MEDINA

Deliverable D4.4

Methodology and tools for risk-based assessment and security control reconfiguration-v1

Editor(s):	Artsiom Yautsiukhin
Responsible Partner:	National Research Council CNR
Status-Version:	Final – v1.0
Date:	30.04.2022
Distribution level (CO, PU):	PU

Project Number:	952633
Project Title:	MEDINA

Title of Deliverable:	Methodology and tools for risk-based assessment and security control reconfiguration-v1
Due Date of Delivery to the EC	30.04.2022

Workpackage responsible for the Deliverable:	WP4 - Continuous Life-Cycle Management of Cloud Security Certifications
Editor(s):	Artsiom Yautsiukhin (CNR)
Contributor(s):	Artsiom Yautsiukhin (CNR) Immanuel Kunz (FhG)
Reviewer(s):	Mika Leskinen (NIXU) Cristina Martínez (TECNALIA)
Approved by:	All Partners
Recommended/mandatory readers:	WP2, WP4, WP5

Abstract:	This deliverable comprises the methodology as well as the prototype implementation of the risk-based auditor component. To follow the approach taken in other tasks, there will be three iterations of the tool integration, an initial prototype, showcasing the methodology, a second release, which will be based on a refinement of the technical architecture and finally the third iteration, which will reflect the implementation of the use cases. This deliverable is the result of Task 4.4.
Keyword List:	Risk Assessment, Compliance, Non-conformity degree, Dynamic risk-based non-conformity assessment
Licensing information:	This work is licensed under Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) http://creativecommons.org/licenses/by-sa/3.0/
Disclaimer	This document reflects only the author's views and neither Agency nor the Commission are responsible for any use that may be made of the information contained therein.

Document Description

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	15.03.2022	Table of Contents	Artsiom Yautsiukhin (CNR)
v0.2	31.03.2022	Implemented comments and suggestions received by consortium partners, introduction to Section 2	Artsiom Yautsiukhin (CNR)
v0.3	08.04.2022	Section 2	Artsiom Yautsiukhin (CNR)
v0.4	13.04.2022	Sections 3 and 4	Artsiom Yautsiukhin (CNR)
v0.5	14.04.2022	Using non-conformity assessment results	Immanuel Kunz (FhG)
v0.6	15.04.2022	Conclusion, Introduction and Executive Summary	Artsiom Yautsiukhin (CNR)
v0.7	19.04.2022	Comments and corrections	Mika Leskinen (NIXU)
v0.8	27.04.2022	Addressed all comments received in the internal QA review	Artsiom Yautsiukhin (CNR) Immanuel Kunz (FhG)
v1.0	30.04.2022	Ready for submission	Cristina Martínez (TECNALIA)

Table of Contents

List of Tables

List of Figures

Terms and Abbreviations

API	Application Programming Interface
DBMS	Data Base Management System
CCE	Continuous Certification Evaluation
CIA	Confidentiality, Integrity, and Availability
CSS	Cascading Style Sheets
CSP	Cloud Service Provider
EC	European Commission
EUCS	European Cybersecurity Certification Scheme for Cloud Services
GA	Grant Agreement to the project
GUI	Graphical User Interface
JSON	JavaScript Object Notation
JSP	Jakarta Server Pages
HTML	Hypertext Mark-up Language
OS	Operating System
RAOF	Risk Assessment Framework
REST	Representational State Transfer
SATRA	Self-Assessment Tool for Risk Analysis
UUID	Universally Unique Identifier
VM	Virtual Machine

Executive Summary

This deliverable reports the initial findings of task 4.4 that is dedicated to the dynamic risk-based assessment and security control configuration. In scope of this task, we develop and implement a risk-based approach for assessment of non-conformity with a selected certification schema. The risk-based approach allows evaluating the deviation from the complete compliance focussing on protecting the most sensitive assets from the likely threats. Such an approach will help the CSP to focus on its concrete needs, justify the distribution of security effort and decrease cyber risks for CSP's customers (e.g., since the customer's trust is one of the assets in our risk computational model).

The deliverable first focusses on the description of the overall approach, which is based on our risk computational model reported in D2.6 [1]. Section 2 describes in the details the methodology for our risk-based assessment which allows automating our approach. Now, the direct human intervention (i.e., the reliance of the tool on manually provided input) is required only before the monitoring. After that, the continuous monitoring phase does not require human to be involved in the loop and the whole process becomes automatic.

In Section 3, we show how the dynamic risk-based non-conformity assessment functionality can be integrated into the MEDINA framework, describing which elements of the framework are involved in the non-conformity assessment process and which data should be exchanged for its execution. We note that the dynamic risk-based non-conformity assessment is implemented as a part of our Risk Assessment and Optimisation Framework, first version of which (focused on static risk-based non-conformity assessment) is reported in D2.6. In this deliverable, although we repeat some concepts from D2.6, we dedicate our attention to the dynamic usage of our risk assessment approach.

Finally, we provide the updated version of the delivery and usage descriptions of our tool, called SATRA, which implements the RAOF functionalities (Section 4). This section is very close to D2.6 (reported only 3 months before delivering D4.4), since it is dedicated to the description of the same tool. Nevertheless, we do our best to focus on the new functionalities and report the most up-to-date information.

This is the first deliverable of task 4.4, covering the first six months. The aim was to develop the core (though, incomplete) methodology for the dynamic risk-based non-conformity assessment and develop a supporting tool implementing it. This approach allows us to start integrating the functionality into the overall MEDINA process, improving its precision and maturity at the same time. The final version of the methodology and supporting tools will be reported in deliverable D4.5 [2].

1 Introduction

The main focus of MEDINA is on the continuous monitoring of compliance with a selected certification schema (and chosen assurance level). During this phase, a cloud service is to be monitored by means of applied metrics and the results of these measurements are used in order to decide whether the service could maintain its certificate, or the certificate should be revoked.

Naturally, once evaluation of every metric succeeds to pass a verification check the certificate is to be maintained. Such situation is ideal, but the real life shows that deviations are frequent. Some of such deviations could be temporary, due to the dynamic nature of the cloud environment (e.g., a new virtual machine could be added and destroyed within a period of several minutes). Other deviations could be long living, but they may relate to a very insignificant asset, which is not secured properly simply because it cannot rise a severe security problem (e.g., non-private data may be sent through an open channel). Although, the later indicates a failure to fulfil strictly all requirements of the selected certification schema, such non-conformities are insignificant and should not lead to the revocation of a certificate.

One way to evaluate non-conformities and to decide which of them could be considered as minor (insignificant for revocation) and which ones should be taken into account seriously (e.g., lead to certificate suspension or revocation) could be to empower the decision making procedure with a risk-based assessment of non-conformities.

1.1 About this deliverable

This deliverable is dedicated to the description of our approach to a risk-based non-conformity assessment and implementation of this approach with a tool in scope of the MEDINA framework.

Our approach is based on the risk computation model reported in D2.6 [1]. On the other hand, the main communication channel used by the static risk-based support described in D2.6 is a GUI, since it is assumed to interact with a human operator. In this deliverable we focus on dynamic, and, thus, automatic operation of our risk-based assessment, which, in its turn, means that human involvement in the process should be minimised. Our dynamic risk-based assessment of non-conformity requires interaction with an operator only in the set-up phase, letting the assessment to be fully automatic during the continuous monitoring.

We should underline once again that our dynamic risk-based assessment is not completely independent, since it uses the same core computational model as the static one uses. On the other hand, this approach ensures that the computation done in both phases is very similar, and that the results of the non-conformity assessment received during the static phase are to be confirmed during the continuous monitoring.

Thus, even though this report has much in common with D2.6, it focusses on the dynamic functionality, and (in some cases) repeats what has already been stated in D2.6 only for providing a complete picture. This deliverable is also linked with other deliverables from WP4 (e.g., D4.1 [3]) as the described assessment is an integral part of the certificate evaluation process.

Finally, we should note that this is the first report of task 4.4, covering the initial six months of its operation. The work done in task 4.4 will continue and the final version of the dynamic risk-based non-conformity assessment approach and supporting tools will be delivered as D4.5 [2] at M30 of the project.

1.2 Document structure

The document is structured as follows. Section 2 describes the methodology for the risk-based non-conformity assessment. This section explains how, when and why this assessment is required for a continuous monitoring of certification and provides the details on how this assessment is realised in the scope of MEDINA. Section 3 describes how the dynamic functionality of Risk Assessment and Optimisation Framework (RAOF) is designed and is integrated into the MEDINA framework. Finally, Section 4 provides the recent updates of the delivery and usage of RAOF.

2 A methodology for risk-based assessment and security control recommendations

This section describes our approach to apply risk-based decision making into the process of a certificate status evaluation. This approach is based on the core risk assessment procedure reported in D2.6 [1] and applies it for the continuous monitoring and evaluation process. The dynamic risk assessment differs from its static counterpart in a number of aspects, including different modelling assumptions, changed of approach the provisioning of the required input and output values, modified workflow in risk calculation, etc. All these aspects will be covered and described in this section.

2.1 Risk-based support during the continuous monitoring of MEDINA

As stated above, the main goal of applying risk assessment during the continuous monitoring phase is to use its results for a more CSP-oriented certificate evaluation. Such an approach is more CSP oriented comparing to many others because it weights the requirements imposed by a selected certification schema with the needs of the CSP (i.e., focusing on protecting the most important assets). Therefore, following a risk-based approach should make the certificate evaluation more flexible, more security focused, and more attractive for CSPs.

Similar to the static risk assessment, dynamic risk assessment model should be based on identification and evaluation of the same three components: assets, threats and vulnerabilities. Moreover, it is required to align static and dynamic risk assessments in a way that they return the same (or very close) results if the same input parameters are provided. In other words, the dynamic and static risk assessments must be based on the same computational model, with the changes that mostly affect the way the parameters are provided and avoid elements which may significantly change the result of the computation. What is important to achieve is the assurance that in case of correct[1] manual provisioning of the input parameters during the static risk assessment, the result (i.e., non-conformity assessment decision) will be the same as during the dynamic risk assessment with input parameters automatically collected by the verification tools.

On the other hand, in contrast to the static risk assessment, the dynamic risk assessment must be automatic, i.e., it should be executed without human intervention. Automation of the risk assessment process requires automatic provisioning of the changing information, its automatic processing and propagation to the decision-making engine. The latter is mostly an implementation problem (instead of displaying the results though GUI, the risk assessment tool should provide it though API). But, the change in the type of the provided information requires re-evaluation of initial assumptions and ensuring their correct processing by the risk computation engine. This step highly depends on the sources of the information.

The required input for the risk assessment is the information about the main assets (or resources) of the cloud (the Target of Certification) and the fulfilment of the security requirements defined by the selected certification schema. It is assumed that these two types of input may change in time (e.g., new VMs could be added to the service or an insecure protocol could be temporary used for transferring data). This information is to be collected by the MEDINA's assessment tools (developed in the scope of WP3), pre-processed and provided to the risk assessment tool via a dedicated API.

[1] Here by "correct" inputs we mean genuine answers provided by the analyst (e.g., compliance manager) which accurately represent the real state of the service (i.e., existing assets and implemented security features).

Unfortunately, some information can only be provided by a human. First, this is the information about the selected certification scheme and the target assurance level. This information is required to set up the risk assessment for further operation and to be provided by the human operator (e.g., compliance manager) before the continuous monitoring starts operating.

Another piece of information which is not possible to collect with automatic means is the sensitivity of the available assets/resources[2]. Since resources could be changed in time, before the continuous monitoring starts the operator is asked to provide sensitivity values for all possible types of resources. These values will be assigned to specific resources depending on their type on the fly.

The result of execution of our computation model (and supporting tool) is the assessment of non-conformity, which could be either *major* or *minor*. In other words, risk assessment should be used only if a non-conformity is detected. The results of the risk-based non-conformity assessment are provided to the engine responsible for evaluation of the certificate, which will make its decision using its internal logic about the state of the certificate (continue, suspend, revoke, etc.).

2.2 A methodology for risk-based assessment during the continuous monitoring of MEDINA

Our methodology for a risk-based non-conformity assessment during the continuous monitoring is grounded in the basic risk assessment computational model described in D2.6 [1], and has the main focus on pre-processing and preparing the required inputs and specific usage of the mentioned computational model. Thus, in this deliverable we are not going to repeat the basic functionality of the model but focus on the methodology for its usage during the continuous monitoring. The main elements and steps of this methodology are indicated in Figure 1.

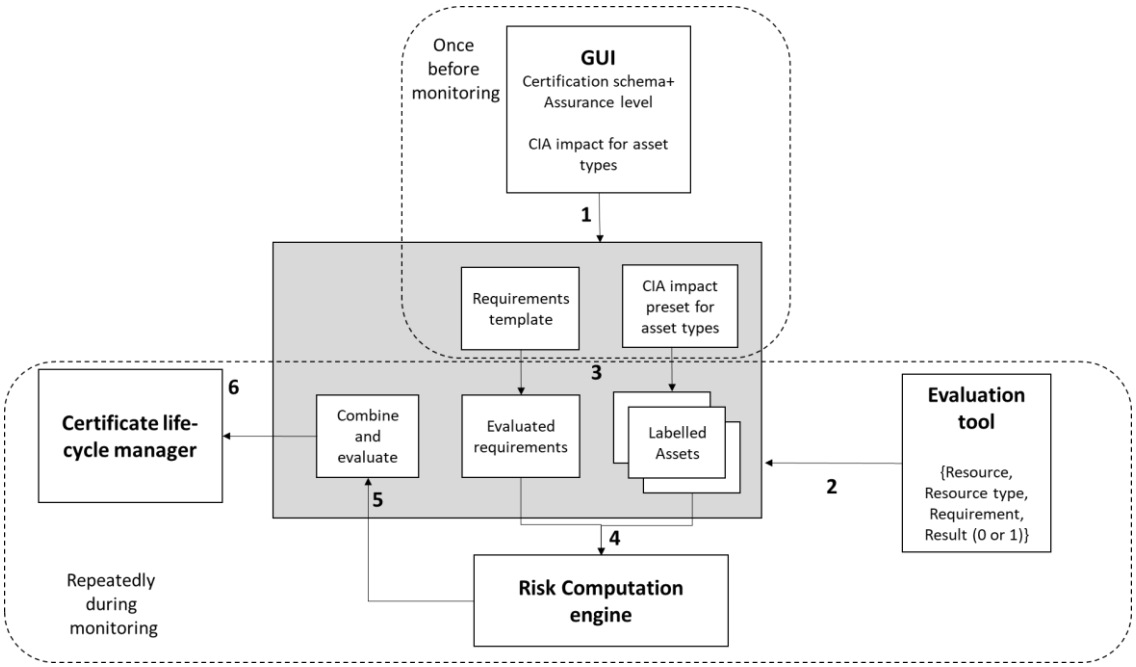


Figure 1. Dynamic risk-based non-conformity assessment workflow

[2] In the scope of the considered task assets are the cloud resources; and, therefore, the terms “asset” and “resource” are used interchangeably in this document.

The input is provided during two different phases: before starting the continuous monitoring (done once) and during this phase (periodically).

Step 1. Set up the model/tool for continuous usage

The first step of the methodology is to set up the model for its usage during the continuous monitoring. By "setting up the model" we mean setting up the elements of the model related to the selected certification scheme and selected assurance level, such as requirements and controls. This is required because within the scope of MEDINA, the user is allowed to select a scheme to comply with.

In particular, regarding to the risk computation model, we should set up the following parameters according to the selection of the user:

- a list of requirements (defined as *R* in D2.6)
- the associated list of Boolean values *RV* of the same size, which denotes fulfilment (1) or not fulfilment (0) of the corresponding requirement
- a list of controls *C*
- the mapping matrix ***RC***, in which every element denotes the degree up to which a requirement *r* contributes to control *c*
- the mapping matrix ***RT,*** with cells denoting the probability for a security control *c* to prevent a threat *t*
- weight vectors $W_{C'}$ and $W_{C''}$ denoting the degree to which management controls affect the power of the controls directly preventing threats from occurrence.

All these values should already be defined in our supporting tool and do not require any modifications from the user. The user should only select the certification scheme (and assurance level) and the tool will use the appropriate values automatically.

The second part of this step is to pre-set the impact values for assets. Since it is assumed that the main resources of the service may vary in time, the expected impact values could be set up only for asset types[3] (e.g., Virtual Machine, Database, etc.). Once the values are set up, the risk assessment tool is able to assign these values to concrete resources once they are detected and reported to the risk assessment engine. This operation can be done automatically during the continuous monitoring phase.

In short, for all asset types *AT*, we ask the user to define the estimated impact in case Confidentiality, Integrity or Availability is compromised, obtaining vectors AT_C, AT_I, AT_A, which are to be used in the next steps to define vectors A_C, A_I, A_A for concrete resources, vectors which are required as input by our computation model.

Step 2. Provisioning of the monitored data

Risk assessment is periodically used to assess the detected non-conformity. Obviously, if no deviation from the certificate requirements is detected, conducting the risk-based assessment is redundant.

Once a non-conformity is detected, the required input parameters should be provided for risk assessment. We are not going to discuss how this information is collected, as it is the topic to be considered in other MEDINA deliverables such as D4.1 [3] / D4.2 [4]. The main information

[3] The asset types used for risk assessment in scope of the MEDINA project are the ones taken from the ontology of cloud resources defined and supported by Clouditor (an integral part of MEDINA).

required, and which is going to be provided as an input for risk assessment is a set of the tuples[4] containing the following information:

- Resource ID
- Type of the resource
- Requirement ID
- Requirement evaluation status: fulfilled (1) or not fulfilled (0).

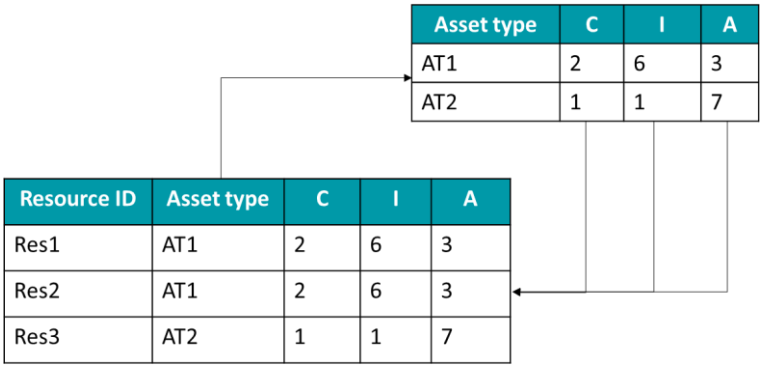
Using this information, the engine is able to identify a set of requirements $R^r \in R$ evaluated in relation to a reported resource r.

In addition to this information, the evaluation tool provides the information about the considered service (UUID).

Step 3. Preparation of input parameters for risk assessment

Once a request for a risk-based assessment of non-conformity is received, the provided input should be pre-processed for the computational engine to perform the assessment. The following actions are to be performed to prepare input parameters for risk assessment.

First, a list of assets is formed. For every reported tuple, distinct Resource IDs are extracted which form the list of assets *A*. For every Resource, the associated asset type is used in order to retrieve the pre-defined impact values (AT_C, AT_I, AT_A) and assign the corresponding impact types in case confidentiality A_C, integrity A_I, and/or availability A_A is violated.



Asset type	C	I	A
AT1	2	6	3
AT2	1	1	7

Resource ID	Asset type	C	I	A
Res1	AT1	2	6	3
Res2	AT1	2	6	3
Res3	AT2	1	1	7

Figure 2. Assigning impact to discovered resources using pre-set values for asset types

The core difference with the static risk assessment approach is that using monitoring functionality, it is possible to identify how certification requirements are addressed for every resource. We remind that for the static risk assessment fulfilment of requirements is assessed for all resources together. Considering different security features applied to different resources can be also done for static assessment, but then, it would require answering a long questionnaire for every resource separately, which makes the service hardly usable. On the other hand, such a tedious operation could be performed by a tool with automatic input provisioning.

With the inputs provided for analysis, it is possible to perform a risk assessment for every resource separately, considering satisfaction for every resource. Thus, we are able to sense different risks in case one virtual machine has a malware protection and another one does not.

Unfortunately, it is often impossible to obtain information about the assessment of *all* requirements for every asset. This may happen because of many reasons: some requirements

[4] A tuple is a finite sequence of elements.

may have no assessment methods and metrics for assessment (i.e., the monitoring system cannot neither confirm or disprove satisfaction of some requirements); some metrics could not be used by a CSP; some requirements should be measured for different resource types (i.e., the strong encryption of the communication channel cannot be checked for a database).

There could be different approaches on how to determine the values for the requirements which are not directly measured for a resource; but at this stage we head for the simplest scenario, which allows focussing the analysis only on the most relevant, i.e., measured requirements. Our initial approach is as follows: we take the values for measured requirements per resource as they are (i.e., assign the requirement status value) and assume other requirements as satisfied (i.e., 1). This approach is described in Figure 3. In the future months of the project we are going to investigate more sophisticated and accurate ways of determining these missing values (they will be briefly discussed in Section 2.3).

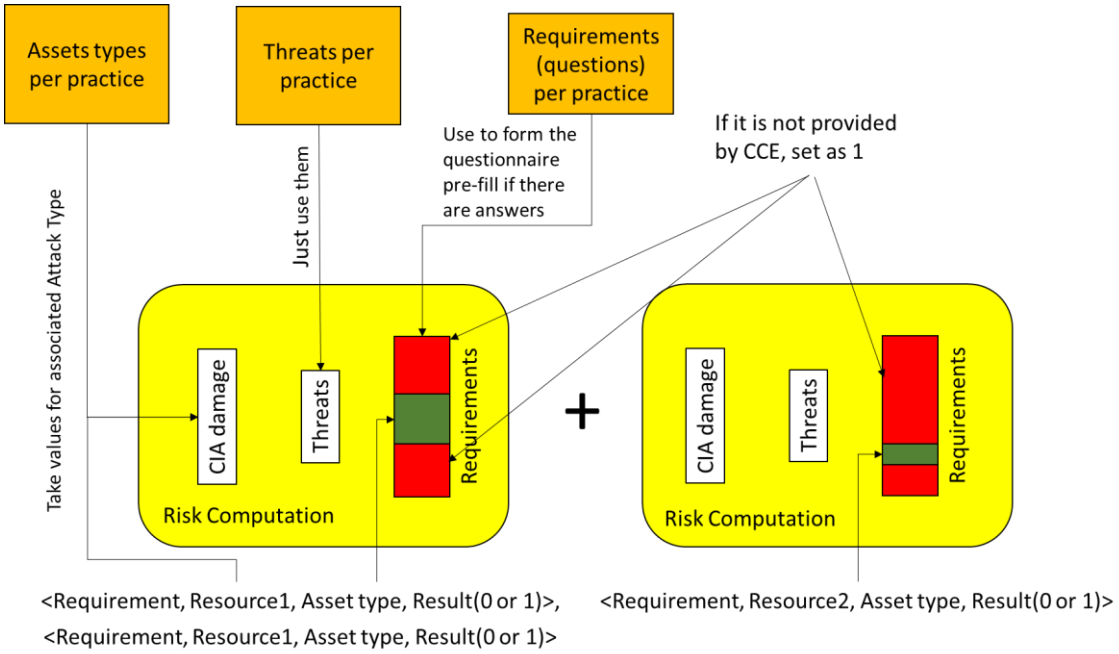


Figure 3. Computation of risk during the dynamic risk assessment

Now, in order to define the required list of requirement values *RV*, we identify those requirements which have the reported value and assign it, leaving others as 1 ("satisfied").

Step 4. Risk computation per resource

Now, we have the list of requirement values *RV*, and the impact values in case confidentiality A_C, integrity A_I, and/or availability A_A is violated for every resource. This is enough to execute risk assessment for every resource separately using the computation model described in the D2.6.

The result of the computation is a list of risk values *R* for every threat and total risk (*Risk*).

Step 5. Combine risks and evaluate non-conformity

Once risk for every resource is computed, it is possible to aggregate these values for the whole service. Since the risk levels are the expected losses, i.e., monetary values, it is possible just to sum up the corresponding values. Finally, we obtain the real risk value ($risk_{real}$) for a service.

The next step is to assess a non-conformity degree. In order to compute it we need the *ideal* risk value ($risk_{ideal}$), which can be computed in the same way as it has been described above, but assuming that all requirements are satisfied for all assets. Since now all requirements are considered as satisfied independently of the considered resource, this simplifies the overall computation: there is no longer need to consider every resource separately, but just consider all resources together, as it is defined by the core computation model. Thus, for computing ideal risk value it is required to compute the risk level only once for the whole service.

Finally, the degree of non-conformity is computed as follows:

$$10*log_{10}(Risk_{real}) - 10*log_{10}(Risk_{ideal}) < threshold$$

The usage of logarithm is required to transform the values into an interval [0;100] (rare values higher than 10^{10} are mapped to 100), which is often seen as more suitable for evaluation by a risk or compliance manager than an absolute value.

The difference (which also can be seen as the ratio of absolute values) now can be compared with a threshold. If the difference exceeds the threshold, the non-conformity is considered as major. If it is less than the threshold, then the non-conformity is minor.

Step 6. Using non-conformity assessment for the decision about the certificate status

The identification of minor and major non-conformities is essential to make decisions about the maintenance of a certificate.

The EUCS [5] defines several certificate maintenance cases, most importantly including the continuance, the suspension, and the withdrawal of a certificate. Additionally, a certificate can be renewed or updated, for example when certain information, like its expiration date, change. Note that the suspension of a certificate means a temporary state where the CSP can still do remedial actions, while the withdrawal of a certificate is a final decision.

The most important decisions with respect to potential for automation are therefore the continuance and suspension of a certificate. This importance stems from the fact that in an automated, continuous process, these are the most common decisions to be made and their automation would therefore replace a considerable amount of manual auditing.

Future deliverables will discuss the different parameters which can be considered to make the decision that a cloud service “does not fulfil the requirements anymore” [5]. Of these parameters, the level of security risk that exists in the cloud service is a critical one: If only a minor non-compliance or no non-compliance is identified, the certificate can be continued. If, however, a major non-compliance is identified, it can be said that there is a significant deviation between the requirements and the cloud service’s security and the certificate should be suspended.

In future work, we will also investigate other maintenance cases regarding their potential for automation, for instance the complete withdrawal or the renewal of a certificate due to a change in its assurance level.

2.3 Future actions

The proposed methodology is the initial step in defining the dynamic risk-based non-conformity assessment procedure. There are a number of ways of how this procedure could be improved.

First, in the future months we are going to investigate more in depth how requirements for every resource could be assessed. Right now, we consider as satisfied (assign value 1) all requirements

which have not been directly evaluated. This approach allows us to focus only on those requirements which are the primary for the considered resource. On the other hand, risk is very difficult to localise, as many aspects are interdependent. Moreover, some of the requirements are generic for the system (e.g., various procedure or organizational practices) and, although cannot be directly attributed to one resource affect the system as a whole.

Several approaches are possible to cope with this problem. One possibility is to re-use manually the provided initial settings in case no objective/measured information is available. In this case, the dynamic risk assessment may use the objective information only when it is available and use the manually provided inputs for the rest (assuming that there is no better solution). With this approach we put more burden on the shoulder of a compliance manager, making static risk assessment obligatory (not optional, as it is now). Moreover, now the computational process becomes much dependent on the integrity and correctness of the manually provided values.

Another possibility is to investigate further dependencies between different resources (as well as between requirements). For example, a malware protection implemented on a virtual machine should protect all resources running (or stored) on this VM. Such dependencies could be taken into account either implicitly by the definition of metrics (e.g., a metric for a resource inherits a requirement value from another related resource), or explicitly by definition of a resource dependency model and defining a special mechanism for inheriting requirement values through this model. This approach much depends on the capabilities of the assessment tools and metrics definitions. In the future months we are going to investigate these possibilities.

At this point, we consider all assets belonging to the same asset type as the same (from the impact point of view). Our tool is capable to go further and consider different sub-groups for the same asset type. This could be useful if resources belonging to the same asset type could have very different sensitivity levels. For example, some VMs could run important processes or store sensitive information, and others could serve only as temporary infrastructure for secondary operations. In this case, the first set of VMs are of high importance, while the others should have low impact values. The current approach does not allow to distinguish such cases, even though the computational model and the supporting tool have the means to treat the resources differently. At this moment, we do not use the full power of our risk assessment approach because of the difficulty to perform such categorisation of resources with the automatic means provided by the partners.

It is also important to note that the project has several use case providers who will help us to evaluate the improvement options also from the usability perspective. In other words, although, in theory some features could be found important to develop, some of them could be difficult to follow in practices (e.g., require too much involvement of a human operator) or could bring only marginal benefit (from a practical point of view). Thus, the possible improvements for usage of our tool will be evaluated also from usability point of view and, probably, additional features will be added.

Last, but not least, we provide a couple of words about another direction related to Task 4.4, which is the optimization of reconfiguration options. In short, next to the assessment of a non-conformity, we plan to support the CSP with recommendations of how to improve their system in the best way. This line of work has not yet been started, as it is planned for the future months.

3 Implementation

The dynamic risk-based non-conformity assessment methodology is implemented as a part of the Risk Assessment and Optimisation Framework (RAOF). It utilises the same risk computation engine used for static risk assessment but focusses on automatic processing of input data and provisioning them further for a more comprehensive decision on the certification status.

3.1 Functional description

The RAOF implements a service for a quick and simple risk assessment, which is used as a background for the assessment of non-conformity. Although, the static risk assessment can be used as a standalone preparation tool, the dynamic risk assessment is an integral part of MEDINA.

For the dynamic risk-based non-conformity assessment, a GUI is used just before the start of the continuous evaluation in order to select a certification scheme (and the associated assurance level) (see Figure 4) and pre-set impact values for asset types (see Figure 5). After providing these settings, the tool is ready for dynamic risk-based non-conformity assessment.

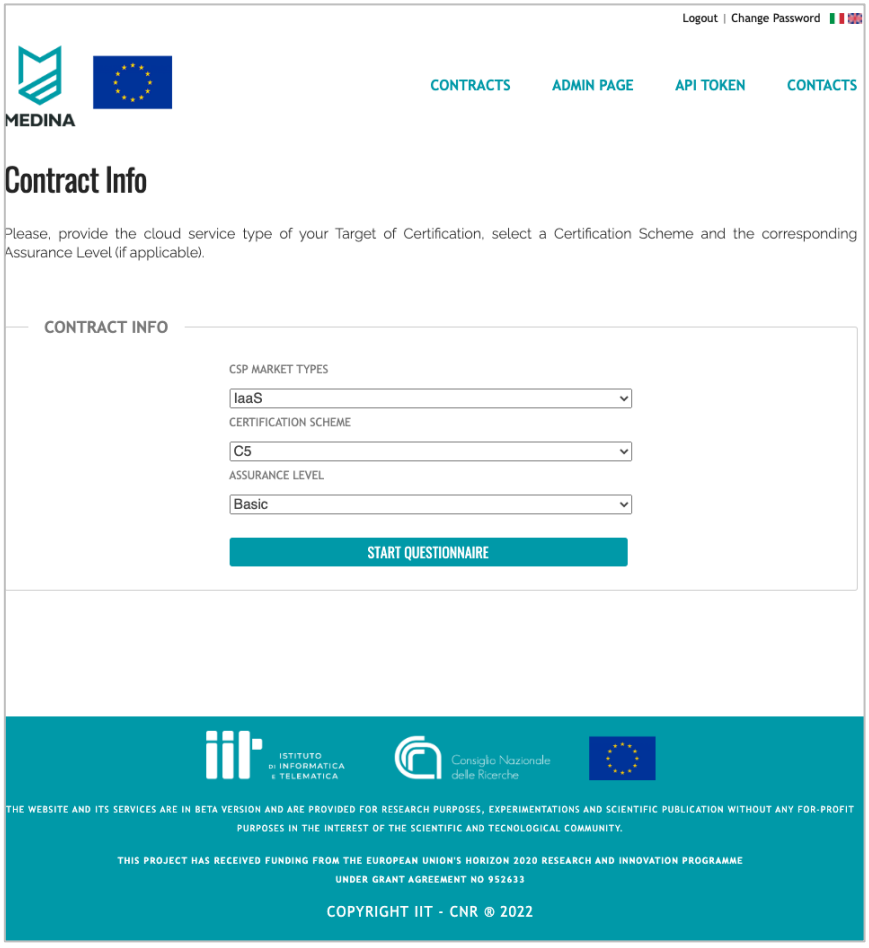


Figure 4. Selection of a certification scheme and associated assurance level

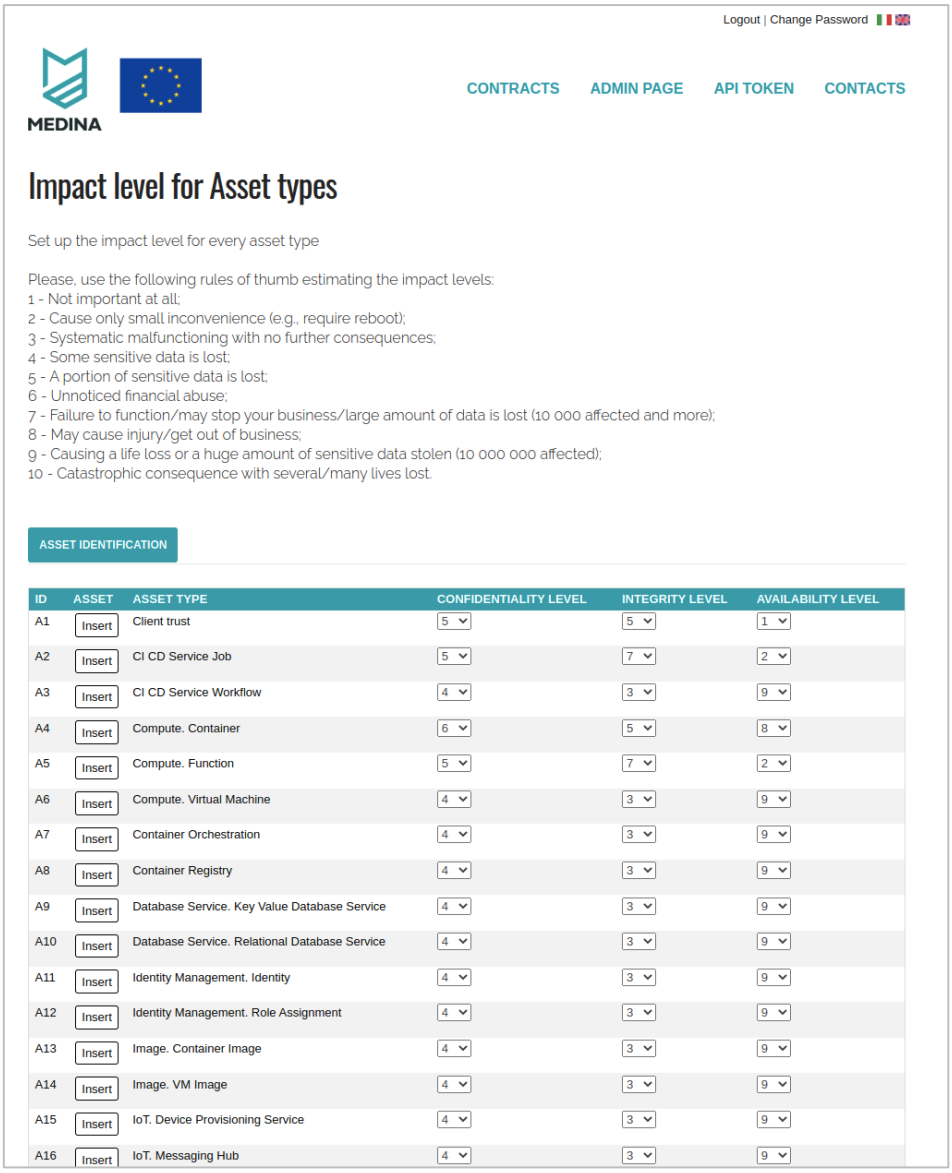


Figure 5. Setting up impact values for asset types

During the continuous monitoring phase, the tool periodically receives the required input data through a predefined API, performs the required assessment according to the methodology described in Section 2, and sends the result of the assessment (major or minor result) further. It should be noted here that the tool does not interact with a human operator at this phase and, thus, does not use a GUI.

It should be noted, that although the main goal for the tool is to evaluate the degree of non-conformity, the tool also may provide the calculated risk level. For example, this information can be stored with the collected evidences or in the log of certification evaluation results. This information can be provided to the user through a compliance dashboard or using other means for browsing such information. This possibility will be analysed and evaluated in the future months of the project.

Deliverable D5.1 [6] defines the following set of requirements for the dynamic functionality of our framework:

Requirement id	RBCA.01
Short title	Dynamic risk assessment
Description	Timely adjust the CSP's risk profile and re-evaluate efficiency of security configuration
Implementation status	Mostly implemented

The framework is set up to automatically re-compute CSP's risk profile and use it in order to assess non-conformity of the current security configuration. As it has been underlined above, this is the first version of this functionality and we will look for more advanced approaches to make it more accurate.

Requirement id	RBCA.02
Short title	Interface to the continuous evidence management tools
Description	Requires to consume the current status of the system configuration to re-adjust risk profile.
Implementation status	Implemented

The framework has an interface for consuming continuous evidence input. The concrete form of the input depends on the evaluation module, but the main information required for re-adjustment of the risk profile is as it is reported in Section 2.2.

3.1.1 Fitting into overall MEDINA Architecture

During the continuous monitoring phase, real measurements are collected by various assessment tools, and various metrics are used to evaluate if the requirements of the selected certification scheme are fulfilled by the considered service. Then, a decision on whether to maintain or revoke the certificate is made. The RAOF takes part in this process and provides its assessment of the detected non-conformity using risk assessment (as it is described in Section 2). Figure 6 focalises the part of the overall diagram of MEDINA on the considered functionality and Figure 7 shows the interfaces used in the dynamic risk assessment.

Figure 6. A part of the overall MEDINA diagram related to the dynamic risk assessment

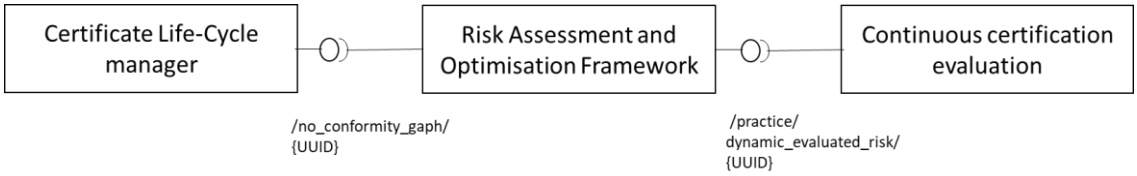


Figure 7. Components and interfaces used in the dynamic risk assessment process

The Continuous Certification Evaluation (CCE) component collects all the evidences available at some moment of time and sends them to the RAOF through a dedicated API (*/practice/dynamic_evaluated_risk/{UUID}*) providing the input for the further risk-based assessment as a JSON file (see Appendix). This file contains a lot of information about the assessment, but the RAOF extracts from every "evaluationAnswers" only the following information:

- resource.id
- resource.resourceType
- requirement.name
- requirement.conformant (status)

This information (together with UUID) is enough to conduct the risk-based non-conformity assessment. The result of this assessment is sent to the certificate Life-Cycle Manager (LCM), where the decision about the state of the certificate is made using */non-conformity_gap/{UUID}* API and sending a simple JSON file, as follows:

```
{
    "certificateid": str(certificate_id),
    "majordeviation": True,
    "description": 'majordeviation for '+str(uuid)
}
```

3.2 Technical description

Since the dynamic risk-based non-conformity assessment is just a part of RAOF we only briefly outline the components of RAOF involved in this process.

3.2.1 Prototype architecture

There is no significant difference with respect to the basic architecture of RAOF presented in D2.6 [1] (see Figure 8). The main component of the framework could be split into the following three parts:

- A *Risk storage* database where the domain layer knowledge and user input are stored.
- *Main engine* with
  - GUI
  - Risk assessment module
  - Non-conformity assessment
  - Dynamic risk evaluation
  - Improvement recommender[5] (to be added in the future)
- *APIs*

[5] The title of the component has been changed (it was called "risk optimizer" in D2.6).

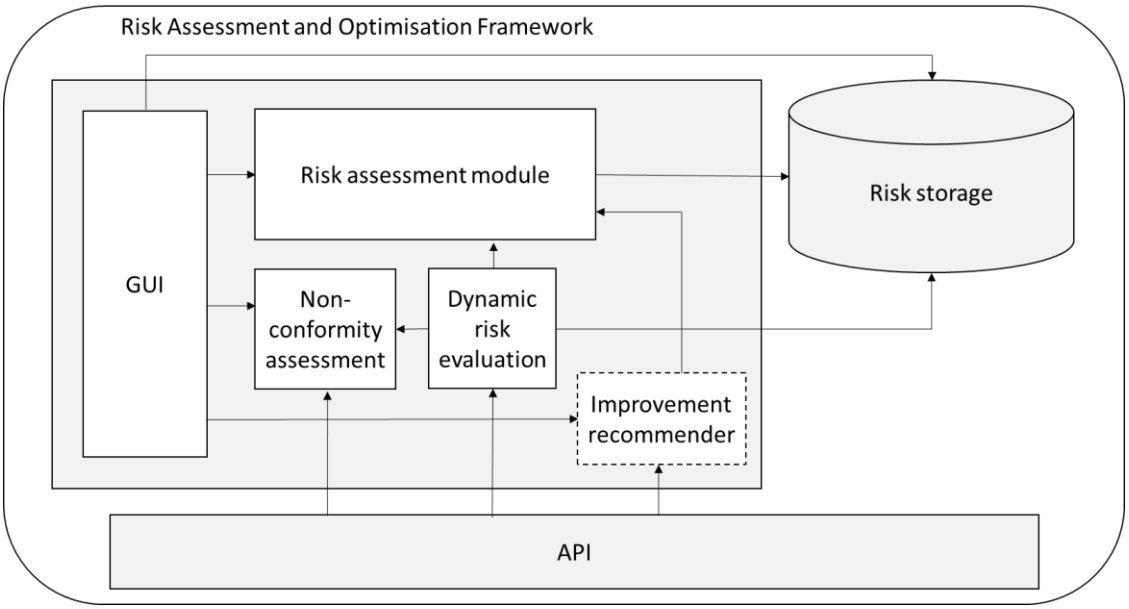


Figure 8. RAOF internal architecture

The only difference with the similar figure reported in D2.6 is the part related to the *dynamic risk evaluation*, which implements the methodology from Section 2.

3.2.2 Components description

The *dynamic risk evaluation* component interacts with the *database*, where the pre-set impact values for asset types are stored, and the *non-conformity assessment* module, which performs the final comparison of real and ideal risk levels and determines if the non-conformity is major or not.

3.2.3 Technical specifications

There are no significant changes in the technical specification of RAOF with respect to D2.6 [1], since the dynamic risk assessment part is just an extension of the core risk assessment tool. In this subsection we only briefly summarise the technical specifications and provide the most up to date information regarding the implementation.

The RAOF component of MEDINA is implemented with the SATRA tool, which is being modified for the needs of the project. Currently it is deployed at the Kubernetes server available provided by the project:

- GUI (engine): https://risk-assessment-engine-dev.k8s.medina.esilab.org/
- APIs (app): https://risk-assessment-app-dev.k8s.medina.esilab.org/

It should be noted that, due to various actions related to the integration with other components of MEDINA, there is no more direct access to our service. Moreover, since our service had its own authentication procedure, the alignment of the internal and project-wise authentication mechanisms is on the way, but currently two logins are required: one log into the MEDINA platform, and then, log in into SATRA. This double authentication process will be resolved in the future version of the tool.

The whole project is delivered using 3 docker containers, one per each part reported in Section 3.2.1. The main service runs over a Tomcat 8 and Apache2 Web Service. The backend is implemented in Java (and Springboot 5 framework). The frontend is developed using JSP,

Javascript, HTML, and CSS. The database is the MySQL DBMS. The third part of the service is Python REST APIs created with swagger documentation. The core communication for the dynamic risk-based non-conformity assessment is executed using this facility.

4 Delivery and Usage

4.1 Package information

The structure of the SATRA tool was only slightly updated. New or changed elements are highlighted in Table 1 and Table 2. In short, the "Risk-Assessment-tool" project is split into 3 folders: frontend-engine (which contains the main part implementing the core logic and the GUI), "app" (the APIs), and "db" (database).

Table 1. Project directory

Folder	Description
-**app/**	Contains the API interfaces source code.
-db/	Contains the database backup.
-engine/deploy_war/	Contains the war file to allow docker-compose of loading this file into correct compose.
-engine/webinterfaces/src	Source code used to connect and communicate with the databases and execute the computation of risks using specific inputs and return specific output.
-engine/webinterfaces/WebContent	Contains all code and media used to implement the GUI (JSP pages/ JavaScript files, CSS, images, WEB-INF configurations).
-engine/tomcat_configs	Contains all files to execute a correctly configuration for tomcat.

Table 2. Project package

Package	Description
api/	Contains the source code for the API interfaces.
api.endpoints/	Contains all endpoint versions for the API interfaces.
api.endpoints.v1/	Contains the first version of the API interfaces.
iit.cnr.it.hibernate.survey/	Source code to manage the connection and communication with the database that contains the survey information.
iit.cnr.it.hibernate.rat/	Source code to manage the connection and communication with the database that contains the user information.
iit.cnr.it.utility/	Useful sub-class and interfaces that contain functions used to perform a particular operation in computation risk class.
iit.cnr.it.security/	Useful sub-class to perform security features.
iit.cnr.it.webtool/	Contains the source code to perform the risk analysis and manage input and output of this operation.

iit.cnr.it.webtool.computation/	Contains the code to compute the risk analysis and csvfe valuation
iit.cnr.it.webtool.computation.csfevaluation/	Contains the code to execute the csvfe valuation
iit.cnr.it.webtool.computation.riskanalysis/	Contains the code to execute the risk analysis
iit.cnr.it.webtool.computation.input/	Contains the code to manage the input
iit.cnr.it.webtool.computation.ouput/	Contains the code to manage the output
utilities	Useful code implemented to compute some operation into the API interfaces code.

It is easy to see that the structure of the project is mostly the same as it was, but only some packages (in bold) have been updated to accommodate the changes. In particular, the "api" package was modified to receive the request for dynamic risk-based non-conformity assessment and make some initial input data pre-processing (this also required some utilities from the corresponding package). Then, the core code for risk and non-conformity assessment (from the frontend-engine) is invoked. Finally, the results is submitted to the life-cycle manager component through an API request.

4.2 Installation instructions

The updated instructions for the SATRA version supporting the risk-based non-conformity assessment is provided below. Once again, although the whole procedure is very close to the one reported in D2.6 [1], we list it here as the most recent one and which captures the latest modifications made in the implementation project.

The following instructions could also be found in the README file uploaded into the repository of the RAOF in TECNALIA's GitLab.

This project uses *docker* to execute and deploy the GUI and the API interfaces. There are three containers:

1. engine: this container contains the GUI
2. app: this container contains the API interface
3. db: this container is a DBMS

Since Docker is compatible with a number of operating systems, below we provide instructions for such OSs as Windows, Mac OS and distro Linux.

First, it is required to create a docker volume for the web services that allows the distribution of GUI and API interfaces.

For Mac OS or Linux

```
sudo docker volume create risk_assessment_web_data
```

For Windows

```
docker volume create risk_assessment_web_data
```

Once risk_assessment_web_data volume is created, it is required to run the containers. There is a folder for every service. The first service to start is the DBMS.

For Mac OS or Linux:

```
cd -db/

sudo docker build . -t risk-assessement-db

sudo docker run -dp 32000:3306  risk-assessement-db
```

For Windows:

```
docker build . -t risk-assessement-db

docker run -dp 32000:3306  risk-assessement-db
```

After the DBMS is started, it's possible to un the engine.

For Mac OS or Linux:

```
cd -app/

sudo docker build . -t risk-assessement-app

sudo docker run -dp 5000:5000 risk-assessement-app
```

For Windows:

```
cd -app /

docker build . -t risk-assessement-app

docker run -dp 5000:5000 risk-assessement-app
```

4.3 User manual

There is not a dedicated user manual for the moment since only the first version of the tool is developed and more changes are planned. Moreover, the dynamic risk-based non-conformity assessment is mostly used as an integral part of the MEDINA process and only a very limited and self-explanatory interaction with a human operator is expected (the initial set up).

4.4 Licensing information

SATRA is licensed under the Apache License 2.0.

4.5 Download

https://git.code.tecnalia.com/medina/wp2/task_2.4/risk-assessment-and-optimisation-framework/-/tree/main

5 Conclusions

This deliverable describes how a cyber risk assessment for a cloud service can be applied in order to support the compliance verification process during continuous monitoring. We have provided the details on how risk-based non-conformity assessment can be performed to support the decision-making process for evaluation of the status of a certificate.

The dynamic functionality reuses the computation model defined in D2.6 [1] and extends it for the automatic operation, during which the input parameters are provided by another tool (instead of a human). Similarly, the supporting tool is not a separate service, but an extension of the exiting one, which is using API for input and output communication instead of a GUI.

In the future months of the project we are going to focus on improving the component in a number of ways. First and the foremost, we will look for the ways to improve the precision in evaluation of requirements for specific resources, considering various possibilities. One possibility is to re-use the values provided manually. Another one is to identify dependencies between assets and use the inherited values. These and other approaches will depend on the information the discovery and evaluation tools are able to provide for the analysis. Also, we will evaluate the approaches using the practical experience of our industrial partners.

Another advancement will be related to suggesting recommendations for correcting actions. These recommendations should help the CSP to efficiently change the non-conformity level from major to minor. The concrete solution for this problem will be evaluated taking into account the possible way of communicating these recommendations to the user and the overall workflow of MEDINA.

Last but not least, we will continue improving the supporting tool, looking for the ways its usability and to reduce the time for assessment. More work is also required in order to ensure tight integration of our tool with other MEDINA components, using the common technologies (e.g., aligned authentication process).

6 References

[1] MEDINA Consortium, "D2.6 Risk-based techniques and tools for Cloud Security Certification-v1," 2022.

[2] MEDINA Consortium, "D4.5 Methodology and tools for risk-based assessment and security control reconfiguration-v2," 2023.

[3] MEDINA Consortium, "D4.1 Tools and techniques for the management and evaluation of cloud security certifications-v1," 2021.

[4] MEDINA Consortium, "D4.2 Tools and techniques for the management and evaluation of cloud security certifications-v2," 2022.

[5] European Union Agency for Cybersecurity, «EUCS – Cloud Services Scheme,» 2020.

[6] MEDINA Consortium, D5.1 MEDINA Requirements, Detailed architecture, DevOps infrastructure and CICD and verification strategy, 2021.

APPENDIX: An example of the input JSON file sent by CCE to RAOF

This appendix contains an example of a JSON file which is sent to trigger the dynamic risk re-assessment process.


```
{
  // "evaluationID": 12345,
  // "cloudServiceId": "16914b5b-803c-4025-bfe9-86350fe8bf54",
  "timeUpdated": "2022-03-25T15:45:49.911Z",
  "evaluationAnswers": [
    {
      "value": 0,
      "weight": 1,
      "threshold": 1,
      "code": "Res. 2 @ OPS-07.2",
      "name": "Res. 2 @ OPS-07.2",
      "state": "SET",
      "timeUpdated": "2022-03-25T15:45:49.909Z",
      "conformant": false,
      "resource": {
        "id": "Res. 2",
        "resourceType": "Container Registry",
        "weight": 1
      },
      "requirement": {
        "value": 0,
        "weight": 0.5,
        "threshold": 0.5,
        "code": "OPS-07.2",
        "name": "OPS-07.2",
        "state": "SET",
        "timeUpdated": "2022-03-25T15:45:49.904Z",
        "conformant": false
      },
      "assessmentResults": [
        {
          "code": "M121 @ Resource{Res. 2}",
          "name": "M121 @ Resource{Res. 2}",
          "state": "SET",
          "conformant": false,
          "id": "36964b5b-803c-4025-bfe9-86350ae8bf54",
          "metricId": "M121",
          "evidenceId": "d1624c6a-499c-4444-a73c-059745cef851",
          "timestamp": "2022-03-25T15:45:49.878Z"
        },
        {
          "code": "M048 @ Resource{Res. 2}",
          "name": "M048 @ Resource{Res. 2}",
          "state": "SET",
          "conformant": true,
          "id": "f112245f-3435-4048-9638-fbbc8f4ae08a",
          "metricId": "M048",
          "evidenceId": "b8d489fc-c363-46bc-a522-30c9172c36dc",
          "timestamp": "2022-03-25T15:45:49.878Z"
```

```
        }
      ]
    },
    {
      "value": 1,
      "weight": 1,
      "threshold": 1,
      "code": "Res. 1 @ OPS-05.1",
      "name": "Res. 1 @ OPS-05.1",
      "state": "SET",
      "timeUpdated": "2022-03-25T15:45:49.909Z",
      "conformant": true,
      "resource": {
        "id": "Res. 1",
        "resourceType": "Database Service. Key Value Database Service",
        "weight": 1
      },
      "requirement": {
        "value": 1,
        "weight": 1,
        "threshold": 0.5,
        "code": "OPS-05.1",
        "name": "OPS-05.1",
        "state": "SET",
        "timeUpdated": "2022-03-25T15:45:49.909Z",
        "conformant": true
      },
      "assessmentResults": [
        {
          "code": "M111 @ Resource{Res. 1}",
          "name": "M111 @ Resource{Res. 1}",
          "state": "SET",
          "conformant": true,
          "id": "6946a235-8a75-4157-ad57-b439d64921b9",
          "metricId": "M111",
          "evidenceId": "319d70b8-1548-4604-8f78-6f7314b279b2",
          "timestamp": "2022-03-25T15:45:49.877Z"
        },
        {
          "code": "M123 @ Resource{Res. 1}",
          "name": "M123 @ Resource{Res. 1}",
          "state": "SET",
          "conformant": true,
          "id": "e0e03386-be4b-4833-83cc-5edaf9dc8c29",
          "metricId": "M123",
          "evidenceId": "4b07c35e-a426-4ed4-b234-146a2d66bd40",
          "timestamp": "2022-03-25T15:45:49.876Z"
        }
      ]
    },
    {
      "value": 1,
      "weight": 1,
```

```
    "threshold": 1,
    "code": "Res. 1 @ OPS-05.2",
    "name": "Res. 1 @ OPS-05.2",
    "state": "SET",
    "timeUpdated": "2022-03-25T15:45:49.879Z",
    "conformant": true,
    "resource": {
      "id": "Res. 1",
      "resourceType": "Identity Management. Role Assignment",
      "weight": 1
    },
    "requirement": {
      "value": 0.5,
      "weight": 1,
      "threshold": 1,
      "code": "OPS-05.2",
      "name": "OPS-05.2",
      "state": "SET",
      "timeUpdated": "2022-03-25T15:45:49.911Z",
      "conformant": false
    },
    "assessmentResults": [
      {
        "code": "M173 @ Resource{Res. 1}",
        "name": "M173 @ Resource{Res. 1}",
        "state": "SET",
        "conformant": true,
        "id": "1d01ab86-da35-4b33-9f81-21d7cc5c8c5d",
        "metricId": "M173",
        "evidenceId": "8e83fdf4-6086-4d71-9197-6d97bf6cc522",
        "timestamp": "2022-03-25T15:45:49.877Z"
      }
    ]
  },
  {
    "value": 0,
    "weight": 1,
    "threshold": 1,
    "code": "Res. 2 @ OPS-05.2",
    "name": "Res. 2 @ OPS-05.2",
    "state": "SET",
    "timeUpdated": "2022-03-25T15:45:49.911Z",
    "conformant": false,
    "resource": {
      "id": "Res. 2",
      "resourceType": "Image. Container Image",
      "weight": 1
    },
    "requirement": {
      "value": 0.5,
      "weight": 1,
      "threshold": 1,
      "code": "OPS-05.2",
```

```
      "name": "OPS-05.2",
      "state": "SET",
      "timeUpdated": "2022-03-25T15:45:49.911Z",
      "conformant": false
    },
    "assessmentResults": [
      {
        "code": "M173 @ Resource{Res. 2}",
        "name": "M173 @ Resource{Res. 2}",
        "state": "SET",
        "conformant": false,
        "id": "72fe57ec-5d2f-4cbd-b899-f195fe9bca9b",
        "metricId": "M173",
        "evidenceId": "c616e38d-d17c-4fe2-8012-a95b6e6ee763",
        "timestamp": "2022-03-25T15:45:49.877Z"
      }
    ]
  },
  {
    "value": 1,
    "weight": 1,
    "threshold": 1,
    "code": "Res. 4 @ OPS-07.1",
    "name": "Res. 4 @ OPS-07.1",
    "state": "SET",
    "timeUpdated": "2022-03-25T15:45:49.907Z",
    "conformant": true,
    "resource": {
      "id": "Res. 4",
      "resourceType": "Image. VM Image",
      "weight": 1
    },
    "requirement": {
      "value": 0.5,
      "weight": 1,
      "threshold": 1,
      "code": "OPS-07.1",
      "name": "OPS-07.1",
      "state": "SET",
      "timeUpdated": "2022-03-25T15:45:49.907Z",
      "conformant": false
    },
    "assessmentResults": [
      {
        "code": "M222 @ Resource{Res. 4}",
        "name": "M222 @ Resource{Res. 4}",
        "state": "SET",
        "conformant": true,
        "id": "2e725c8c-7bdf-4074-9a72-b8d951a86804",
        "metricId": "M222",
        "evidenceId": "ceac8765-d086-40ed-b674-46220d24daae",
        "timestamp": "2022-03-25T15:45:49.878Z"
      }
```

```
    ]
  },
  {
    "value": 0,
    "weight": 1,
    "threshold": 1,
    "code": "Res. 3 @ OPS-07.1",
    "name": "Res. 3 @ OPS-07.1",
    "state": "SET",
    "timeUpdated": "2022-03-25T15:45:49.902Z",
    "conformant": false,
    "resource": {
      "id": "Res. 3",
      "resourceType": "IoT. Messaging Hub",
      "weight": 1
    },
    "requirement": {
      "value": 0.5,
      "weight": 1,
      "threshold": 1,
      "code": "OPS-07.1",
      "name": "OPS-07.1",
      "state": "SET",
      "timeUpdated": "2022-03-25T15:45:49.907Z",
      "conformant": false
    },
    "assessmentResults": [
      {
        "code": "M222 @ Resource{Res. 3}",
        "name": "M222 @ Resource{Res. 3}",
        "state": "SET",
        "conformant": false,
        "id": "27e3ebcf-b9bc-46b4-9260-09c15b9f05f8",
        "metricId": "M222",
        "evidenceId": "0d4fd3ef-5629-4bc7-9328-ab5d673dcb7f",
        "timestamp": "2022-03-25T15:45:49.877Z"
      }
    ]
  },
  {
    "value": 0,
    "weight": 1,
    "threshold": 1,
    "code": "Res. 3 @ OPS-07.2",
    "name": "Res. 3 @ OPS-07.2",
    "state": "SET",
    "timeUpdated": "2022-03-25T15:45:49.900Z",
    "conformant": false,
    "resource": {
      "id": "Res. 3",
      "resourceType": "Logging. Infrastructure Logging",
      "weight": 1
    },
```

```
      "requirement": {
        "value": 0,
        "weight": 0.5,
        "threshold": 0.5,
        "code": "OPS-07.2",
        "name": "OPS-07.2",
        "state": "SET",
        "timeUpdated": "2022-03-25T15:45:49.904Z",
        "conformant": false
      },
      "assessmentResults": [
        {
          "code": "M121 @ Resource{Res. 3}",
          "name": "M121 @ Resource{Res. 3}",
          "state": "UNSET",
          "conformant": false,
          "id": "0320215e-3645-4df7-b9dd-6da9daf81d5a",
          "metricId": "M121",
          "evidenceId": "a7877d3b-7f06-4586-8d55-4adfa117c15f",
          "timestamp": "null"
        },
        {
          "code": "M048 @ Resource{Res. 3}",
          "name": "M048 @ Resource{Res. 3}",
          "state": "SET",
          "conformant": true,
          "id": "0320215e-3645-4df7-b9dd-6da9daf81d5a",
          "metricId": "M048",
          "evidenceId": "a7877d3b-7f06-4586-8d55-4adfa117c15f",
          "timestamp": "2022-03-25T15:45:49.878Z"
        }
      ]
    }
  ]
}
```