]	https://osm.etsi.org
[16]	ETSI OSM Northbound API - https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/02.05.01_60/gs_nfv-sol005v020501p.pdf
[17]	http://www.egov.ee/media/1374/martinovic-blockchains-design-principles-applications-and-case-studies.pdf
[18]	https://dzone.com/articles/apache-kafka-mqtt-end-to-end-iot-integration-githu
[19]	https://www.enisa.europa.eu/publications/standards-and-tools-for-exchange-and-processing-of-actionable-information
[20]	https://oasis-open.github.io/cti-documentation/
[21]	http://www.wseas.us/e-library/conferences/2013/Rhodes/COMPUTE/COMPUTE-36.pdf
[22]	https://idea.cesnet.cz/en/definition
[23]	https://www.enisa.europa.eu/publications/using-taxonomies-in-incident-prevention-detection

ID	REFERENCE
[24]	https://idea.cesnet.cz/en/classifications
[25]	Node-RED Flow-based programming - https://nodered.org/about/
[26]	https://www.owasp.org/
[27]	ONOS REST API - https://wiki.onosproject.org/display/ONOS/Appendix+B%3A+REST+API

Table 8 - Reference Table