FP7-610416-PANOPTESEC

Dynamic Risk Approaches for Automated Cyber Defence

D5.4.2: Response System for Dynamic Risk Management Integration Prototype Report

Work-Package	WP5	Deliverable	D5.4.2
Due Date	30-06-2016	Submission Date	30-06-2016
Main Author(s)	Alcatel-Lucent Bell Labs France (ALBLF)		
Contributors	IMT, UzL, UROME, SUPELEC, ALBLF.		
Version	V1.0	Status	Final
Dissemination Level	PU	Nature	R
Keywords	Dynamic Risk Management, Response System, Verification, Experimentation and Integration.		

Part of the Seventh

Framework Programme

Funded by the EC - DG Connect

www.panoptesec.eu

EXECUTIVE SUMMARY

The work package 5 of the PANOPTESEC project has the ambition to deliver a beyond state-of-the-art *Dynamic Risk Management Response System* prototype, which will be integrated as a centrepiece of the global *Security Management System* researched in the PANOPTESEC project.

In this deliverable we give a synthesis and analysis of the tests and experiments that were conducted within the implementation and refinement phases of the work package 5. Those tests enable to verify that the DRMRS prototypes are working as expected regarding the Specialized Requirements established during the early specification phases of the project (See [D5.1.1] deliverable). The experimentations reported also assess their scalability and performance.

The tests and experiment synthesises in this report allow assessing one of the main achievement of the work package 5, which is an integrated, verified and tested DRMRS at the Milestone 6 of the project, whereas the initial schedule expected to start the integration of the work package 5 sub-system on the Demonstration System of the project at this milestone.

The produced DRMRS and its components provide significant contribution beyond the state-of-the-art on Response Systems. Some of them already produced papers for valuable scientific venues. More of the contributions and experimentations synthesised in this deliverable will be exploited by the PANOPTESEC Consortium as basic inputs to publish scientific papers to valuable journals and conferences venues.

HISTORY

Version	Date	Name/Partner	Comment
V0.1	11-03-2016	Samuel Dubus/ALBLF	Initial creation of the document, with a first Table Of Content proposal.
V0.2	25-03-2016	Samuel Dubus/ALBLF	Add the stakeholders' definition in the Methodology Section 3.1. And, an initial list of Acronyms with their respective definition and some basic references about the PANOPTESEC project.
V0.3	17-05-2016	CIS-UROME, IMT, SUPELEC, ALBLF	IMT: Initial contribution that report on the produced SRD component prototype provided for sub-section 4.2. CIS-UROME: Initial contribution that report on the produced QB-HOC prototype for sub-section 5.2. SUPELEC: Initial contribution reporting on the produced AB-HOC component prototype for sub-section 5.1. ALBLF: Integration and review of Partners contribution, some remarks and comments provided.
V0.4	27-05-2016	IMT, SUPELEC, ALBLF	IMT: Augmented contribution reporting on the SRD in sub-section 4.2. Revised contribution following some ALBLF comments and remarks. SUPELEC: Augmented contribution provided on AB-HOC component. Revision of sub-section 5.1 following some comments of ALBLF. ALBLF: Initial contribution provided to report on the TRD software component prototype provided to sub-section 5.4.
V0.5	13-06-2016	IMT, SUPELEC, CIS-UROME, ALBLF	IMT: Refined contribution reporting on the SRD in sub-section 4.2, including reworked sub-section Contribution, an additional introduction description of the SRD following ALBLF comments and remarks. SUPELEC: Reworked and augmented contribution provided on AB-HOC component (mainly introduction and individual component V& sub-sections). CIS-UROME: Reworked QB-HOC contribution regarding the introduction, experimentation strategy, individual component V&V and additional experimentation. Augmented sub-system integration V&V sub-section. Provision a new contribution on experimentation, done on the Emulation Environment during Integration and Experimentation workshop in ACEA premises done between 30/05 and 01/06 2016, for Integration in ACEA Emulated Environment sub-section 5.2.7. ALBLF: Initial contribution provided to report on the AGG-TRQ component prototype for sub-section 3.1. Augmented contribution on TRD component prototype provided in sub-section 5.4.

V0.6	21-06-2016	UzL,IMT, SUPELEC, CIS-UROME, ALBLF	UzL: Contribution on the reporting about the ROIA component prototype V&V, experimentations and integration. IMT: Revision of the contribution on the SRD component prototype including additional experimentations and reworking of other sub-sections according to comments and remarks. SUPELEC: Revised contribution on AB-HOC, with additional experimentation of the prototype on Emulation Environment. CIS-UROME: Revised contribution on QB-HOC, with reworked sub-sections on V&V additional experimentation and experimentation of the prototype on Emulation Environment. ALBLF: Augmented contribution on the reporting of the AGG-TRQ component including revised contribution sub-sections and an Additional Experimentations sub-section. Revised contribution on the reporting about the TRD prototype including a new subsection with scientific experimentations.
V0.7	28-06-2016	UzL,IMT, SUPELEC, CIS-UROME, ALBLF	UzL: Slight rewording and references in Section 3.2 about ROIA provided for Sections 7. IMT: Rework of sub-section 4.1.4 about sub-system integration V&V of SRD component. Addition of Section 4.1 acronyms to table 1. SUPELEC: Slight rewording in Section 5.1 and addition of Section 5.1 acronyms to table 1. CIS-UROME: Slight rewording and figures referencing in Section 5.2. Addition of Section 5.2 acronyms to table 1. ALBLF: Reworking of sub-sections 3.1.3, 3.1.4 and 3.1.5 about V&V process of AGG-TRQ component. Provision of sub-section 5.4.3 and 5.4.4 on individual and sub-system integration V&V for TRD component. Provision of a Methodology Section 2.
V1.0	30-06-2016	CIS-UROME, SUPELEC, ALBLF	CIS-UROME & SUPELEC: Additional section 5.3 comparing the QBE and ABE approaches provided to answer QA Review. ALBLF: Revisions after QA Review process. Provision of proper Conclusion, Introduction and Executive Summary.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

ACRONYMS AND DEFINITIONS

Table 1: Acronym List

Acronym	Meaning
ABE	Automaton Based Engine
ACEA	ACEA S.p.A.
ACL	Access Control List
AGG	Attack Graph Generator
AGG-TRQ	Attack Graph Generator – Tactical Response Quantifier
AIV	Annual Infrastructure Value
ALBLF	Alcatel-Lucent Bell Labs France
ALE	Annual Infrastructure Value
ARC	Annual Response Cost
AS01HV	Attack Scenario 01 High Voltage
CEP	Complex Event Processing
CS	Cosine Similarity
CIS-UROME	Universita Degli Studi Di Roma La Sapienza
CVE	Common Vulnerabilities and Exposures
DRMRS	Dynamic Risk Management Response System
EPIST	Epistematica SRL
EAP	Enriched Attack Path
EAG	Enriched Attack Graph
HOC	High-level Online Correlation
IAP	Instantiated Attack Path
ICS	Industrial Control System
ICT	Information and Communication Technology
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IMT	Institut Mines-Telecom
JS	Jaccard Similarity
LLC	Low Level Correlation
MA	Mitigation Action

MIM	Mission Impact Model
PEP	Policy Enforcement Point
PRS	Proactive Response System
P_LLC_FN	Probability of raising False Negatives by the Low Level Correlator
P_LLC_FP	Probability of raising False Positives by the Low Level Correlator
QA	Quality Assurance
QAM	Quality Assurance Manager
QBE	Query Based Engine
RFIA	Response Financial Impact Assessor
RHEA	RHEA System S.A.
RM	Risk Mitigation
ROIA	Response Operational Impact Assessor
RORI	Return On Response Investment
RP	Response Plan
RRS	Reactive Response System
RTU	Remote Terminal Units
SPI	Security Policy Instantiation
SR	Specialized Requirements
SRD	Strategic Response Decider
STU	Supervisor Terminal Unit
SUPELEC	Ecole Supérieure D'Électricité
SVN	Subversion repository
UoL	Universität zu Lübeck
WP	Work Package
1-ED	1 - Edit Distance

Table 2: Definitions

Word or Phrase	Meaning
False negatives	Alerts that are not emitted when an attack occurs
False positives	Alerts that are emitted when no attack occurs
WP5	The Work Package 5 of the PANOPTESEC project in which the DRMRS is researched, designed, implemented and experimented.

1 INTRODUCTION

1.1 Context

This deliverable reports on the realisation of a beyond state-of-the-art *Dynamic Risk Management Response System* prototype. This integration prototype is a centrepiece of the global *Security Management System* researched in the PANOPTESEC project (i.e. the PANOPTESEC System) as described in its High-Level Design deliverable [D3.1.2].

1.2 Purpose

The purpose of this deliverable is twofold.

Give a synthesis and analysis of the tests and verification that were conducted during the implementation and refinement phases on each prototypes of the DRMRS sub-system and the integrated DRMRS prototype.

Report on the experimentation that was done, within the work package 5, on the DRMRS software prototypes to assess their scalability and performance. The experimentation results and measurements presented in this deliverable should be exploited by the PANOPTESEC Consortium as basic inputs to publish scientific papers to valuable journals and conferences venues.

1.3 Scope

The scope of this deliverable includes the tests, verification and experimentations of prototypes which implement the functional architecture of the Dynamic Risk Management Response System and covers the Specialized Requirements reported in the [D5.1.1] deliverable.

The deliverable also reports on some preliminary experimentation of the DRMRS sub-system integrated on the Demonstration test bed of the User Partner. This includes preliminary experimentation with components of the other work packages (e.g. work package 4).

The test and experimentation of the integrated DRMRS as a compound of the global PANOPTESEC Security Monitoring System is beyond the scope if this deliverable and should be reported in the [D7.4.2] at the end of the project.

***Note*:** Details of the PANOPTESEC Consortium approach and the mapping of the work package 5 sub-system (i.e. the DRMRS) in the global architecture of the PANOPTESEC System can be found in the Project Description of Work [DoW2013] and the PANOPTESEC System High-Level Design described in the [D3.1.2] deliverable. Whereas, further details and background on the *Dynamic Risk Management* approach are available in the [D2.1.1] Deficiency Analysis deliverable.

1.4 Document Structure

This D5.4.2 deliverable is structured in the following manner:

Section 1 *Introduction*: describes the context, purpose and scope of the deliverable.

Section 2 *Methodology*: describes the methodology followed in the development of the deliverable.

Section 3 *Dynamic Risk Management Response System Common Components*: reports on the tests, verification, integration and experimentation of the software components common to the two treatment chains of the DRMRS sub-system.

Section 4 *Dynamic Risk Management Response System Proactive Components*: reports on the tests, verification, integration and experimentation of the software components specific to the *proactive* treatment chain.

Section 5 *Dynamic Risk Management Response System Reactive Components*: reports on the tests, verification, integration and experimentation of the software components specific to the *reactive* treatment chain.

Section 6 *Conclusion*: summarizes the findings, results and recommendations.

Section 7 *References*: provides a list of references applicable to the D5.4.2 deliverable.

2 METHODOLOGY

2.1 Stakeholders

A group of stakeholders are involved in the frame of the WP5, the Dynamic Risk Management Response System (DRMRS) and the PANOPTESEC Project. The [D2.2.1] identifies extensively stakeholders' categories, together with actors that apply in the context of the global PANOPTESEC Project. Several profiles and user roles, which are recalled after, are identified for these stakeholders and apply in the context of the WP5. Some specific actors, which apply more specifically to the DRMRS, are also specified.

Note: Those stakeholders are those initially identified in the WP5 Specialized Requirement deliverable [D5.1.1].

WP5 stakeholders

- *Solution Provider*: a Partner of the PANOPTESEC Consortium that proposes scientific or technical solutions for which he is skilled and recognized in his community, to fulfill one or several of the objectives of the sub-system (e.g. Dynamic Risk Management Response System) researched, designed and developed in the purview of the Work Package. Within the WP5, UoL, CIS-UROME, SUPELEC, IMT and ALBLF are identified as Solution Providers.
- *User Partner*: a Partner of the PANOPTESEC Consortium which provide to other Partners the operational context and requirements (e.g. use cases, scenarios, experiment dataset and test bed) and control the appropriateness of the solution proposed by Solution Providers to this operational context and requirements. Within the WP5, ACEA is also involved as the User Partner (i.e. User Partner of the PANOPTESEC Project).
- *Work Package Leader*: a Partner of the PANOPTESEC Consortium that coordinate the work between the other Partners involved within a Work Package, validate the produced results according to the technical objectives of the Work Package as described in the [DoW2015] to ensure technical high quality, and enforce the schedule according to the [DoW2015]. Within the WP5, ALBLF assumes the Work Package Leader role.
- *Deliverable Editor*: a Partner of the PANOPTESEC Consortium that organizes and coordinate the writing of a deliverable between the other Partners within a Work Package, validate the contributions to ensure the technical high quality of the deliverable, and enforce the schedule to respect the due date of the deliverable according to the [DoW2015]. For the D5.4.2, ALBLF assumes the Editor role.

PANOPTESEC Project stakeholders

- *Technical Project Manager*: a Partner of the PANOPTESEC Consortium that manages the technical work between the other Partners involved in the various Work Packages, propose processes that ensure the smooth running of the technical progress, validate the produced results according to the global objectives of the Project as described in the [DoW2015] to ensure technical high quality and consistency, and enforce the technical processes and the technical schedule of the project according to the [DoW2015]. After a decision voted by the project Steering Committee in January 2016, the Technical Project Manager role is now assumed by IMT.

- *Project Coordinator*: a Partner of the PANOPTESEC Consortium that Coordinate all the aspect of the work between the other Partners involved in the Project, propose processes that ensure the smooth running of the Project within the Consortium and outside the Consortium, validate the produced results according to the global objectives of the Project as described in the [DoW2015] to ensure high quality, consistency and pertinence, and enforce the processes and the schedule of the project according to the [DoW2015]. IMT assumes the Project Coordinator role.

Dynamic Risk Management Response System stakeholders

- *Monitored System Administrator:* A person who, for an organization, is responsible for the inventory, deployment or/and configuration management of hardware and software systems that compose the monitored system(s) on a day to day basis, with the focus to keep the monitored system(s) running and fulfilling their missions. In the context of the WP5, are identifies more specifically two sub-roles.
    - *Network Administrator*: A person who, for an organization, is responsible for the inventory, deployment and configuration management of hardware and software systems that compose the monitored system(s) on a day to day basis, with the focus to keep the monitored system(s) up and running.
    - *Security Administrator*: A person who, for an organization, is responsible for the inventory, deployment and configuration management of hardware and software systems that compose the protection system(s) of the monitored system(s) on a days to day basis, with the focus of keeping the monitored system(s) secure according to rules derived from a security policy established for the monitored system(s) according to defined security objectives and policies of the organization.
- *Business Owner/Manager*: A person with an executive level function within the organisation interested in understanding the security status of the business (mission) processes and possible business impact due to cyber-attacks. He or she is also interested in improving the security level of the business he of she owns/manages for it to better fulfil its missions.

Beyond the user roles defined in the [D2.2.1] for identified actors in the context of the organization of the PANOPTESEC User Partner. An additional common user role in the security domain is identified for the WP5:

- *Security Officer*: A person who, in an organization, is responsible of the security and the management of security resources for monitored systems supporting missions and businesses of the organization. In particular, in order to achieve his objective, he usually has the responsibility to establish security objectives and a security policy relative to missions and businesses of the organization. A Security Officer is usually responsible for the enforcement of a security policy on monitored systems under its responsibility. In the context of the User Partner of PANOPTESEC, this role is carried out by Business Owners/Managers.

Definitely, a last actor applies also to the DRMRS, even if it is not a stakeholder per-se. The *Threat Agent Actor*, also called shortly in the context of the WP5:

- *Attacker*: a person, system or entity that performs advert actions on the monitored system of an organization that bypass the security policy in order to gain unauthorized information or privileges with the focus of harming the businesses or prevent the organization from achieving one or several of its missions.

2.2 Dynamic Risk Management Response System Design

Functional design

The first phase of design of the WP5 Dynamic Risk Management Response System (DRMRS) consisted in establishing a *functional design* (i.e. functional architecture) and the Specialized Requirements associated to each identified functional modules and knowledge in this functional architecture.

This functional architecture of a Dynamic Risk Management Response System and the Specialized Requirements are reported in details in the [D5.1.1] deliverable.

High-Level design

In a second phase of design, the established functional architecture of a DRMRS was decomposed into several Software Components that should implement the identified functions. Each Software Component covering a part, the totality or several functions of the DRMRS architecture was detailed.

The resulting Instantiation of the DRMRS functional architecture and the functional specification of the interfaces of each component composing it, were specified using the SysML formalism to comply with the work package 3 process of specifying the global PANOPTESEC System High-Level Design (See [D3.1.2]).

Components detailed design

A third phase of design had the purpose of specifying the detailed design of each component of the WP5 DRMRS.

The specific paradigms of design adopted during this phase has been to provide as much details, description and specification for each identified software component of the WP5 DRMRS that could help the implementation phase.

2.3 Dynamic Risk Management Response System Components Implementation and Refinement

After the design phases, the WP5 Partners engaged the development of the DRMRS sub-system, which is the part of the PANOPTESEC System (See [D3.1.2]) researched within the work package 5. According to decisions of the Project's Steering Committee, we adopted an iterative development approach based on the scheduling of Sprints of 4 weeks. Although not formally followed, we adopted a light AGIL approach inspired from the [Scrum] methodology.

The initially scheduled tasks of implementation and refinement in the project's description of Work [DoW2015] were split in eighteen Sprints up to the end of the WP5. During the eighteen Sprints, actions were scheduled all kinds of activities for implementation, unitary testing, sub-system integration, experimentations and integration for components of the WP5 DRMRS and for the sub-system itself.

On a management point of view, the Sprints were formalized in a Sprint Plan. Each Sprint ended with a Review organized by the WP5 Leader (acting as *Scrum Master*) during which the Producing Partners demonstrated the progress on the expectation scheduled in the Sprint Plan. The Review

usually occurred in presence of the Deputy Technical Manger of the PANOPTESEC project (acting as *Business Owner*). After the Review, a period occurred to reschedule the activities of future Sprints. A back log was also used to reschedule activities that were not fulfilled at the expected Sprint Review.

The Sprint Plan of the WP5 were scheduled in accordance with the Project's Milestones and enabled to deliver on time, first prototypes of the WP5 DRMRS components (i.e. at Milestone 4, 31 July 2015), second prototypes of the WP5 DRMRS components (i.e. at Milestone 5, 31 October 2015), and a final integrated prototype of the WP5 DRMRS (i.e. at Milestone 6, 30 June 2016).

2.4 Dynamic Risk Management Response System Verification & Validation

During the eighteen Sprints of implementation and refinement phases, the WP5 Partners adopted a continuous testing of their components, lightly formalized for demonstrating the progress on their component of the DRMRS during Sprint Reviews.

A more formal testing activity was also conducted in parallel, in order to assess that software prototypes released at each delivery Milestones (i.e. 5 and 6) cover their functional specification and was working as expected.

A Verification & Validation (V&V) process in three steps was then conducted over established version of the software components delivered for each Milestone:

1. *Requirement and design review*: during every software project lifecycle, it is common to have requirement changes and/or updates in the design due to the always-increasing understanding of the functionalities and their challenges. As a consequence, a fundamental step before proceeding in the "real" software verification is the assessment of the functional and non-functional requirements and the system design.
2. *Test Cases definition*: once requirements and design are assessed, it is possible to proceed with the definition of several test cases whose aim is to verify the software conformance with respect to (i) its functional specification following from functional requirements and (ii) its deployment and any other non-functional aspect following from non-functional requirements and design choices.
3. *Test Execution*: every defined test case is finally executed, by using multiple inputs, to verify the conformance of the actual results with respect to the expected ones.

The [Redmine] tool has been properly adapted and used as the online platform to store and manage all details of the Verification & Validation process.

A V&V report based on the results of the process has been produced at each delivery Milestone with the software prototypes.

2.5 Dynamic Risk Management Response System Integration and Experimentation

The integration process in the WP5 followed an iterative approach decomposed in two steps.

In a first phase, each software prototypes were tested, verified and validated according to its interfaces with the other components of the PANOTPESEC system. The behaviour and format of data exchanged on each interfaces were tested individually with data produced by peer components (i.e. other interacting components).

In a second phase, although the objective of the WP5 was to deliver a DRMRS sub-system to be integrated in the PANOPTESEC system starting at Milestone 6 (i.e. June 2016), we iteratively

integrated and experimented the WP5 DRMRS components directly in the test bed of the PANOPTESEC system. This allowed organizing several experimentation sessions on the Demonstration test bed of the User Partner [D7.4.2] (i.e. Emulation Environment of the project), in order to do experimentation involving several components on the two treatment chains (i.e. *proactive* and *reactive*) of the WP5 DRMRS in presence of attack scenarios.

2.6 Synthesis of results

The approach has been to provide a synthesis and analysis of the experimentation activities conducted during each testing phase of the implementation and refinement phases of the WP5 (i.e. V&V, additional experimentations and integration). The tests and experimentations were designed and executed with the goal to produce the required measurements to feed experimentation sections of scientific papers that the WP5 Partners have submitted (or will submit) to valuable conferences venues and journals.

Some of the tests and experimentations of this report were already inputted in five scientific papers published, presented or accepted in international conferences and journals.

2.7 Quality assurance

2.7.1 Quality criteria

The QA in the PANOPTESEC project relies on the assessment of a work product (i.e. deliverable) according to lists of QA checks (QA checklists) established by a QAM, validated at a Consortium level and centralized in the Project Handbook [PH15].

For the purpose of the QA of the D5.4.2, the deliverable MUST be assessed according to the following checklist:

- PEER REVIEW (PR) QA CHECKLIST: the D5.1.1 deliverable is a report, it then requires a proper peer review according to the checks defined in this checklist;

Note: the QA checklist that the WP5 MUST uses during the QA validation process of the D5.4.2 is available on the Project SVN (https://gotika.ifis.uni-luebeck.de/panoptesec/WP01/Project%20Handbook/Quality%20Assurance/QA%20Checklists).

2.7.2 Validation process

For the final validation of work products (i.e. deliverables) within the PANOPTESEC project, a final QA review process MUST be used before the issuing of a final version.

This QA validation process follows the Quality Review Procedure established by the QAM and validated by the Consortium in order to guarantee the high quality level of work products and to validate its adequacy according to the defined quality criteria chosen and defined for each deliverable (see Section 2.7.1). The Quality Review Procedure itself and the selection of the QA Review Committee are described in the PANOPTESEC Project Handbook [PH15]. It is specifically detailed in a PANOPTESEC Quality Review Procedure document available on the Project SVN (https://gotika.ifis.uni-luebeck.de/panoptesec/WP01/Project%20Handbook/Quality%20Assurance/QA%20Procedure).

The QA validation process is scheduled in the QA Schedule [QAS15] managed by the QAM. And, the detailed results obtained after the process took place are captured and stored in the Project log in a Quality Review Summary Report also available on the Project SVN (https://gotika.ifis.uni-luebeck.de/panoptesec/WP01/Project%20Handbook/Quality%20Assurance/QA%20Reports).

3 DYNAMIC RISK MANAGEMENT RESPONSE SYSTEM COMMON COMPONENTS

3.1 Attack Graph Generator - Threat Risk Quantifier (AGG-TRQ)

The Attack Graph Generator and Threat Risk Quantifier (i.e. AGG-TRQ) software component is a unified component that implements two functions of the global Functional Architecture of the DRMRS:

- Attack Graph Generator function as described in Section 3.1.1 of D5.1.1, and covers the corresponding established Specialized Requirements as defined in Section 4.2 of D5.1.1.
- Threat Risk Quantifier function as described in Section 3.2 of D5.1.1, and covers the corresponding established Specialized Requirements as defined in Section 4.5 of D5.1.1.

First, AGG-TRQ calculates the exposure of the monitored system to threats. This is achieved by calculating an Attack Graph, as a set of Attack paths. Each Attack path depicts an attack scenario that starts from a predefined entry point, and reaches to a critical machine in the system (monitored system). This exposure is captured on two levels:

- Proactive level: AGG-TRQ calculates attack graphs corresponding to all potential attack scenarios for the monitored system. Hence, the exposure of the monitored system is captured regardless whether there are ongoing attack attempts in the monitored system. This exposure is relevant to assess the proactive risk posture of the monitored system, which characterises the risk profiles of the monitored systems on the mid-long term.
- Reactive level: AGG-TRQ calculates attack graphs corresponding to observed and ongoing attack scenarios. Hence, the exposure is captured considering detected attack events and corresponding alerts detected by the intrusion detections systems. Such exposure is relevant to assess the reactive risk posture of the monitored system, which characterizes the risk profile of the monitored system on the short-term while considering ongoing and observed attacks.

Table 3: Functions of the AGG-TRQ component on proactive and reactive levels

	Proactive Level	Reactive Level
Attack Graph Generation	Calculates the potential exposure of the monitored system	Calculates current exposure of the monitored system considering ongoing attacks
Threat Risk Quantifier	Derives the profile of potential risks of the monitored system	Derives the profile of immediate risks induced by ongoing attacks in the monitored system

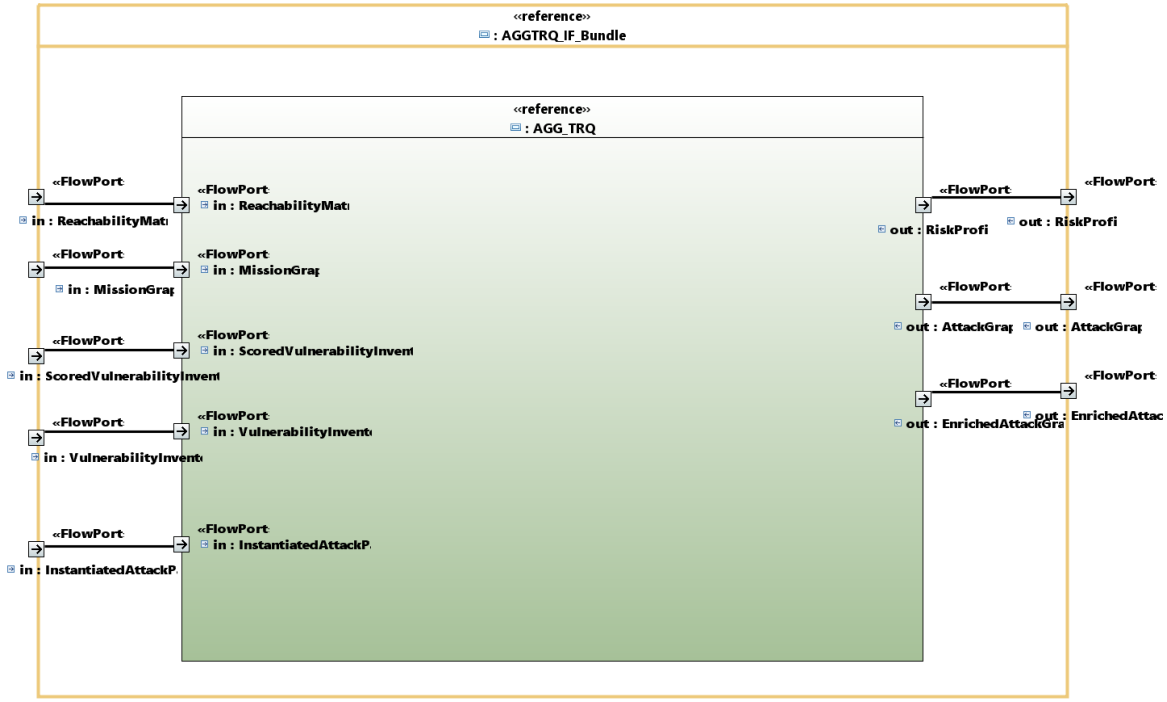

Figure 1 - High-level view of the AGG-TRQ component design with its various interfaces

In order to conduct the Attack Graph Generation and Threat risk quantifier, the AGG-TRQ components leverage the *Mission Graph* produced by the Mission Impact Module. The *Mission Graph* contains needed information such as entry points, supporting assets (i.e. critical machines), organizational model needed for the risk profile, etc.

The AGG-TRQ retrieves the *Reachability Matrix* from the Reachability Matrix Correlator (RMC), which is a sub module of Data Collection and Correlation System. The reachability matrix depicts the connectivity between the machines of the monitored system.

Furthermore, the AGG-TRQ retrieves two crucial TYPES OF information from Data Collection and Correlation System: (i) The *Vulnerability Inventory* (REAH) which depicts the vulnerabilities existing on certain machines in the monitored system, and (ii) the *Scored Vulnerability* which contains technical characteristics & metrics concerning the vulnerabilities.

For the reactive part, and in addition to aforementioned input, the AGG-TRQ leverages instantiated Attack Paths. This information is provided by the two High Level Online Correlators (HOC): AB-HOC et QB-HOC.

On the proactive level, the AGG-TRQ produces the following output: *Proactive Attack Graph*, *Proactive Risk Profile*. These output are by leveraged by the Strategic Response Decider (SRD) to derive the best proactive response. Moreover, the AGG-TRQ produced *Enriched Attack Graphs* that are mainly used by the Visualization component and High-level Online Correlators (HOC).

On the reactive level, the AGG-TRQ works tightly with TRD (for the reactive level) by delivering the following output:

- *Levelled Ongoing Attack Graphs*, in response to a Risk Contract, representing the Attack Graph corresponding to Attack Paths leading to risk levels above or equal the Risk Contract;

- *Residual Risk Profiles*, in response to Abstract Response Plans, representing the level of Risks on the Monitored System that may be reached with a corresponding Abstract Response Plan;

NOTE: Each of those interfaces has been strictly defined with regards to the format of the data and the sequence of messages exchanged during a preliminary design phase of the Work Package 5 components before starting the implementation. As we adopted a light Agile methodology for implementation, the design of the various interfaces have slightly evolved during the 18 Sprints we had for implementing, testing, verifying, refining and integrating the TRD software component.

3.1.1 Contributions

Traditional Risk Assessment are rather organizational (business-aware) than technical, and enable security officers to manage risks on the long run. However, both ICT systems and threat landscape do not cease to evolve, and dynamic cyber security management becomes paramount to address potential breaches. The operational security management is based on technical processes, executed by administrators who are not necessarily aware of organization's business and strategic aspects. This gap between technical and organizational levels renders traditional risks assessment methods cumbersome and obsolete. The AGG-TRQ leverages a novel concept of Elementary Risk (ER) that represents a quantum of risk for an organization. Composite Risks (CRs) enable dynamic calculation of risk posture while considering the system's state. The theoretical and fundamental contribution is presented in:

> **Waël Kanoun, Serge Papillon, and Samuel Dubus. Elementary Risks: Bridging Operational and Strategic Security Realms Elementary, 11th International Conference on Signal-Image Technology & Internet-Based Systems, Thailand, 2015.**

3.1.2 Testing and experimentation strategy description

In order to test, verify and validate our AGG-TRQ component, we devised a twofold approach:

- Case-study based experimentation
- Emulated data based experimentation.

The first approach (i.e. case-study based) aims at experimenting the core functions of the AGG-TRQ: attack graph generation, and risk profile assessment. For this objective, we devised a generic case-study which serves as synthetic data to validate our work and associated implementation. AGG-TRQ expected results are therefore established by a security expert, which will serve as reference in order to evaluate the accuracy of the AGG-TRQ. Such sound reference cannot be obtained by emulated data, since the expected output is beyond establishment by human effort. In other words, establishing expected output manually by human expert on emulated data is tedious, cumbersome and error prone.

The case study (depicted in below Figure) was devised as fine tuned compromise: a case study too large won't be useful since expected output cannot be pre-established by a security export. On the other hand, a case study too small/limited will undermine a genuine experimentation attempts. Hence, the devised case study provides the best trade-off between complexity and accuracy to experiment the AGG-TRQ core functions.

The second approach relies on emulated data (provided via SVN), which represents realistic data of the case study for the user (i.e. ACEA). This approach is used to experiment that our AGG-TRQ can be fully integrated within the integration framework (and therefore with other modules). Furthermore,

this approach is employed in order to verify that the AGG-TRQ retrieves and exports data with other modules as per pre-established models, formats, and exchange schemes.

These two approaches covers, in complementary fashion, the two major experimentation aspects for the AGG-TRQ. The first focuses on the core functions of AGG-TRQ, and verifies that output results are produced correctly and accurately considering well mastered input. On the other hand, the second focuses on the integration of the AGG-TRQ with the rest of the modules, and all information exchange are performed correctly and timely fashion.

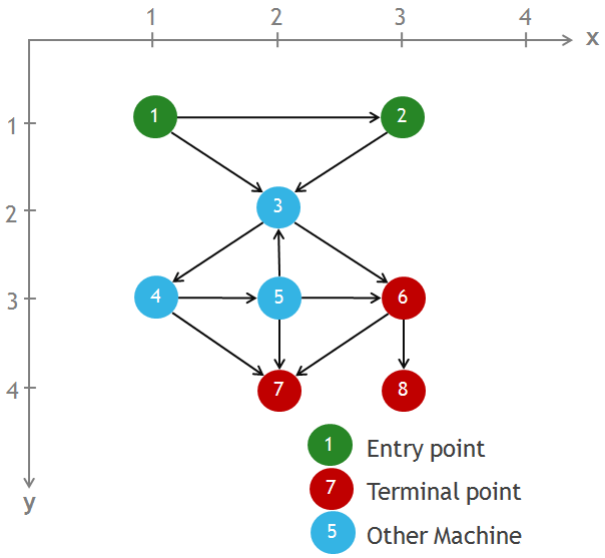

Figure 2 - Generic Case-Study used to experiment core functions of AGG-TRQ

3.1.3 Individual component V&V

Since the AGG-TRQ results from the merge of the following components:

- Attack Graph Generation (AGG),
- Risk Quantifier (RQU),
- Threat Impact Assessment (TIA), and
- Likelihood Assessment (LA).

In consequence, we verified that each requirement of the aforementioned modules is covered by our AGG-TRQ. In order to verify a given requirement, at least one test case is devises. Each test case defined the test to be performed on the AGG-TRQ in order to check whether the related requirement is converged. When a test case is executed, a test execution is therefore created which includes the exact input and output of the test. A requirement is covered when all test cases are verified, which means that their corresponding test executions are satisfactory.

Broadly speaking, each requirement addresses one of the two following aspects of the AGG-TRQ:

I. The core functions of the AGG-TRQ, which consists of (i) capturing the cyber exposure via attack graph generation, and (ii) calculation of the risk profile.

II. The integration of the AGG-TRQ with other components, and ensuring effective and convenient exchange between such modules.

In order to illustrate our approach, let's take for example requirement 6 of AGG (WP5.AG.R6), which states that the *Attack Graph Generation MUST compute as accurately as possible, the possible attack paths (i.e. direct and backtracking attack paths), which an attacker could use by exploiting vulnerabilities existing on devices (i.e. nodes) of each monitored system in the organization, from all identified Entry Points up to all known Supporting Assets*.

In order to verify that our AGG-TRQ generates a relevant, complete and accurate set of attack paths, we proposed the test case WP5.AGG.R6.TC1. This test case consists of comparing the output of AGG-TRQ to a known reference (i.e. expected output) as established by other means. In this case, the reference output is established by a security expert manually relying on the same input provided to the AGG-TRQ.

Afterwards, we proceed by executing this test case and thus instantiating the corresponding test execution (WP5.AGG.R6.TC1.TE1). The objective of a test execution is to specify the exact conditions, provided input, expected result/output, and actual result/output.

Provided input

A complete snaptshot, which contains the MissionGraph, ScoredVulnerability, ReachabilityMatrix, and the MissionGraph related to the use case presented in the previous section.

Description

Upon receiving and processing the snapshot, the AGG-TRQ calculates the attack paths list, and prints in the trace:

mars 29, 2016 4:36:20 PM com.alu.bl.alblf.scn.agg.Parse launchCalculation INFOS: Attack Graph Generator [CORE] : recalculating the attack pathes mars 29, 2016 4:36:21 PM com.alu.bl.alblf.scn.agg.Parse writeOutputAttackGraphEdgeEdges

Actual Result

The AGG-TRQ produced the following attack paths list:


```
AttackPath": [

{ "attackPath_Ident": "1", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "0", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "1", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "2", "vuln_rank": 1 }, { "rank": 4, "edge_ident": "3", "vuln_rank": 1
}, { "rank": 5, "edge_ident": "4", "vuln_rank": 1 } ], "likelihood": "0.152" },


{ "attackPath_Ident": "2", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "0", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "1", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "2", "vuln_rank": 1 }, { "rank": 4, "edge_ident": "3", "vuln_rank": 1
}, { "rank": 5, "edge_ident": "5", "vuln_rank": 1 } ], "likelihood": "0.152" },


{ "attackPath_Ident": "3", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "0", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "1", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "2", "vuln_rank": 1 }, { "rank": 4, "edge_ident": "3", "vuln_rank": 1
} ], "likelihood": "0.193" },


{ "attackPath_Ident": "4", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "0", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "1", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "2", "vuln_rank": 1 }, { "rank": 4, "edge_ident": "6", "vuln_rank": 1
} ], "likelihood": "0.193" },
```

```
{ "attackPath_Ident": "5", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "0", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "1", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "7", "vuln_rank": 1 } ], "likelihood": "0.263" },


{ "attackPath_Ident": "6", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "0", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "8", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "4", "vuln_rank": 1 } ], "likelihood": "0.263" },


{ "attackPath_Ident": "7", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "0", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "8", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "5", "vuln_rank": 1 } ], "likelihood": "0.263" },


{ "attackPath_Ident": "8", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "0", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "8", "vuln_rank": 1 } ],
"likelihood": "0.415" },


{ "attackPath_Ident": "9", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "9", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "1", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "2", "vuln_rank": 1 }, { "rank": 4, "edge_ident": "3", "vuln_rank": 1
}, { "rank": 5, "edge_ident": "4", "vuln_rank": 1 } ], "likelihood": "0.152" },


{ "attackPath_Ident": "10", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "9", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "1", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "2", "vuln_rank": 1 }, { "rank": 4, "edge_ident": "3", "vuln_rank": 1
}, { "rank": 5, "edge_ident": "5", "vuln_rank": 1 } ], "likelihood": "0.152" },


{ "attackPath_Ident": "11", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "9", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "1", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "2", "vuln_rank": 1 }, { "rank": 4, "edge_ident": "3", "vuln_rank": 1
} ], "likelihood": "0.193" },


{ "attackPath_Ident": "12", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "9", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "1", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "2", "vuln_rank": 1 }, { "rank": 4, "edge_ident": "6", "vuln_rank": 1
} ], "likelihood": "0.193" },


{ "attackPath_Ident": "13", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "9", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "1", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "7", "vuln_rank": 1 } ], "likelihood": "0.263" },


{ "attackPath_Ident": "14", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "9", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "8", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "4", "vuln_rank": 1 } ], "likelihood": "0.263" },


{ "attackPath_Ident": "15", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "9", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "8", "vuln_rank": 1 }, {
"rank": 3, "edge_ident": "5", "vuln_rank": 1 } ], "likelihood": "0.263" },


{ "attackPath_Ident": "16", "attackPath_Action": "new", "attackPathEdges": [ { "rank": 1,
"edge_ident": "9", "vuln_rank": 1 }, { "rank": 2, "edge_ident": "8", "vuln_rank": 1 } ],
"likelihood": "0.415" }

],
```

Expected output

The AGG-TRQ must generate the following list of Attack paths as per Expert expectation.

	Node #1	Node #2	Node #3	Node #4	Node #5	Node #6	Node #7
AP #1	1	2	3	4	5	6	
AP #2	1	2	3	4	5	6	8
AP #3	1	2	3	4	5	6	7
AP #4	1	2	3	4	5	7	
AP #5	1	2	3	4	7		
AP #6	1	2	3	6			
AP #7	1	2	3	6	7		
AP #8	1	2	3	6	8		
AP #9	1	3	4	5	6		
AP #10	1	3	4	5	6	8	
AP #11	1	3	4	5	6	7	
AP #12	1	3	4	5	7		
AP #13	1	3	4	7			
AP #14	1	3	6				
AP #15	1	3	6	7			
AP #16	1	3	6	8			

Upon comparison between the Expected results to Actual results, we verified that the generated list of attack paths, (outputAttackGraphEdge0.json), corresponds to the list of attack paths as established by an expert. Therefore, WP5.AGG.TC1.TE1 is passed. In consequence, the test case WP5.AGG.TC1 and corresponding requirement WP5.AGG.R6 are verified successfully.

For each requirement, we devised at least a test case in order to verify whether we are covering the requirement. Since AGG-TRQ components results from the merge of four previous components (AGG, RQU, TIA, and LA), we addressed the requirements of these four component. We note that because of such merge, some of the requirements are covered by design because they relate to internal aspects/calculation/treatment of the new components AGG-TRQ.

3.1.4 Sub-system integration V&V

The AGG-TRQ as central part in the DRMRS interacts with other components intra-WP5 (HOC, SRD and TRD), and extra-WP5 such as MIM and DCC. Therefore for the sub-system integration V&V, we adopted a testing approach similar to the individual component V&V, but focused on interface features of the AGG-TRQ software component.

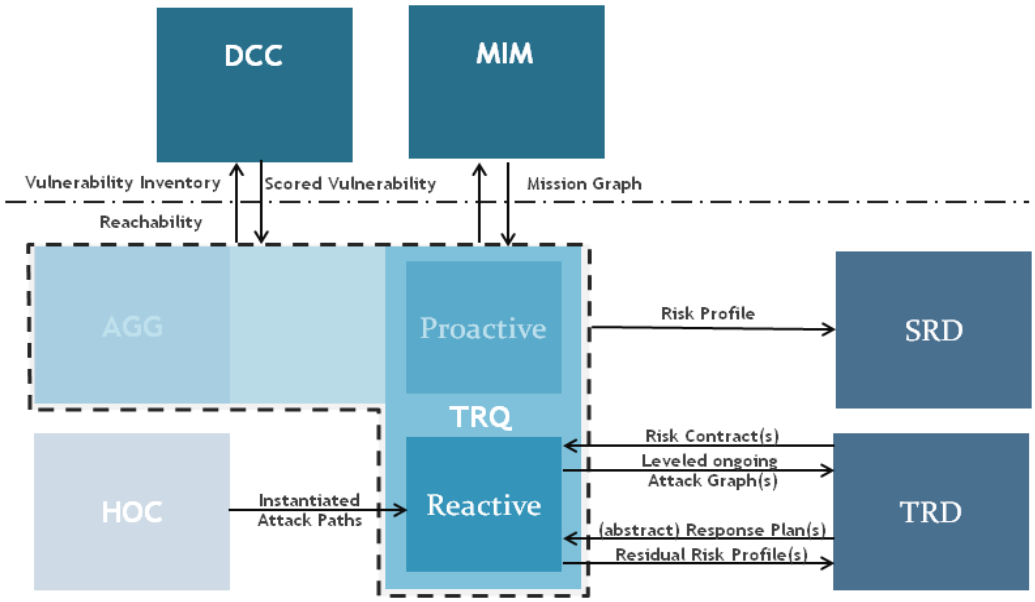

Figure 3 - Interaction between AGG-TRQ and other components

Therefore test cases relating to integration features verify the following:

- the correct processing of formatted data at the input of the component: we relied on formatted datasets produced by other components of the WP5 or components produced in other work packages (e.g. MIM component of the WP4 for formatted Mission Graph, and DCC of the WP4 for the Authorized Mitigation Actions);
- the behaviour of the TRD software component with regards to the input and output interfaces with the Integration Framework (i.e. the main interfacing middleware of the PANOPTESEC system).

3.1.5 Uncovered Specialized Requirements

During the refinement phase, the four ex-components AGG, RQU, TIA and LA were merged into one new single component (AGG-TRQ). Such major modification of the global design of the PANOPTESEC system was guided by the evolution of the understanding of the needs for such security monitoring system, and to optimize process time and memory usage. This had some impact on the need of some of the specified Specialized Requirement of the [D5.1.1], established at the beginning of the project. Hence, some requirements were deprecated during the V&V process as presented in the next table.

Requ. ID	Requirement Description	Motivation of Depreciation
WP5.AGG. R11	The Attack Graph Generation MAY receive (e.g. from the Potential Attack Identification Module) the list of nodes of a monitored system that have been observed as source nodes of attacks, and which are not currently identified as Entry Points of currently computed possible attack paths.	An interface is specified between AGG-TRQ and PAI modules in the high level design, but we did not specify the low lever design. We chose not to implement this requirement since the component PAI has not been developed (as it was defined as an optional component by its set of requirements "PAI").
WP5.AGG. R12	When the list of additional potential Entry Points associated to a monitored system has changed, the Attack Graph Generation MAY compute, as accurately as possible, the possible attack paths (i.e. direct and backtracking attack paths), which an attacker could use by exploiting vulnerabilities existing on devices (i.e. nodes), from all	This requirement is deprecated since the PAI (defined as an optional component by its set of requirements) did not implemented associated feature.

Requ. ID	Requirement Description	Motivation of Depreciation
	new additional potential Entry Points up to all identified Supporting Assets associated to this monitored system.¶	
WP5.AGG.R13	The Attack Graph Generation SHOULD stop an attack paths generation process on request, before the end of the computation of all attack paths for a monitored system.	This requirement is deprecated since the attack graph generation algorithm is conveniently fast, and stopping the process is irrelevant. Each computation will then ever go until the end before the component can process a new one.
WP5.AGG.R16	The Attack Graph Generation MAY receive a list of additional previously unidentified vulnerabilities potentially existing on nodes of a monitored system which are not currently identified in the regular vulnerabilities inventory, from a module of the PANOPTESEC system responsible of the establishment of this knowledge	By design, AGG-TRQ is always capable of receiving the latest state of monitored system and corresponding vulnerabilities via the ReachabilityMatrix.xml. When new vulnerability(ies) are discovered on one or several machines in the monitored system, the ReachabilityMatrix.xml is updated accordingly and sent to the AGG-TRQ. Subsequently, AGG-TRQ considers read the latest version of ReachabilityMatrix.xml (as per WP5.AGG.R2), and therefore considers the newly identified vulnerable machines. These unidentified vulnerabilities are supposed to be produced by the PotentialAttackidentified component, which has not been developed since it is optional. The requirement, then, cannot be fulfilled
WP5.AGG.R17	The Attack Graph Generation MAY compute, as accurately as possible, the additional possible attack paths (i.e. direct and backtracking attack paths), which an attacker could use by exploiting (i) vulnerabilities existing on devices (i.e. nodes) (ii) plus the additional previously unidentified vulnerabilities potentially existing on devices of a monitored system, from all identified Entry Points up to all identified Supporting Assets associated to this monitored system.	As per WP5.AGG.R8, AGG-TRQ always recomputes the list of attack paths considering the most up-to-date version of the ReachabilityMatrix.xml, including the case when new vulnerabilities are discovered on one or several machines. These unidentified vulnerabilities are supposed to be produced by the PotentialAttackidentified component, which has not been developed since it is optional. The requirement, then, cannot be fulfilled
WP5.AGG.R19	The Attack Graph Generation SHOULD protect the Vulnerability Inventory, the identified Entry Points and Supporting Assets, the Reachability and computed possible attack scenarios of each system it monitors or protect, both within the process memory or during their external storage, from unauthorized disclosure.	AGG-TRQ does not perform any storage operation. Concerning the protection runtime memory, this requirement is deprecated.
WP5.AGG.R20	The Attack Graph Generation MAY request necessary information of exploitability (i.e. for each vulnerability: supposed protocols or ports, and supposed level of privilege required to exploit it; together with, supposed protocols or ports, and supposed level of privilege gained if successfully exploited) for supposedly existing vulnerabilities on nodes of a monitored system, from a module of the PANOPTESEC system responsible of the establishment of this knowledge	This requirement is deprecated since the relevant information will be pushed to the AGG-TRQ via the ScoredVulnerabilityInventory.json and VulnerabilityInventory.xml files, and AGG-TRQ does not need additional information.
WP5.RQU.R7	The Risk Quantification MAY retrieve the risk crossing function to be used to compute a Risk level based on an impact value (e.g. as computed by the Impact Assessment for each Detrimental Event) and, a Likelihood value assessed on a proactive or a reactive perspective (e.g. as computed by the (Success) likelihood assessment) from the PANOPTESEC module that manages this knowledge	This requirement is deprecated since the Mission Graph does not contain the risk crossing function, which is not needed in any actual computation of AGG-TRQ.

Requ. ID	Requirement Description	Motivation of Depreciation
WP5.RQU.R20	The Risk Quantification SHOULD protect the possible and ongoing attack scenarios, the dependency model entities (e.g. Assets, Supporting Assets, and Detrimental Events), and the various computed values and levels of risks, both within the process memory and during their external storage, from disclosure to unauthorized module or user	AGG-TRQ does not perform any storage operation. Concerning the protection runtime mempry, this requirement is deprecated.
WP5.TIA.R8	The Threat Impact Assessment SHOULD protect the possible and ongoing attack scenarios, the dependency model entities (e.g. Assets, Supporting Assets, and Detrimental Events), and the computed impact characteristics associated to each Detrimental Events, both within the process memory or during their external storage, from disclosure to unauthorized module or user	AGG-TRQ does not perform any storage operation. Concerning the protection runtime mempry, this requirement is deprecated.
WP5.LA.R14	The Likelihood Assessment SHOULD protect the possible and ongoing attack scenarios, and their computed likelihood and success likelihood values, both within the process memory and during their external storage, from disclosure to unauthorized module or user	AGG-TRQ does not perform any storage operation. Concerning the protection runtime memory, this requirement is deprecated.

In total, the AGG-TRQ has 64 requirements, among which 10 has been deprecated for the reasons stated in the table above. Such relatively high number of deprecated requirements results from the merger of AGG, RQU, TIA, LA into a single component AGG-TRQ. The deprecated requirements covered data exchange between the old four components. Thus, the merge rendered these requirements obsolete.

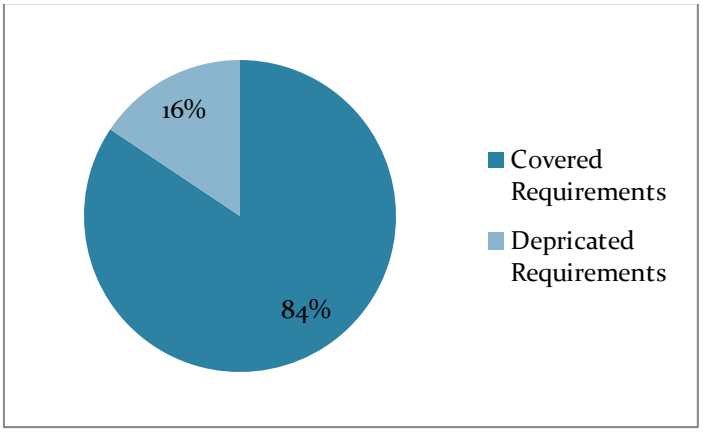

Figure 4 - Repartition of Covered vs Deprecated requirements of the AGG-TRQ

3.1.6 Additional experimentations

We conducted a thorough experimentation using data retrieved from scans in the Emulated environment. A dataset is composed of several files which are formatted as currently specified in the project SysML design project:

- A Reachability Matrix XML data file, produced from the Simulation Environment configuration = (i.e. ReachabilityMatrix).
- A Vulnerability Inventory XML data file, with the vulnerabilities collected on the Simulation Environment (i.e. VulnerabilityInventory).

- A Mission Graph JSON file, which enables to get the possible Entry Points and the devices that are important for the Business Processes of the organization simulated in the Simulation Environment.
- A Scored Vulnerability Inventory JSON file, with the exploitability and difficulty characteristics of the vulnerabilities in the Vulnerability Inventory file (i.e. ScoredVulnerabilityInventory).

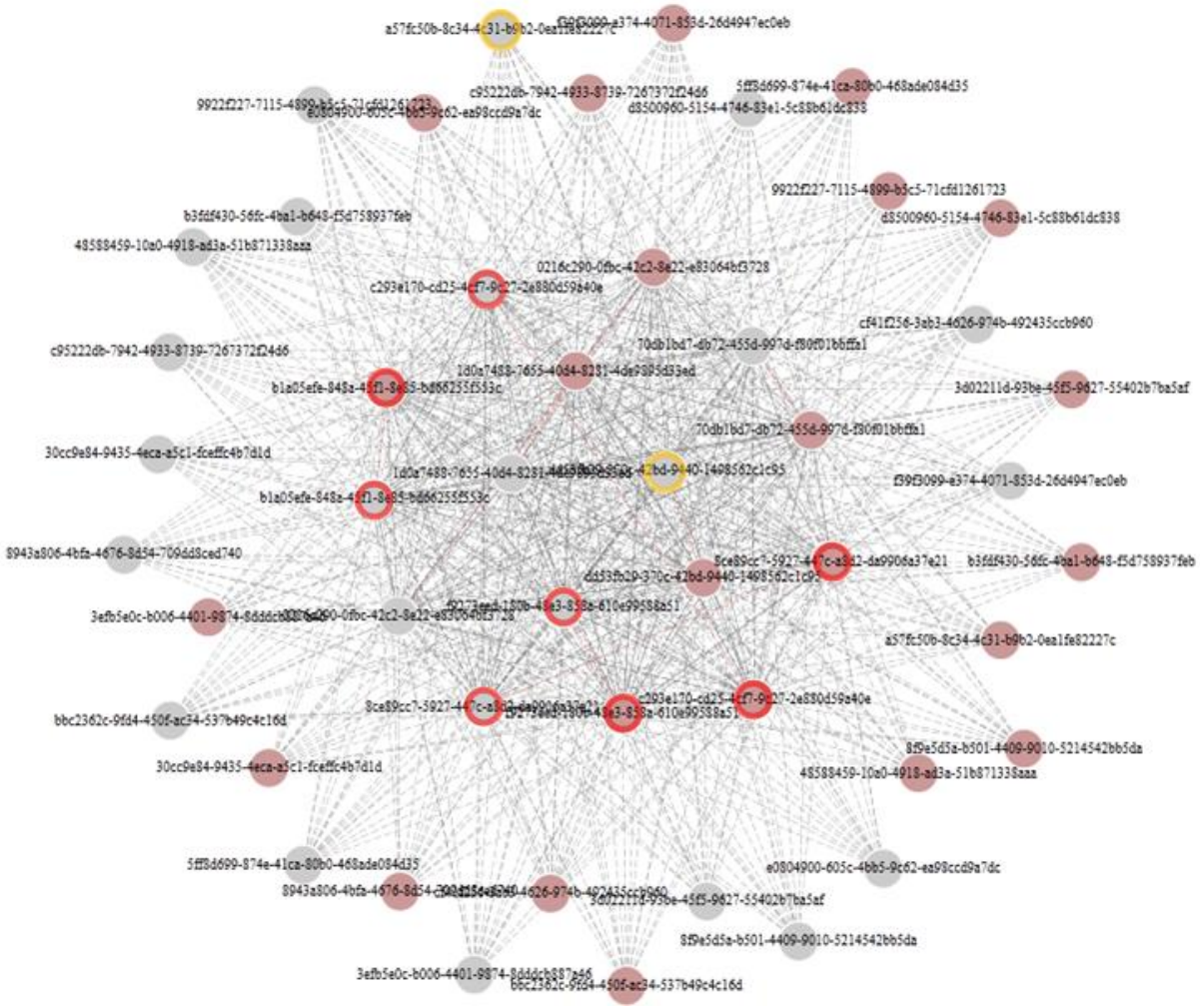

Figure 5 - Original dataset vulnerability exploitation graph with Entry Points and Critical Devices

First, we assess the order of magnitude of the attack paths that would be generated by our exhaustive generation algorithm based on the vulnerability exploitation graph presented in Figure 5.

From a theoretical point of view, this graph actually is formed of 16 nodes fully meshed (at the centre of the graph).

If we consider the sources (i.e. Entry Points) and the target (i.e. Critical Devices) are among these nodes we are looking at a combination of **14! = 87,178,291,200** possibilities for the longest paths only. If the source is outside of this group of 16 fully meshed nodes and still connected to all of the 16 nodes (i.e. in one direction only, so the source node is not fully meshed with the others), we are looking at 15 times more. This is just a number of all possible combinations for a connection.

Additionally, we have to take into account the fact that each connection has 2 possible vulnerabilities. In term of attack paths, it means that the numbers should be multiplied by a factor 2^14 for the first case, and 2^15 for the second case. We note that the mathematical exhaustive calculation is not adapted to full meshed or even strongly connected network.

We then proposed and implemented a **first optimization** of the **Attack Graph Generation algorithm**:

- We took into account the fact that an attacker with a "root" access on a machine has at least all the exploitation possibilities of an attacker with a simple "user" access.
- In our vulnerability exploitation graph (i.e. connection graph of Figure 5), if an attacker could go from the machine/device A to the machine/device B gaining a "root" privilege on the machine/device B, we created a possible connection between the node of the graph representing the machine/device A (i.e. denoted node A in the rest of the paper) and the node representing the machine/device B with "root" privilege (i.e. denoted node B/root in the rest of the paper) but also a possible connection between the node A and the node representing the machine/device B with "user" privilege (i.e. denoted node B/user in the rest of the paper). This is correct from a mathematical exhaustive point of view, but useless from our security/risk perspective as all the possibilities of an attacker with "user" privilege are included in the possibilities of an attacker with "root" privilege. Then, all the paths that go through a node B/user would be also expressed with a node B/root in another path.
- The optimization we did is that when an attacker can go from the node A to the node B gaining a "root" privilege on the node B, we create a possible connection between the node A and the node B/root only.

3.1.6.1 Analysis of Attack Graph Generation after first optimization

We conduct an experimentation for the first optimisation we proposed on the original dataset. The vulnerability exploitation graph with Entry Point and Critical Devices generated by the new visualization feature of the AGG-TRQ component is represented in Figure 6.

On this dataset, the optimization works perfectly as we have only vulnerabilities that grant "root" access in the original Vulnerability Inventory file. Hence, the first optimization lowers the possible combinatory to only the nodes of the graph representing the machine/devices of the Simulation Environment with "root" privilege.

This reduces the vulnerability exploitation graph to a full mesh of 8 machines. And, based on the Mission Graph used for the experimentation, we have 4 targets among them. We also have 2 sources outside of the group of 8 fully meshed nodes, but those 2 nodes reach each of the 8 nodes of the fully meshed group. For a couple of [source; target] in this condition, the number of connection paths in the graph that should be found for a given length "n" is then:

(1) $$Number_{Communication\ path} = \frac{(number\ of\ node\ in\ full\ mesh-1)!}{[(number\ of\ node\ in\ full\ mesh-1)-n+1]!}$$

Which means that the total number of paths generated should be:

(2) $$Total\ number_{Communication\ path} = \sum_{n=1}^{N} \left[\frac{7!}{(7-n+1)!}\right]$$

The total number of attack path with the original dataset should then be:

(3) $$Total\ number_{Communication\ path} = 1 + 7 + 42 + 210 + 840 + 2520 + 5040 + 5040 = 13{,}700$$

As we have 2 sources and 4 targets, we are looking at a number of connection paths (we will call "clustered path" in the rest of the paper) of (2 x 4) = 8 times this number, which is exactly: 109,600 connection paths (i.e. clustered paths).

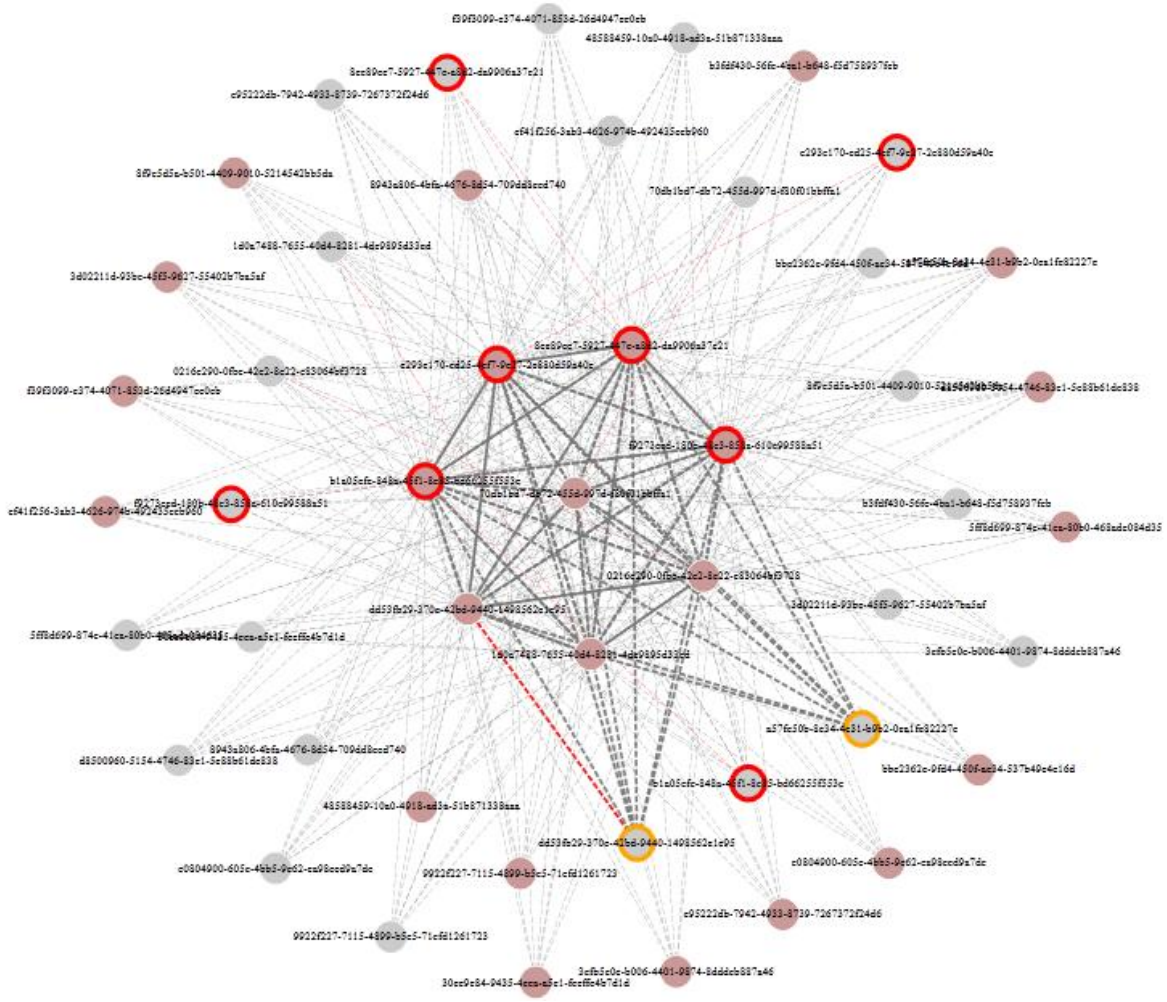

Figure 6- Original dataset vulnerability exploitation graph with Entry Points and Critical Devices using the first optimization

Experimental results

Here after is a trace of the AGG-TRQ component showing the result after the computation of the Attack Paths:

```
time:11.412s  longest path:8 clusteredpathes:109600 PathNumbner:17017968  PathNumbnerForCorrelators:13665024 nb of path filtered:0
 paths(1 hop)=8 paths(2 hops)=56 paths(3 hops)=336 paths(4 hops)=1680 paths(5 hops)=6720 paths(6 hops)=20160 paths(7 hops)=40320 paths(8 hops)=40320 paths(9 hops)=0
```

We find **109,600 clustered paths**. Taking the combinatory of vulnerability, we reach a number of **17,017,968 Attack Paths**. The multiplication factor is around 155, which bring an average of 128=2^7 possibilities, which is consistent with a longest path of 9 nodes (8 hops) in this case.

3.1.6.2 Analysis of Attack Graph Generation after second optimization

In the modified dataset, the two vulnerabilities specified in the Vulnerability Inventory only enable to gain "user" privilege on the devices/machines on which they are present in the Simulation Environment.

Using this modified dataset, the AGG-TRQ produced the vulnerability exploitation graph, including the Entry Points (i.e. source nodes) and Critical Devices (i.e. target nodes), is presented in the figure below.

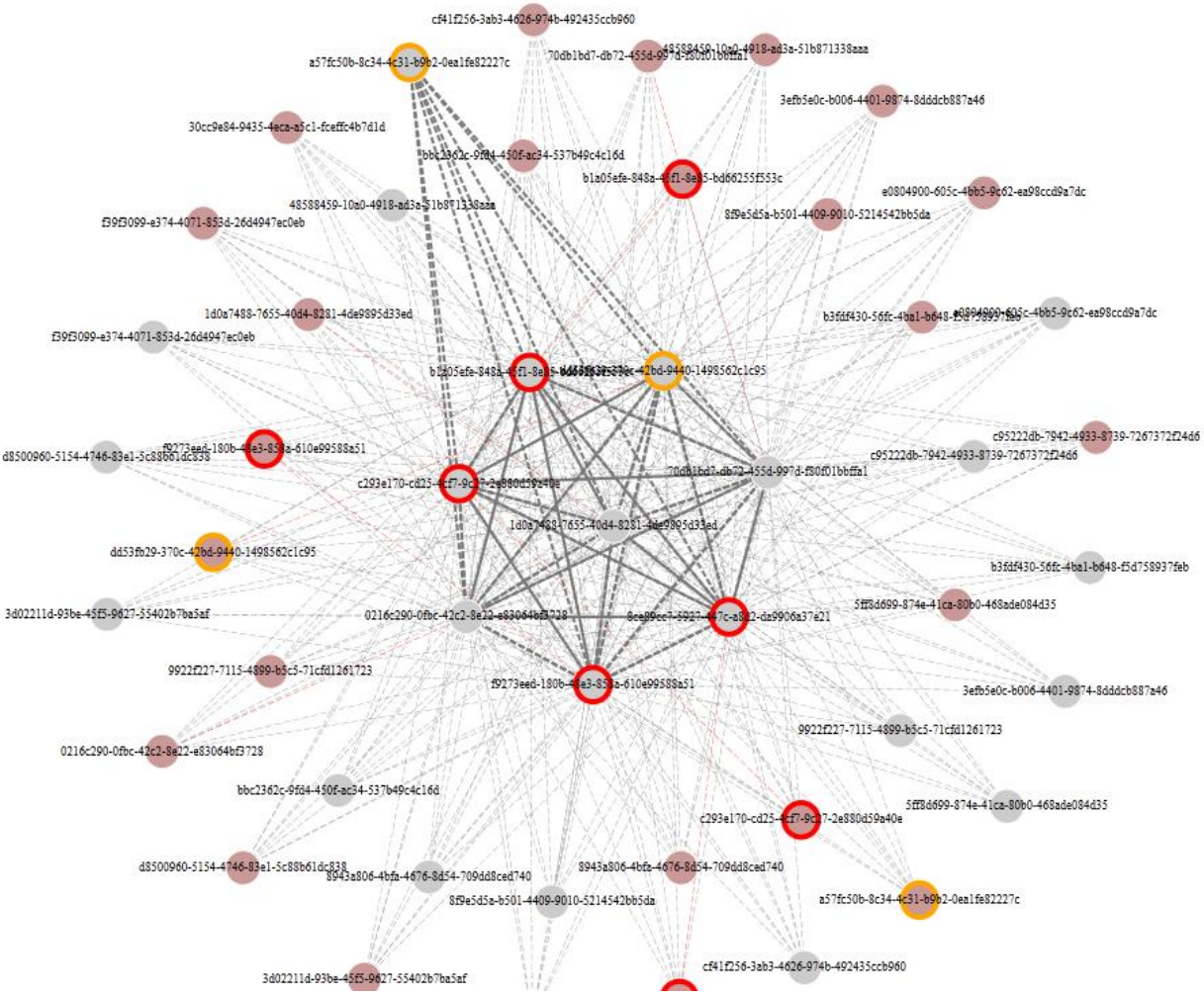

Figure 7- Modified dataset vulnerability exploitation graph with Entry Points and Critical Devices

The new vulnerability exploitation graph is simpler than the original dataset. Nevertheless, the core of the graph is still composed of a group of fully meshed nodes. We can also remark that a source node (i.e. a disk circled in orange) is linked in the graph with another source node.

Experimental results

Here after is the AGG-TRQ component output log showing the results after the computation of the Attack Paths on the modified dataset:

```
time:6.344s  longest path:8 clusteredpathes:62628 PathNumbner:9116768  PathNumbnerForCorrelators:7302912 nb of path filtered:0
 paths(1 hop)=8 paths(2 hops)=52 paths(3 hops)=288 paths(4 hops)=1320 paths(5 hops)=4800 paths(6 hops)=12960 paths(7 hops)=23040 paths(8 hops)=20160 paths(9 hops)=0
```

With the modified dataset, the AGG-TRQ implementing the first optimization finds 62,628 clustered paths. Taking the combinatory of vulnerability, we reach a number of 9,116,768 Attack Paths.

By changing the privilege consequences of the two vulnerabilities in the Vulnerability Inventory, we" then notice that less Attack Paths are generated. Actually, this is due to the behavior of the AGG-TRQ generation algorithm which implements an heuristic that considers that a node device/machine which is compromised with "root" privilege has the possibility to access the other devices/machines to which it is linked with a maximum connectivity even if only few protocols or TCP/IP ports are available in the Reachability Matrix. This heuristic is classically considered in the Attack Graph Generation literature. But, whereas it enables to find more possible Attack Path, it usually over evaluate the number of Attack Paths. It is usually a perfectly valid hypothesis when considering a pessimistic exposure situation when adopting a Risk Management approach.

When analyzing the produced Attack Paths for the modified dataset, we notice that there are some paths which start with edges that hop from one source to another source. While perfectly valid and appropriate on a graph theoretical point of view, those edges are useless in an ICT security and Risk Management perspective as we consider the source as being already compromised in Attack Paths. We have them implemented a **second optimization** that gets rid of all the paths starting with edges that hops from a source to another source.

With the second optimization implemented in the AGG-TRQ, the component generates what we call now aggregated clustered path and aggregated Attack Paths.

Experimental results with second optimization

Here after is the output log showing the results after the computation of the Attack Paths on the modified dataset using the AGG-TRQ component implementing the second optimization:

```
time:1.884s  longest path:8 clusteredpathes:23484 PathNumbner:2431136  PathNumbnerForCorrelators:1881600 nb of path filtered:0
 paths(1 hop)=8 paths(2 hops)=52 paths(3 hops)=264 paths(4 hops)=1080 paths(5 hops)=3360 paths(6 hops)=7200 paths(7 hops)=8640 paths(8 hops)=2880 paths(9 hops)=0
```

With the modified dataset, the AGG-TRQ implementing the second optimization (i.e. additionally to the first optimization) finds 23,484 aggregated clustered paths. Taking the combinatory of vulnerability, we reach a number of 2,431,136 aggregated Attack Paths.

The second optimization reduces the number of produced Attack Paths with a noticeable factor (2.66 for the clustered paths, and 3.75 for the Attack Paths), which is a satisfactory performance in term of number of paths to consider.

Similarly to the first optimization, this second optimization impairs the exhaustiveness of the Attack Paths list generated in a graph theoretical perspective. On an ICT security and Risk Management perspective, it does not impair the value of the Attack Paths generated by the AGG-TRQ component (i.e. we don't miss any dangerous attack scenarios).

3.1.7 Integration in ACEA Emulated Environment

The AGG-TRQ was successfully integrated in the ACEA Emulated Environment. For the proactive plan, the AGG-TRQ was able to process the scan (MissionGraph, Reachability, Scored Vulnerabilities and Vulnerability Inventory) of the Emulation Environment and generate the corresponding attack graphs.

On the reactive counterpart, the AGG-TRQ was able to process the Instantiated Attack Paths, and generate accordingly the instantiated attack graphs. We noted each time the attacker progresses and a new IAP is sent to the AGG-TRQ, the size of the instantiated attack graphs decreases. This is

due to the fact that with each IAP, the attacker is getting closer to the supporting assets, and therefore reducing the number of potential attack paths.

3.2 Response Operational Impact Assessment (ROIA)

A response operational impact assessment (ROIA) is used to assess potentials of collateral damage onto a company and their associated business processes. It is a subcomponent of the mission impact module (MIM). The MIM is part of work package 4 (WP4). In order to perform an assessment it requires information from a shallow network dependency analyzer (SNDA), the network inventory processor (NIP), and, for some methods, from the vulnerability inventory processor (VIP) and vulnerability database processor (VDP). All of these components are part of work package 4, but the ROIA component is associated with WP5 for legacy reasons.

In this section we briefly describe testing, experimentation, and integration strategies and results. For any further details, please refer to the corresponding documents of WP4, namely, [D4.3.1] and [D4.3.2].

3.2.1 Testing and experimentation strategy description

In order to verify the functionality of the ROIA and in order to assure the validation of obtained results, we use a three step approach.

1) Code inspection tests, identifying crucial regions of code providing and fulfilling specialized requirements. This is used to assure that mathematical principles are correctly embedded, which are required by Step 3.
2) Automatic tests: Functional syntactic tests, testing correct syntactic behaviour when given syntactically correct input data. Functional behaviour tests, testing the intended behaviour and reaction to specific kinds of input data. All of these tests are automated by using Junit test and a direct API to [Redmine] , i.e., the central Panoptesec project verification and requirements' management and documentation tool. Every run and result is automatically reported to [Redmine] in a test execution of an associated Junit test.
3) Semantic tests, testing the usefulness and correctness of obtained results by the MIM and ROIA component. The MIM and ROIA are based on a probabilistic graphical model, which provides local semantics that allow validating individual parameters. Based on probabilistic inference we assure that obtained results are validated as well. We describe this approach deeply in [D4.3.2], in a journal artide (currently under review) given in Appendix E of [D4.3.2].

3.2.2 Individual component V&V

Code inspection tests regarding the ROIA component have been conducted by an independent partner of WP4 not in charge of development of the MIM, ROIA or SNDA component. These tests are documented in [Redmine] under the following test case IDs: *TC.WP5.ROI.R1.1, TC.WP5.ROI.R2.1, TC.WP5.ROI.R4.1, TC.WP5.ROI.R5.1, and TC.WP5.ROI.R7.1*.

All test cases were executed successfully, and show that the ROIA assessment was implemented as intended from a code presence perspective.

Automatic tests have been conducted on artificially generated datasets, beyond the complexity of the ACEA use case. Artificial data was used to precisely test individual behaviour. These tests include TC.WP5.ROI.R3.1, TC.WP5.ROI.R1.R2.2, TC.WP5.ROI.R5.2, and TC.WP5.ROI.R1.R2.3.

All test cases were executed successfully, and show that the MIM component behaves as expected for ROIA.

Semantic tests have been conducted in TC.WP5.ROI.R4.2 testing the accuracy of an employed approximation algorithm and TC.WP5.ROI.R6.1 testing the linear scalability of a ROIA. See further Section 3.2.5.

3.2.3 Sub-system integration V&V

The sub-system of ROIA is part of WP4, as explained in the introduction and is therefore explained in deliverables [D4.3.1] and [D4.3.2]. For reference, we outline results in Section 3.2.6.

3.2.4 Uncovered Specialized Requirements

As evident from [D4.3.1], from [D5.4.1], and from Section 3.2.2all specialized requirements of the MIM and ROIA components are fully covered.

3.2.5 Additional experimentations

Various scalability, performance and accuracy tests have been executed on the ROIA evaluation, MIM, and SNDA components. All of them are deeply described in Appendix E of [D4.3.2].

In particular, we show that an employed approximation algorithm is verified against exact inference and provides an expected convergence depending on specific parameters. The latter has also been documented in TC.WP5.ROI.R4.2. Moreover, we show the linear scalability of a ROIA, i.e., that the computation time required for one ROIA evaluation scales at most linearly with each parameter in various experiments. For these experiments we use artificially generated data to produce test sets way beyond the complexity of the ACEA environment. In particular, we show the scalability beyond a network consisting of 400 000 dependencies, i.e., dependencies between individual resources, which is the most deciding parameter of complexity in our approach.

3.2.6 Integration in ACEA Emulated Environment

The deep integration of NIP, SNDA, MIM, ROIA and the PM is described in [D4.3.2], Section 5. In particular, all five components are integrated and running inside the ACEA environment, from which we extracted two results: a mission dependency model (MDM) and a resource dependency model. A mission dependency model was created and validated by business experts to the ACEA Distribucione division from a production perspective. The process of this is described in [D4.3.2]. As the integration is running on emulation data, and the MDM is based on “production” data, this shows that both are complimentary to each other. Furthermore, we automatically learn and generate a resource dependency model (RDM) identifying individual dependencies between devices. For example, a RDM identifies dependencies of a web-server on a database-server. A generated RDM was presented to an external IT consultant to ACEA and validated to represent the infrastructure of ACEA Distribucione.

A visualized demonstration of MDM and RDM is given in the following Figure 8. In this figure, ACEA is represented in dark green, critical devices are highlighted in green, while business functions are in blue and business processes in orange.

Figure 8 - Visualized dependency and mission dependency models extracted from running Integration Framework inside ACEA Distribucione

Moreover, we demonstrate and evaluate the merits of multiple assessments inside the PANOPTESEC framework. Response plans are proposed by the strategic and tactic response decider (cf. following sections). In particular we demonstrate and evaluate the symbiosis of SRD and ROIA in a paper accepted at ARES 2015 entitled "Selection of Mitigation Actions Based on Financial and Operational Impact Assessments" (cf. Section 4.1). In this paper, given as an Appendix F in [D4.3.2], we demonstrate how response plans are selected based on an unweighted best compromise from multi-dimensional assessments.

4 DYNAMIC RISK MANAGEMENT RESPONSE SYSTEM PROACTIVE COMPONENTS

4.1 Strategic Response Decider (SRD)

The Strategic Response Decider (SRD) consists of a proactive management software component that maps detrimental events and attack graph evidences (reported by the AGG-TRQ component) to potential attack scenarios and proactive conditions that were previously defined as strategic policies by the security officers of the organization. The design of the SRD interfaces and sub-components has evolved during the implementation phases of the project. Figure 9 depicts the current status of the high-level view of the global SRD component design. It encompasses the necessary interfaces and sub-components to address the requirements defined for the Strategic Response Decision (cf. Section 4.1.3, SRD component) and Security Policy Instantiation functions (cf. Section 4.1.3, SPI sub-component) of the global Functional Architecture of the DRMRS (i.e. Dynamic Risk Management Response System) sub-system, as defined in Sections 3.3.1 and 3.3.2 of [D5.1.1]. It also covers the corresponding established Specialized Requirements defined in Sections 4.5.4 and 4.6.1 of [D5.1.1] for the calculation of response financial impact assessment (cf. Section 4.1.3, RFIA sub-component).

Figure 9 - Strategic Response Decider Inputs and Outputs

The goal of the SRD component is to anticipate the occurrence of potential attacks. It conducts an initial evaluation of the reported proactive evidences based on a quantitative metric, hereinafter referred to as RORI (Return On Response Investment). The SRD component evaluates and selects

mitigation actions from a pool of candidates, by ranking them in terms of RORI values. The higher the RORI value associated to a mitigation action, or to a combination of mitigation actions, the higher the associated ranking. The purpose of this process is to preselect sets of combined mitigation actions that are identified as optimal from a financial perspective and propose them to reduce the risk of threats against the monitored system. Preselected sets are sent to the Response Operational Impact Assessor (ROIA) and to the Visualization Environment, prior their eventual deployment over the monitored system.

4.1.1 Contributions

The process undertaken by the Strategic Response Decider (SRD) extends initial work reported in the following publication:

> **G. Gonzalez-Granadillo, M. Belhaouane, H. Debar, G. Jacob. RORI-based countermeasure selection using the OrBAC formalism, International Journal of Information Security, Springer, 13(1):63-79, February, 2014.**

The approach proposes the combination of authorization models and quantitative metrics, for the selection of mitigation actions. The actions, modeled in terms of contextual rules, are prioritized based on a cost-sensitive metric that extends the return on investment (ROI) concept. The goal is finding an appropriate balance between the financial damages associated to a given threat, and the benefits of applying some mitigation actions to handle the threat, with respect to the loss reduction. The RORI metric addresses such a goal. It is calculated for each mitigation action, according to following expression:

$$\frac{(ALE \times RM) - ARC}{ARC + AIV} \times 100$$

where ALE (Annual Loss Expectancy) refers to the financial cost expected from the threat, in the absence of applying mitigation; RM (Risk Mitigation) estimates the effectiveness and coverage of an action in mitigating the threat; ARC (Annual Response Cost) expresses the expected cost of applying the mitigation action; and AIV (Annual Infrastructure Value) is a fixed cost associated to the system infrastructure (e.g., cost of equipment, services, etc.), regardless of applying or not mitigation.

With regard to the previous publication, a first improvement has been to enhance the Risk Mitigation (RM) function of the RORI expression. The work, reported in the following two publications:

> **G. Gonzalez-Granadillo, J. Garcia-Alfaro, E. Alvarez, M. El-Barbori, H. Debar. Selecting optimal countermeasures for attacks against critical systems using the Attack Volume model and the RORI index. Computers and Electrical Engineering, Elsevier, 47(2015):13-34, October 2015.**
>
> **G. Gonzalez-Granadillo, J. Garcia-Alfaro, H. Debar. A Polytope-based approach to measure the impact of events against critical infrastructures. Journal of Computer and System Sciences, Elsevier, March 2016.**

extends the concept of attack surface used in previous versions of the RORI metric. It identifies authorization and contextual dimensions that may directly contribute to the exposition of system vulnerabilities. New properties associated to the vulnerabilities, such as temporal conditions (e.g., granted privileges only during working hours), spatial conditions (e.g., granted privileges when connected within the company premises), and historical conditions (e.g., granted privileges only if

previous instances of the same equivalent events were already conducted) can now be included and combined with the RORI cost-sensitive metric.

The adaptation of the selection process, based on financial and operational assessment functions, has been presented in the following publication:

> **G. Gonzalez-Granadillo, A. Motzek, J. Garcia-Alfaro, H. Debar. Selection of Mitigation Actions Based on Financial and Operational Impact Assessments. 11th International Conference on Availability, Reliability and Security (ARES 2016), Salzburg, Austria, August 2016.**

which reports the combination of both assessment approaches, over a representative set of mitigation actions. The combination, based on a multi-dimensional minimization proposal, proposes the choice of semi-optimal responses that, on the one hand, bear the highest financial attractiveness on return on investment; and, on the other hand, bear the lowest probability of conflicting with the organization's missions. This is seen as beneficial for its application in the scenarios of the project, where highly critical missions and resources must be protected, without sacrificing missions in favour of security.

4.1.2 Testing and experimentation strategy description

The testing and experimentation strategy consists on demonstrating the accomplishment of the different functional and non-functional requirements that have been defined in the project. This leads to the individual and integration V&V activities. More specifically, we focus on defining a set of tests that are conducted to verify that each requirement is covered by the SRD implementation. These experimentations cover two main aspects: code inspection and test execution for functional and non-functional requirements.

Strategic Response Decider (SRD): The functional requirements addressed by the SRD are the following: The SRD evaluates the list of mitigation actions per potential attack, and obtains the corresponding RORI value (Requirement SRD.R1), The SRD requests the RORI evaluation for every mitigation action with the inputs provided by the impact assessment (Requirement SRD.R2), The SRD determines a threshold to be used as a reference point to select candidates to be combined (Requirement SRD.R3), The SRD evaluates combined mitigation actions that can mitigate a potential attack, with respect to their RORI index (Requirement SRD.R4), The SRD requests approval from the security administrator before deploying the mitigation actions (Requirement SRD.R5).

The non-functional requirements addressed by the SRD are the following: The SRD evaluates combinations of mitigation actions within minutes (Requirement SRD.R6), Communication with other components is encrypted and authenticated (Requirement SRD.R7), and The SRD ensures the security of resources associated to the evaluation process in terms of storage and computation (Requirement SRD.R8).

Response Financial Impact Assessor (RFIA): The functional requirements addressed by the RFIA are the following: RFIA provides financial loss considering both the consequences incurred by attacks as well as those provided by mitigation actions (Requirement RFI.R1), The RFIA provides the annual loss expectancy (ALE), the Annual Response Cost (ARC), the Annual Infrastructure Value (AIV), and the Risk Mitigation (RM), as required for the RORI computation (Requirements RFI.R2, RFI.R3, RFI.R4, and RFI.R5 respectively).

The non-functional requirements addressed by the RFIA are the following: The RFIA determines combinations of mitigation actions within minutes (Requirement RFI.R6), Communication with other

components is encrypted and authenticated (Requirement RFI.R7), The RFIA ensures the security of resources associated to the evaluation process in terms of storage and computation (Requirement RFI.R8).

Security Policy Instantiation (SPI): The functional requirements addressed by the SPI are the following: The DRMRS uses contextual security rules defined in terms of policy violations (Requirement SPI.R1), The SPI uses contextual policy representations for the activation of rules (Requirement SPI.R2), The SPI uses the most up-to-date state of the policy contexts (Requirement SPI.R3), The SPI requests the state of policy context updates with a constant frequency (Requirement SPI.R4), The SPI receives an up-to-date list of policy contexts at any time (Requirement SPI.R5), Contextual policy representation uses proactive evidences from the attack graphs constructed via the PANOPTESEC system (Requirement SPI.R7).

The non-functional requirements addressed by the SPI are the following: Communication with other components is encrypted and authenticated (Requirement SPI.R8), and The SPI ensures the security of resources associated to the evaluation process in terms of storage and computation (Requirement SPI.R9).

In addition, as part of the experimentation strategy, we estimated the performance of the module in terms of computation speed, while performing the combination of mitigation actions. As a result, we successfully cover all design, functional and non-functional requirements for the RFIA, SRD, and SPI components.

4.1.3 Individual component V&V

The testing strategy consists in demonstrating we cover the defined functional and non functional requirements for the RFIA, SRD, and SPI components. The functional requirement tests can be organized in three different parts: the component design is compliant with the requirements, the component is able to generate successfully the expected outcome, and the RORI evaluation is compliant with the specifications. Table 4, Table 5 and Table 6 summarize the tests and main results for the SRD component, and the RFIA and SPI sub-components respectively.

SRD Test Cases and Executions

Test cases and their associated executions, evaluate the component capabilities to analyse a list of mitigation actions per potential threat, obtain the corresponding RORI value and generate the response plans containing the evaluated mitigation actions. The evaluations are made for individual and combined mitigations actions, based on a list of mitigation actions given as input or using thresholds values (e.g., the average of the RORI index for all the evaluated mitigation actions). Web Service interfaces are used to exchange information between the component and the Integration Framework, e.g., in order to request approval prior deployment of mitigation actions. Table 4shows the requirements for the SRD component.

Table 4 - SRD Test Cases

Case	Description of the tests	Results of the test
SRD.R1	The SRD evaluates the list of mitigation actions per potential attack, and obtains the corresponding RORI value	The output obtained is a Response Plan in a Json format. The response plan is said to be valid since it is compliant to the latest Response Plan schema given in the enriched response plan schema and contains the RORI index for the evaluated

		mitigation action(s).
SRD.R2	The SRD requests the RORI evaluation for every mitigation action with the inputs provided by the impact assessment	The test produces a valid set of Response Plans (conformed to the Response Plan JSON schema) for each individual mitigation action assigned to a given threat. Each response plan provides information of the RORI index associated to the mitigation action.
SRD.R3	The SRD determines a threshold to be used as a reference point to select candidates to be combined	The test generates a set of response plans for the combined evaluation of the mitigation actions for which the individual RORI index is greater than a predefined threshold. They are all valid response plans since they are compliant to the latest response plan schema.
SRD.R4	The SRD evaluates combined mitigation actions that can mitigate a potential attack, with respect to their RORI index	The SRD component communicates with the RFIA component in order to execute a RORI evaluation for a given threat on an organization (monitored system) requesting to combine part or all the mitigations actions assigned to such a threat. The process produces a response plan (or a set of them) with the RORI index value of the evaluated mitigation action(s).
SRD.R5	The SRD sends an enriched response plan in order to request approval from the security administrator before deploying the mitigation actions	The Enriched Response Plan is correctly received by the server in the Integration Framework (server response is OK). Additionally, the log registered in the Integration Framework shows the successful communication between the SRD and the Integration Framework.
SRD.R6	The SRD evaluates combinations of mitigation actions within minutes	The output for all tests execution is produced within minutes. For instance, A combination of 12 candidates with some restrictions generates a total number of 796 of candidates in a total elapsed time of: 0:00:00.507 (less than 1 second). For more information, please refer to Figure 9.
SRD.R7	Communication with other components is encrypted and authenticated	The SRD module communicates with other PANOPTESEC components and requests/collects the information required by the SPI and RFIA modules. The test case is accepted for this requirement by design inspection. In the Pull mode, for instance, the SRD makes a successful connection to the Integration framework and retrieves information about the target system. Additionally, the data is securely transferred (encrypted) between the components over a cryptographic protocol. Additionally, in the Push mode, the integration framework connects to the

		SRD via the SFTP server using a user and password (authenticated). Additionally, the data is securely transferred (encrypted) between the components over a cryptographic protocol.
SRD.R8	The SRD ensures the security of resources associated to the evaluation process in terms of storage and computation	The SRD component is executed on a Linux based system, which allows setting security policies over the objects composing the module thanks to Linux's DAC mechanism. Additionally, the authentication/authorization mechanisms provided by Linux permits only to authorized users the execution of the RFIA module.

RFIA Test Cases and Executions

Test cases and their associated executions, evaluate the component capabilities in order to compute the RORI value associated to mitigation actions. They test the requirements shown in Table 5.

Table 5 - RFIA Test Cases

Case	Description of the tests	Results of the test
RFI.R1	The RFIA provides financial loss considering both the consequences incurred by attacks as well as those provided by mitigation actions	The RORI evaluation given by the execution of this test is approximated to the results given in tables II and III of a published scientific article. The results are not equal to the ones in the paper since the RORI indexes given in it are rounded up to the nearest whole number.
RFI.R2	The RFIA provides the annual loss expectancy (ALE) value as required for the RORI computation	The output obtained doesn't contain any errors or exceptions. From this execution it can be seen that an organization is successfully created, and a set of threats is created. Each Threat is assigned to the organization by computing its ALE value.
RFI.R3	The RFIA provides the Annual Response Cost (ARC) as required for the RORI computation	A set of Mitigation Actions is created and assigned to a Threat. For each Mitigation Action an ARC value is computed and assigned.
RFI.R4	The RFIA provides the Annual Infrastructure Value (AIV) as required for the RORI computation	A set of PEPs is created. Each PEP is assigned to the organization by computing its AEV value. The sum of all AEVs results into the AIV parameter.
RFI.R5	The RFIA provides the Risk Mitigation (RM) as required for the RORI computation	A set of Mitigation Actions is created and assigned to a Threat. For each Mitigation Action an RM value is computed and assigned.
RFI.R6	The RFIA determines combinations of mitigation actions within minutes	The output for all tests execution is produced within minutes. For instance, A combination of 12 candidates with some restrictions generates a total number of 796 of candidates in a total elapsed time of: 0:00:00.507 (less than 1 second). For more information, please refer to Figure 1.
RFI.R7	Communication with other components is encrypted and authenticated	The RFIA module communicates only with the SRD module; such communication is locally done via command lines and scripts execution. The SRD

		module is the one that communicates with other PANOPTESEC components and requests/collects the information required by the RFIA module. Hence, this requirement is fulfilled by the test cases and test executions done for requirement SRD.R7
RFI.R8	The RFIA ensures the security of resources associated to the evaluation process in terms of storage and computation	The RFIA module is executed on a Linux based system, which allows setting security policies over the objects composing the module thanks to Linux's DAC mechanism. Additionally, the authentication/authorization mechanisms provided by Linux permits only to authorized users the execution of the RFIA module.

SPI Test Cases and Executions

Test cases and their associated executions for the SPI sub-component are presented in Table 6.

Table 6 - SPI Test Cases

Case	Description of the tests	Results of the test
SPI.R1	The system uses contextual security rules defined in terms of policy violations	The SPI component uses information gathered from the target system that allows creating contextual policy rules, instantiating security policies and activating security rules using proactive evidences extracted from that information.
SPI.R2	The SPI uses contextual policy representations for the activation of rules	The SPI retrieves the most up-to-date information from the target system and uses that information to create contextual policies and instantiate security policies to further active security rules.
SPI.R3	The SPI uses the most up-to-date state of the policy contexts	By design, the Integration framework provides the most up-to-date information of the system to the SRD and SPI components. This is done using the Pull/Push transfers modes to retrieve/gather information from the target system. The test cases show how the SPI request the state of policy context updates in PULL mode and how the PUSH mode is used to receive from the PANOPTESEC system the state of policy context at any time.
SPI.R4	The SPI requests the state of policy context updates with a constant frequency (i.e. pull mode).	The SPI component, via the SRD component, retrieves information from the target system (e.g., valid Network Inventory file, valid Proactive Risk Profile, valid Reachability Matrix, etc.), in a Pull mode using a set of Web Service client interfaces.
SPI.R5	The SPI receives an up-to-date list of policy contexts at any time (i.e. push mode).	The Integration Framework, using the SFTP server, makes a successful connection and pushes correctly the new generated information required by the SPI component, e.g., a new Network Inventory, a new Proactive Risk Profile, a new Reachability Matrix, etc.

SPI.R7	Contextual policy representation uses proactive evidences from the attack graphs constructed via the PANOPTESEC system	The SPI component creates contextual policies and instantiates security policies using proactive evidences extracted from the Attack Graph.
SPI.R8	Communication with other components is encrypted and authenticated	The SPI module communicates only with the SRD module, such communication is locally done via command lines and scripts execution. The SRD module is the only one that communicates with other PANOPTESEC components and requests/collects the information required by the SPI module.
SPI.R9	The SPI ensures the security of resources associated to the evaluation process in terms of storage and computation	The SPI module is executed on a Linux based system, which allows setting security policies over the objects composing the module thanks to Linux's DAC mechanism. Additionally, the authentication/authorization mechanisms provided by Linux permits only to authorized users the execution of the RFIA module.

4.1.4 Sub-system integration V&V

As it was already introduced in the previous section, the integration and communication requirements between subcomponents were tested and successful results were obtained. More specifically, the test cases state that the different SRD inner sub-components are executed on a Linux based system and communicate among them using Command Line Interfaces, scripts and Linux system calls.

Moreover, due to the fact that the SRD module is a standalone component developed in Python and running under a GNU/Linux Virtual Machine, the integration and communication with other system modules is done via a proxy (deployed in the Integration Framework module) that handles the communications with the other Java developed PANOPTESEC components. As depicted in Figure 10, the SRD components communicate with the proxy in order to request/received the needed data types that will further be analysed to produce the expected proactive results.

Successful integration results were obtained from the test where the integration and interaction between the component and the Integration Framework was tested. In those test cases is stated that:

- The component is able to receive information from the PANOPTESEC system in a Push mode at any time. In this mode the Integration Framework (via the SRD Proxy) transfers the most up-to-date information of the PANOPTESEC system to the component using a SFTP server. The component is also able to receive the following data types: Network Inventory, Proactive Risk Profile, Reachability Matrix, Authorized Mitigation Action, Abstract Default Security Policy, Abstract Response Policy Context, TRD Enriched Response Plan, SRD Selected Response Plan and Attack Graph.
- The component retrieves information from the PANOPTESEC system in a Pull mode using a set of Web Services client interfaces. Those interfaces request to the Integration Framework (via the SRD Proxy) the most up-to-date or previous information of the PANOPTESEC system. The module is able to request the following data types Network Inventory, Proactive Risk

Profile, Reachability Matrix, Authorized Mitigation Action List, Abstract Default Security Policy, Abstract Response Policy Context and Attack Graph.

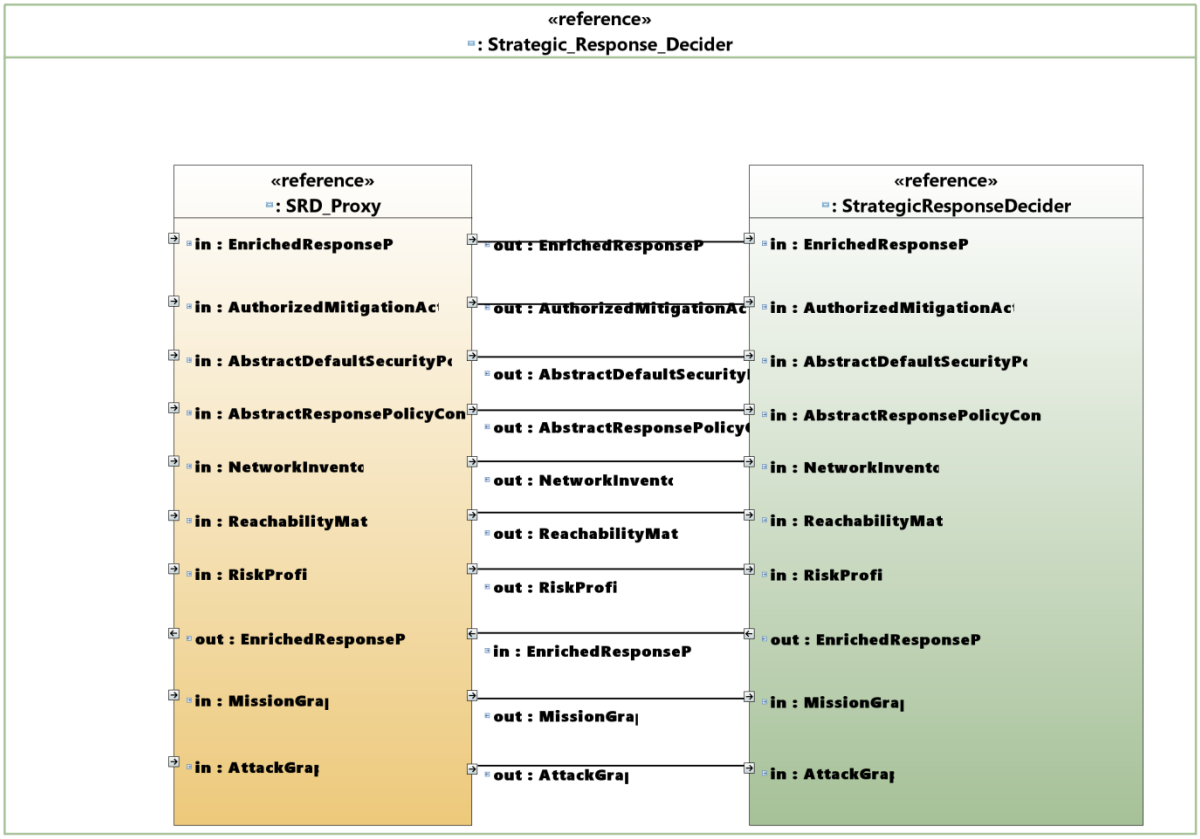

Figure 10 - Proxy High Level Design

4.1.5 Uncovered Specialized Requirements

All specialized functional and non-functional requirements of the SRD component have been covered.

4.1.6 Additional experimentations

Several test cases have been executed in order to evaluate the computation speed in the combined evaluation of mitigation actions. The number of combination for a set of non-restrictive candidates is given by the expression $X= (2^N) – (N+1)$, therefore, in case N=4, the number of combinations will be equivalent to 11, in case N=12, the number of combinations is equivalent to 4083. Since the total number of combinations grows exponentially, we measured the time at which the system is able to perform the evaluation of multiple candidates. Results are plotted in a curve as shown in Figure 11.

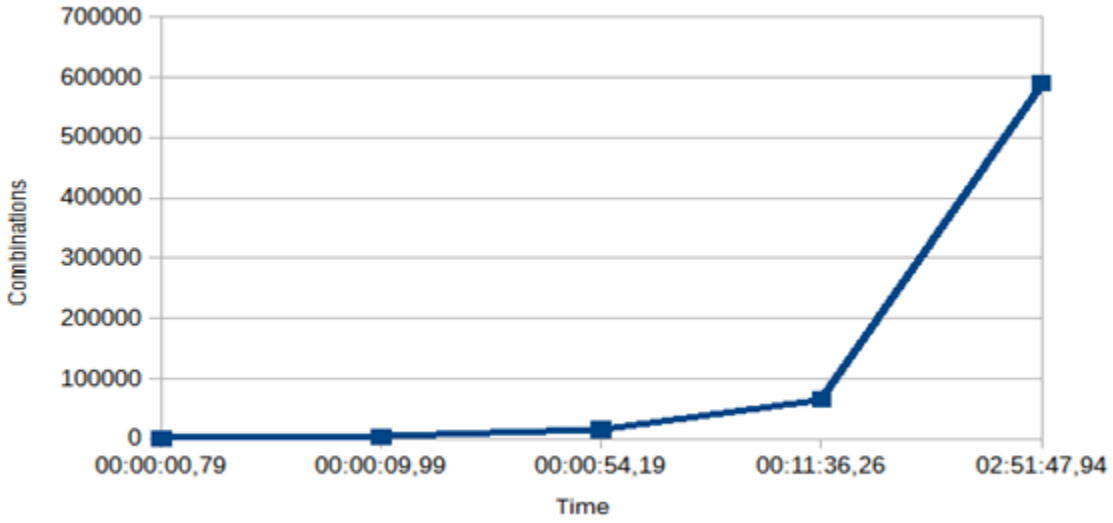

Figure 11 - Computation Speed in the Combined Evaluation of Mitigation Actions

As depicted in Figure 11, a combination of 12 restrictive mitigation actions results into 796 combinations. Results are obtained in less than one second. For 12 non-restrictive mitigation actions, there are a total of 4,082 combinations, which are performed in less than 10 seconds. Similarly, for 16 restrictive mitigation actions, there are a total number of 15,534 combinations performed in less than one minute, whereas for 16 non-restrictive mitigation actions, there are a total of 65,518 combinations, which are performed in a total elapsed time of 00:11:36,26 (less than 12 minutes). Going up to 24 restrictive mitigation actions, results into 590,464 combinations. Such number of combinations is executed in almost three hours. As a result, in order to keep the evaluation process within a reasonable time (less than one minute), the system must process up to 14 non-restrictive mitigation actions (16,369 combinations). Beyond this threshold, the elapsed the performance of the evaluation is highly degraded.

In addition, we have tested the interaction with the Response Operational Impact Assessor (ROIA) module in order to select the best response plan in financial and operational terms. As such, we reduce the number of response plans to only one that best satisfies the RFIA and the ROIA modules. The method searches for the best semi-optimal response plan with the lowest operational impact assessment and the highest RORI index.

4.1.7 Integration in ACEA Emulated Environment

We sketch in this section a sample evaluation conducted by the Strategic Response Decider (SRD) in the ACEA Emulated Environment. The evaluation starts by processing information about the threat scenarios defined by the *strategic abstract policies* and the information of the *detrimental events* reported in the *Proactive Risk Profile*. The SRD compares the set of conditions, such as minimum likelihood or impact of a detrimental event, defined in the policies. For each detrimental event matching the conditions, the list of attack paths associated to the detrimental event is processed. The list of attack paths is used to extract from the attack graph the list of vulnerable system nodes that can lead to the exploitation of vulnerabilities in the business devices identified by each threat scenario.

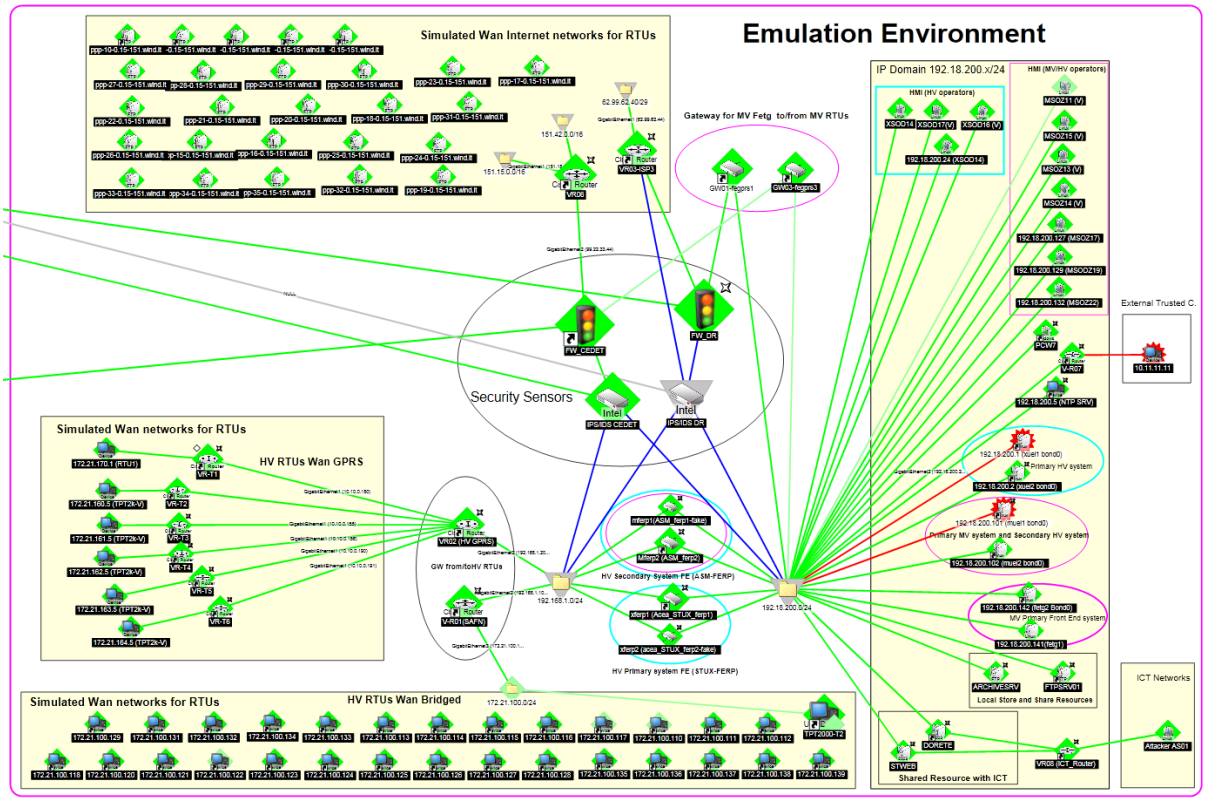

Figure 12 - SCADA Emulation Environment

Let us start by presenting one of the scenarios tested using the ACEA Emulated Environment. The environment consists of a distributed network of Remote Terminal Units (RTU) in energy stations of medium voltage (MV = 20,000 Volts) and high voltage (HV = 150,000 Volts), that acquire data from electrical equipments (e.g., PLC, sensors, etc), and send them to the Supervisor Terminal Unit (STU) of the headquarters. The system uses Supervisory Control and Data Acquisition (SCADA) protocols.

In the absence of security compromise, operators review the security status of the Monitored System (SCADA and ICT environment). Security status indicators may note the presence of one or more system vulnerabilities due to known software security flaws as posted by publicly available vulnerability advisory services. Attack paths from hypothetical attack sources to known mission critical systems are analyzed and the impact on critical business functions (e.g., energy distribution) is assessed resulting in a quantified risk assessment.

The threat to analyze is a Denial of Service against a high voltage node of the C&C infrastructure with the objective of taking the C\&C offline. The threat will cause an out of service condition on Front End Servers (e.g., FE-X1) which breaks communication path from SCADA Servers to RTUs. There exists an attack vector via ICT Network (via VR-08) targeting first the file server (i.e., File-SRV), second, the archive server (i.e., Archive-SRV), and third, the high voltage Front End devices (i.e., FE-X1, FE-X2), as depicted in Figure 12.

We assume now that a detrimental event that fulfills the matching conditions of the SRD policies reports as target business devices, nodes *ftpSrv01*, *archiveSrv*, and *dorete*. By using the list of attack paths associated to the detrimental event, nodes *languard*, *xferp1* and *stweb* are identified as vulnerable nodes that can potentially affect the mission of the business devices.

The threat has a severity defined as 'grave', which corresponds to a single loss expectancy SLE = 10,000,000 €, and a likelihood defined as `medium', equivalent to an annual rate of occurrence ARO=2. The Annual Loss Expectancy is therefore equivalent to ALE = 20,000,000 €/year.

The list of nodes extracted in the previous step are mapped to the abstracts entities defined in the policies (e.g., Policy Enforcement Point types), and mapped to the list of authorized mitigation actions. From the proactive chain, the following three types of mitigation actions can be applied:

- Patching, refers to a piece of software designated to update a computer program or its supporting data, to fix or improve it. This includes fixing or removing security vulnerabilities and other bugs with such patches and improving the usability or performance.
- Reboot, refers to the process of restarting a device or a computer program.
- Shutdown, refers to completely remove any possibility to access a device by powering off a device.

Table 7shows some samples of the RORI computations based on our case scenario.

Table 7 - RORI Evaluation Results for Individual Mitigation Actions

MA	MA Type	PEP Type	RM	ARC	Restrictions	RORI
MA_1	Shutdown	WEBSCADA	0.0141	15.00	MA_2, MA_3	4.07
MA_2	Reboot	WEBSCADA	0.0009	15.00	MA_1	0.26
MA_3	Patching	WEBSCADA	0.0937	25.00	MA_1	27.09
MA_4	Shutdown	FEXSCADA	0.0750	25.00	MA_5	21.66
MA_5	Reboot	FEXSCADA	0.0050	200.00	MA_4	1.44
MA_6	Shutdown	RTUSCADA	0.0234	200.00	MA_7	6.76
MA_7	Reboot	RTUSCADA	0.0016	15.00	MA_6	0.46
MA_8	Shutdown	FTPSRV	0.0281	15.00	MA_9, MA_{10}	8.11
MA_9	Reboot	FTPSRV	0.0019	15.00	MA_8	0.55
MA_{10}	**Patching**	**FTPSRV**	**0.1875**	**25.00**	**MA_8**	**54.15**

As shown in Table 7the mitigation action with the highest RORI index is **MA_{10}**, which requires installing a patch for the PEP Type 'FTPSRV'. More specifically, the node 'dorete' requires a patching against two vulnerabilities (i.e., CVE-2008-4250, and CVE-2006-3439).

Considering the previous information about mitigation actions, a total of 214 combinations have been performed to evaluate the RORI metric. Table 8 presents the top 5 combination results.

Table 8 - RORI Evaluation Results for Combined Mitigation Actions

Mitigation Actions	RM	ARC	RORI
$MA_2, MA_3, MA_4, MA_6, MA_9, MA_{10}$	**0.3085**	**295.00**	**89.07**
$MA_3, MA_4, MA_6, MA_9, MA_{10}$	0.3080	280.00	88.94
$MA_2, MA_3, MA_4, MA_6, MA_{10}$	0.3075	280.00	88.80

$MA_3, MA_4, MA_6, MA_{10}$	0.3071	265.00	88.67
$MA_2, MA_3, MA_4, MA_7, MA_9, MA_{10}$	0.2975	295.00	85.91

From the previous example, the highest RORI evaluations would suggest that, from a financial perspective, the optimal response is to combine mitigation actions $MA_2, MA_3, MA_4, MA_6, MA_9, MA_{10}$, which proposes the following concrete actions:

- Install patches to the node 'STWEB' against CVE-2008-4250, and CVE-2006-3439;
- Reboot node 'STWEB';
- Shutdown the node mferp2;
- Shutdown the node 'TPT2000-T2';
- Reboot nodes 'ARCHIVESRV' and 'FTPSRV01';
- Install patches to the node 'dorete' against CVE-2008-4250, and CVE-2006-3439.

For each evaluated mitigation action (including all possible combinations), a response plan has been generated. Each response plan contains the identification of the mitigation action(s), the PEP responsible for its enforcement, and the associated RORI index. The Response Plans contain mitigation actions applied only to the nodes obtained in the Attack Graph parsing (e.g., STWEB, ARCHIVESRV, FTPSRV01, dorete, etc). For the previous scenario, a total of 224 response plans were generated. An example of a response plan is given in Listing 1.

Listing 1 - Individual Response Plan for AS01HV

```
{
 "mitigationActions":[
  {"properties": {
    "enforcementPoint": "c9fa4086-d979-4794-9b6e-cd0478040856",
    "mitigationActionID": "d23088a7a3eecd860f855c60d7282036"
    },
   "type": "MitigationActionShutdown"
  }],
   "monitored_System_Ident": "AceaSim_Env",
   "responsePlanID": "SRD_RP_2016-06-13_15-05-29-209167",
   "rori_Index": 4.07,
   "scope": "proactive",
   "snapshot_Ident": "05/20/2016 12:26:42"
}
```

Listing 1 shows that a response plan has been generated for a mitigation action of type `shutdown' with a RORI index of 4.07. In case of multiple actions, each of them will provide identification and the associated properties. For instance, if the mitigation action type is patching, the response plan must indicate the vulnerability that is intended to be removed and the PEP where the mitigation action should be implemented.

5 DYNAMIC RISK MANAGEMENT RESPONSE SYSTEM REACTIVE COMPONENTS

5.1 Automata Based High-level Online Correlation (AB-HOC)

In the context of the PANOPTESEC project, two different approaches have been proposed to design the correlation engine. In both cases, the goal is to analyze the flow of Low Level Alerts (generated by the LLC component) to detect, within this flow, patterns that are corresponding to the known attack scenarios (inferred by the AGG component). This section focuses on the Automata based approach while the next one will consider the Query based approach.

In the following figure, the inputs and outputs of the Automata-Based Engine (ABE for short) are represented. The Enriched Attack Graph contains the description of all the Attack Paths which have been computed by the Attack Graph Generator. Each Attack Path describes some possible attack scenarios. The analysis of the incoming flow of LLCAlerts leads to generate two kinds of outcomes. An Unmatched Alert is an element of the input flow that seems to be useless: it does not allow to progress in the recognition of any attack scenario. An Instantiated Attack Path corresponds to a partial or complete recognition of a particular attack scenario.

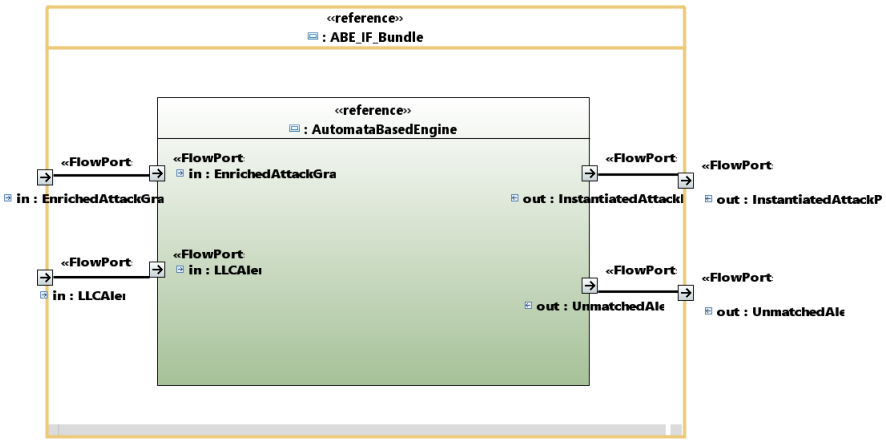

Figure 13 - Automata-Based Engine Inputs and Outputs

The Automata Based approach relies on the recognition of multi-step attack scenarios by Automata. This means that:

The ABE approach is able to transform each Enriched Attack Path (EAP) into an automaton whose role is to recognize any (partial or complete) occurrence of the depicted multi-step attack scenario;

The set of automata handled by the ABE-HOC can be updated dynamically, by modifying/inserting new EAPs. When such a dynamic update occurs (i.e., when a new Enriched Attack Graph is provided by the AGG module), new automata are created. After their initializations, instead of starting immediately the analysis of the new LLC Alerts, some previously received LLC Alerts are replayed. More precisely a time window (called history) is managed to keep the most recent LLC Alerts. They're used to initialize the state of the new generated automata in a more accurate configuration that takes into account the occurrences of these past LLC Alerts: The LLC Alerts that have been taken into account recently are analyzed again.

When an LLCAlert is processed, ABE determines if such an alert can help to progress in the recognition of some attack scenarios. For each EAP, the states already reached by the automaton are

considered: the goal is to check if the new LLCAlert allows to transit from one of these previous states into a new state corresponding to a more advanced stage in the execution of the attack scenario. In other words, the algorithm checks if a new LLCAlert matches with one of the next expected steps of the scenario. In that sense, the recognition process is purely sequential: the matching between an alert and a step of the scenario is considered if and only if all the previous steps have been already recognized: the recognition process identifies the prefixes of attack paths followed by the attackers.

Regarding the matching test, two cases can be considered: (i) an LLCAlert can be an approximation of what is expected or (ii) it can be exactly what is expected. In the ABE approach, an approximated matching is allowed only in two particular cases: when the algorithm tries to see if the new LLCAlert can match with the first or with the last step of a scenario. In the first case (respectively the second case), we consider that the source IP (respectively the target IP) field is acceptable whatever its values. This leads to the notion of an approximate attack path (called respectively new entry point and new target).

1. In the case of an exact matching, the three fields of an LLCAlert have to be equal to the expected triple <*source_IP*, *CVE_ID*, *target_IP*> associated to a transition of the automaton. This choice is fully consistent with the definition of an attack step in an attack path: an adversary who has already attacked the mentioned source node exploits the vulnerability identified by the CVE on the mentioned target node.
2. The recognition supports missing alerts (that can be due to false negatives at the IDS level) by defining an algorithm that can tolerate up to *s* missing alerts for a scenario of length *l* (*s* depends on *l and is bound by a threshold denoted k:* $0 \leq s \leq k < l$).
3. The ABE module is able to generate Instantiated Attack Paths (IAP) of three kinds: *approximate IAP* (approximate attack path or missing alert), *exact IAP* (exact attack path recognition), and *on-going IAP* that emphasize the evolution of the attacks against the system (partial recognitions).

ABE is optimized to support a large number of alerts per second. However, both for performance and for memory consumption reasons, it cannot keep in memory the whole history of all alerts that have occurred since the beginning of the process. This implies the definition of a memory cleaning processes called *purges* that allow:

1. To remove old LLC Alerts from the history of received LLC Alerts;
2. To discard old partial recognitions which are probably no more of interest.

This process is essential, as ABE has not an unlimited memory available. Moreover, it has an heavy impact on performances: (1) if the number of old events in history is lower, the replay of history is quicker when you insert new EAP; (2) when a new LLCAlert occurs, finding all the partial recognitions that can be extended with this alert is quicker if the number of partial recognitions in memory is lower.

5.1.1 Contribution

The scientific contributions of the ABE component are twofold, compared to the state of the art in the field of explicit alert correlation:

- The ability to detect approximate attack paths, i.e. attack paths that are not explicitly defined, but are variants of the existing known attack paths;

- The ability to detect attack paths with missing alerts. That means that our component is able to tolerate faults into IDSes it relies on, in particular it can tolerate false negatives (i.e., the absence of alerts when an attack occurs). This is an important contribution of our component.

Compared to a preliminary prototype (i.e., the one described in the paper « A language driven intrusion detection system for event and alert correlation » by E. Totel, B. Vivinis and L. Mé, IFIP Sec conference, 2004), the challenge is now to handle a huge number of attack paths. While the prototype was only able to cope with a few complex correlation rules, the ABE component is able to manage thousand of attack paths generated automatically while exhibiting good performances. The language used to describe the attack paths is still quite rich (each attack path can be expressed by a combination of elementary attack steps and operators AND, OR, SEQ): current SIEM provide very simple alert correlation languages and consider usually only a few short sequences. To obtain good performances, several mechanisms have been used to control the use of the memory and to adapt some functions of the component to the amount of alerts to treat.

In parallel of the Panoptesec project, tools have been developed to visualize the internal behavior of the correlation engine (for education purposes). As the ABE component is sufficiently mature now, we expect to include it into real production environments.

A paper dedicated to the description and the evaluation of the ABE component is currently under preparation.

5.1.2 Testing and experimentation strategy description

The testing and experimentation strategy consists in two different parts.

The first part aim at demonstrating we cover the different requirements that have been defined in the project. This leads to both individual and integration V&V activities. In this part we focus on defining for each requirement a set of tests that are conducted to assess that the requirement is covered by the component implementation. These experimentations presented in a functional logical manner consist in verifying design requirements and functional requirements:

- The Design requirements are the following ones: the HOC module is independent from the underlying IDS, IPS and probes (Requirement 6); the format of alerts and the data coming from the attack graph generator are providing the same type of information (Requirement 4); the communications with this component are asynchronous (Requirement 5);
- The Functional and Non-Functional requirements we addressed are the following ones: an AG is transformed into a correlation rule usable by HOC (requirement 8); the module must be able to adapt to changes in attack paths (Requirement 7); the HOC module should trigger unmatched alerts (Requirement 3); the HOC must be able to identify attack paths currently active and provide a dynamic ranking of ongoing attack paths (Requirements 1 and 2); the HOC module may recognize some approximate attack paths (Requirement 9); the HOC module should recognize approximate attack paths with missing alerts (Requirement 12); the HOC module should adapt to switch to a new attack graph (Requirement 14); the module should react to performance degradation (Requirement 13); the HOC module should automatically be able to scale in terms of incoming low level alerts rate (Requirement 15); the analysis of alert must be performed in (near) real-time (GEN Requirement 2).

The second part of the experimentation strategy consists in estimating the performances of the correlation engine, both in terms of computation speed and in terms of detection capabilities. The

last experimentations consist in estimating if the mechanisms (especially the purge mechanism) implemented have an impact on the false positive or false negative rate.

We can note here that all design, functional and non-functional requirements for the ABE component have been covered by tests.

5.1.3 Individual component V&V

The first part of the testing strategy consists in demonstrating we cover the defined functional and non-functional requirements for the ABE component. The functional requirement tests can be organized in three different parts: the component design is compliant with the requirements, the component is able to generate successfully the recognition rules from the EAG component and the attack recognition is compliant with the specifications.

5.1.3.1 Component Design

The component Design must be compliant with 3 requirements: the inputs must be independent from the data source (Alerts) formats (WP5.HOC.R6), the alerts must provide information compatible with the EAG formats (WP5.HOC.R4), and communications between the components must be asynchronous.

These requirements are fulfilled for the following reasons. The alerts are normalized in a format that is described in the Figure 14.

This format holds all pieces of information required to decide if an alert is part of a scenario. In particular, the source, destination and exploit required to make an attack are hold by the alert. These pieces of information are normalized whatever the source of the alerts (i.e., whatever the type of the IDS alert taken as input). This normalisation is performed by the Low Level Correlator.

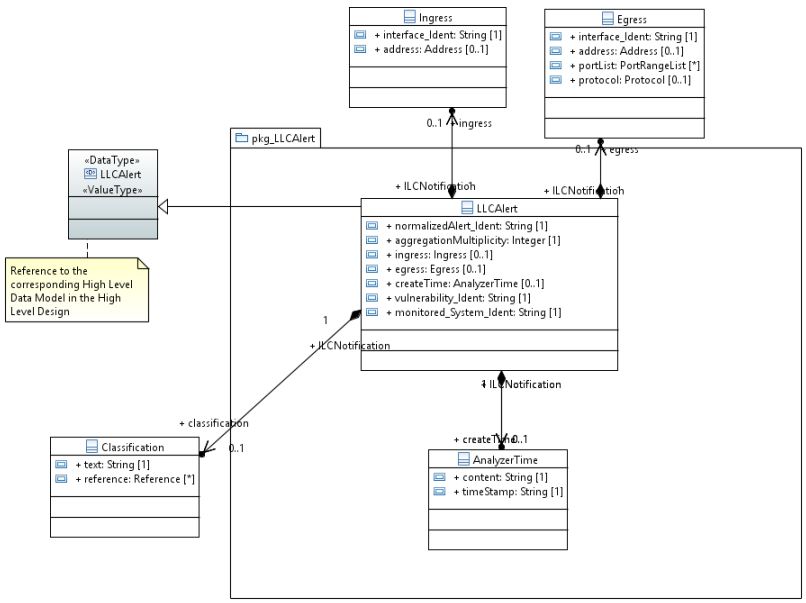

Figure 14 - Format of LLC Alerts

As a matter of fact, this alert holds the same semantic information as the correlation rules (called Attack Paths) that are provided by the Enriched Attack Graph, as seen in the Figure 15.

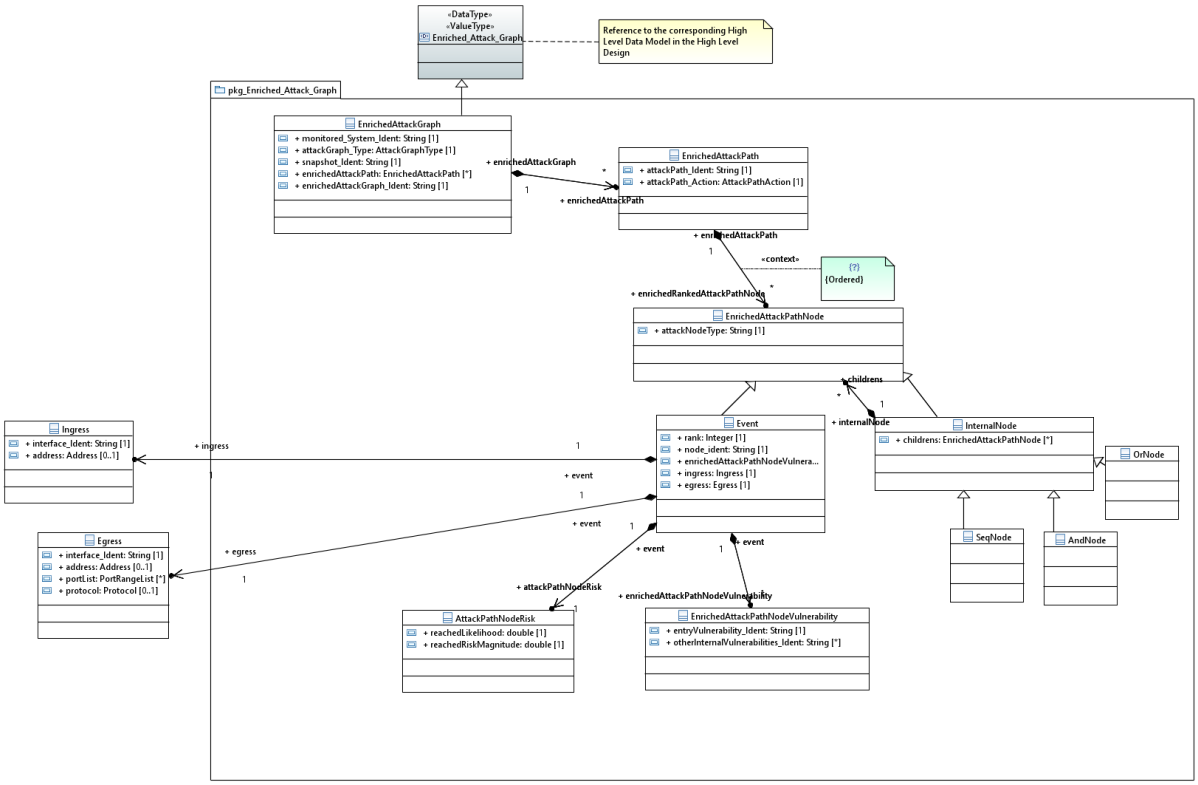

Figure 15 - Format of Enriched Attack Graph

In particular, the nodes in the EAG are holding the source, destination and exploit required to perform the attack. These pieces of information are compatible with those contained in the LCC alerts.

Moreover, by construction, the communications are asynchronous, because, as a communication layer we use an ActiveMQ component that is a publish and subscribe message passing mechanism. By definition this communication component provides asynchronous communications.

5.1.3.2 Rule Generation

The ABE component must translate an EAG into correlation rules that are able to recognize any multi steps attack scenario described in the EAG. To fulfill this goal, we translate the EAG into an automaton, either at initialization stage (WP5.HOC.R8) or dynamically during the component execution (WP5.HOC.R7).

To demonstrate these functionalities, we provided the component with some Enriched Attack Graphs, and displayed the built automata. As a matter of fact, it can be illustrated here on an example that demonstrates the function. The following Figure shows an EAG as input, that contains a single attack path under the form of a correlation tree.

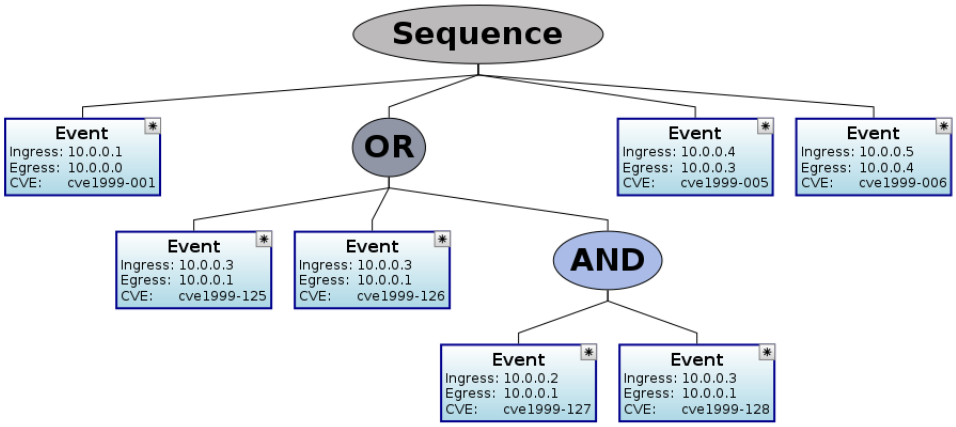

Figure 16 - Attack Path included into the EAG

This attack path is translated into an automaton that is able to recognize the different attack steps. The obtained automaton is displayed in the following Figure.

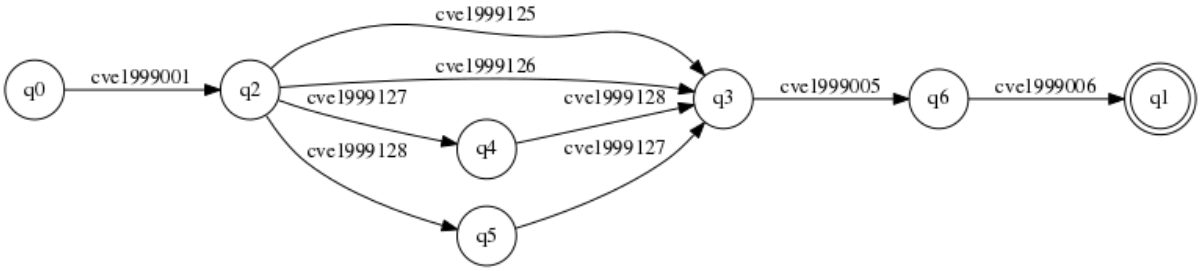

Figure 17 - Automaton corresponding to the Attack Path

This graphical representation of the automaton displays in a simplified way the transitions, as it only shows the exploited vulnerabilities, and not the source and destination node. However, we can see that the automaton corresponds to the correlation tree displayed in the previous Figure.

This demonstrates the capability of the ABE to translate the EAG structure into automata recognizing the attack scenario, at initialization stage.

The tests have been conducted either on synthetic data, covering all the logical tree nodes (SEQ, AND, OR) in order to demonstrate that we cover all functionalities. Moreover, to ensure that the automata are correctly generated in a production environment, the tests have been performed on EAG coming from the emulated environment.

The second part of the rule generation tests targets the dynamic functionalities of the EAG. We conducted several tests to verify that (1) new attack path are correctly dynamically added on demand, and (2) old attack paths are correctly suppressed.

5.1.3.3 Attack Recognition

When an alert is received by the ABE, we can face with five different cases:

1. The alert is an unmatched alert. In this case, we just have to store an *unmatched* alert (WP5.HOC.R3)

2. If an attack recognition evolves, we must produce *ongoing alerts* that notify the administrator that a new step has been reached by the attacker (WP5.HOC.R1 and R2)
3. If the alert is the first alert of a given scenario recognition, we can tolerate that the source attribute of the alert is not the expected one. This permits to detect new attack Entry points. In this case, even if the source is incorrect, we consider the alert contributed to the recognition of the multi-step attack scenario. (WP5.HOC.R9)
4. In the same way, we tolerate that the last alert of a scenario has a wrong destination attribute. We thus detect new attack targets. (WP5.HOC.R9)
5. The recognition algorithm must be able to detect scenarios where some alerts are missing. For example, we can tolerate in a scenario with three attack steps, that one alert is missing. In this case, we inform the administrator that a partial recognition has been performed, and emit what we call a *missing alert*.

All these cases have been covered by several tests, either with synthetic data or data coming from the emulated environment. The following table summarizes the results of the tests that have been conducted.

Table 9 - Results of the tests conducted for the attack recognition section

Case	Description of the tests	Results of the test
1	Several tests have been conducted, either by giving to the ABE unknown alerts, alerts at the wrong place in the recognition, alerts that should match a step recognition	When an alert does not match with a current step recognition, an unmatched alert is emitted. If the alert allows to progress in a scenario recognition, no unmatched alert is emitted.
2	We produced several tests with synthetic data, where a given percentage of the alerts is matching with the scenario steps. Additionally, manual production of synthetic data that triggers the recognition algorithm have been produced.	Each time an alert matches, if sufficient steps have been recognized in the scenario, an ongoing alert is always emitted.
3	For a given scenario, we gave as input an alert with a wrong source attribute.	The alert is accepted as first alert of the scenario, detecting thus a new entry point.
4	For a given scenario, we gave as input an alert with a wrong destination attribute.	The alert is accepted as last alert of the scenario, detecting thus a new target.
5	A sequence of inputs corresponding to a defined scenario, but with a missing alert is given as input of ABE. Moreover, tests have been conducted on data coming from the emulated environment.	A missing alert is emitted at the end of an incomplete sequence. On emulated data, missing alerts are also produced.

5.1.3.4 Non-Functional Requirements

The non-functional requirements target essential the capability of the component to adapt its performances in the context of a highly pressured number of LLC Alerts. This is divided into three main requirements:

1. The module should adapt its time required to switch to a new attack graph according to the current rate of the flow of alerts (WP5.HOC.R14).
2. The module should react to performance degradation (WP5.HOC.R13). This part is tightly linked to the fact that we accumulate partial recognitions in memory. As a consequence the recognition becomes slower, and the memory available for new recognitions can become low. This is solved by introducing a memory purge mechanism.
3. The HOC module should automatically be able to scale in terms of incoming low-level alerts rate (WP5.HOC.R15).

These requirements are handled by different mechanisms. The existence of these mechanisms is in some cases sufficient to comply with the requirements. In other cases, performance tests must be conducted to ensure that the requirement is fulfill.

Requirement WP5.HOC.R14

In the case of WP5.HOC.R14, two mechanisms are implemented. A « buffer » is used to identify the incoming LLCAlerts that have been received but never analyzed till now. An « history » is used to refer to a sliding window of at most x elements that contains the most recent LLCAlerts that have been already analyzed at least once. When a new Attack Graph is taken into account, for all the new paths, the elements of the history can be analyzed again. This replay aims at reducing the possible occurrences of false negatives: the attack recognition process will not ignore the most recent LLCAlerts already analyzed in the past (in the context of a previous Attack Graph). The mechanisms used in the ABE-HOC component rely on two parameters. The first parameter (called herein x) corresponds to the maximal number of LLCAlerts that can be analyzed again: by construction, it is the maximal size of the history. The second parameter (called herein y) is the maximal number of LLCAlerts that can wait in the buffer when a replay mechanism is launched. In other words, as the replay is a time consuming action, it is done only when the current size of the buffer is smaller than the threshold. We consider two different executions. For both executions, the value of x is set to 500. During the first execution, the value of y is set to 500 while it is set to 6000 during the second execution. The sequence of incoming LLCAlerts is composed of 10262 LLCAlerts. The second Attack Graph is provided when half of the sequence has been analyzed. At that time, 5132 LLCAlerts are still waiting in the buffer. Consequently, no replay is activated during the first execution (because 5132 > 500) while the 500 LLCAlerts contained in the historic are executed again during the second execution (because 5132 <= 6000).

Requirements WP5.HOC.R13

The test conducted is the following one: the Enriched Attack Graph (generated by the AGG module on 15/04/2016) contains 6466 Attack Paths. Each time a progress occurs in the recognition of an attack path, a new data structure called a plan is created. Our goal is to focus on executions where a lot of plans are created and many of them are logged. Consequently, as a lot of memory space is necessary, a purge mechanism has to be used. First, we analyze all the Attack Paths to construct a synthetic sequence of LLCAlerts such that each of these alerts has a high probability to allow a progress in the recognition of one (or several) scenario(s) depicted in this Attack Graph. Despite our selection, some alerts that appear in the constructed sequence of LLCAlerts are still corresponding to Unmatched Alerts. They have no real interest but they do not impact our tests. Their analysis has only an impact on the execution time but not on the consumed memory. During our tests, we identify and use 30000 LLCAlerts and less than 10% are corresponding to Unmatched Alerts. Note that a distinction has to be done between the number of created plans (which continuously

increases) and the number of logged plans. The number of logged plans may decrease (when a purge is performed) or may remain stable (a new plan often replaces an old one. More precisely, as all the generated Attack Paths are corresponding to sequences, the choice has been made to keep only the most recent information at each level of the sequence). The purge mechanism (evaluated through this test) aims at controlling the heap consumption. Obviously, for a given graph, there exists an upper bound and during an execution without any purge, the number of logged plans converges progressively towards this upper bound. Consequently the heap consumption will grow. As this heap consumption may be too high, purges have to be performed. A purge has always a positive impact on the used memory space. It may also have a positive impact on the time required to analyse the whole sequence of LLCAlerts. As information related to old partial recognitions of attack scenario are suppressed, a purge mechanism may have a negative impact on the false negative rate. We analyze the behavior of the ABE-HOC component in different configurations, both in terms of memory consumption and created plans (partial recognitions).

The first Figure shows that when a lot of alerts match (90%) with expected attack scenarios, a lot of new plans are created, and thus this implies a heavy consumption of the memory. As a consequence, we triggered a purge mechanism when the heap consumption was too high. This is shown on Figure 18, where two purges were triggered. The purge speeds up the detection time used for each incoming alert.

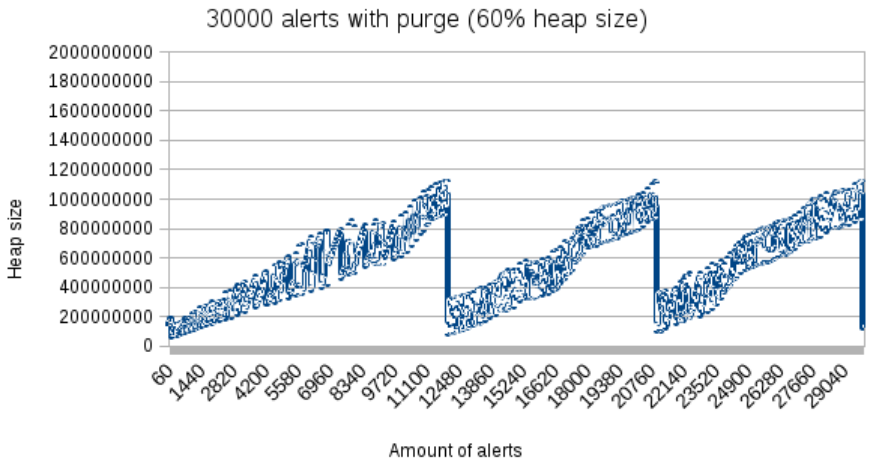

Figure 18 - Heap Consumption with 100% of alerts match (2 purges)

Then, we considered a realistic flow of inputs with at least 70% alerts that were not matching. Figure 19 shows that the heap consumption was stable for a purge with a 60% heap consumption threshold. Thus, no purge has been performed

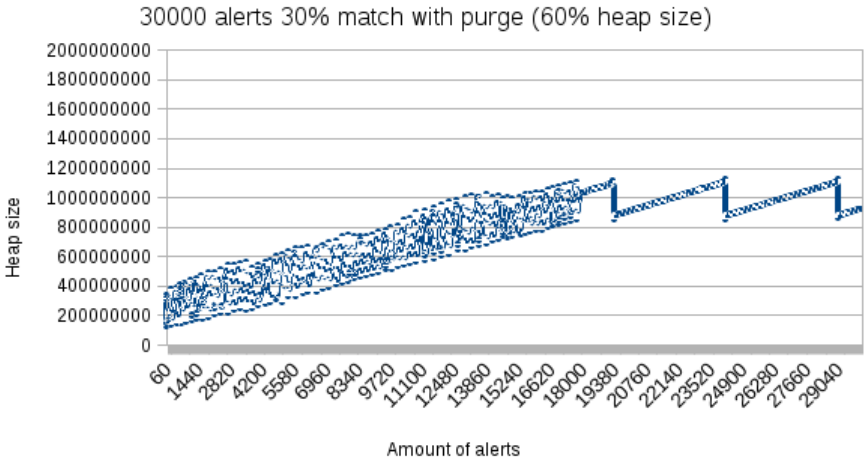

Figure 19 - Heap Consumption with 30% of alerts match (no purge)

Moreover, when no purge is performed, a limit is attained in the number of partial recognitions (plans) (Figure 20).

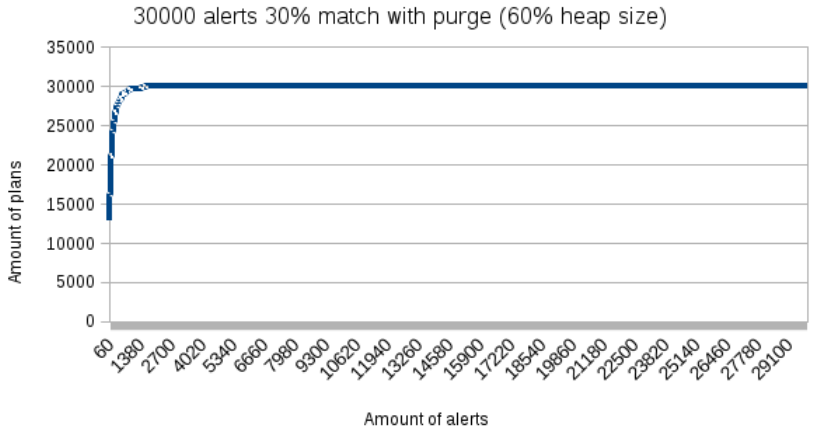

Figure 20 - Number of plans in memory with 30% alerts match (no purge)

The impact on the alert treatment time on this last example is important. If no purge is triggered, the treatment of the 30000 alerts takes one minute. If the threshold is set to 50% heap consumption, one purge is triggered and the treatment takes less than 40s. Thus the mechanism has an heavy impact on the performances.

Requirement WP5.HOC.R15

The Enriched Attack Graph (generated by the AGG module on 04/02/2016) contains 1616 Attack Paths. First, we analyze all the Attack Paths to construct a first synthetic sequence of LLCAlerts such that each of these alerts has a high probability to allow a progress in the recognition of one (or several) scenario(s) depicted in this Attack Graph. This incoming sequence of LLCAlerts (called the "Suitable sequence") contains 10262 elements. Then we construct a second sequence of 10262 LLCAlerts such that all these alerts are corresponding to Unmatched Alerts. This second sequence of LLCAlerts is called the "Unsuitable sequence". The mechanism proposed to reduce the heap consumption (and consequently to reduce the time required to analyze a whole sequence of

LLCAlerts) relies on the purge mechanism depicted in the test related to the requirement R13. The purge mechanism can be launched when the heap exceeds a threshold (requirement R13) or when the size of the buffer that contains the LLCAlerts received but never analyzed exceeds a given threshold (requirement R15). In the latter case, the threshold evolves dynamically. Each time the threshold is crossed, it is dynamically increased: in that case, we add 200 to its current value. To avoid a continuous fluctuation of the threshold, it may also decrease but only when the current size of the buffer becomes at least 400 less than the current threshold: in that case we suppress 200 from its current value. We will consider a particular value for the initial threshold. It is equal to 200: some purges may occur during the executions.

We consider the Suitable sequence of LLCAlerts. To observe only the impact of the purges that are launched due to an increase of the size of the buffer, we inhibit the second possible triggering factor (a very high limit on the heap consumption is defined). Our goal is to analyze the behavior of the ABE-HOC component when the arrival rate is very high. During this test, we consider that we observe a fixed delay denoted d between the arrival dates of two consecutive LLCAlerts and a second delay denoted c (used to trigger the mechanism) between the consumption of two consecutive LLCAlerts by the component. Two particular values of d are considered d= 0ms and d= 1ms and one particular value for c is considered c = 10ms. We analyze the reports obtained during three executions. Each time a new LLCAlert is received and inserted in the buffer, the current size of the buffer is written in the report. As the arrival rate is very high (d= 0ms or d=1ms) and the consumption rate is quite low (c=10ms), the size of the buffer increases. 3 executions have been performed: 1) one with no purge 2) one with some purges and d = 0ms 3) one with some purges and d=1ms. It appears that the total execution time is respectively equal to approximately 1) 14 minutes, 2) 7 minutes, and 3) 5 minutes. This clearly attests 1) the impact of the arrival rate on the performances and 2) the interest of the purge mechanism when the objective is to keep short execution time whatever the circumstances.

5.1.4 Sub-system integration V&V

The ABE component has been successfully integrated with other components of the PANOPTESEC system. This success is due to the conformance of the component with the communication interfaces of the other components it communicates with.

The components in interaction with the ABE-HOC are the following ones:

1. The AGG component provides Enriched Attack Graphs that are used as inputs to generate the scenario recognition automata;
2. The LLC component provides the LLC Alerts that are used to detect multi step scenario attacks;

The ABE component uses a precise format as inputs, these two formats being described in Figure 14 and Figure 15.

The ABE component also provides High Level Alerts called Instantiated Attack Paths (IAP) to other components such as TRQ, Visualization components and the Persistency Manager. The format of the IAP is given in the following Figure 21.

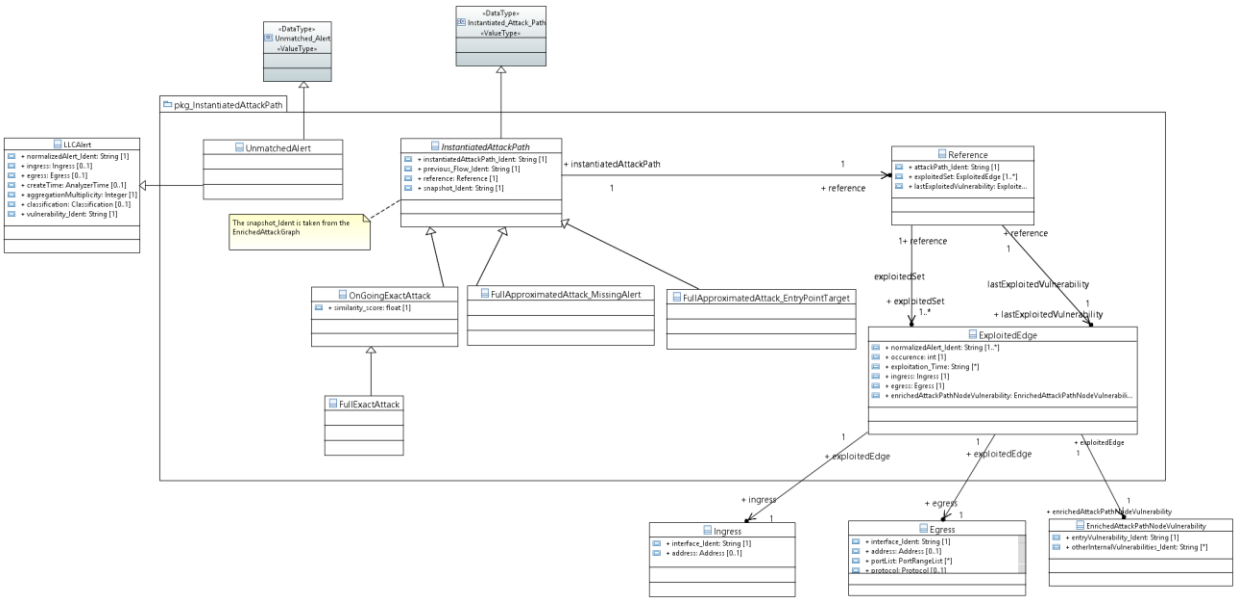

Figure 21 - Instantiated Attack Path Format

The ABE component has been executed for days in the emulated environment, without playing attacks. As a result, despite the presence of LLC Alerts, no alert was emitted by ABE. Consequently, we can expect a very low false positive rate for this component.

5.1.5 Uncovered Specialized Requirements

All specialized functional and non-functional requirements of the ABE component have been covered.

5.1.6 Additional experimentations

The purpose of this section is to discuss about the design choices due to performance reasons on risk to miss the detection of attacks.

5.1.6.1 Impact of the partial EAP enumeration on the false negative rate

The EAG should describe all the possible attack paths on the system. The completeness of this information is a quality parameter: as all the possible paths between two arbitrary nodes in the graph are considered, the path really followed by an attacker is necessarily in the exhaustive list of computed attack paths. However, such an exhaustive enumeration implies a combinatorial explosion of the number of paths. Due to the fact that memory and CPU resources are limited, it may be necessary to discriminate the attack paths to select only a subset of them.

First of all, during the correlation process, each EAP is taken into account independently: in the ABE-HOC component, one automaton is created per attack path and the management of an automaton never interferes with that of another. Consequently, by dividing the set of attack paths into n subsets, it is possible to distribute the correlation process over a farm of n machines that do not have to cooperate while each of them analyzes the same stream of LLCAlerts (with one of the subsets of attack paths). This first approach ensures that we will have no false negative. Let us note that, such approach requires no modification of the proposed high level correlator but only an increase of the computing resources used to run independently several instances of the correlator in parallel. In the context of this project, we do not follow this possible approach and consider the

extreme case where a single instance of the correlator exists. Our goal is to observe the impact caused by a limitation of the resources (CPU and memory) on the proposed solution.

Obviously the needs in terms of resources (CPU and memory) increase when the number of attack paths depicted in a graph increases. To keep reasonable needs, the idea is to select only the EAP that seem to be those that will be most likely chosen by an attacker. The goal is to minimize the number of paths that are used by the correlator. Of course, the fact that some attack paths are ignored may lead to have false negative.

An attacker often tries to reach quickly his target and to minimize the number of alerts raised due to his presence. First, we have to attest that the longest path has (nearly) always a corresponding shorter path that reaches the same target from the same entry point in less steps. If this property is true, the most common paths followed by the attackers are the shortest ones.

Second when the attacker does not behave like this and selects a long path that has been ignored, we have to check if the fact that we consider a shortest path which is included in the more complicated followed attack path (i.e., with more steps) may allow to detect the attack. This may lead us to relax the matching test between an LLCAlert and a step of the attack path: in particular, the matching of the two sources may be ignored.

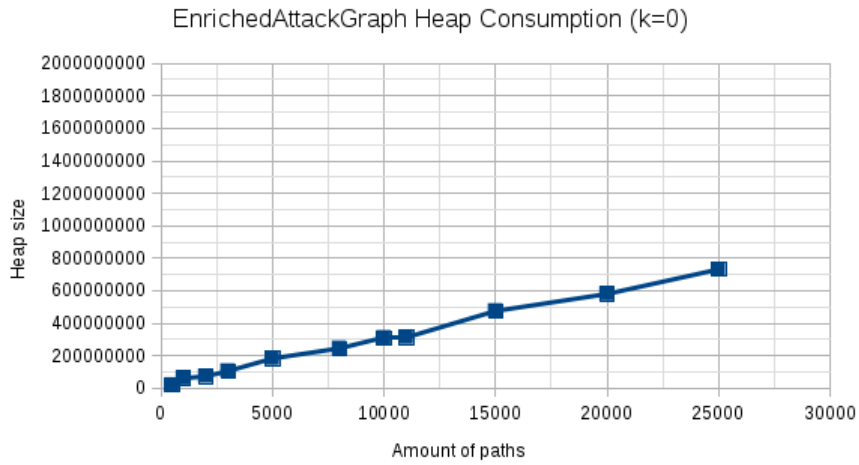

Figure 22 - Amount of memory required to transform EAPs

The number of attack paths that can be managed is limited by the available memory. Figure 22 shows that transforming 25000 Attack Paths into automata requires almost 800MB of memory, which can be considered as a limit. Note that this result is obtained when the mechanism to cope with missing alerts is not activated (k=0). This means that only one detection plan is created for each automaton at initialization. The time required to transform these 25000 Attack Paths remains reasonable (less than 2s).

5.1.6.2 Impact of the history size on the false negative rate

We keep in memory a history of previous LLC Alerts. This is used to replay old alerts when an attack path is added dynamically. However, the size of this history cannot be infinite, and must thus be scaled according to (1) the rate of alerts that are raised per second and (2) the duration we want to keep any alert. If we adopt a too small size for the history, we can increase significantly the risk of false negative (especially if the new attack graph is corresponding to an exhaustive description rather than a differential description). In the case of an exhaustive description, for any attack path

that is both in the previous graph and the new graph, all the partial recognition previously done are lost. The replay mechanism is the only way to counterbalance this bad choice.

As the replay mechanism has a computation cost (and can slow down the analysis), a trade-off has to be found. In the ABE component the following mechanism has been implemented. The word « buffer » is used to identify the incoming LLCAlerts that have been received but never analyzed till now. The word « history » is used to refer to a sliding window of at most x elements that contains the most recent LLCAlerts that have been already analyzed at least once. When a new EAG is taken into account, for all the new paths, the elements of the history can be analyzed again. The mechanism used in the ABE component relies on two parameters. The first parameter (called herein x) corresponds to the maximal number of LLCAlerts that can be analyzed again: by construction, it is the maximal size of the history. The second parameter (called herein y) is the maximal number of LLCAlerts that can wait in the buffer when a replay mechanism is launched. In other words, as the replay is a time consuming action, it is done only when the current size of the buffer is smaller than the threshold y. In some sense, when the recognition process is late, we take more risk to have false negative. In practice to estimate this risk we have to analyse first if the correlator is able to handle a normal flow of alerts without being late. Experiments that have been conducted show that the size of the buffer increases when the inter-arrival rate of the LLCAlerts increases. In the following Figures, we show the size of the buffer during a computation similar to the one already considered: 10262 LLCAlerts arrives. We indicate the size of the buffer after each arrival of a new LLCAlert. In the middle of the computation, an new EAG is installed. In a first experiment, the parameter y is set to the value 0 to avoid a replay. In a second case, an high value is adopted for y: all the LLCAlerts already analyzed are taken into account during a replay. The results show that the impact on the size of the buffer is very limited. During the replay, the new incoming LLCAlerts are logged in the buffer (rather than being consumed immediately). Yet the impact on the size of the buffer is very small. Therefore, the difference between the plot “with replay” and the plot “without replay” is barely observable. Compared to the problem created by the fast arrival speed of the LLCAlerts, this additional increase can be ignored. Thus a high value can be adopted for both the parameter x and the parameter y (they are both greater than 5000 in our examples). As a consequence, the replay mechanism is inhibited rarely (and the risk of false negative is limited).

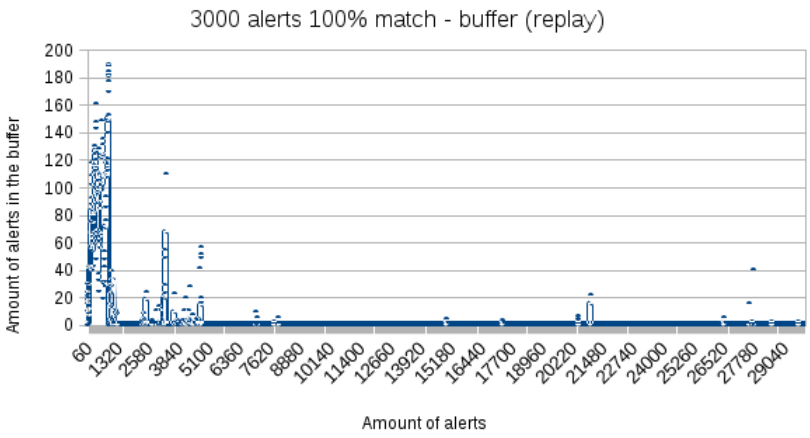

Figure 23 - Buffer size

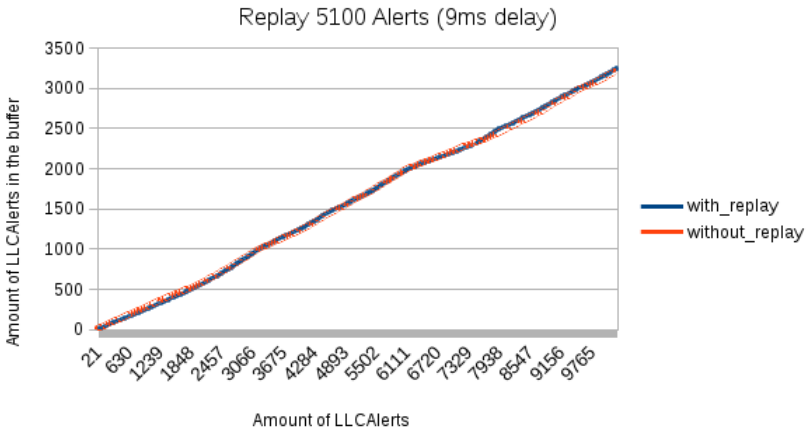

Figure 24 - Buffer size (inter-arrival rate 9 ms)

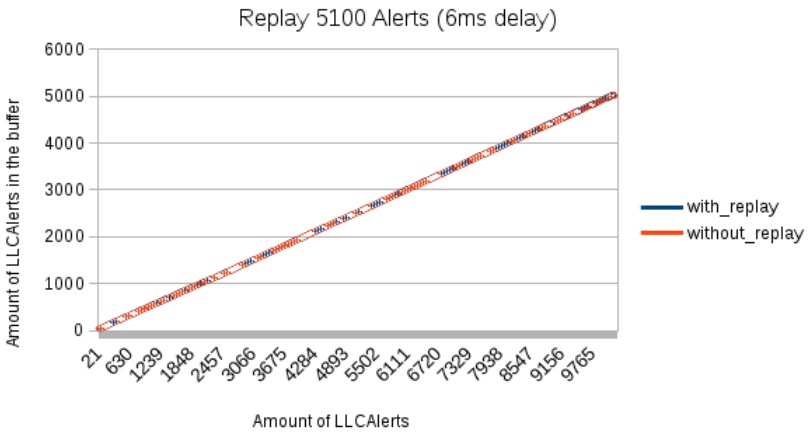

Figure 25 - Buffer size (inter-arrival rate 6 ms)

5.1.7 Integration in ACEA Emulated Environment

The ABE component has been successfully integrated into the ACEA Emulated Environment. As such, the first tests have been conducted. A first scenario of attack consisting in a four step attack has been successfully executed in the emulation environment to test if the LLC component and ABE component are working perfectly together, and that an attack scenario is detected.

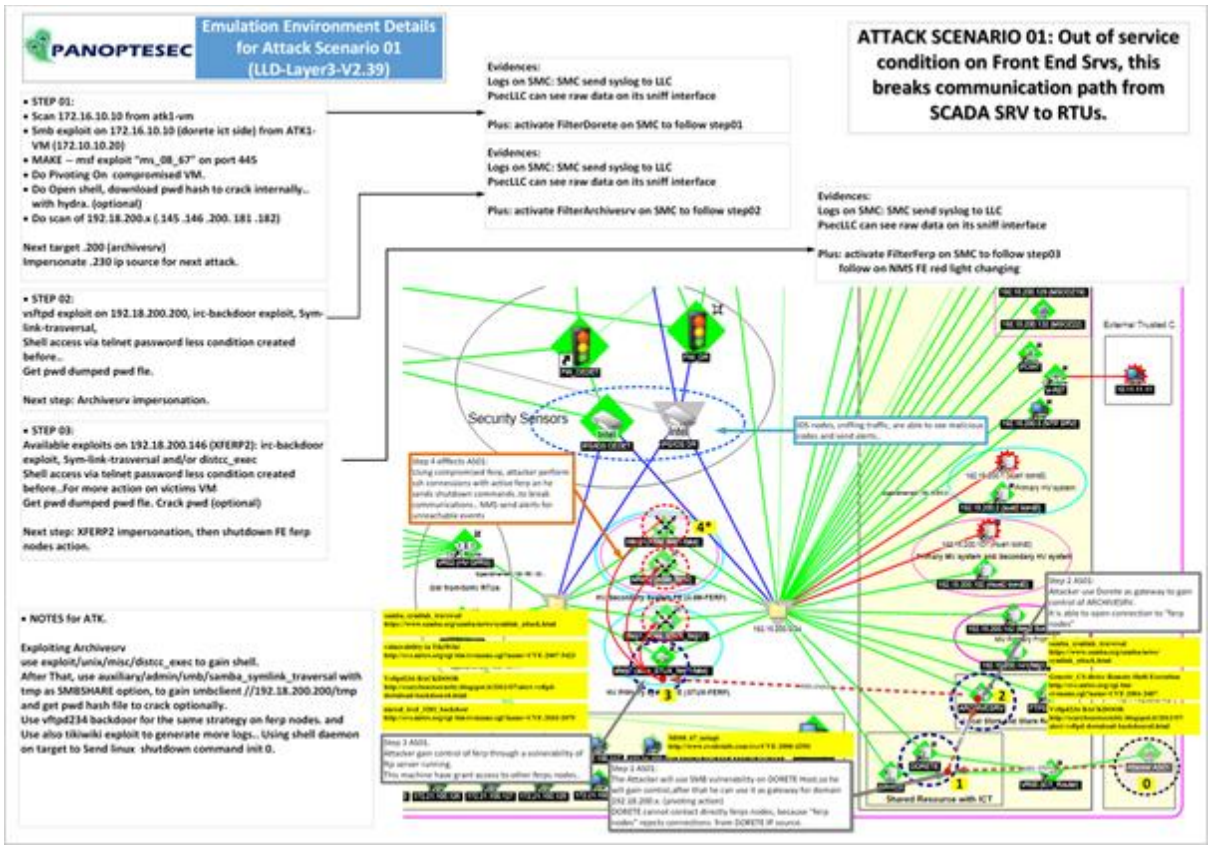

Figure 26 - Emulation Environment and Attack Description

The same attack scenario has been performed in two different ways, giving birth to two different experimentations. These experimentations have been applied on the same dataset on both ABE and QBE components. Their results can thus be compared.

Experimentation 1 - ABE

- Wrong sources for Step 3 and 4
- Number of attack paths: 39513
- Step 1: 172.16.10.20 -> 172.16.10.10
- Step 2: 192.18.200.230 -> 192.18.200.200
- Step 3: 192.18.200.230 -> 192.18.200.146
- Step 4: 192.18.200.230 -> 192.18.200.181
- Step 5: 192.18.200.200 -> 192.18.200.181
- Total Syslog messages: 2200
- Total LLCAlerts: 67

The first experimentation is composed of the sequence of steps defined before. This experimentation does not refer to a known attack path. However, it is part of known attack paths. As a result, we obtain the following alerts emitted by the ABE component.

Table 10 - Number of alerts emitted

5 steps	ABE k=1 t=50%	ABE k=1 t=60%	ABE k=0 t=60%
Step 1	3900	0	0
Step 2	298	149	0
Step 3	298	149	0
Step 4	298	149	0
Step 5	43	43	0
TOTAL	4837	490	0

The three columns of the Table above provide the number of alerts that are produced for a given configuration of the ABE.

The first column consists in a configuration where we can miss 1 alert in an attack path, with the production of an alert for each step when at least 50% of the attack path has been recognized. In this configuration, at the first step of the scenario, we have 10% of the attack paths that reach this condition, i.e., 3900 alerts. This means that all these paths with at most 5 steps with one missing alert reach the 50% threshold. Thus we have a lot of ongoing alerts that are emitted. But this result was expected. As a matter of fact, on step 5, only 43 alerts are emitted, corresponding only to missing alerts or approximated attack paths. This configuration shows that the correlator is able to send alerts even if LLC alerts corresponding to an attack path are missing.

The number of alerts can be reduced when the threshold is growing. This is the purpose of column 2, where no alert is emitted on step 1.

An important experimentation is exhibited by column 3. If we do not consider missing alerts, there is absolutely no alert emitted, because no exact known attack path has been recognized.

This experimentation shows that the correlator can be tuned to fit different functionalities. As a matter of fact the integration of LLC component and ABE works pretty well.

Experimentation 2 - ABE

- Number of attack paths: 39513
- Step 1: 172.16.10.20 -> 172.16.10.10

- Step 2: 192.18.200.230 -> 192.18.200.200
- Step 3: 192.18.200.200 -> 192.18.200.146
- Step 4: 192.18.200.146 -> 192.18.200.181
- Total Syslog messages: 2076
- Total LLCAlerts: 64

The second experimentation consists in following an exact known attack path during the attack scenario. The scenario is composed of 4 steps. The results are summarized in the Table below.

Table 11 - Number of alerts emitted

4 steps	ABE k=1 t=50%	ABE k=1 t=60%	ABE k=0 t=60%
Step 1	3900	0	0
Step 2	298	149	0
Step 3	43	43	12
Step 4	372	24	4
TOTAL	4613	216	16

The first configuration (column 1) of the correlator generates a lot of alerts, as the threshold is too low, and a lot of paths are very similar to the path we try to detect. However, if we don't consider missing alerts (column 3), with a threshold of 60%, at the fourth step, we emit only 4 alerts. These alerts correspond to 3 approximated attack paths (i.e., attack paths that terminate with a different target IP address), and 1 exact recognition. Thus we detect exactly the attack scenario.

5.2 Query-based High-level Online Correlation (QB-HOC)

This section describes one of the two proposed approaches responsible of analyzing the flow of Low Level Alerts (generated by the LLC component) and of detecting, possible information that are corresponding to the known attack scenarios (described by AGG output).

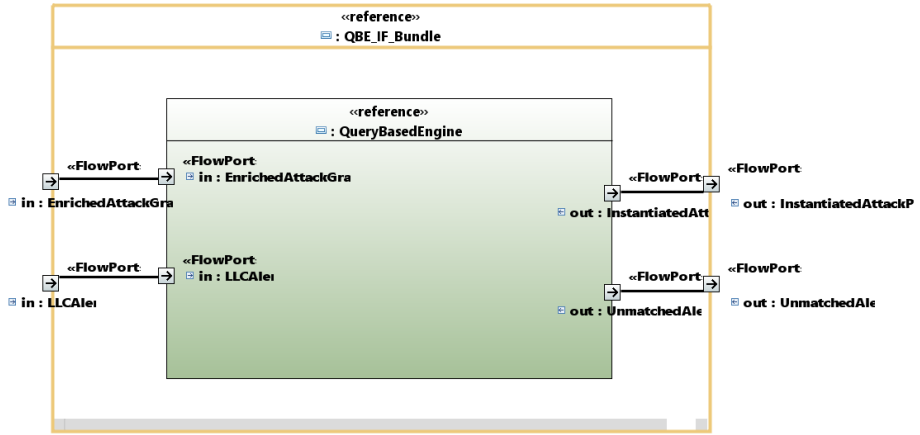

Figure 27 – Query Based Engine Inputs and Outputs

Figure 27 describes inputs and outputs of the Query-Based Engine. In the following we provide some highlights on the modelling and algorithms used by this component.
In order to estimate on-line the on-going attacks, QBE takes the following steps:

1. *Establish if an LLCAlert matches one or more edges in the current EAG*.
2. *Estimation of the attack progress on each Enriched Attack Path*.
3. Generation *of IAPs*.

In order to prevent an attacker to reach its target, an IAP should be generated early enough to take the proper countermeasures. As a consequence, an IAP must be generated before QBE observe all the exploits of a given path. This is done by defining a *Threshold* on the similarity metric: when an EAP exists such that its associated similarity is above the threshold, then the corresponding IAP is generated. The selection of a good threshold is thus mandatory to have a good trade-off between the numbers of false positive/false negative. This point will be analysed later in Section 5.2.4.

In addition, QBE must also be able to take into account new EAGs computed during the evolution of the system (e.g. due to new vulnerabilities detection, new devices added, changes in the routing configuration etc.). To this aim, QBE also implements a mechanism to switch the computation from the current EAG to the new one. In the current implementation, when a new EAG is notified to QBE, it suspends the computation of incoming LLCAlerts and it starts to parse and allocate data structures needed to handle the new EAG. Therefore, the processing of LLCAlerts and the processing of the new EAG work in mutual-exclusion. It is important to note that meanwhile QBE processes new EAGs it does not drop new LLCAlerts, but it stores them in a specific queue. At the end of the EAG's computation task, QBE unlocks the matching process restarting from the alerts that have been queued. Moreover, the QBE component is able to manage new EAGs without discarding all the information collected before the arrival of the new graph. When a new EAG is sent to QBE, before leaving the computation of the new graph, it starts a "Reverse procedure" able to manage again important old LLCAlerts, without losing previous detection of the attacks.

In order to take into account the components scalability, some internal design choices have been done. In particular, the following considerations hold:

- *The total number of EAP composing an EAG is quite large (tens of thousands of EAP) while the total number of edges is relatively small (hundreds different edges)*. We decided to represent information and evaluate matches on an edge-base. This is done by using a

Complex Event Processing (CEP) engine, Esper [Esper], which is able to filter the stream of LLCAlert by considering only those corresponding to existing edges. This has two main advantages: (i) the filtering time is optimized by the usage of Esper and (ii) the number of data structures used to estimate progress on paths remains small making the memory usage limited.

- However, there is the drawback of having a large number of paths insisting on the same set of edges, with a corresponding trade-off between the computation time required for evaluating progress of the attacks on multiple paths and the maximum matching rate that the engine can sustain. This point will be evaluated in section 5.2.4.3.
- *Multiple LLCAlerts may match over the same edge*. This implies that information related to multiple matches must be stored locally causing a progressive increase of the memory usage. This phenomenon exposes the same trade-off discussed in the previous point.

5.2.1 Contribution

QBE employs in the same place different techniques coming from different fields.

As far as we know, QBE is the first attempt to employ Complex Event Processing techniques and string recognition metrics to correlate alerts.

Such approach gives us the possibility to perform efficiently on-line correlation. We do not need to search through data structures.

As an example, previous techniques used to store the most significant matching alerts in a graph in order to search on it possible paths. Such implies a graph search each time a new alert is detected. In our case, employing a Complex Event Processing, for each new alert, we are already focused on the alert interested paths.

QBE, contrarily to most of the other approaches, takes advantages of the attack graph provided as input and the correlation is performed directly on line. The establishment of the matches between *LLCAlerts and the edges of the EAG is done by considering two types* of matches:

- *Exact match*: An exact match happens when, given an edge of an EAP characterized by the triple < source IP, target IP, CVE > and an LLCAlert described by the triple < source IP', target IP', CVE', we have that source IP = source IP', target IP = target IP' and CVE = CVE'.
- *Approximate match*: given an edge of an EAP characterized by the triple < source IP, target IP, CVE >> and an LLCAlert described by the triple < source IP', target IP', CVE' >, we have an approximate math when two out of three equalities hold, except for the match on the pair < source IP, CVE > that it does not provide important information on the on-line attack and it contributes only to generate a lot of false positives. Let us note that, such type of match allows to take in to account possible inaccurate or wrong information (e.g., a missing source IP address in the LLCAlert, a mismatch in the CVE due to different classifications used by the underline IDS and the vulnerability scanner etc…).

Approximate matches and exact matches associate a different weight to the match so that, during estimation of the attack progress, it is possible to consider differently the two cases.

Another important contribution is the employment of techniques used for similarity metrics. This gives us more flexibility since we can consider even alerts that, due to network communication delays, are delivered in an order different from the generation one. So basically instead of doing a search in a graph we compare how similar are the attack scenarios that we are observing with a

known one. Once QBE identified which is the subset of edges corresponding with observed exploits, it estimates, for each EAP, if there is a progress of the attack. In order to estimate the progress, QBE uses some similarity metrics generally used in the domain of string recognition and text mining. The basic idea is to represent each EAP as a reference string. As soon as LLCAlerts matches over edges of a given path P, a current path string is constructed based on the matches. QBE then evaluates the similarity between the two stings and assign the corresponding score to the path. In the current implementation, we consider the following metrics: Jaccard Similarity, Cosine Similarity and 1- Edit Distance. In particular, the first two metrics does not make any difference between a stream of alerts that is ordered with respect to the edge sequences and a non-ordered stream. Conversely, the third metric is able to make the difference between the two cases.

5.2.2 Testing and experimentation strategy description

The proposed tests have three main purposes: to determine if the QBE component is able to satisfy its requirements; to determine which performances can be reached; to determine the detection capability. Obviously, some tests have been performed in a standalone version, since the complete control of the inputs and outputs is needed, and other ones in a sub-system integration way.

More in detail, Section 5.2.3 considers internal aspects of QBE component such as evaluations on the metrics used in order to identify possible attacks on specific paths, either in presence of exact matching and partial ones. Section 5.2.6.1 instead evaluates QBE performances (memory usage and CPU performances) under different input conditions. To this aim we vary the LLCAlerts input rate and we change the input dataset to impact in terms of alerts match the graph, and alerts that does not.

In Section 5.2.4, we will evaluate the integration with other components of the PANOPTESEC system, in particular the integration with the AGG-TRQ and LLC components. The evaluation will be made in terms of false positives and false negatives raised, according to a specific characteristic of the LLC component and according to the graphs generated by AGG.

Final tests for the QBE component will be made on the ACEA Emulation Environment, in Section 5.2.7, showing which is the actual behaviour of QBE working on a real scenario. This evaluation will be made in terms of InstantiatedAttackPaths generated, starting from a specific input for the LLC component. Therefore, a complete analysis of all the correlation process will be showed.

5.2.3 Individual component V&V

5.2.3.1 Evaluation of metrics used to generate IAP

The generation of IAP from QBE is based on the similarity estimation between the LLCAlerts set notified by LLC and each EAP contained in the EAG. Such estimation can be done in different ways computing different similarity metrics. Depending on the specific metric considered, it is possible to obtain different results in terms of number of IAP (#IAP) generated from the same attack scenario on the same EAP.

In the current prototype version of QBE, three different metrics have been implemented:

- Jaccard Similarity (JS);
- Cosine Similarity (CS);
- 1- Edit Distance (1-ED).

For every EAP expressed in the input EAG, all those metrics can be computed just by knowing the path length (denoted as |EAP|) and the number of LLCAlerts matching on such path (denoted as x).

JS and CS do not consider explicitly temporal information about LLCAlerts (i.e., they do not take into account timestamps of LLCAlerts, as a consequence they consider equally EAPs expressed using a SEQ operator or using an AND operator) while 1-ED does. This means that JS and CS order the set of notified LLCAlerts according to their arrival order in the EAP without considering if they really happened in such order. On the contrary, 1-ED orders the set of notified LLCAlerts according to their timestamp.

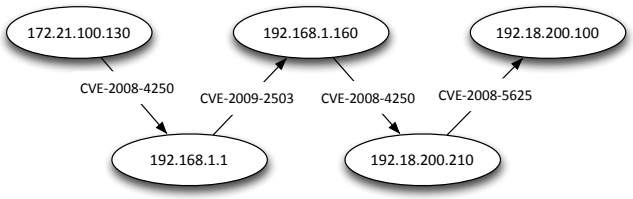

Figure 28 - Example of EAG extracted from the PANOPTESEC Emulation Environment

As an example, let us consider the EAP shown in Figure 28 and the following three triples corresponding to three LLCAlerts matching on the considered EAP:

- LLCAlert 1 = < 172.21.100.130, 192.168.1.1, CVE-2008-4250 >
- LLCAlert 2 = < 192.168.1.1, 192.168.1.160, CVE-2009-2503 >
- LLCAlert 3 = < 192.18.200.210, 192.18.200.100, CVE-2008-5625 >.

When computing JS and CS, QBE always considers LLCAlert 1, LLCAlert 2, and LLCAlert 3 in this order as this is the order in which they appear in EAP. Contrarily, when computing 1-ED, QBE orders LLCAlert 1, LLCAlert 2 and LLCAlert 3 according to their timestamp.

This is done to take into account the possibility of non-synchronized data sources and to evaluate the absence of synchronization impact on the path detection.

Let us note that when the temporal order of LLCAlerts is equal to the path order, 1-ED corresponds to JS. As a consequence, talking about 1-ED it makes sense to consider a parameter x_FP that consider the percentage of LLCAlerts matching over the considered EAP but notified out of order with respect to the sequence.

Figure 29 and Figure 30 show respectively how JS and CS evolve when *x* LLCAlerts match on a given EAP. This is done for different possible lengths of the EAP.

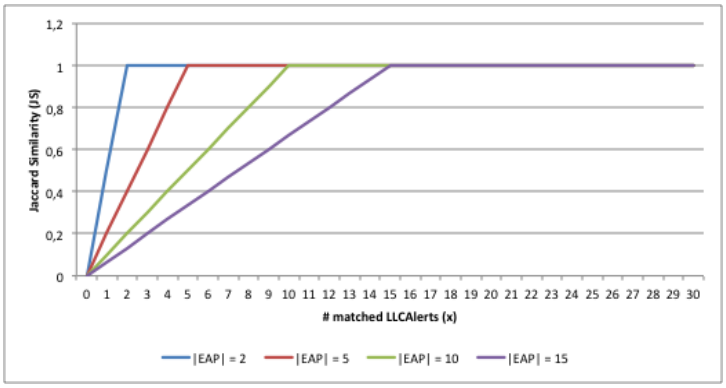

Figure 29 – Jaccard Similarity Metrics evolution for different EAP length

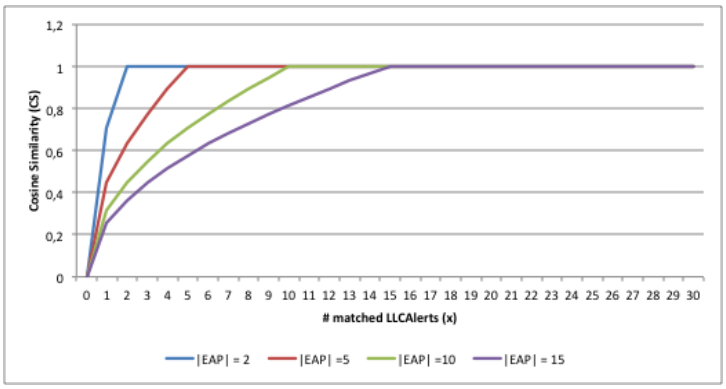

Figure 30 – Cosine Similarity Metrics evolution for different EAP length

It is possible to see that both JS and CS converge to similarity 1 as soon as new matches on new edges are found. The main difference is in the convergence speed that depends on the length of the path (linear for JS, logarithmic for CS).

The same analysis has been done for the 1-ED metric (Figure 31). In this case, given x LLCAlerts matching on an EAP, we considered different proportion of unordered ones and we evaluated their impact on the metric computation (i.e., x_FP=20% and x_FP=50% of x).

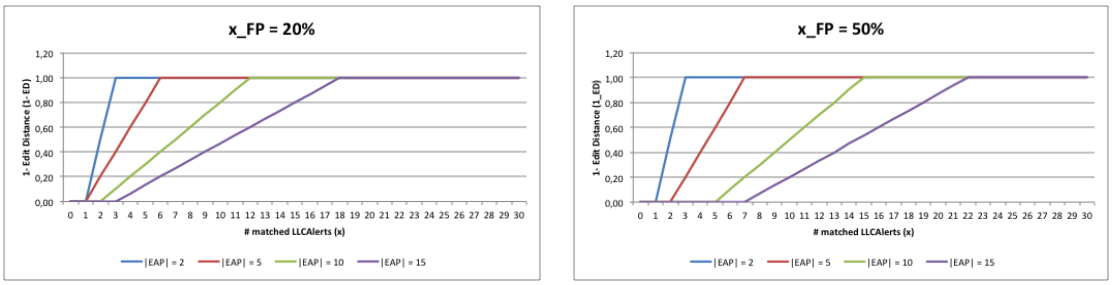

Figure 31 – 1-ED Metric evolution for different EAP length (scenario with perfect matching)

Also in this case, it is possible to see that the metric is able to converge to the similarity 1 as soon as the number of ordered matching LLCAlerts grows. Such convergence is still linear but it is strongly influenced by the proportion of unordered LLCAlerts.

A comparison of the three analysed metrics is shown in Figure 32.

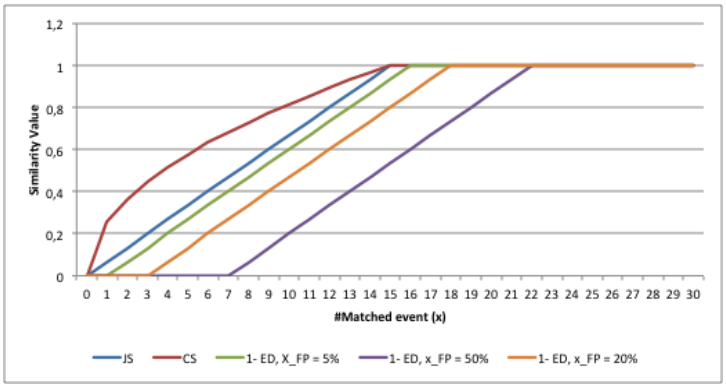

Figure 32 – Metrics comparison for |EAP| = 15 (scenario with perfect matching)

Comparing all the metrics considered so far, it is possible to see that, independently from the considered metric, QBE is able to match and recognize on-going attacks as soon as their intermediate steps are recognized by the LLC. The main difference between the three considered metrics is in the convergence speed i.e., how many matching alerts are needed to have a "good" assessment.

Currently, QBE uses the three metrics independently to estimate the presence of an on-going attack. However, considering the impact that disordering may have on estimation, we are currently defining a mechanism to combine together JS (or CS) with 1-ED in order to get benefit from both. In addition, we are also investigating a variation of 1-ED whose aim is to analyse where the violations of the ordering happen and assign to them different weights. The intuition behind this extension is that having order inversion (or missing event) at the end of the path is very much different from having the same phenomenon at the beginning of a path and the two scenarios should be evaluated differently.

5.2.3.2 Evaluation of metrics in presence of partial matching

Let us recall that an approximate matching occurs when, given an edge of an EAP characterized by the triple <sIP, CVE, tIP> and given the triple <sIP', CVE', tIP'> extracted from an LLCAlert, only two out of three fields are equal.

When computing JS and CS, we are not considering any difference between exact matching and approximate matching. On the contrary, we consider explicitly the approximate matching when computing 1-ED.

Figure 33 shows the same evaluation done in the previous section but now considering that each LLCAlert matches an edge of and EAP approximately.

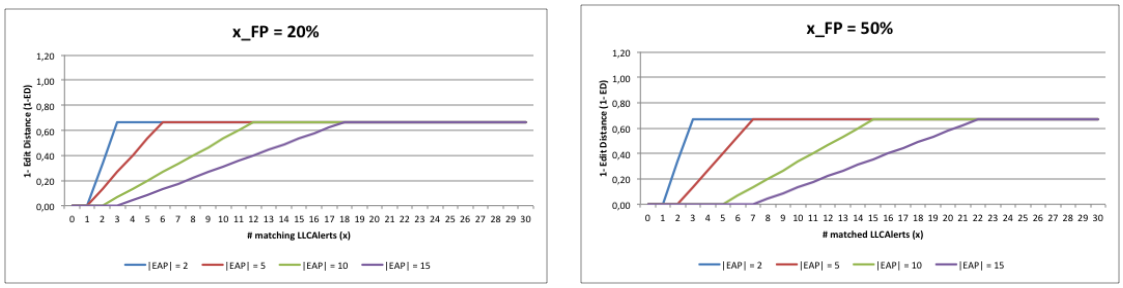

Figure 33 – 1-ED Metric evolution for different EAP length (scenario with approximate matching)

It is possible to see that, in this worse case, the metric does not converge to similarity 1 and correctly takes into account inaccurate or incomplete LLCAlerts.

Figure 34 shows the comparison between the metrics in presence of all approximate matches.

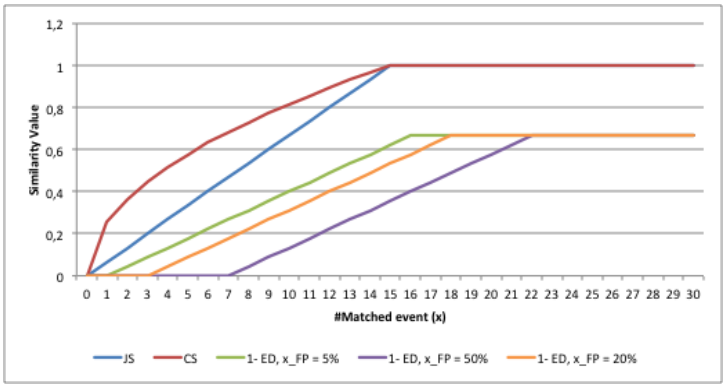

Figure 34 – Metrics comparison for |EAP| = 15 (scenario with approximate matching)

From the comparison, it is possible to see that when we consider together both inaccurate or missing LLCAlerts and the possible absence of synchronization, it is fundamental to choose properly a threshold T that triggers generation of IAP. This point is analysed in detail in the following Section.

5.2.4 Sub-system integration V&V

5.2.4.1 Evaluation of False Positive IAP generation depending on the Threshold T

As said before, QBE uses a threshold T in order to decide if an IAP has to be generated or not.

However, if the threshold is not set correctly, its usage may imply the generation of a lot of false positives/negatives:

- if the threshold T is too small, very few alerts on a path *x* are necessary to generate an IAP (even with incorrect information). As a consequence, QBE may generate a false positive.
- if the threshold T is too high, we need almost all the LLCAlerts corresponding to the path in order to generate an IAP. However, due to incorrect fields in the LLCAlerts, it is possible to remain behind T even if the attack succeeded and QBE may generate a false negative.

For these reasons the management of the threshold needs to be carefully investigated.

Looking to the general definition of *False Positive* we have that it is defined as *an error in data reporting in which a test result improperly indicates the presence of a condition when in reality it is not*.

As a consequence, the specific meaning of False Positive is different when associated to LLC and when associated to HOC.

DEF. At QBE level we consider as a false positive as the generation of an IAP, associated to an EAP P, when no attacks is on-going over P.

Let us note that, if the LLC is perfect (i.e., it does not notify any LLCAlert not related to a real on-going attack) then QBE will not generate false positive. In fact, independently from the selected threshold T, any IAP generated will correspond to an EAP with a real on-going attack. In this case, the only impact of the threshold T is on the time interval between the time of the on-going attack detection (i.e., the time at which the first IAP is generated) and the time of the completed attack.

On the contrary, if LLC is not perfect (e.g., it identifies as an LLCAlert a suspicious activity generated by the system itself or it does not filter correctly information coming from IDSs) then QBE evaluates LLCAlerts that may be not relevant and, depending on the value of T, may induce the generation of false positive IAPs.

In order to evaluate the accuracy of QBE in terms of false positive IAPs generated, we considered the following parameters:

- The probability *P_LLC_FP* that LLC generates and propagates LLCAlerts not corresponding to any real on-going attack. In this way, our analysis is independent from the current implementation of LLC as it is considered as a black box characterized by its P_LLC_FP.
- The threshold *T* used to generate IAP.
- The length *|EAP|* of the path we are monitoring.

Given these three parameters, we computed the probability that the first generated IAP is a false positive. Let us note that this happens when all the LLCAlerts used to estimate the on-going attack (and thus to generate the IAP) are false positive generated by LLC.

In particular, using the evaluation result, done in section 5.2.3, we computed for each considered threshold T, the minimum number *x_min* of matches on a path of a given length |EAP|, needed to generate the first IAP. Then we used such number to compute the probability that all the x_min LLCAlerts are false positive.

Figure 35, Figure 36 and Figure 37 show the probability of generating a false positive IAP for different types of LLC (each characterized by its probability of generating false positive) and different values of the threshold T.

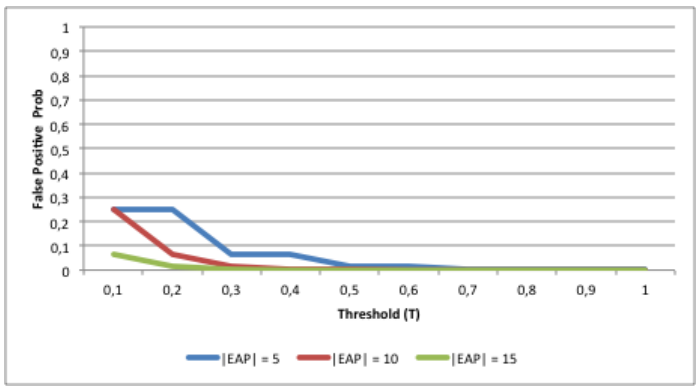

Figure 35 – Probability of generating the first False Positive IAP for different threshold T and an LLC characterized by P_LLC_FP = 0,25

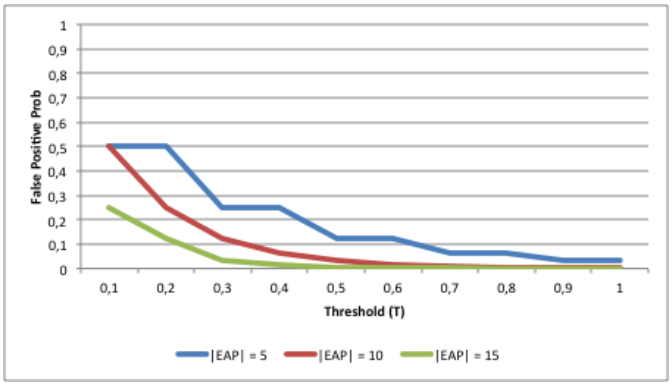

Figure 36 – Probability of generating the first False Positive IAP for different threshold T and an LLC characterized by P_LLC_FP = 0,5

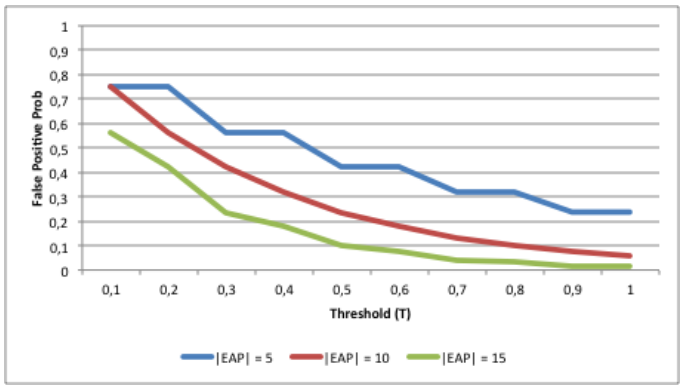

Figure 37 – Probability of generating the first False Positive IAP for different threshold T and an LLC characterized by P_LLC_FP = 0,75

Figure 38 shows the probability of generating the first IAP as a false positive for different values of the Threshold T and different (always less) accurate LLC. Let us note that this comparison is done by considering an EAP of length 5 that is a situation similar to the one we can observe in the emulation environment.

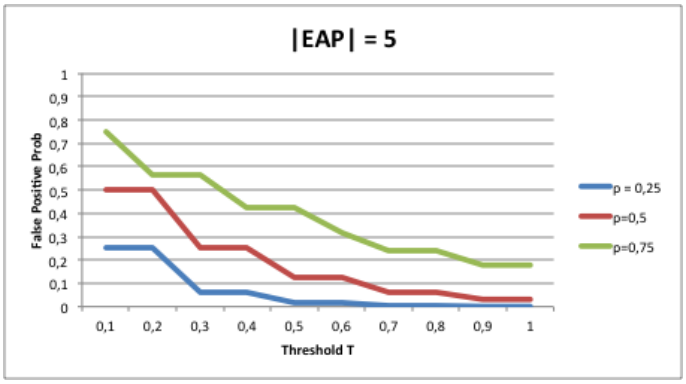

Figure 38 – Comparison of False Positive IAP generation for different LLC

5.2.4.2 Evaluation of False Negative IAP generation depending on the Threshold T

Looking to the general definition of *false negative* we have that *it is an error occurring when a test result indicates that a condition failed, while it actually was successful* i.e., erroneously no effect has been assumed.

Also in this case, the specific meaning of False Negative is different when associated to LLC and when associated to HOC.

DEF. At QBE level we consider as a false negative as the missing generation of an IAP, associated to an EAP P, when an attack is on-going over P.

Again, if the LLC is perfect (i.e., it notifies only LLCAlert related to a real on-going attack and does it in an accurate way) then QBE does not generate false negative. In fact, independently from the selected threshold T, any IAP generated will correspond to an EAP with a real on-going attack. In this case, the only impact of the threshold T is on the time interval between the time of the on-going attack detection (i.e., the time at which the first IAP is generated) and the time of the completed attack.

On the contrary, if LLC is not perfect some false negative may be generated due to:

1. Alerts filtered by LLC because they have been interpreted as not relevant or
2. Too much inaccuracy in the LLCAlerts generated.

Let us note that, considering the evaluation of the current implemented metrics used by QBE (cf. section 5.2.2), it is possible to see that scenario 2 can be avoided using together one metric weighting the inaccuracy (i.e., 1-ED) and one metric ignoring such weight (e.g., one between JS and CS).

On the contrary, if LLC is not perfect (e.g., it filters alerts due to misclassification, misconfiguration etc.) then QBE may miss to evaluate LLCAlerts that may be relevant and, depending on the value of T, may prevent the generation of an IAP (incurring in a false negative).

In order to evaluate the accuracy of QBE in terms of false negative (IAP not generated), we considered the following parameters:

- The probability *P_LLC_FN* that LLC filters and does not propagate a proper LLCAlerts to QBE. This is done to let our analysis independent of the current implementation of LLC as it considers LLC as a black box characterized by its P_LLC_FN.
- The threshold *T* used to generate IAP.
- The length *|EAP|* of the path we are monitoring.

Given these three parameters, we computed the probability of generating a false negative i.e., the probability of missing the issue of an IAP for a real on-going attack. Let us note that this happens when QBE misses (or is not able to match) a number of LLCAlerts that prevent it to go beyond the fixed threshold and to estimate the on-going attack.

In particular, using the result of the evaluation done in Section 5.2.3 we computed, for each considered threshold T, the maximum number *x_max* of matches needed to prevent the generation of an IAP on a path of a given length |EAP| and used such number to compute the probability that all the x_max LLCAlerts are false negative (i.e., that LLC fails in notifying x_max useful LLCAlerts).

Figure 39, Figure 40 and Figure 41 show the probability of generating a false negative for different types of LLC (each characterized by its probability of generating false negative) and different values of the threshold T.

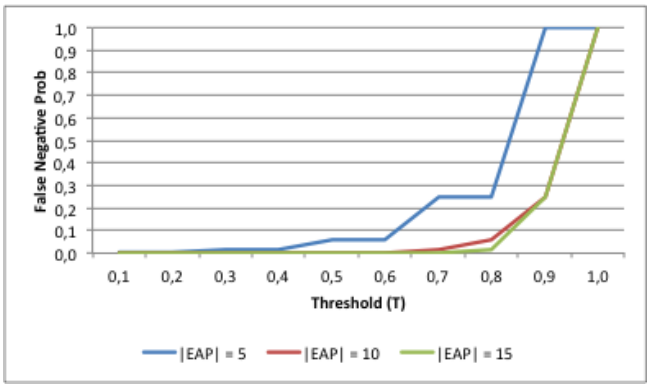

Figure 39 - Probability of generating a False Negative for different threshold T and an LLC characterized by P_LLC_FN = 0,25

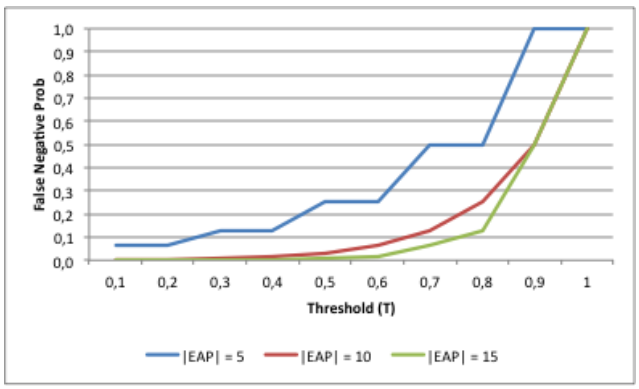

Figure 40 - Probability of generating a False Negative for different threshold T and an LLC characterized by P_LLC_FN = 0,5

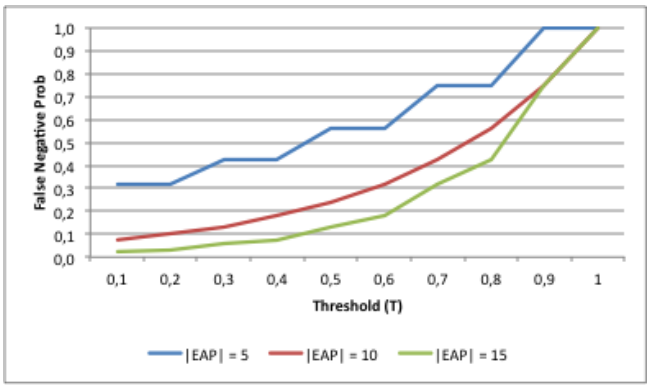

Figure 41 - Probability of generating a False Negative for different threshold T and an LLC characterized by P_LLC_FN = 0,75

5.2.4.3 Impact of the EAG Topology on the False Positive / False Negative IAP

All the analysis done in the previous Sections consider the case of a single EAP. However, when looking to the whole graph, it is easy possible that more than one EAP share a common subset of edges.

As an example, if we consider the EAG from the emulation environment generated on March 3rd and we consider the EAP with ID 4.2 having length 4 and depicted in Figure 42, it is possible to see that it shares a sub-sequences of length 3 (highlighted in red) with the 7 following paths: 4.1, 4.3, 5.1, 5.2, 6.1, 6.2, 6.3.

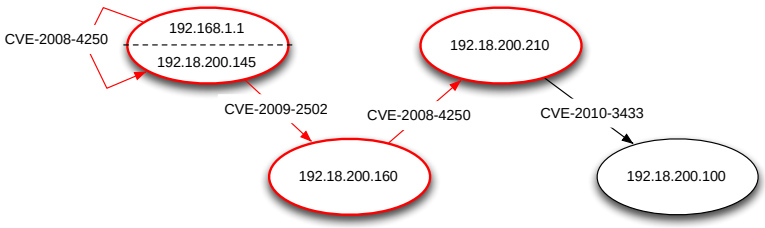

Figure 42 - Example of EAP with shared edges

The immediate consequence is that there are sub-paths shared by more than one EAP (8 in the previous example) that are undistinguishable and that may cause the generation of an IAP for each of the sharing EAP. Let us note that, in general, attacker follows one particular EAP and not all the shared ones. As a consequence, given a set of n paths sharing a sub-sequence able to generate an IAP we have 1 real on-going attack (with its corresponding IAP) and n-1 false positive IAP. This phenomenon is amplified if we consider that the threshold generating IAP depends on the length of the path.

This means that the topology of the EAG has a strong impact on the number of IAP generated and consequently on the number of false positive IAP.

In order to evaluate impact of the sub-sequence sharing, we first run an experiment on a family of synthetic EAG shaped as shown in Figure 43.

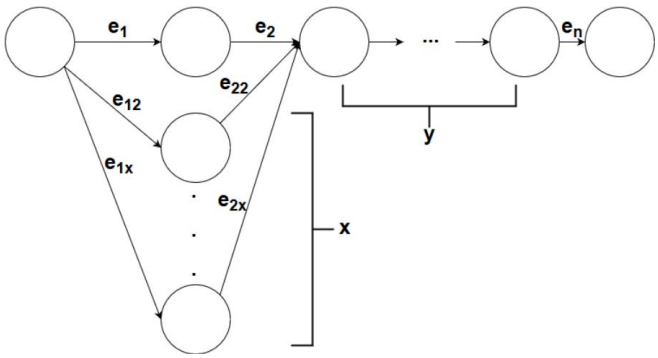

Figure 43 - Topology Schema for Synthetic EAG

The topology is characterized by a "main path" e_1, e_2, … e_n and by x other sub-paths sharing n-2 edges e_3, … , e_n.

Thus, and EAG in this family can be characterized by a pair (X, Y) where:

- **X** defines the number of sub-paths sharing a common sequence.

- **Y** defines the length of all the shared sub-paths.

Given such topologies, we evaluated how the metrics increase for the attacked path (Figure 44) and for all the other paths sharing a sub-sequence (Figure 45).

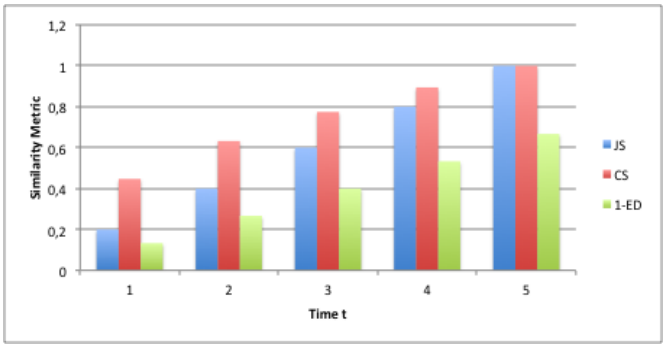

Figure 44 - Metric evolution on the "attacked" path for an EAG (5, 3)

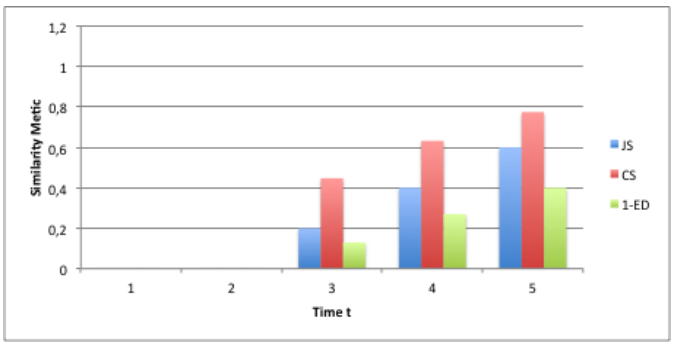

Figure 45 - Metric Evolution on "secondary" EAP for an EAG (5, 3)

It is clear that in order to distinguish the attacked path from the others, we need a threshold higher than the metrics computed by considering matches over all the shared sub-sequence.

In order to assess how much our synthetic topologies are representative of the EAGs built in the emulation environment, we did a topological analysis of the EAG computed on March 3rd.

Such EAG is composed by 116 edges defining 9098 different EAPs. Among the 9098 different EAP, 306 has length 3 and 8792 has length 4.

Figure 46 shows the EAP-per-edge distribution i.e., the relation between the number of EAP sharing the same edge and the percentage of edges having such specific sharing factor.

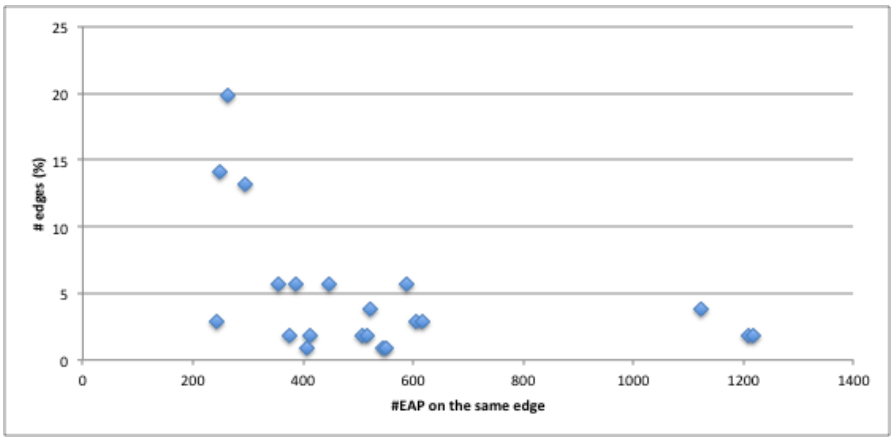

Figure 46 - EAP per edge Distribution

It is interesting to note that there are small clusters of edges (between 1% and 5%) that are shared by a huge number of EAPs (between 400 and 600 EAP) and a relatively high percentage of edges (between 15% and 20%) that are shared by a smaller number of EAP (around 200).

It is even more interesting to see, for each EAP how many sub-paths (or sub-sequences) including more than one edge are shared. Figure 47 shows such distribution.

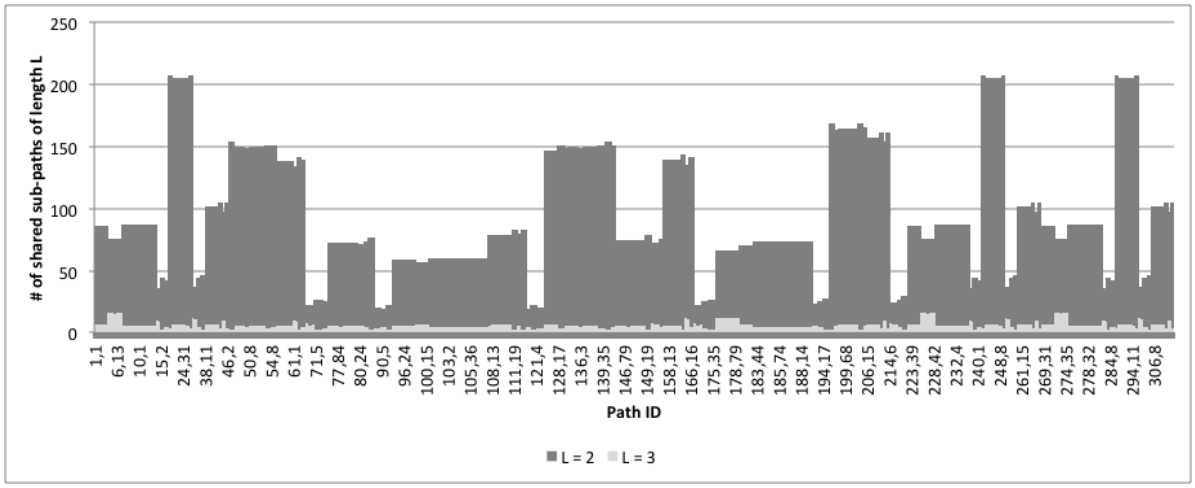

Figure 47 - Distribution of shared sub-path of length L for any EAP

It is possible to see that, for any path in the EAG, there are around 10 others EAP that shares almost all the edges and around the 10% of EAP that shares half of the EAP.

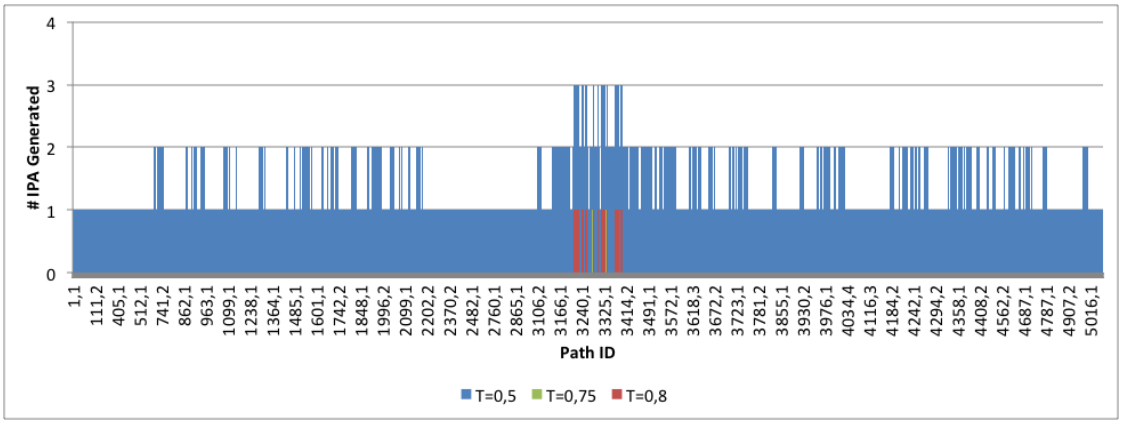

Figure 48 - Number of IAP Generated by using different thresholds

Figure 48 shows the relation between the threshold used to generate IAP and the number of generated IAP considering the set of LLCAlerts and the EAG extracted from the Emulation Environment on March 3rd.

5.2.4.4 Reduction of Instantiated Attack Paths generated by QBE using Temporal Correlation

During the experimentation made on the integration system we noticed that given exploits that lead to more than one LLCAlert by the LLC component (for example exploits on the SMB protocol), a lot of IAPs were generated by QBE.

These LLCAlerts refer the same edge on the EnrichedAttackGraph used at that time, and therefore their computation generates the same similarity score for a specific path.

This is why it may happen that more than one alert generates more than one IAP, even if the alerts are related to the same attack and it would be reasonable that the component generates only one IAP.

However, we could not consider this as a *problem*, but it would be better to understand a possible way to manage it. In fact, this management would be useful for the components that receive in input the QBE output as, for example, the visualization module. There is no reason to manage a lot of IAP that carries on the same similarity score since they belong to the same attack performed in a *small time window*.

This is why the actual version of the component is able to manage this kind of situation. The reduction is based on the usage of different timestamps for each path of the Graph used (ts_p). These timestamps are updated every time a new IAP for the path p is raised. Whenever a new similarity score is computed on a specific path p, we take into account ts_p. If the new similarity score for the path p is less than the previous one stored, we raise the IAP and we update ts_p. In the case in which the new similarity score is equal to the previous one stored instead, only if the difference between the actual time t and ts_p is greater of a certain amount, we generate a new IAP and we update ts_p with t.

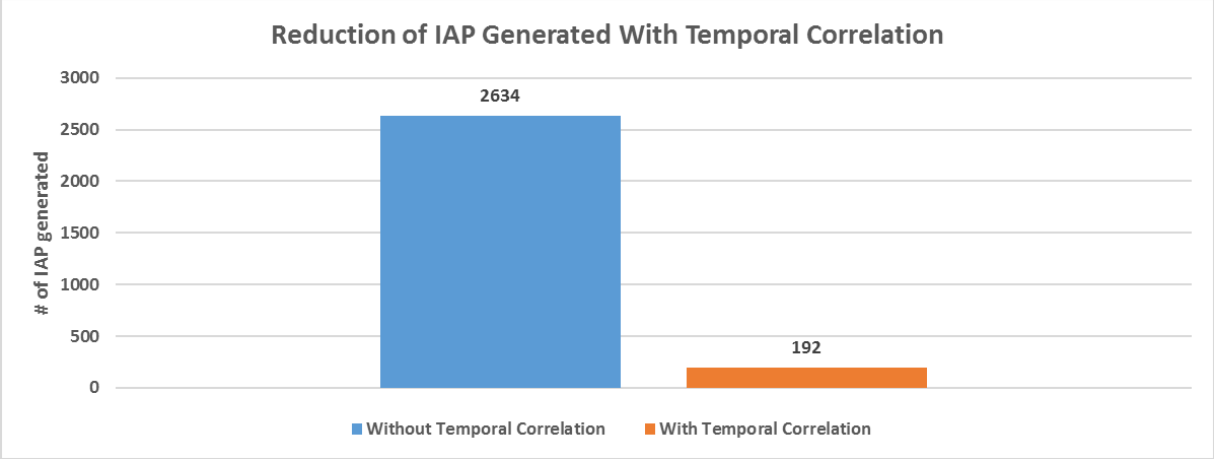

Figure 49 – The image shows the reduction of generated IAP in a specific attack scenario tested in the emulation environment

Figure 49 shows the results obtained managing this situation. In this case, the test consisted in the attack of three different machines on the infrastructure. Without the usage of this reduction the component raised 2634 IAP (when either the first second and third machines were exploited). Using the reduction instead, the component raised only 192 IAPs (93% less).

5.2.4.5 Reduction of False Positives generated avoiding the match on the pair < sourceIP, CVE >

As we already said, QBE manages both exact matches and partial matches. The reason for which we allowed also the partial matching is very simple, and can be easily explained with examples.

- Let us say that for a specific LLCAlert we do not have associated CVEs of the exploit. This is a possible situation, since it may happen that the exploit made by source A and having destination B has not an associated CVE know by the system. For this reason, the match over only the source address and the destination address can handle this kind of situation.
- The same can happen for the match over only the destination address and the CVE of the exploit. Let us suppose that, for any reason, the attacker has been able to arrive to a specific machine X, and the PANOPTESEC system has not recognized his previous attacks before the arrival on X. In this situation the attacker is able to change as he wants the IP address of a specific machine, and consequently if we take into account either the source, destination and CVE of an alert, we will not able to identify the attack. The matching on the source will never be contained on the graph that represents the topology.

Until now, we used also the matching over the fields source address and CVE, but during the latest tests on the integrated system, we have seen that this information is not important to detect attacks. In fact, let us consider the LLCAlert with source address A, destination address B and vulnerability C. If we use this type of matching, this information will highlight also paths containing edges of the type < A, x, C >, with x different from B. This is not correct, since it is not useful to know that the attacker from the same machine and the same exploit will be able to compromise other machines of the infrastructure. Maybe it could be useful for other kind of analysis, but not for the purpose of the HOC modules. Consequently, this kind of matching can raise a lot of false positives.

In the images below it is possible to find a demonstration of the false positives generated considering the match over the pair < sourceIP, CVE >, in a real attack scenario.

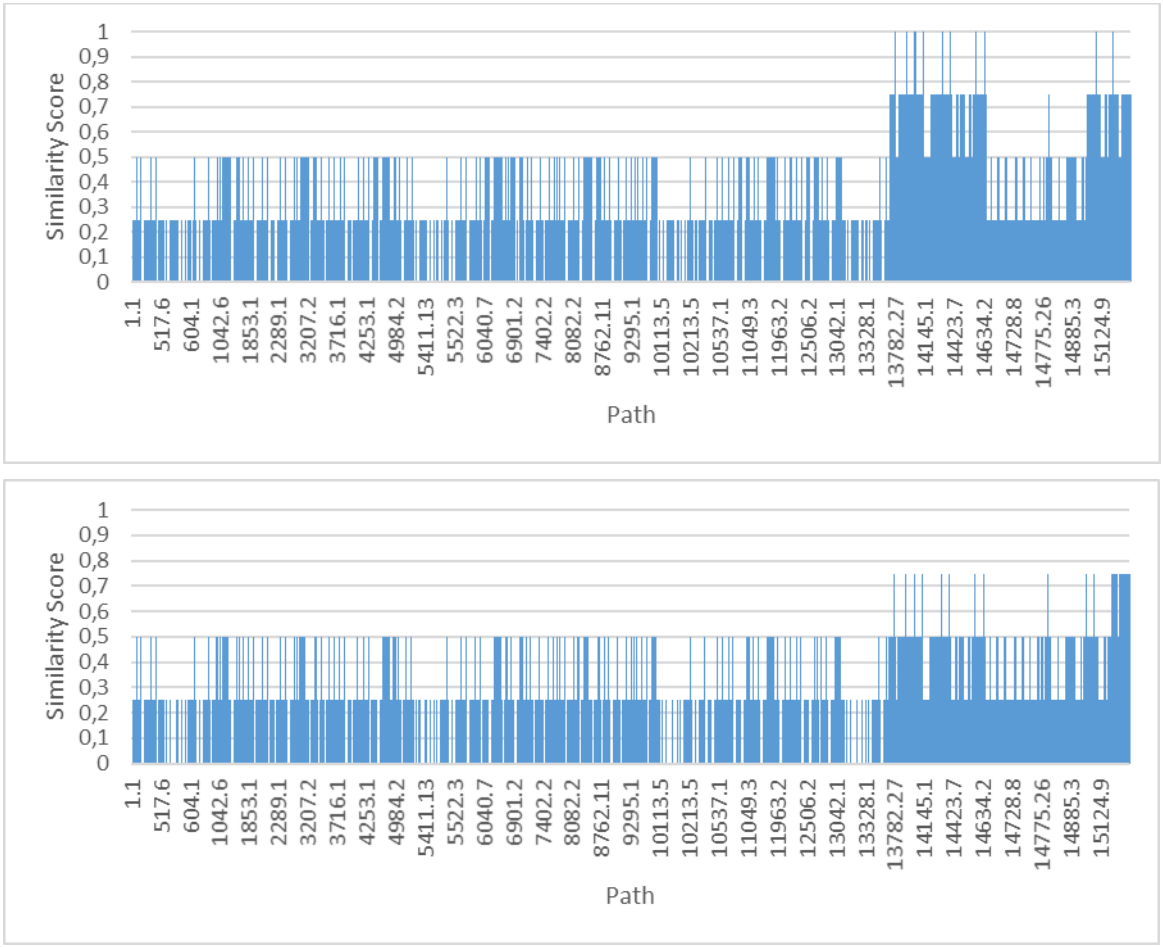

Figure 50 – Similarities score per path at 'second' attack step (without reduction upside and with reduction downside)

The attack scenario used for this experiment is constituted by four different attack steps, and the AttackGraph describing the possible exploits in the network was constituted by 39513 different paths. Figure 50 shows the differences in using or not the matching over the sourceIP and the vulnerability, for the second step. If we use this type of match, we already have paths with a very high similarity score, and consequently a huge number of false positives rose.

Figure 51 shows the same difference but for the fourth step, where the benefits gained from the reduction are more evident.

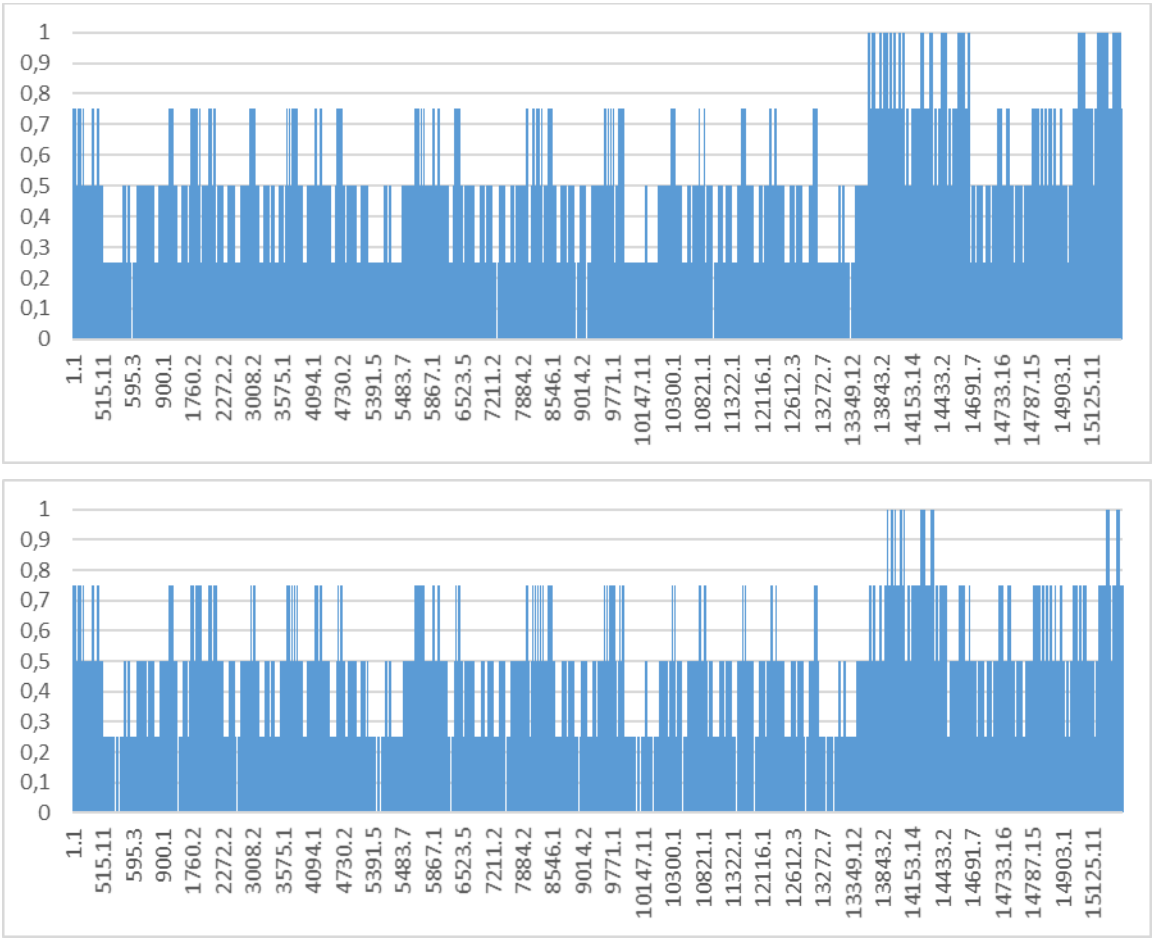

Figure 51 – Similarities score per path at 'fourth' attack step (without reduction upside and with reduction downside)

5.2.5 Uncovered Specialized Requirements

All specialized functional and non-functional requirements of the QBE component have been covered.

5.2.6 Additional experimentations

With the following experiments, we analyse how QBE responds (in terms of memory consumption and throughput) to changes in the input scale.

Monitoring the memory is fundamental, as QBE needs to store (i) the list of EAPs to be monitored and (ii) all information about matched alerts to estimate the presence and progress of an on-going attack.
In addition, it is also interesting to evaluate the performances in terms of CPU consumption.

The scaling parameters considered in our experiments are:

- LLCAlerts arrival rate;
- Proportion between matched and unmatched alerts;
- Size of the Enriched Attack Graph.

In order to assess the performances of QBE at runtime, we measure the following scalability metrics:

- Heap Memory Space Utilization;
- LLCAlerts queue size.

In order to correctly measure our selected scalability metrics, we run scalability experiments by using the stand-alone version of QBE prototype running in a dedicated VM. Such VM has been hosted on a physical machine provided of two physical processors Intel Xeon X5560 2.8GHz with 24GB RAM. Specifically, on this VM we had four cores and 6GB RAM.

The dataset used has been extracted from the Emulation Environment. In particular, we use EAG resulting from the computations done on March 3rd 2016. It carries on ~50MB of information.

Since we need to control the stream of LLCAlerts, we take a sample from the Emulation Environment that is consistent with the EAG used. Then, we inject such stream with synthetic LLCAlerts in order to reach the proper proportion between matched and unmatched ones.

5.2.6.1 Scalability Evaluation

In a previous version of QBE, we considered different LLCAlerts arrival rate to test the performances of the component. In particular, these rates are the following:

- 10 LLCAlerts/sec
- 50 LLCAlerts/sec
- 100 LLCAlerts/sec
- 200 LLCAlerts/sec

The same concerning the proportion between matched and unmatched LLCAlerts, we considered the following ones (the dataset has been sent in loop to QBE):

- First dataset: 0% matched and 100% unmatched;
- Second dataset: 10% matched and 90% unmatched (130 total LLCAlerts, 13 matching selected uniformly at random on the edges and 117 non-matching);
- Third dataset: 20% matched and 80% unmatched (130 total LLCAlerts, 26 matching selected uniformly at random on the edges and 104 non-matching);
- Fourth dataset: 40% matched and 60% unmatched (130 total LLCAlerts, 52 matching selected uniformly at random on the edges and 78 non-matching).

The same experiments are repeated for the new version, and viewing a boost in the performances of the components, we decided to repeat the test related only to the more stressful situation. For example, once stated the relation between the 10, 50, 100 and 200 alert rate (with different datasets), the same relation exists for the actual QBE version. Therefore, it is enough to execute the same test only with the 200 Alert rate.

However, for the completeness of the documentation, in the proposed testes we show both the performances of the previous and the new version.

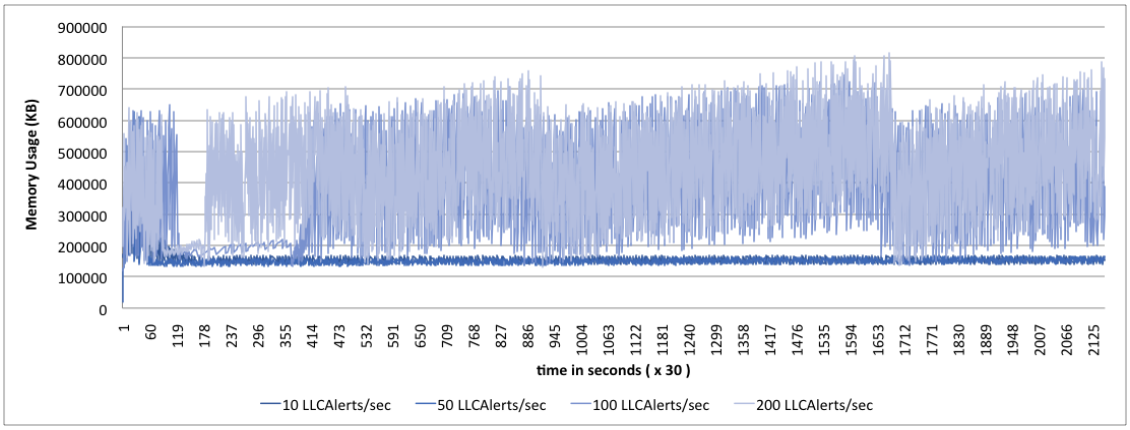

Figure 52 - Previous version: Heap Memory Space Utilization for 0% matched - 100% unmatched scenario

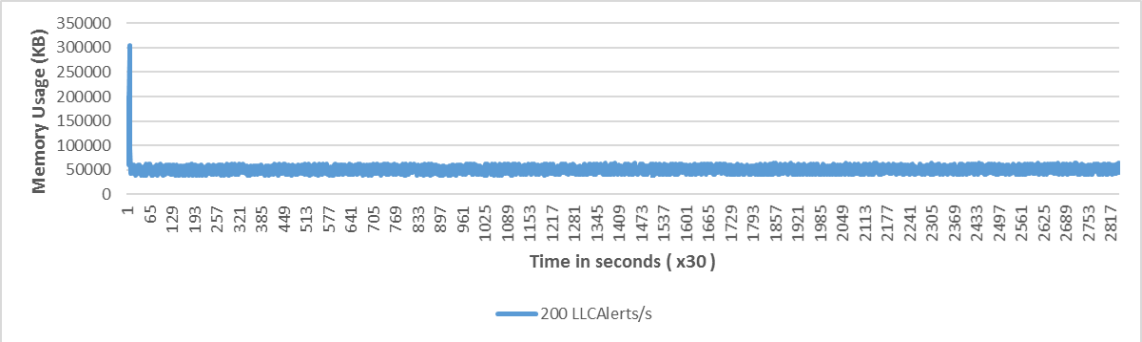

Figure 53 - New version: Heap Memory space utilization for 0% matched - 100% unmatched scenario

As we can see in Figure 52, the memory usage is very low for the first two alert rates, and for the second ones the memory is more "unstable": it grows for a while and then falls down. This is due to the action of the JAVA Garbage Collector that let the component take a large part of the memory and then de-allocate useless data structures when needed. This means that the *real* memory usage is low also for the higher alert rates.

Figure 53 instead, shows the situation for the new version of the component using the same dataset and the higher LLCAlert rate used in the previous version (200 alerts per second). It is possible to state that the amount of memory used is far better than before.

In order to measure the throughput, we monitored the queue size of incoming LLCAlerts (Figure 54).

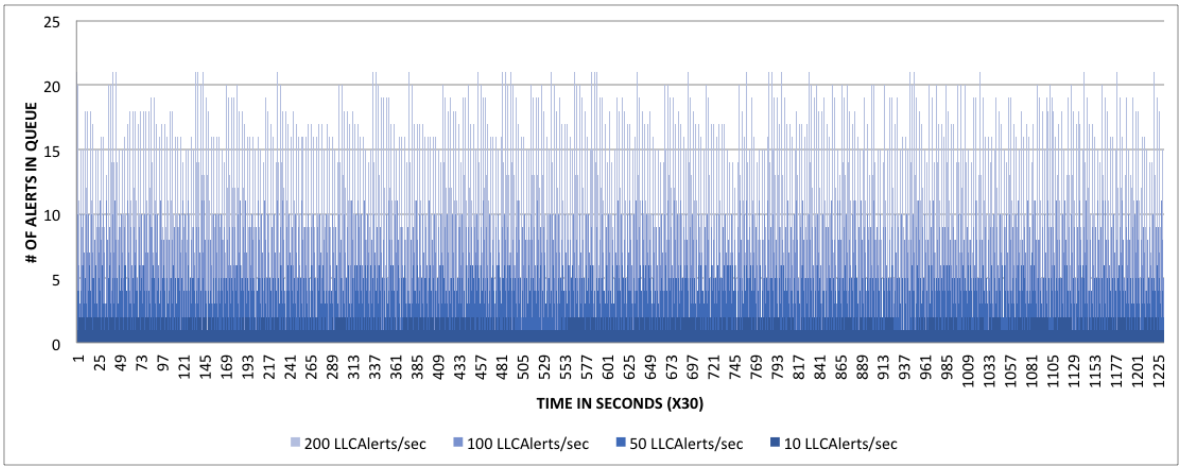

Figure 54 - Previous version: LLCAlerts incoming queue size for 0% matched - 100% unmatched scenario

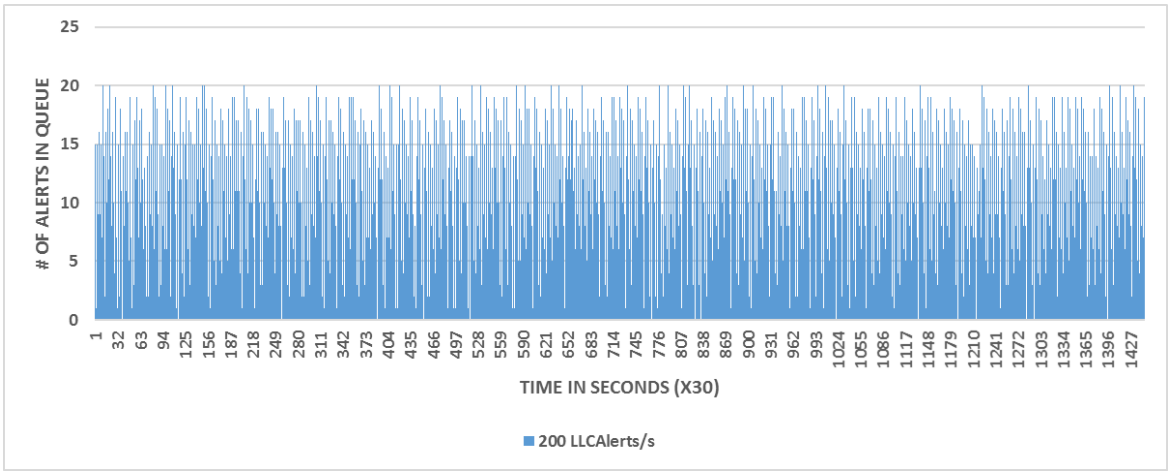

Figure 55 - New version: LLCAlerts incoming queue size for 0% matched - 100% unmatched scenario

When the CPU is not able to sustain the incoming alerts rate, the queue size would be increased forever. However, Figure 55 shows that for the considered rates, this does not happen.

Considering the scenario 10% matched and 90% unmatched, the 4 considered input rates could be mapped in to rates of matching LLCAlerts/sec as follows:

1. 10 Alerts per second: **~1** matching alerts per second
2. 50 Alerts per second: **~5** matching alerts per second
3. 100 Alerts per second: **~10** matching alerts per second
4. 200 Alerts per second: **~20** matching alerts per second

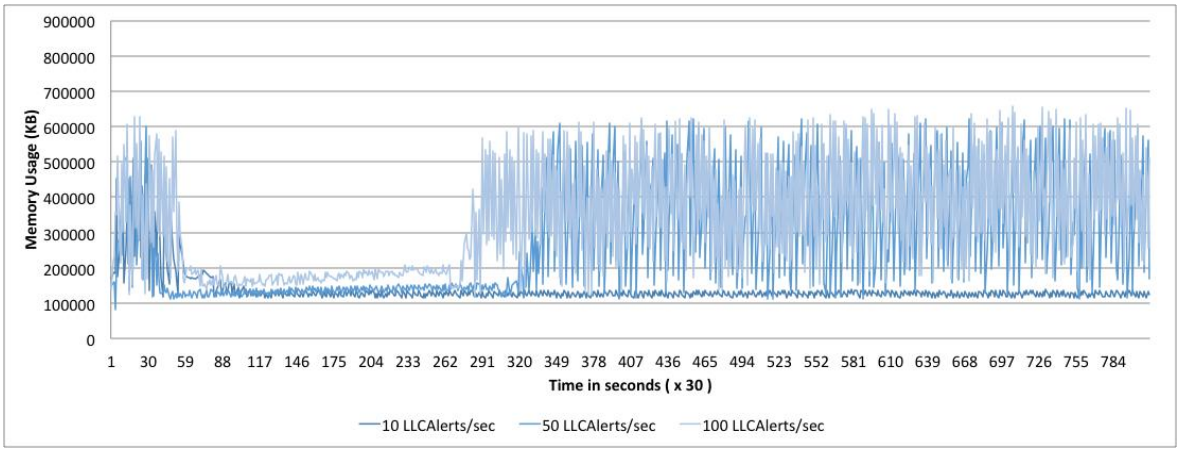

Figure 56 – Previous version: Heap Memory Space Utilization for 10% matched - 90% unmatched scenario

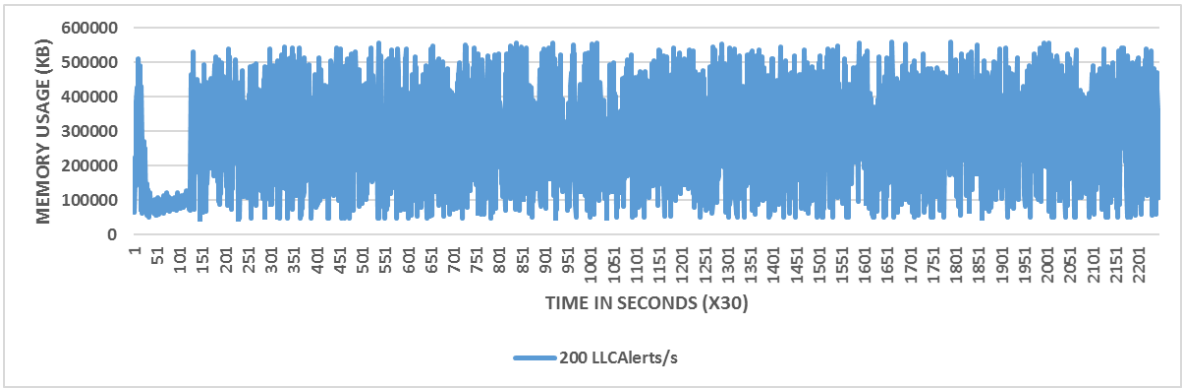

Figure 57 - New version: Heap Memory Space Utilization for 10% matched - 90% unmatched scenario

Figure 56 and Figure 57 show instead the experiments made with a different dataset, composed by the 10% of matching alerts and the 90% of unmatching ones. It is possible to notice that this change in the dataset used has an impact on the memory performances. Moreover, for the previous version of QBE, the 200 LLCAlerts/sec rate is not showed since the component was not able to support the ~20 matching alerts per second on the VM used for the test. For the new version instead, QBE can reach this rate, also using less memory than the previous tests made on the older version.

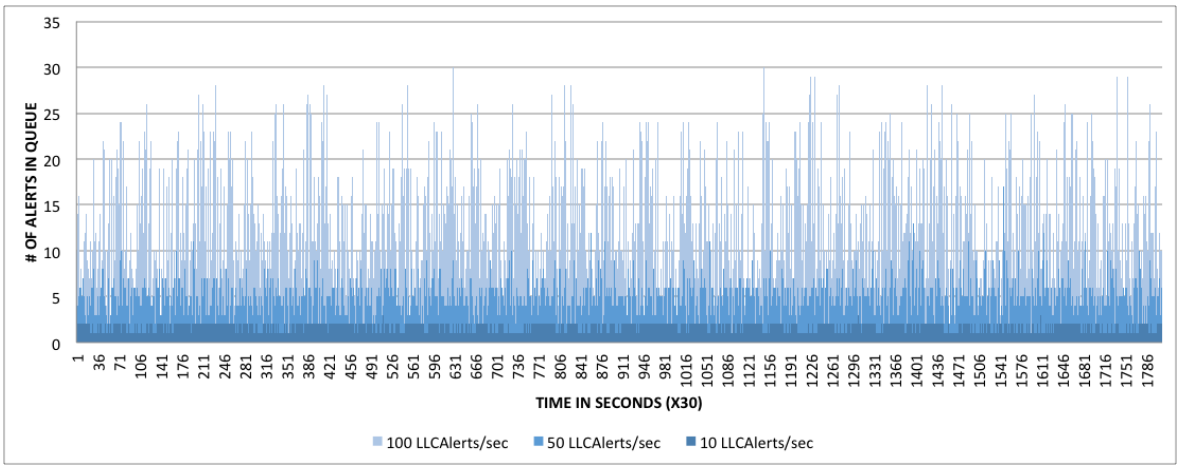

Figure 58 - Previous version: LLCAlerts incoming queue size for 10% matched - 90% unmatched scenario

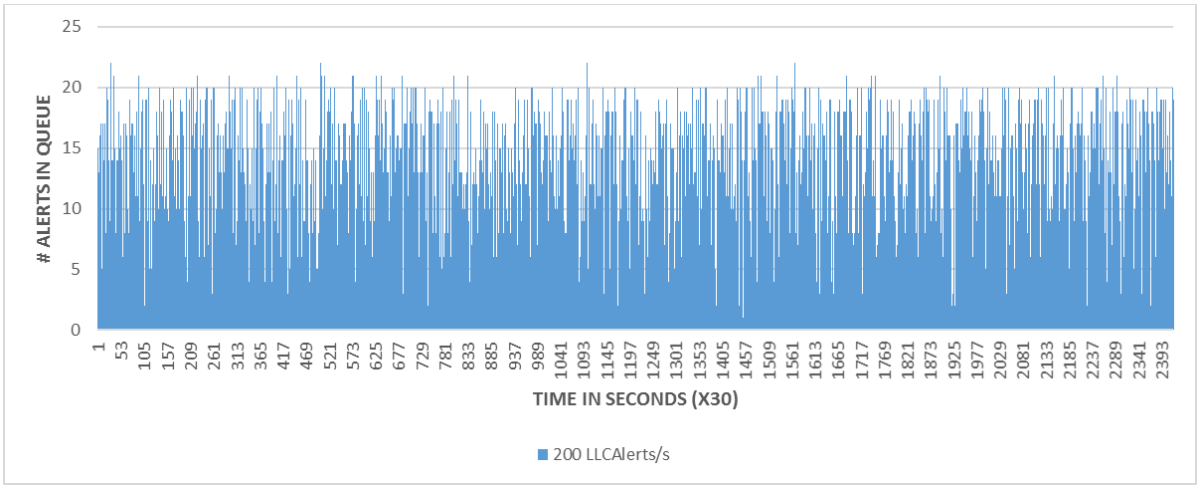

Figure 59 - New version: LLCAlerts incoming queue size for 10% matched - 90% unmatched scenario

Figure 59 shows that, for the higher rate used the input alerts' queue never increases forever, highlighting the fact that the component is not suffering with such alert rate. This was not achievable before (Figure 58).

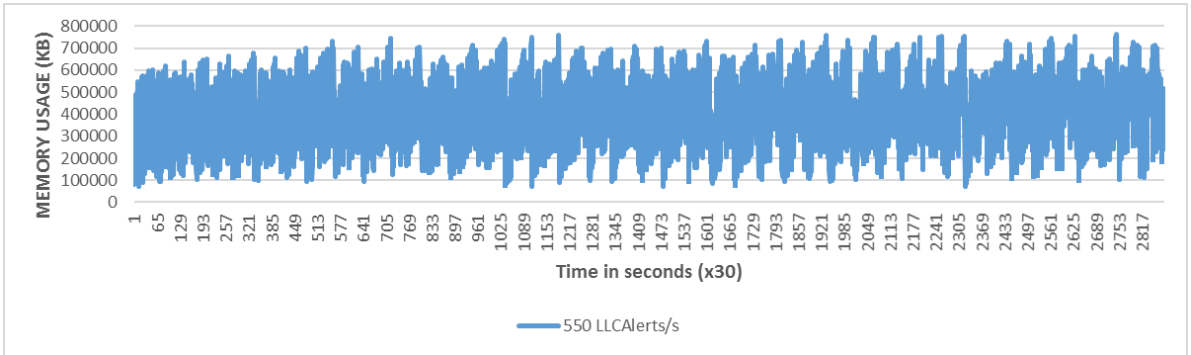

Figure 60 - New version: Heap Memory Space Utilization for 40% matched - 60% unmatched scenario

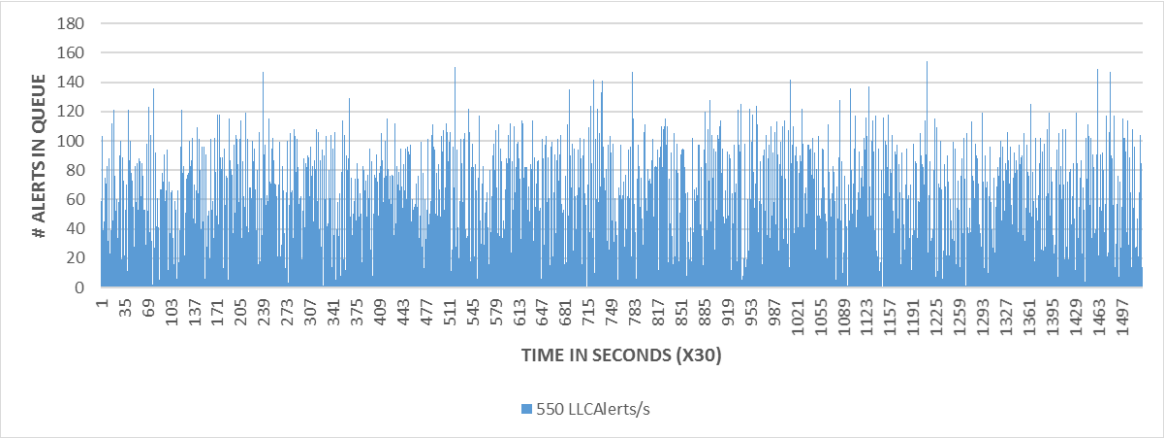

Figure 61 - New Version: LLCAlerts incoming queue size for 40% matched - 60% unmatched scenario

At the end, with this experiment we use a dataset composed by the 40% of matching alerts and the 60% of unmatching ones, and an alert rate of 550 LLCAlerts per second. In Figure 61 we see that the queue is more stressed during this test. However, since it never increases forever QBE can sustain such work load, also without stressing a lot the memory of the component (Figure 60).

We have shown that the component is able to *reactively* carry on the proposed work load. After 220 matching alerts per second threshold (40% of 550) we introduce a delay factor in the processing of the incoming alerts. This factor is linear dependent in the length of the incoming matching alerts sent in burst mode.

However, due to the fact that the component buffers all the incoming alerts, QBE eventually processes all the incoming alerts.

Considering that in the previous version this threshold was ~20 Matching Alerts/s, we reach a huge boost in performances (about 1000%).

5.2.7 Integration in ACEA Emulated Environment

The QBE component has been successfully integrated into the ACEA Emulation Environment. After its integration a lot of tests have been conducted to identify the component effectiveness in such environment.

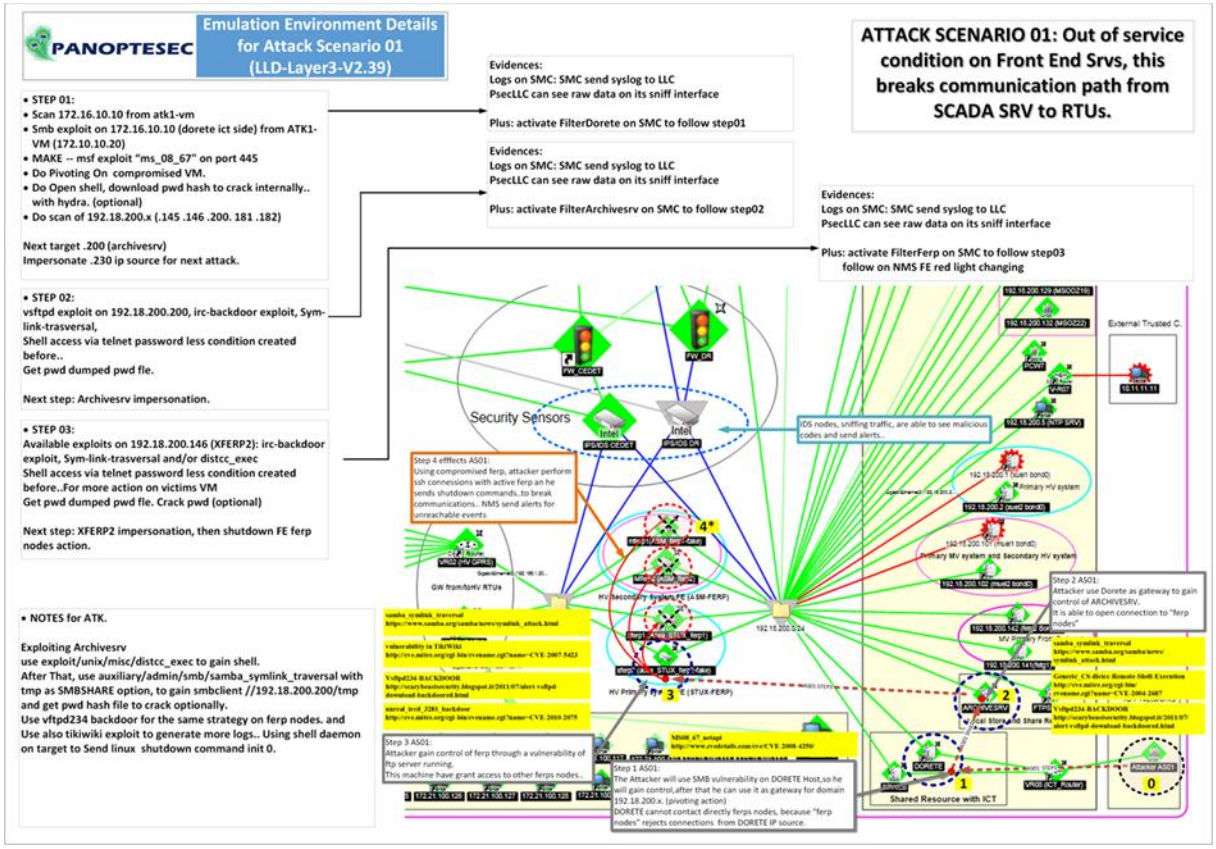

Figure 62 – Representation of the attack scenarios tested in the emulation environment

Figure 62 shows a snapshot of the emulation environment part involved in the attacks performed to evaluate the component effectiveness.

The tests reported below have been performed in date 01/06/2016 using a threshold set to 80%. After this, in order to perform the same tests with the same data and different thresholds, we re-played these ones in offline mode in the stand-alone version. The data have been gathered from the Emulation Environment while performing the attack steps.

Experiment #1:

Steps:

1. From machine 172.16.10.20 to machine 172.16.10.10
2. From machine 192.18.200.230 to machine 192.18.200.200
3. From machine 192.18.200.230 to machine 192.18.200.146
4. From machine 192.18.200.230 to machine 192.18.200.181
5. From machine 192.18.200.200 to machine 192.18.200.181

In this experimentation the AttackGraph used was composed by **39513** paths, and the total number of Syslog messages generated by the emulation environment was **2200**. These messages have been collected and analysed by the **LLC** component which, produced **67 LLCAlerts**. These alerts analyzed by HOC components, leaded to the results listed in the table below:

Table 12 - Number of alerts emitted from QBE

Steps	# of QBE IAPs raised Threshold = 50%	# of QBE IAPs raised Threshold = 60%	# of QBE IAPs raised Threshold = 80%
1	0	0	0
2	3428	124	0
3	5579	907	80
4	7562	1863	372
5	7030	1809	212

As we can see in Table 12, the experimentation has been repeated each time using a different QBE threshold to understand the impact to the other components of the PANOPTESEC system.

Experiment #2:

Steps:

1. From machine 172.16.10.20 to machine 172.16.10.10
2. From machine 192.18.200.230 to machine 192.18.200.200
3. From machine 192.18.200.200 to machine 192.18.200.146
4. From machine 192.18.200.146 to machine 192.18.200.181

In this experimentation the AttackGraph used was the same as the previous experimentation (composed by 39513 possible paths), and also the total number of Syslog messages generated by the emulation environment is near the previous number of Syslogs generated. Exactly, they are **2076**. Also this time these messages have been collected and analysed by the **LLC** component which produced **64 LLCAlerts**. These alerts analyzed by HOC components leaded to the results listed in the table below:

Table 13 - Number of alerts emitted from QBE

Steps	# of QBE IAPs raised Threshold = 50%	# of QBE IAPs raised Threshold = 60%	# of QBE IAPs raised Threshold = 80%
1	0	0	0
2	3428	124	0
3	5579	907	32
4	7562	1863	160

As expected in the proposed scenario, that consists in a less number of steps respect to the previous one, QBE generates less IAPs (Table 13), and even in this case QBE has been able to recognize the complete attack scenario.

5.3 Differences between Automata-Based Engine and Query-Based Engine

The Automata-based approach and the Query-based approach are complementary: the first allows minimizing the generation of false positive Instantiated Attack Paths while the second aims to

minimize the probability of having false negative in the Instantiated Attack Path generation. In the following, we will explain how the two approaches leads to this double effect.

Let us recall that the Automata Based Engine (ABE) generates, for each attack path, a finite state automaton. As a consequence, ABE has to process events (alerts in this case) in the same order as they appear in the automata transitions. Thus, ABE works basically processing sequences (where no omissions are allowed). The immediate consequence is that missing events and notification inversions induce ABE to fail in the detection of an on-going attack.

To limit this drawback, ABE has been designed to tolerate a small number of unordered or missing events. This means that if an instantiated attack path is raised then ABE processed almost all events related to such attack path and in the expected order.

Let us observe that when the Low Level Correlator (LLC) is perfect and events notifications are ordered, ABE does not generate any false positive and false negative. Conversely, when LLC is not perfect or the events stream does not preserve the event generation order, ABE generates a small number of false positive and a lot of false negative.

Contrarily, the Query Based Engine (QBE) does not consider attack paths as sequences but rather as sets of events. Then, when an alert is notified, it is matched against the available attack paths and a similarity metric is computed between the set of delivered alerts and attack paths.

The immediate consequence is that QBE is tolerant to alerts inversion and can tolerate missing alerts. Therefore with respect to ABE, the number of false negative is lower but there is a higher number of false positive (due to the approximate matching and to the unordered processing).

Let us note that, in case of ordered alert notification, if QBE is configured to evaluate only exact matching, we have that the number of false positive generated can be lowered to the case of ABE.

To clarify the situation, let us make an example. Suppose to have the following attack paths:

1. A B C D E
2. B A D E

Then, let us suppose to receive by the LLC the following alerts in this order: B A D E.

To simplify the explanation let us suppose that only one Instantiated Attack Path (IAP) per path is generated. Considering the ABE component, it triggers an IAP related to the second path. QBE instead, triggers two IAPs, either for the first and second path. This means that, if the attack is on the first path, the both components, ABE and QBE, raise a false positive related to the second path. Otherwise, if the attack is on the second path, only QBE raises a false positive. Taking into account this, we can state that in all the considered cases QBE will be able to detect attacks. However, this detection capability has a cost in terms of false positives generated. On the contrary, ABE is not able to detect attacks if the previously specified situations occur, but it raises less false positives than QBE.

Experimental results confirm such intuitions. Section 5.1.7 and Section 5.2.7 report the number of IAPs that ABE and QBE arise in the tested attack scenarios. For what we said before, it is not astonishing that ABE generate less IAPs than QBE. In fact, the adopted configuration during the test executions is the worst we can have. This means that we are using an approximate matching in order to identify alerts and that the events, provided by LLC, are generated in a casual order respect to the attack graph used.

Several tests have been carried out considering different configurations for each component, therefore referring to the average of all the obtained results there are more or less 2700 IAPs arose by QBE against 9700 IAPs generated by ABE.

5.4 Tactical Response Decider

The Tactical Response Decider (i.e. TRD) software component address the challenges defined for the Tactical Response Decision function of the global Functional Architecture of the DRMRS (i.e. Dynamic Risk Management Response System) sub-system as defined in Section 3.3.3 of [D5.1.1], and covers the corresponding established Specialized Requirements as defined in Section 4.7 of [D5.1.1].

Its main objective is to provide to the security officer a comprehensive view of possible Response Plan options, effective to mitigate the ongoing Attack Paths leading to unacceptable risks at a given time, with the associated Operational Impact assessment of each Response Plan on the Monitored System.

In order to achieve its objective, the specified, designed, implemented and tested TRD software component works tightly with the AGG-TRQ component, particularly with its reactive part, which delivers two kinds of Input knowledge, as represented by Figure 63:

- *Levelled Ongoing Attack Graphs*, in response to a Risk Contract, representing the Attack Graph corresponding to Attack Paths leading to risk levels above or equal the Risk Contract;
- *Residual Risk Profiles*, in response to Abstract Response Plans, representing the level of Risks on the Monitored System that may be reached with a corresponding Abstract Response Plan;

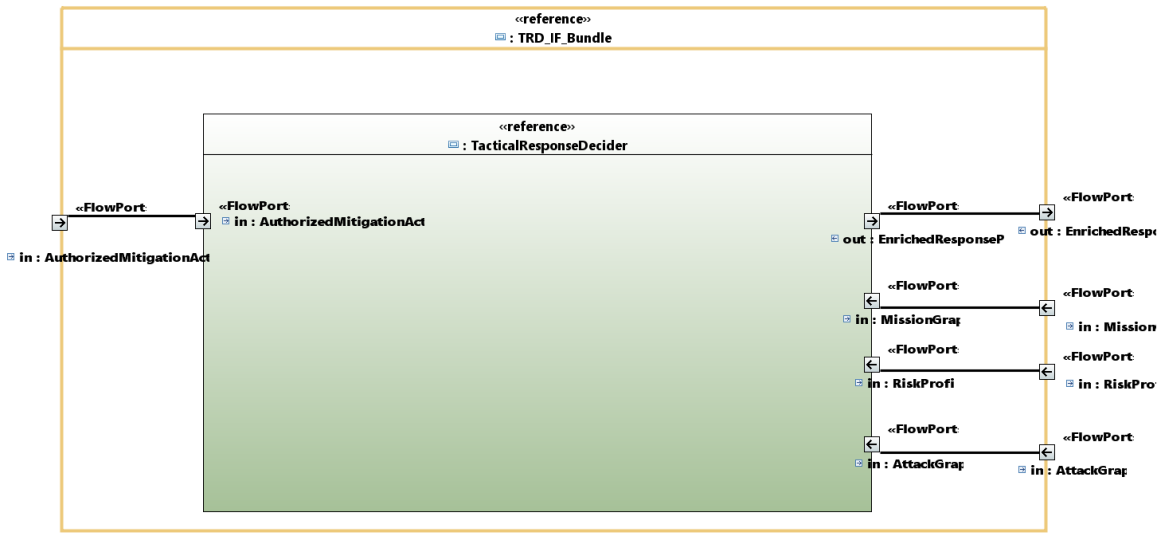

Figure 63 - High-level view of the TRD component design with its various interfaces

To process the Abstract Response Plans relating to each Levelled Ongoing Attack Graph requested for a given Risk Contract, the TRD component must receive a list of the Mitigation Actions that are authorized on the reactive perspective. In the frame of the PANOPTESEC project, the Mitigation Action can be of three abstract kinds on the reactive perspective:

- *Patching*, meaning removing a vulnerability on a device): this kind may be implemented by the deployment of a patch;
- *Filtering*, meaning removing the network access to a remotely exploitable vulnerability on a device: this kind may be implemented by deploying firewalling rules or ACLs on gateways;

- *Quarantining*, meaning completely remove any possibility to access a device: this kind may be implemented by shutting down the device, or isolating it on a specific VLAN;

On another hand, the TRD component also works in collaboration with the ROIA, a sub-component of the MIM component, which provides in response to each computed Response Plans presented on its designed interface:

- *Operational Impact Evaluated Mission Graphs*, which represents the potential negative side effects that the enforcement of the requesting Response Plan may induce on the Monitored System at the assessment time.

Definitely, *Enriched Response Plans* are issued and sent on its Output interface, in order to be further delivered to the Visualization sub-system (i.e. developed in the frame of the Work Package 6) of the global PANOPTESEC system. Administrators may then have the opportunity to analyse them and choose the one to be deployed on the Monitored System. An Enriched Response Plan aggregates for a computed Response Plan:

(i) The computed list of effective Mitigation Actions recommended in the Response Plan to reach a given Risk Contract;

(ii) The assessments of the Residual Risk Profile of the Monitored System, if the effective Mitigation Actions are deployed at the current state;

(iii) And, the assessment of the Operational Impact, if the effective Mitigation Actions are deployed at the current state;

NOTE: Each of those interfaces has been strictly defined with regards to the format of the data and the sequence of messages exchanged during a preliminary design phase of the Work Package 5 components before starting the implementation. As we adopted a light Agile methodology for implementation, the design of the various interfaces have slightly evolved during the 18 Sprints we had for implementing, testing, verifying, refining and integrating the TRD software component.

5.4.1 Contribution

The TRD component computes a set of tactical Response Plans effective to reduce the level of the risk on the Monitored System taking into account the current know status of ongoing attacks along possible attack paths in the Monitored System. The knowledge of this status is given to the TRD through ongoing Attack Graphs.

Unlike other solutions from the literature (see [CUP2006] [CHU2013] [SAM2015]) which proposes also a set of Mitigation Actions (i.e. countermeasures) effective to mitigate an Ongoing Attack Graph, the TRD developed in the PANOPTESEC project finds the minimum set of Mitigation Actions minimizing the total cost of a Response Plan based on the knowledge of the individual cost of each single Mitigation Action. Moreover, while the other solutions in the literature only compute the set of efficient Mitigation Action to mitigate an ongoing Attack Graph, our solution enables minimizing the level of risk on the Monitored System.

In order to achieve this risk minimization, the TRD component works in collaboration with the AGG-TRQ component (i.e. its reactive part) using an interface enabling the TRD to request the status of ongoing attacks inducing risks on the Monitored System (i.e. on each identified Business Processes of the Mission Graph [D4.3.2]) beyond a given level of risks. This risk limit is called a Risk Contract, and is the request to which the AGG-TRQ response with a subset of the global Ongoing Attack Graph (i.e. the Ongoing Attack Graph that induce all levels of risks), that we call the Levelled Ongoing

Attack Graph corresponding to the Risk Contract. Figure 64 illustrate this exchange, the Risk Contract and the resulting Levelled Ongoing Attack Graph.

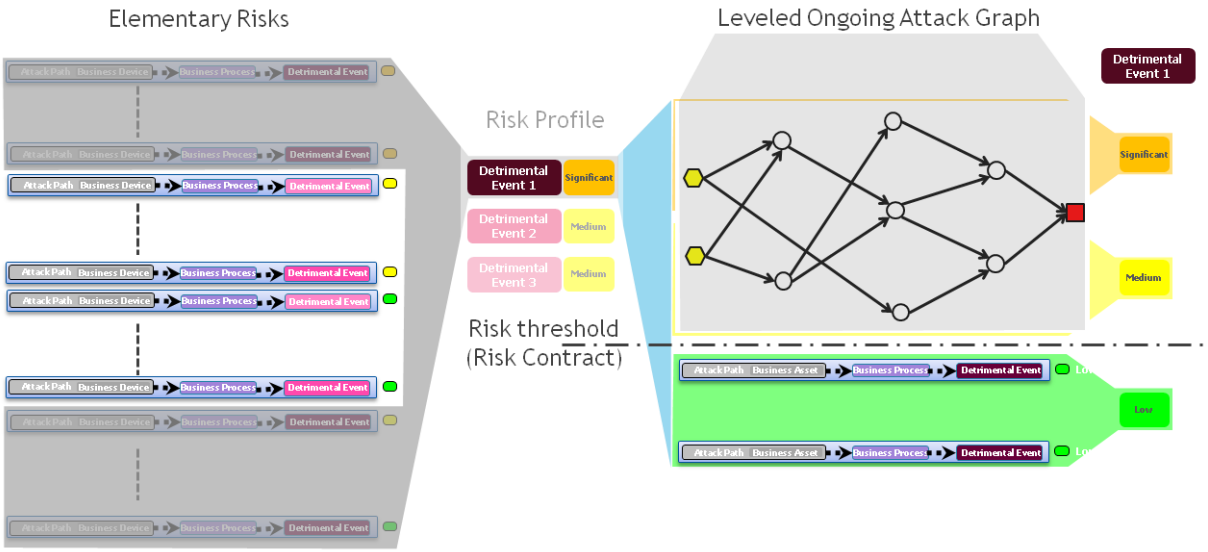

Figure 64 - Extracting a Leveled Ongoing Attack Graph corresponding to a Risk Contract

By requesting the Levelled Ongoing Attack Graphs for several stereotyped Risk Contracts (i.e. one for each level of the Risk Scale used in the PANOPTESEC project: Extreme, High, Medium, Significant and Low), the TRD can determine for which Risk Contract it exists tactical Response Plans mitigating the corresponding Levelled Ongoing Attack Graphs. This computation of the Response Plans is based on transforming Attack Graph in a flow graph and finding the S/T MinCut (see [FOR1956][DIN1970] [EDM1972][GOL1988][KAR1993][STO1997][ORL2013]). The Attack Graph is transformed based on the availability of Mitigation Actions kinds that are authorized on some devices of the Monitored System and the weights of the edges of the transformed graph are derived from the fixed cost of each Mitigation Action on authorized devices. All these information are pushed to the TRD component through an interface delivering Authorized Mitigation Actions.

On the tactical perspective, unlike the classical and generic anti-correlation approach which enables defining any kind of Mitigation Actions, we adopted a simplification approach and only support three kinds of Mitigation Actions:

- Applying a patch on a device;
- Apply a firewall rule on a filtering device to block the remote exploitation of a vulnerability;
- And, definitely, shutting down a device;

Based on those three kinds of Mitigation Actions, the TRD searches up to four Response Plans representing different *mitigation strategies, which are:*

- Response Plan using only Patch kind of Mitigation Actions;
- Response Plan using only Firewalling kind of Mitigation Actions;
- Response Plan using only Shutdown kind of Mitigation Actions;
- Response Plan using a combination of the three kinds of Mitigation Actions.

The TRD component then transforms the problem of finding a tactical Response Plan minimizing the risks on the Monitored System to a problem of transforming an Attack Graph in a flow graph and fixing the adequate weights to perform an S/T MinCut. The complexity of computing a tactical Response Plan is the similar to the computational complexity of the used S/T MinCut algorithm (e.g. $O[V*E*\log(V^2/E)]$ for the pushed-relabed algorithm [GOL1988]).

Because of the way the TRD component requests Levelled Ongoing Attack Graph, it may happen that a computed Response Plan is so efficient that it enables reducing the levels of risk on the Monitored System beyond the related Risk Contract. Another interface has the also been designed between TRD and AGG-TRQ component in order to request the exact levels of risk that is implied by a computed tactical Response Plan beyond the expected Risk Contract. The achieved levels of risk are then the Residual Risk Profile considering a tactical Response Plan.

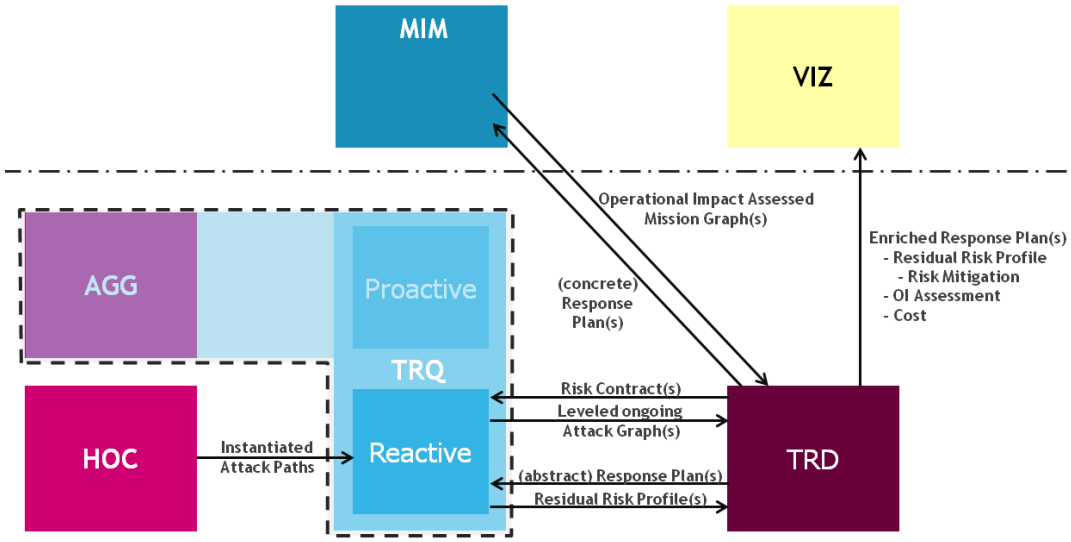

Figure 65 - Communications between TRD and peer components

Also, each of the computed Response Plans are assessed regarding the potential Operational Impact it may induce on the Monitored System, by requesting an Operational Impact assessed Mission Graph to the ROIA (i.e. Response Operational Impact Assessor) part of the MIM component [D4.3.2].

Definitely, the TRD component sends each computed tactical Response Plans for each stereotyped Risk Contract, including the achieved Residual Risk Profile and the corresponding assessment of the Operation Impact of the Response Plan. These Enriched Response Plans are computed on a regular basis, and sent for further analysis and deployment decision to the Visualization of the PANOPTESEC monitoring system. The Figure 65 shows the communications of the TRD component with its various peer components.

5.4.2 Testing and experimentation strategy description

The global testing strategy we implemented for the TRD component software verification and validation relied on two kinds of experimentation datasets and cases:

- Semi-Syntactic data: build from a basic case-study used across all the prototypes we implemented within the PANOPTESEC project;

- Semi-Emulated data: derived from emulated data generated from other component of the PANOPTESEC system, collected on the Emulation Environment of the project (i.e. the Demonstration test bed of the User Partner) and sometimes augmented or modified for the purpose of a test execution;

The Semi-Syntactic kind of dataset had the goal to put the software in controlled conditions in order to test each core feature of the implemented TRD component.

To achieve this objective, we constructed generic case studies, derived from the Syntactic data used for the AGG-TRQ component core features testing. Provided the case studies for the AGG-TRQ were scaled so that an expert were able to check that output results on its various interfaces were the expected one, we build on top of these results the TRD component case studies. By relying on basic AGG-TRQ case studies, we then managed to construct TRD case studies for which an expert could verify that the TRD produces the expect output results. Additionally to the datasets produced by a running AGG-TRQ component for the TRD component, which are several Leveled Ongoing Attack Graphs and several Residual Risk Profiles corresponding to the TRD generated Response Plans on the reactive perspective, several Authorized Mitigation Actions dataset for the various generic case studies were created. Proceeding this way managed to tune the TRD generic case studies as a satisfactory compromise which enabled verifying the implemented TRD core features were working as expected. Definitely, some data sources, supposed to be generated by components outside the DRMRS sub-system, were created completely syntactically according to the purpose of each generic case study: this was the case for the Mission Graph, and also for the Operational Impact Evaluated Mission Graphs corresponding to each generated Response Plan by the TRD component, which are computed by the Mission Impact Module component in the scope of the WP4.

On another hand, we also relied on data derived from the PANOPTESEC Emulation Environment for the testing of the general behaviour of our TRD component, the verification of the correctness of its interfaces (i.e. both behaviour and data formats) and additional experimentation of wide scale.

To achieve this objective, we constructed case studies by deriving the data from datasets collected on the project Emulation Environment that we tuned to enable replaying or enhancing their scale. For this kind of case studies, we injected the datasets in the TRD component with a controlled testing environment that simulated the expected behaviour of the PANOPTESEC Integration Framework regarding the TRD component's input and output interfaces. Additional datasets were needed to perform the perfect simulation of the PANOPTESEC Integration Framework regarding the TRD component. In particular, the Operational Impact Evaluated Mission Graphs corresponding to Response Plans generated by the TRD component during the tests and experimentations were established in collaboration with the MIM component Solution Provider (i.e. Univesität zu Lübeck) by exchanging and executing offline requests build by the TRD component during early phases of testing with Emulated Data.

5.4.3 Individual component V&V

The Verification & Validation (V&V) strategy we developed for the TRD software component consisted in decomposing the verification of the coverage of each Specialized Requirements of the D5.1.1. We defined for each of them a set of features that the component must implement to achieve the defined goals. In these sets of features, we defined two kinds:

- Core features of the component which achieve a goal, or a part of the goal, of a Specialized Requirement;

- Interface features which achieve the input and output communication between the component and the PANOPTESEC Integration Framework;

Then, for each features we established test cases, defining for each a test objective and specifying the expected behaviour or produced results.

Definitely, we sat-up for each test case one or several executable tests, by defining the test conditions (e.g. the version of the software, the test environment) and specifying the testing data (i.e. Semi-Syntactic or Semi-Emulated datasets derived from our defined case studies) to execute a test.

The tests were automated (i.e. scripted) and executed on a dedicated test bed. This test bed was created to stub and simulate the behaviour of the project's Integration Framework, in order to inject the datasets required for each test in a controlled test environment. This permitted to verify the expected behaviour or results of the component during the implementation, but also to produce evidences that the component is working as expected at delivery milestone (i.e. milestone 5 and milestone 6). Definitely, this enabled re-executing the tests each time a new version were produced at each delivery or at the end of a Sprint for non-regression evaluation.

From the twenty-six test cases that we creating during implementation and refinement phases for the testing of our TRD software component, seventeen are related to core features. Over the nine Specialized Requirements defined in [D5.1.1] for the TRD domain, seven were covered by implemented core features in the TRD software components that we verified with several test executions relating to each of the defined test cases. Figure 66 shows the proportion of test cases relating to core features that were verified with test executions. The unverified test cases were related to two Specialized Requirements that we did not covered with our TRD implementation. The status of these Specialized Requirements is discussed in Section 5.4.5.

Note that for each covered Specialized Requirements with an implemented core feature at least two test cases were defined. Definitely, although many test executions were performed during the 18 Sprints of the implementation and refinement phases, we formalized at least two test executions for each verified test cases: one executed at a Sprint Review to demonstrate the correct implementation of an expected core feature; one formalized in the tool used by the PANOPTESEC project to trace the V&V process progression (i.e. [Redmine]).

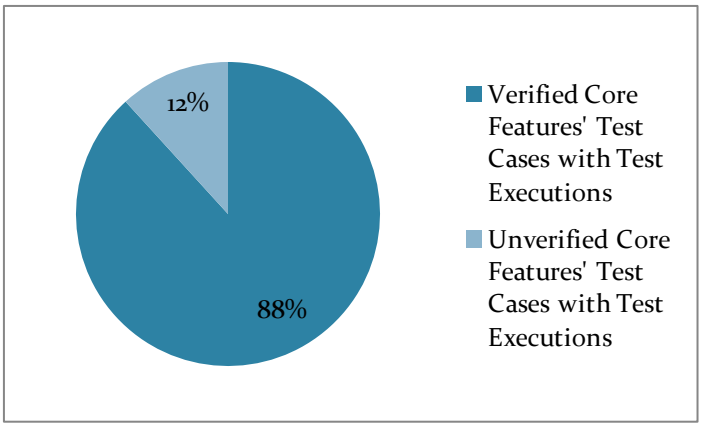

Figure 66 - Test cases relating to core features of the TRD software component verified with test executions

5.4.4 Sub-system integration V&V

For the sub-system integration V&V, we adopted a testing approach similar to the individual component V&V, but focused on interface features of the TRD software component.

We defined tests for each test cases relating to integration features, in order to verify two kinds of parameters implemented by the software component:

- the correct processing of formatted data at the input of the component: we relied on formatted datasets produced by other components of the WP5 or components produced in other work packages (e.g. MIM component of the WP4 for formatted Mission Graph, and DCC of the WP4 for the Authorized Mitigation Actions);
- the behaviour of the TRD software component with regards to the input and output interfaces with the Integration Framework (i.e. the main interfacing middleware of the PANOPTESEC system);

From the twenty-six test cases that we creating during implementation and refinement phases, nine relates to interface features. Over the nine Specialized Requirements defined in [D5.1.1] for the TRD domain, five were covered by implemented interface features that were verified with several test executions: all defined test cases for interface features are verified (cf. Figure 66).

Like the case of the test cases for individual component V&V, we formalized at least three test executions for each verified test cases: one executed at a Sprint Review to demonstrate the correct implementation of an expected interface feature; one formalized in the tool used by the PANOPTESEC project to trace the V&V process progression (i.e. [Redmine]); another one executed on the Emulation Environment of the Project during one of the WP5 integration experimentation sessions.

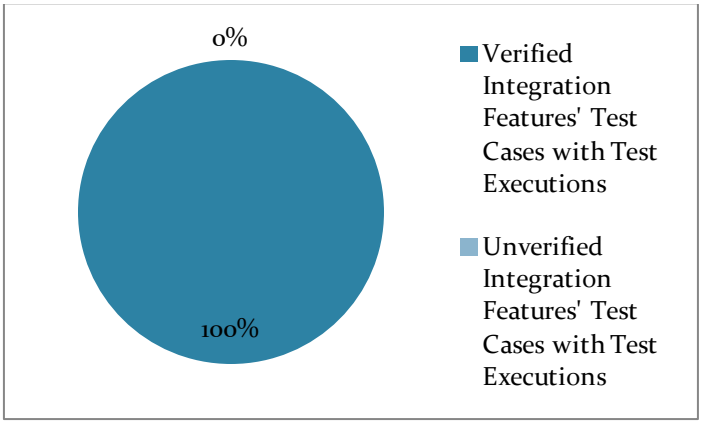

Figure 67 - Test cases relating to integration features of the TRD software component verified with test executions

5.4.5 Uncovered Specialized Requirements

During the refinement phase, some minor modifications of the global design of the PANOPTESEC system were guided by the evolution of the understanding of the needs for such security monitoring system. This had some impact on the need of some of the specified Specialized Requirements of the [D5.1.1], established at the beginning of the project. In particular for the TRD software component, two specific Specialized Requirements were not verified during the V&V process: WP5.TRD.R3 and WP5.TRD.R4. The Table 14 below provide the reasons and an assessment of the potential impact for those two requirements.

Table 14 - TRD Specialized Requirements not verified during V&V process

Requirement		Motivation		Expected Impact
ID	Description	I		
WP5.TRD.R3	The Tactical Response Decision SHOULD sort several lists of mitigation actions (i.e. several tactical response plans), based on their possible impact on the organization.	2	Main Reason(s): The main reason for deprecating WP5.TRD.R3 is that the interface design changed between the TRD, the VIZ (i.e. Visualisation) and the Policy Deployer. There is no possibility to indicate a priority between ResponsePlans proposals. Sorting the computed ResponsePlans based on their Operational Impact is then no more required (or is less important) at the TRD level. Rational: Based on the expressed need of the User Partner in the Project (i.e. ACEA), the TRD should not request to deploy automatically a TacticalResponsePlan to the policyDeployer, but it should ask to a Human Operator of the PANOPTESEC system to choose between several TacticalResponsePlans proposals at the Visualisation level. Recommended action(s): We then should make a change request in order to rewrite this Specialized Requirement or at least lower its importance.	ResponsePlans, while still evaluated in terms of Operational Impact, are not sorted at the TRD level, but they will be sorted at the Visualization Level
WP5.TRD.R4	The Tactical Response Decision MUST provide to any module of the PANOPTESEC system requiring it, the currently computed list of mitigation actions selected as the best trade-off between the risk reduction and the impact on the organisation for the tactical perspective (i.e. the selected tactical response plan).	3	Main Reason(s): The main reason for deprecating WP5.TRD.R4 is that the interface design changed between the TRD, the VIZ (i.e. Visualisation) and the Policy Deployer. Now, all the efficient computed ResponsePlans, assessed regarding their Operational Impact must be transmitted to the VIZ component before a Human Operator decide to enforce the deployment of the one that represent the best trade-off between Risk reduction and Operational Impact. Making this choice automatically at the TRD level is then no more a requirement. Rational: Based on the expressed need of the User Partner in the Project (i.e. ACEA), the TRD should not request to deploy automatically a TacticalResponsePlan to the policyDeployer, but it should ask to a Human Operator of the PANOPTESEC system to choose between several TacticalResponsePlans proposals at the Visualisation level. Recommended action(s): We then should make a change request in order to rewrite this Specialized Requirement or at least lower its importance.	TRD actually sends its computed ResponsePlans to the Visualization for the human evaluation and selection

5.4.6 Additional experimentations

In order to measure the performance and assess the scalability of the TRD software component, we analyzed its behavior compared to several input parameters. Whereas during the V&V process the datasets used was tuned to enable a human expert to verify the correctness of the results or the correct behavior of the software, or derived from the datasets produced on an Emulation Environment to verify that the software was operating as expected within the bound of the use case of the PANOPTESEC project, the datasets here are constructed to put the software component in the conditions of growing scale.

5.4.6.1 Scalability according to the Attack Graph size

For this experiment, we constructed a number of Attack Graphs with growing number of nodes and edges using semi-syntactic data derived from the datasets collected on the Emulation Environment.

We derived several Reachability Matrix datasets with growing proportion of vulnerable devices, from the Reachability Matrix generated on the PANOPTESEC Emulation Environment during a 3 days experimentation workshop organized in ACEA premises between the 30 May 2016 and the 01 June 2016. The initial proportion of vulnerable nodes in the Reachability Matrix was around 12.65% (i.e. 21 nodes over 166 nodes), and we generated Reachability Matrix datasets for eleven additional proportions up to 100% of vulnerable nodes: this means for proportion of vulnerable nodes of 20% (34 nodes over 166), 25% (42 nodes over 166), 33% (55 nodes over 166), 40% (66 nodes over 166), 50% (83 nodes over 166), 60% (100 nodes over 166), 66% (110 nodes over 166), 75% (125 nodes over 166), 80% (133 nodes over 166), and 90% (148 nodes over 166).

By injecting in the AGG-TRQ the different build Reachability Matrix datasets, together with the corresponding other required datasets (Vulnerability Inventory, Mission Graph and Scored Vulnerability Inventory), we managed to generate Attack Graph with growing numbers of nodes and edges. The resulting Attack Graphs were generated on the reactive perspective (i.e. Leveled Ongoing Attack Graphs) using the interface between AGG-TRQ and TRD components.

Measurements for the computation time was then collected in the TRD for the processing of the full Attack Graph on the reactive perspective: meaning when the TRD was processing a Leveled Ongoing Attack Graph requested using a Risk Contract of the lowest risk level in the Risk scale, and whereas no Instantiated Attack Path alerts was already processed by the AGG-TRQ. The measurement taken represents only the computation time to generate Response Plans for the 4 main strategies the TRD is searching for. The Table 15 represents the parameters of the various Leveled Ongoing Attack Graphs generated by the AGG-TRQ, on which we performed our measurements.

Table 15 - Parameters of Attack Graphs used for experimentations on scalability

% of nodes with vulnerabilities in Reachability Matrix	12,65%	20%	25%	33%	40%	50%	60%	66%	75%	80%	90%	100%
Nb of Nodes in corresponding Attack Graph	17	28	36	44	50	61	77	83	91	99	111	135
Nb of edges in corresponding Attack Graph	204	424	724	1017	1297	1722	2660	3235	3938	4466	5544	6307

Figure 68 shows the resulting computation time of the TRD component for the 4 mitigation strategies for a growing number of edges in the corresponding Attack Graph. Similarly, Figure 69

shows the resulting computation time compared to the growing number of nodes in the corresponding Attack Graph.

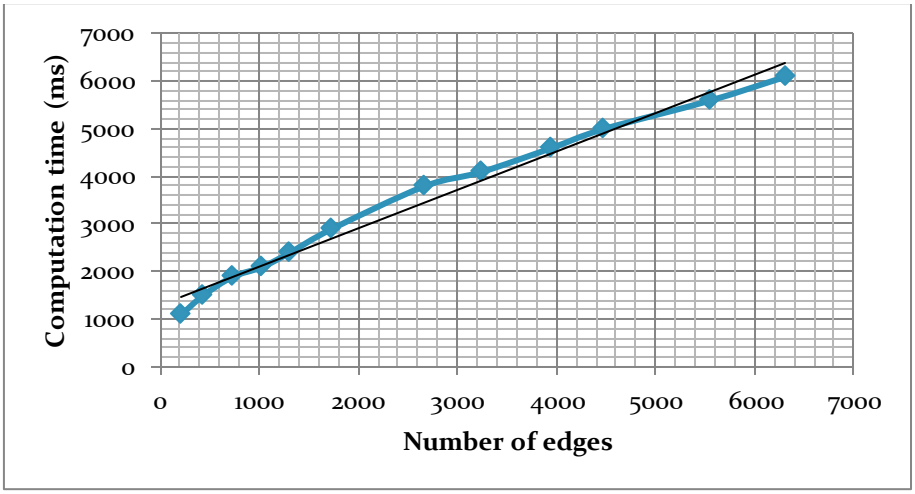

Figure 68 – TRD Response Plan computation time (ms) for growing number of Attack Graph edges

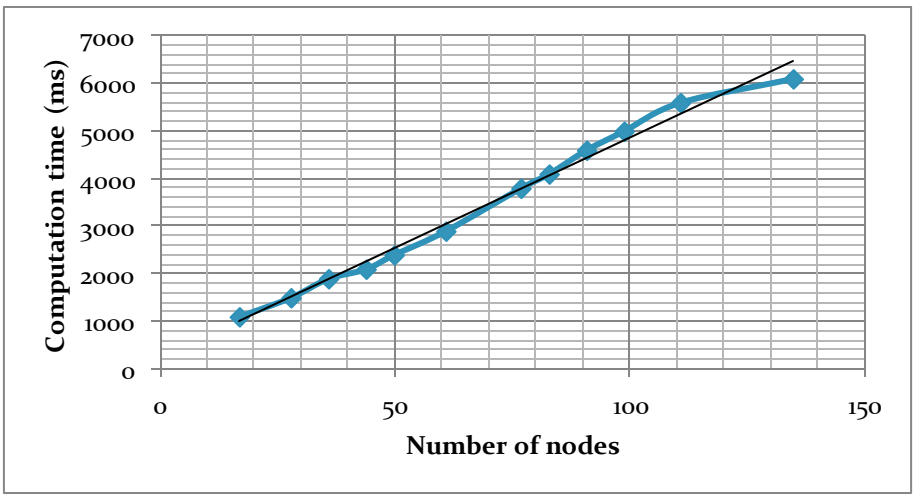

Figure 69 - TRD Response Plans computation time (ms) for growing number of Attack Graph nodes

Note that, for those experiments, the TRD was configured to use an Authorized Mitigation Actions datasets which authorized 825 different Mitigation Actions (i.e. abstract Mitigation Actions) on the Monitored System represented by the Reachability Matrix. Also, using this datasets and each of the input Attack Graphs, our TRD component generated a Response Plan for at least one researched mitigation strategy. Important to note is that the values of measurements were repeated each time one hundred times In order to display on diagrams of Figure 68 and Figure 69 the mean computation time. We managed also to verify that the standard deviation for each measurement point always stayed below 15%. All the experiments has been conducted in a virtual machine on a computer equipped with a i5-2520M CPU clocked at 2.5MHz and equipped with 12Gb of RAM. The virtualization software used was VirtualBox v4.3.8, and was configured to use a dedicated core of the i5 CPU and 4Gb of RAM. The TRD component was executed over a GNU/Linux Ubuntu 1404 LTS distribution configured to use the Oracle JDK version 1.8.

Analyzing the trends in the two diagrams of Figure 68 and Figure 69, we can conclude these experimentations manage to demonstrate that the Response Plans computation algorithm implemented in our TRD component grow linearly with both the number of edges and the number of nodes in the computed Attack Graph.

This performance is particularly remarkable considering that the Attack Graphs derived from the PANOPTESEC use case are dense (i.e. highly interconnected). Indeed, the Attack Graph for a full vulnerable Reachability Matrix is composed of 6307 nodes, which is near to the half of the theoretical number of possible edges for a graph composed of 166 nodes (i.e. 13695). This density supposes that, as a mean each node of the Reachability Matrix is connected to half of the other nodes. This is due to the flat topology of the kind of network used as a use case for the PANOPTESEC project (i.e. a legacy SCADA network). We may expect in a more traditional IT network to have a sparser Reachabiility Matrix, and then Attack Graphs that will probably keep these sparse property. If this is the case, we can expect that our TRD component will be even more efficient in such kind of traditional IT use case and will manage to address larger network topology in a reasonable time.

5.4.6.2 Scalability according to the number of Authorized Mitigation Actions

We also analyzed the performance of the implemented TRD component regarding the size of the Authorized Mitigation Actions dataset it has to address. More precisely, we determined the evolution of the computation time of our core algorithm that computes the search for the four different mitigation strategies, according to growing number of abstract Authorized Mitigation Actions.

Note that, for the TRD component, an abstract Mitigation Action is a possible action that could be enforced on a single device of the Monitored System, whereas the Mitigation Actions in the entry Authorized Mitigation Actions dataset of the TRD uses a more concise formatting that reduced the volume of data sent through the TRD input interface. But, this concise representation is expanded in the TRD component to a list of abstract Authorized Mitigation Actions.

In order to perform this experimentation, we derived from the Authorized Mitigation Actions dataset of the PANOPTESEC emulation Environment semi-syntactic datasets of Authorized Mitigation Actions which correspond to growing number of abstract Authorized Mitigation Actions. Table 16 shows the characteristics of the datasets established.

Table 16 - Number of abstract Authorized Mitigation Actions for various Authorized Mitigation Actions datasets

Number of abstract Mitigation Actions	75	273	613	954	1170	1454

The measurements in this experimentation were collected while the TRD component was processing a full Attack Graph on the reactive perspective, the same way we did it for the experiment reported in Section 5.4.6.1: when the TRD was processing a Leveled Ongoing Attack Graph requested using a Risk Contract of the lowest risk level in the Risk scale, and whereas no Instantiated Attack Path alerts was already processed by the AGG-TRQ. Moreover, for this first campaign, the Leveled Ongoing Attack Graph used was generated using a Reachability Matrix composed of 40% of vulnerable nodes (i.e. 66 nodes over 166), which means the full Attack Graph processed by the TRD component was composed of 50 nodes and 1297 edges.

The measurements collected with those experimentation conditions, are the processing time taken for the computation of the four kinds of Response Plans searched by the TRD component for each abstract Authorized Mitigation Actions sets, and are represented in the Figure 70.

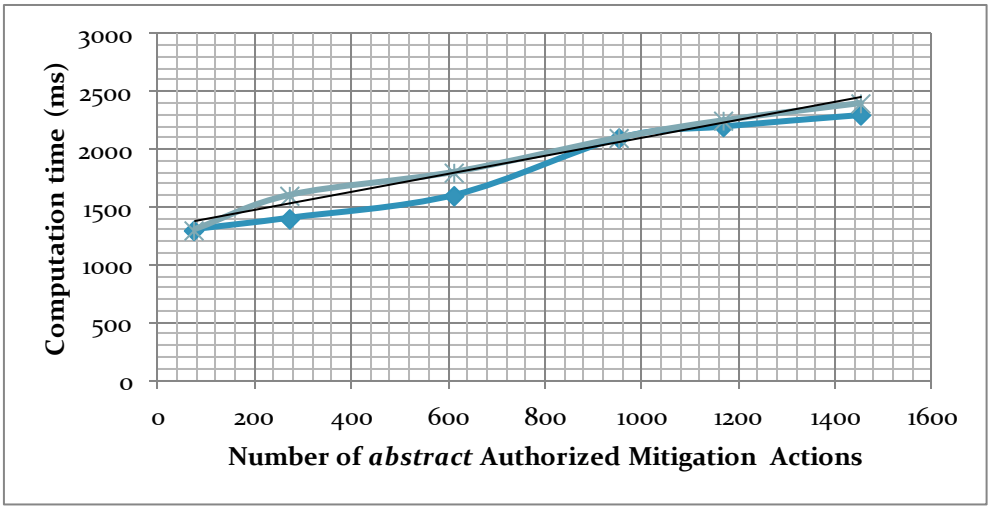

Figure 70 - TRD Response Plans computation time (ms) for growing number of abstract Authorized Mitigation Actions

For this experimentation, two collection campaigns were done. The first one, using an Authorized Mitigation Actions dataset always causing TRD component to generate at least one tactical Response Plan. In Figure 70, the measurement corresponding to this first collection campaign are represented by the curve with diamond points. During the second collection campaign, we decided to modify the Authorized Mitigation Actions dataset in order to avoid TRD component to generate any Response Plan. Results for this measurement campaign are represented in Figure 70 by the curve with cross points.

Like the experimentation reported in Section 5.4.6.1, the measurements were repeated one hundred times each In order to display on the diagram of Figure 70 the mean TRD component computation time for which the standard deviation always stayed below 15%. All our experiments were conducted in a virtual machine on a computer equipped with an i5-2520M CPU clocked at 2.5MHz and 12Gb of RAM. The virtualization software used was VirtualBox v4.3.8, and was configured to use a dedicated core of the i5 CPU and 4Gb of RAM. The TRD component was executed over a GNU/Linux Ubuntu 1404 LTS distribution configured to use the Oracle JDK version 1.8.

From the profile of the two curves, we can draw two conclusions. From the cross point curve, we can conclude that, globally, the computation time of the main algorithm implemented in our TRD component grow linearly when the number of abstract Authorized Mitigation Actions increase, when the whole space of abstract Authorized Mitigation Actions is analyzed. On another hand, the second curve (i.e. with diamond points) shows that the computation may be lower when the whole space of abstract Authorized Mitigation Actions is not sifted through, because the algorithm find appropriate Mitigation Actions which will be used as part of the found tactical Response Plans.

We can then conclude that the cross point curve is the limit above which the TRD component computation time is capped. Which means this curve represents the worst case for our algorithm.

We then exploited this remark in a last experimentation, in order to assess if the scalability of the TRD computation algorithm compared to the growing number of abstract Authorized Mitigation Actions when the size of the Attack Graph grows up. We did several measurement of the computation time of the TRD base algorithm for the same number of abstract Authorized Mitigation Actions, but for Leveled Ongoing Attack Graph generated by the AGG-TRQ with Reachability Matrix composed of growing number of vulnerable nodes: which means, for 40%, 60%, 80% and 100% of vulnerable nodes (cf. Section 5.4.6.1 for corresponding number of nodes and edges in the corresponding Attack Graph).

We took measurements with the same protocol used for all our experimentation (i.e. measurement done one hundred times, values drawn are mean computation time with standard deviation under 15%, same test bed used), and the results are drawn in Figure 71.

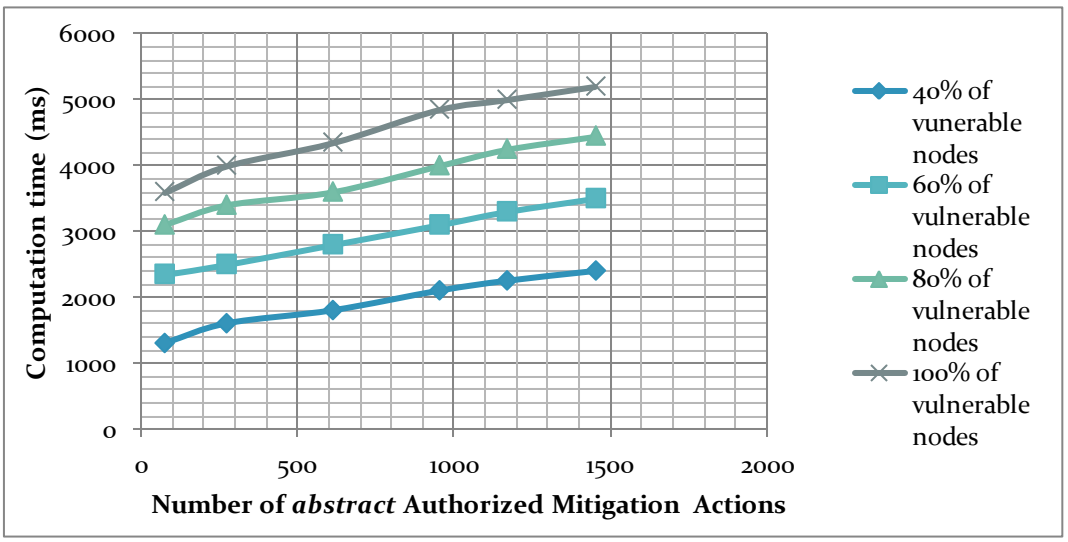

Figure 71 - TRD Response Plans computation time (ms) for growing number of abstract Authorized Mitigation Actions and different Leveled Ongoing Attack Graph sizes

The results in Figure 71, confirms the linear scalability of the TRD computation algorithm with the number of abstract Authorized Mitigation Actions, for various size of the processed Leveled Ongoing Attack Graph. This performance is good provided that the Authorized Mitigation Actions dataset composed of the highest number of abstract Mitigation Actions represent an average of eight to nine different Mitigation Actions per node of the Reachability Matrix. This value should be compared to the PANOPTESEC use case, for which the number of vulnerability per node is limited to an average of three on one third of the devices. Considering our experimentations, we then expect good average performance of the TRD component implementation for reasonable networks' scale probably composed of several thousand of nodes.

5.4.7 Integration in ACEA Emulated Environment

During the refinement phase, several versions of the TRD software component were integrated and experimented on the Emulation Environment of the project (i.e. the Demonstration test bed of the User Partner, ACEA).

Several WP5 integration experimentation sessions were organized, during which we tested and experimented the TRD component with the other software component of the WP5 involved in the reactive treatment chain: HOCs and the AGG-TRQ on its reactive part.

During those WP5 integration experimentation sessions, an attack scenario was executed to produce the datasets required by other WP5 components (i.e. in particular, the two HOCs components) to produce the data used to feed the TRD component with its input datasets. A succinct description of the attack scenario used during the integration experimentation sessions can be found in Section 5.2.7.

Note that, whereas the datasets required at the input of the various WP5 components on the reactive treatment chain were produced by components outside the WP5 (i.e. Mission Impact Model and Data Collection and Correlation modules), the intend of the integrated experimentation sessions were not to experiment or test the global PANOPTESEC system on its reactive perspective.

Those WP5 integration experimentation sessions managed to confirm that the currently integrated TRD software component behaves as expected. Moreover, during the execution of the attack scenario, the TRD component generated tactical Response Plans recommendations that the experts of the Emulation Environment assessed as appropriate.

6 CONCLUSIONS

6.1 Significant results achieved

During the process of researching and producing a beyond-state-of-the-art Dynamic Risk Management Response System (DRMRS), the work package 5 (WP5) Partners adopted an iterative approach for designing, specifying, developing, integrating, verifying and experimenting the software prototype.

A light AGIL process, inspired from the Scum methodology, was adopted during the implementation and refinement phases to manage iteratively the development, integration, verification and experimentation activities. This approach mapped relatively well with the classical applied research process, which is usually based on rapid iteration of prototyping, experimentation and design revision.

The eighteen Sprints scheduled following the design phase allowed the WP5 Solution Provider Partners to achieve the following:

- Production of the different software prototypes composing the designed WP5 sub-system (i.e. DRMRS);
- Testing of the DRMRS prototypes and their individual verification compared to the Specialized Requirements defined during the design, and reported in [D5.1.1];
- Testing and verification of the interfaces of the DRMRS prototypes, both regarding their behaviour and the correctness of the processing of the data formatted as specified;
- A structured V&V process was followed to produce and formalise evidences that the prototypes are working as expected;
- Extensive experimentation were conducted to assess the performance and scalability of each software component of the DRMRS;
- Integration of the DRMRS software prototypes directly on the project Demonstration test bed of the User Partner;
- Testing and experimentation of the integrated DRMRS prototype that enable to verify its software components are working as expected with each others, when attack scenarios (designed in the frame of the work package 7) are executed on the project Demonstration test bed of the User Partner.

Following an iterative approach for all development, integration, verification and experimentation enabled the WP5 Partners to have a DRMRS already integrated, verified and tested as a sub-system at the Milestone 6 of the project, whereas the initial schedule expected to start the integration of the WP5 sub-system at this milestone.

Moreover, the produced components provide significant contribution beyond the state-of-the-art on Response Systems. This assessment is enforced by the papers the WP5 Partners already produced for valuable scientific venues:

- Waël Kanoun, Serge Papillon, and Samuel Dubus. Elementary Risks: Bridging Operational and Strategic Security Realms Elementary, 11th International Conference on Signal-Image Technology & Internet-Based Systems, Thailand, 2015.

- G. Gonzalez-Granadillo, M. Belhaouane, H. Debar, G. Jacob. RORI-based countermeasure selection using the OrBAC formalism, International Journal of Information Security, Springer, 13(1):63-79, February, 2014.
- G. Gonzalez-Granadillo, J. Garcia-Alfaro, E. Alvarez, M. El-Barbori, H. Debar. Selecting optimal countermeasures for attacks against critical systems using the Attack Volume model and the RORI index. Computers and Electrical Engineering, Elsevier, 47(2015):13-34, October 2015.
- G. Gonzalez-Granadillo, J. Garcia-Alfaro, H. Debar. A Polytope-based approach to measure the impact of events against critical infrastructures. Journal of Computer and System Sciences, Elsevier, March 2016.
- G. Gonzalez-Granadillo, A. Motzek, J. Garcia-Alfaro, H. Debar. Selection of Mitigation Actions Based on Financial and Operational Impact Assessments. 11th International Conference on Availability, Reliability and Security (ARES 2016), Salzburg, Austria, August 2016.

6.2 Recommendations

Definitely, even if it was not an expected achievement of the work package 5, some experimentation were conducted between the integrated DRMRS and components of other work packages on the Demonstration test bed of the User Partners. Indeed, some component of the WP5 sub-system need to be fed with input datasets established by work package 4 components: on the reactive perspective, HOC components had to be fed with LLC alerts produced within the work package 4; while on the proactive perspective, the Reachability Matrix, Vulnerability Inventory and Scored Vulnerability Inventory, the Mission Graph and the Authorized Mitigation Actions are datasets established by components of the work package 4.

Nevertheless, the verified and integrated DRMRS still has to be extensively verified and experimented with the other components of the global PANOPTESEC system in the frame of the work package 7. This should prepare the demonstration of its appropriateness, effectiveness and added value for security management of an IT monitored system during a workshop at the end of the project, as scheduled in the frame of the work package 8.

Despite the already published scientific papers, a number of innovations produced within the WP5 have not made the object of publications yet. Some of them, already considered for future papers, should be submitted in the following months. In particular, papers on the Tactical Response Decision, and papers on each perspectives of the DRMRS (or more widely including some work package 4 components) should be envisaged.

6.3 Deliverable validation

The content of this deliverable has been validated at several levels. Internally to the WP5 a validations (i.e. by the Work Package Leader and Deliverable Editor) of the content has been carried out all along the writing process on a periodic basis. Then, a two Review phases QA validation occurred before the publishing of the final version of the D5.4.2 deliverable.

7 REFERENCES

[DoW2015] PANOPTESEC Consortium – *"Annex I - Description of Work"* – European Union Seventh Framework Programme, DoW of the PANOPTESEC project, Grant Agreement no: 610416, 30 Nov 2015.

[D2.2.1] PANOPTESEC Consortium - *"Operational Requirements Analysis"* - Project deliverable D2.2.1, version 2.1, March 27th, 2015.

[D3.1.2] PANOPTESEC Consortium – *"System High-Level Design"* - Project deliverable D3.1.2, Version 2.0, March 27th, 2015.

[D4.3.1] PANOPTESEC Consortium – *"Data Collection and Correlation Integration Prototype"* - Project deliverable D4.3.1, Version 1.0, June 30th, 2016.

[D4.3.2] PANOPTESEC Consortium – *"Data Collection and Correlation Integration Prototype Report"* - Project deliverable D4.3.2, Version 1.0, June 30th, 2016.

[D5.1.1] PANOPTESEC Consortium - *"Response System for Dynamic Risk Management Requirements"* - Project deliverable D5.1.1, version 2.1, March 27th, 2015.

[D5.2.3] PANOPTESEC Consortium - *"Proactive/Reactive Response System Components Prototypes II"*, Project deliverable D5.2.3, Version 1, October 31st, 2015.

[D5.3.3] PANOPTESEC Consortium - *"Proactive/Reactive Response System Components Prototypes II"*, Project deliverable D5.3.3, Version 1, October 31st, 2015.

[D5.4.1] PANOPTESEC Consortium – *"Response System for Dynamic Risk Management Integration Prototypes"* - Project deliverable D5.4.1, Version 1.0, June 30th, 2016.

[D7.4.2] PANOPTESEC Consortium – "*Demonstration System Prototype Report"* - Project deliverable D7.4.2, Version 1.0, *yet to be produced*.

[CHU2013] Chung, Chun-Jen, et al. – "NICE: Network intrusion detection and countermeasure selection in virtual network systems" - IEEE transactions on dependable and secure computing 10.4 (2013): 198-211.

[CUP2006] Cuppens, Frédéric, et al. – "Anti-correlation as a criterion to select appropriate counter-measures in an intrusion detection framework" - Annales des télécommunications. Vol. 61. No. 1-2. Springer-Verlag, 2006.

[DIN1970] Dinic, E. A. – "Algorithm for solution of a problem of maximum flow in a network with power estimation" - Soviet Math. Doll. 11 (5), 1277-1280,(1970)." English translation by RF. Rinehart.

[EDM1972] Edmonds, Jack, and Richard M. Karp – "Theoretical improvements in algorithmic efficiency for network flow problems" - Journal of the ACM (JACM) 19.2 (1972): 248-264.

[Esper] Esper project web page, 2011. http://esper.codehaus.org/

[FOR1956] Ford, Lester R., and Delbert R. Fulkerson – "Maximal flow through a network" - Canadian journal of Mathematics 8.3 (1956): 399-404.

[GOL1988] Goldberg, Andrew V., and Robert E. Tarjan – "A new approach to the maximum-flow problem" - Journal of the ACM (JACM) 35.4 (1988): 921-940.

[GON2014] G. Gonzalez-Granadillo, M. Belhaouane, H. Debar, G. Jacob – *"RORI-based countermeasure selection using the OrBAC formalism"* - International Journal of Information Security, 13(2014):63-79, February 2014.

[GON2015] G. Gonzalez-Granadillo, J. Garcia-Alfaro, E. Alvarez, M. El-Barbori, H. Debar – *"Selecting optimal countermeasures for attacks against critical systems using the Attack Volume model and the RORI index"* - Computers and Electrical Engineering, 47(2015):13-34, October 2015.

[GON2016] G. Gonzalez-Granadillo, J. Garcia-Alfaro, H. Debar – "A Polytope-based approach to measure the impact of events against critical infrastructures" - Journal of Computer and System Sciences, Elsevier, March 2016.

[GON2016+] G. Gonzalez-Granadillo, A. Motzek, J. Garcia-Alfaro, H. Debar – "Selection of Mitigation Actions Based on Financial and Operational Impact Assessments" - 11th International Conference on Availability, Reliability and Security (ARES 2016), Salzburg, Austria, August 2016.

[KAN2015] Waël Kanoun, Serge Papillon, and Samuel Dubus – "Elementary Risks: Bridging Operational and Strategic Security Realms Elementary" - 11th International Conference on Signal-Image Technology & Internet-Based Systems, Thailand, 2015.

[KAR1993] Karger, David R. – "Global Min-cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm" - SODA. Vol. 93. 1993.

[ORL2013] Orlin, James B. – "Max flows in O (nm) time, or better" - Proceedings of the forty-fifth annual ACM symposium on Theory of computing. ACM, 2013.

[PH15] PANOPTESEC Consortium – "PANOPTESEC Project Handbook" – Project internal document, version 0.1, March 27th 2015.

[QAS15] PANOPTESEC Consortium – "PANOPTESEC QA Schedule" - Project internal document, available online at https://gotika.ifis.uni-luebeck.de/panoptesec/WP01/Project%20Handbook/Quality%20Assurance/QA%20Schedule

[SAM2015] Samarji, Léa, et al. – "On the Fly Design and Co-simulation of Responses Against Simultaneous Attacks" - European Symposium on Research in Computer Security. Springer International Publishing, 2015.

[Redmine] http://www.redmine.org/

[Scrum] https://www.scrum.org/

[STO1997] Stoer, Mechthild, and Frank Wagner – "A simple min-cut algorithm" - Journal of the ACM (JACM) 44.4 (1997): 585-591.