MUlti-cloud Secure Applications

Deliverable title

Initial security assurance mechanisms and tools

Deliverable ID:
D4.1

Preparation date:
30/11/2016

Editor/Lead beneficiary (name/partner):
Wissam Mallouli / MI

Internally reviewed by (name/partner):
Massimiliano Rak / CERICT, Stefan Spahr / LHS

Abstract:

This deliverable presents different security monitoring, notification and enforcement mechanisms for multi-cloud applications. A state of the art of existing mechanisms and deployments strategies is presented. The security metrics that are targeted in the context of MUSA project are listed. The MMT-based monitoring agents as well as example of enforcement agents are also presented in detail. These agents are integrated in the MUSA security assurance platform presented in deliverable D4.2 Initial MUSA Security Assurance Platform.

Dissemination level

<table>
<thead>
<tr>
<th>PU</th>
<th>Public</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO</td>
<td>Confidential, only for members of the consortium and the Commission Services</td>
<td></td>
</tr>
</tbody>
</table>
MUSA consortium

Fundación Tecnalia Research & Innovation (TECNALIA, Spain)  
Project manager: Erkuden Rios  
erkuden.rios@tecnalia.com  
+34 664 100 348

Centro Regionale Information e Communication Technology (CER ICT, Italy)  
Contact: Massimiliano Rak  
massimiliano.rak@unina2.it

CA Technologies Development Spain SAU (CA, Spain)  
Contact: Victor Muntes  
Victor.Muntes@ca.com

Montimage (MI, France)  
Contact: Edgardo Montes de Oca  
edgardo.montesdeoca@montimage.com

AIMES Grid Services (AIMES, UK)  
Contact: Prof Dennis Kehoe  
dennis.kehoe@aimes.net

Lufthansa Systems (LSY, Germany)  
Contact: Dirk Muthig  
dirk.muthig@lhsystems.com

TTY-säätiö (TUT, Finland)  
Contact: José Luis Martínez Lastra  
jose.lastra@tut.fi
Table of contents

MUSA consortium ................................................................. 3
Table of contents .................................................................... 4
List of figures ........................................................................ 5
List of tables .......................................................................... 6
Executive summary ................................................................. 7
1 Introduction ......................................................................... 8
  1.1 Objective of this document .............................................. 8
  1.2 Structure of this document .............................................. 8
  1.3 Relationships with other deliverables .............................. 8
  1.4 Acronyms and abbreviations ......................................... 9
  1.5 Revision history ........................................................... 9
2 Monitoring tools in multi-cloud applications ...................... 11
  2.1 Monitoring agents deployment architectures for multi-cloud based applications ... 11
  2.1.1 Monitoring in virtualized environments ....................... 11
  2.1.2 State of art on monitoring virtualized applications .......... 12
  2.1.3 Distributed Monitoring Architecture for multi-cloud based applications .......... 14
  2.2 Security monitoring techniques for cloud based applications ....... 15
  2.2.1 Pattern-based approach and techniques ....................... 15
  2.2.2 Behaviour-based approach and techniques .................... 17
  2.2.3 Hybrid-based approach ............................................ 19
  2.2.4 Data acquisition ..................................................... 19
  2.3 MUSA monitoring agents ............................................... 20
  2.3.1 Network monitoring agent ......................................... 20
  2.3.2 Application monitoring agent ...................................... 22
  2.3.3 System monitoring agent .......................................... 23
  2.4 Security metrics that can be collected .............................. 23
3 Security enforcement and mitigation in multi-cloud applications .... 25
  3.1 Existing security enforcement mechanisms for multi-cloud application .......... 25
  3.1.1 Resource Isolation and Management ................................ 25
  3.1.2 Authentication, Authorisation and Accounting ................ 25
  3.1.3 Encryption and Key management ................................ 26
  3.1.4 Denial of Service attack mitigation ............................ 26
  3.1.5 Network/Web-based protection ................................. 26
  3.1.6 SVA Security and Control ........................................ 29
  3.2 MUSA enforcement agents examples .............................. 29
    3.2.1 High availability framework ................................. 29
    3.2.2 Access control framework ...................................... 30
4 Notification .......................................................................... 32
  4.1 Principle ........................................................................ 32
  4.2 MUSA Web based notification ........................................ 32
5 Conclusion ........................................................................... 34
References ............................................................................. 35
Appendix A. MUSA motivation and background ....................... 41
Appendix B. The MUSA Metric Catalogue .............................. 42
List of figures

Figure 1. Network-based protection ........................................................................................................ 12
Figure 2. Virtual machine introspection .................................................................................................. 13
Figure 3. Host-based protection ................................................................................................................ 13
Figure 4. Distributed MMT architecture deployment ................................................................................ 14
Figure 5. Taxonomy of threat detection approaches .................................................................................. 15
Figure 6. Relationship between the threats and existing detection techniques ....................................... 17
Figure 7. MMT monitoring agent architecture .......................................................................................... 21
Figure 8. Application monitoring agent interfaces .................................................................................... 22
Figure 9. Linux top command for system resources monitoring ............................................................... 23
Figure 10. Workflow of DaMask .............................................................................................................. 27
Figure 11. A taxonomy of mitigation tactics against DDoS attack ............................................................. 28
Figure 12. A taxonomy of mitigation tactics strategies against DDoS attack .......................................... 28
Figure 13. Client control solution architecture ......................................................................................... 29
Figure 14. MUSA high availability framework ......................................................................................... 30
Figure 15. MUSA access control framework ............................................................................................ 31
Figure 16. Web based notification in MUSA Security Assurance Platform .............................................. 32
List of tables

Table 1: MUSA security metric catalogue - summary ................................................................. 42
Executive summary

This deliverable summarises the outcomes of three tasks of WP4, tasks T4.1, T4.2 and T4.3 where we define different mechanisms for security monitoring, enforcement and notification adapted to multi-cloud applications.

Section 2 presents a state of the art of existing security monitoring solutions and deployment architectures that allow to measure different security metrics defined in multi-cloud applications SLAs. The final aim is to learn the real-time behaviour of the multi-cloud environment and early detect the security issues that are violations of the SLA, in order to be able to inform the multi-cloud application DevOps team and minimize their impact. The security monitoring solution adopted in the context of MUSA relies on the MMT (stands for Montimage Monitoring Tool) agents that are able to collect data from network communication between distributed application components. These agents can also monitor the VM or container system resources usage and the application internals at runtime. The collaboration between different agents and other open source monitoring solutions deployed in different CSPs allows the DevOps team to have better visibility on the whole application security and the protection and privacy of sensitive data it manage.

Section 3 describes, by relying on a state of art analysis, the needed set of security enforcement and security risk mitigation mechanisms for the MUSA framework. Some of these mechanisms are to be included in the MUSA Security Assurance Platform SaaS and are to be delivered as a set of embedded libraries which will be used by the multi-cloud application components following a non-intrusive approach. The whole set of enforcement mechanisms will work in an integrated way. The security enforcement mechanisms are built on existing open-source solutions for data encryption, data signing, data obfuscation etc. Through these MUSA security mechanisms, the data will be able to travel along the application components and multi-cloud provider chain with the needed storage and handling security policies enforced.

The last section 4, defines the principles for Web-based notification and response to detected security issues in the multi-cloud application operation. Two kinds of notifications are defined: alerts and violations. Alerts are notified when we are about to violate the SLA. The target of the violation as well as the recommendation strategy to follow in order to mitigate it is also studied in this section.
1 Introduction

1.1 Objective of this document

The issue of data security and privacy in the cloud requires different solutions for implementing and enforcing security policies. In cloud computing environments, many security aspects must be faced, including risk management, data privacy and isolation, security-by-design applications, and vulnerability scans etc. Moreover, it also becomes necessary to have a system that interrelates and operates all security controls which are configured and executed independently on each component of the system being secured and monitored.

In addition, thanks to the large diffusion of cloud computing systems, new attacks are emerging every day, so threat detection systems play a key role in the security schemes, identifying possible attacks. These systems handle enormous volume of information, as they detect unknown malicious activities by monitoring different activities, from different points of observation, as well as adapting to new attack strategies and considering techniques to detect malicious behaviours and react accordingly.

This document is deliverable D4.1 “Initial security assurance mechanisms and tools” of MUSA project (see Appendix A). It is part of the operation phase of the MUSA framework where we define security mechanisms for runtime monitoring, security enforcement and alerts/violations notification included. These security mechanisms are integrated into the MUSA security assurance platform SaaS presented in D4.2 deliverable. This platform is an external entity that allows to monitor the multi-cloud application already deployment in different CSPs. It detects potential deviations from security SLAs and triggers counter-measures to enforce security during application runtime.

The document presents the state of the art of existing security monitoring and enforcement mechanisms and provides first details about the monitoring agents to be integrated in the MUSA security assurance platform SaaS. It also present two example of security enforcement agents integrated into this same platform: The high availability framework and the XACML based access control framework.

1.2 Structure of this document

This document is composed of 3 main sections:

- **Section 2** presents a state of the art of existing security monitoring solutions for virtualized environments as well as the possible deployment of such monitoring solutions. MMT monitoring agents are presented and a list of relevant security metrics for MUSA project is defined.
- **Section 3** presents a state of the art of security controls that can be integrated in cloud based environments. Two examples of such security enforcement mechanisms are presented in details to illustrate the mechanisms that can be offered by the MUSA security assurance platform SaaS.
- **Section 4** presents the Web-based notification mechanism and the possible alerts, violations as well as countermeasures status that are taken at runtime.

1.3 Relationships with other deliverables

The deliverable D4.1 “Initial security assurance mechanisms and tools” presented in this document relates on the following deliverables:

- **D1.1 “Initial MUSA framework specification”**: This deliverable describes the different requirements of the MUSA framework including the security monitoring and assurance mechanisms to be defined in WP4.
D1.2 “Guide to security management in multi-cloud applications lifecycle”: This deliverable defines the step-by-step guide that needed by the assurance mechanisms to fit the MUSA approach.

D2.1 “Initial SdD methods for multi-cloud application”: Security SLAs concepts as well as security risk, controls, metrics relationship is described in this deliverable. This SLAs need to be monitored at runtime thanks to the monitoring agents described in this D4.1 document.

D4.2 “Initial MUSA security assurance platform”: Part of the security mechanisms identified in this D4.1 deliverable is integrated into the MUSA security assurance platform SaaS”. D4.1 mainly present a state of the art of existing security mechanisms for cloud and multi-cloud based environments and D4.2 implements them and integrates them into a software solution deployed as a service.

D5.2 “MUSA case studies implementation”: Outcome software adapted, extended and developed in D4.1 are integrated into the MUSA case studies for evaluation.

1.4 Acronyms and abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>HA</td>
<td>High Availability Framework</td>
</tr>
<tr>
<td>MMT</td>
<td>Montimage Monitoring Tool</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</td>
</tr>
<tr>
<td>CSP</td>
<td>Cloud Service Provider</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SLO</td>
<td>Service Level Objective</td>
</tr>
</tbody>
</table>

1.5 Revision history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date issued</th>
<th>Author</th>
<th>Organisation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>01-May-15</td>
<td>Wissam Mallouli</td>
<td>Montimage</td>
<td>Initial ToC</td>
</tr>
<tr>
<td>0.2</td>
<td>30-May-15</td>
<td>Wissam Mallouli</td>
<td>Montimage</td>
<td>Monitoring deployment architectures</td>
</tr>
<tr>
<td></td>
<td>30-June-15</td>
<td>Wissam Mallouli</td>
<td>Montimage</td>
<td>New version of monitoring deployment architectures</td>
</tr>
<tr>
<td>0.3</td>
<td>01-May-16</td>
<td>Wissam Mallouli</td>
<td>Montimage</td>
<td>New ToC</td>
</tr>
<tr>
<td>0.4</td>
<td>30-June-16</td>
<td>Pamela Carvallo</td>
<td>Montimage</td>
<td>State of the art of security monitoring solutions for cloud based environments</td>
</tr>
<tr>
<td>0.5</td>
<td>30-Sept-16</td>
<td>Pamela Carvallo</td>
<td>Montimage</td>
<td>State of the art of security enforcement mechanisms for cloud based environments</td>
</tr>
<tr>
<td>0.6</td>
<td>04-Nov-16</td>
<td>Antony Shimmin</td>
<td>AIMES</td>
<td>Distributed monitoring deployments</td>
</tr>
<tr>
<td></td>
<td>08-Nov-16</td>
<td>Wissam Mallouli</td>
<td>Montimage</td>
<td>MMT monitoring agents’ description</td>
</tr>
<tr>
<td></td>
<td>09-Nov-16</td>
<td>Petr Belyaev</td>
<td>TUT</td>
<td>High availability framework</td>
</tr>
<tr>
<td>Version</td>
<td>Date issued</td>
<td>Author</td>
<td>Organisation</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>--------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0.7</td>
<td>23-Nov-16</td>
<td>Wissam Mallouli</td>
<td>Montimage</td>
<td>Notification section + Consolidation of the document</td>
</tr>
<tr>
<td>0.8</td>
<td>25-Nov-16</td>
<td>Erkuden Rios</td>
<td>Tecnalia</td>
<td>Security enforcement mechanisms</td>
</tr>
<tr>
<td>0.9</td>
<td>28-Nov-16</td>
<td>Massimiliano Rak</td>
<td>CERICT</td>
<td>Review 1</td>
</tr>
<tr>
<td>1.0</td>
<td>31-Nov-16</td>
<td>Stefan Spahr</td>
<td>LHS</td>
<td>Review 2</td>
</tr>
<tr>
<td>1.1</td>
<td>02-Dec-16</td>
<td>Eider Iturbe</td>
<td>Tecnalia</td>
<td>Final release</td>
</tr>
</tbody>
</table>
2 Monitoring tools in multi-cloud applications

2.1 Monitoring agents deployment architectures for multi-cloud based applications

This section presents different state of the art architectures for security SLA monitoring in multi-cloud based environment (called also virtualized environment) to guarantee security and efficiency of the overall complex application. Besides, a list of requirements, features and challenges to be faced when deploying this kind of monitoring solutions in real-world is presented.

2.1.1 Monitoring in virtualized environments

Monitoring is a solution that is required to ensure the correct operation of the whole system. Malfunctioning or even minor problems in a virtual machine could introduce vulnerabilities and instability of other virtual machines, as well as the integrity of the host machine. In MUSA, the monitoring function is needed to be able to precisely understand what is going on in the network, system and application levels, with a twofold objective. First, it is necessary for improving the security in the communications and services offered by the virtual environments. Second, from the administration and management’s point of view, it will help ensure the environment’s health and guarantee that the system functions as expected and respects its security SLAs.

Existing monitoring solutions to assess security and performance can still be used in virtualized network environments. Nevertheless, existing solutions need to be adapted and correctly controlled since they were meant mostly for physical and not virtual systems and boundaries, and do not allow fine-grained analysis adapted to the needs of cloud and virtualized networks. The lack of visibility and controls on internal virtual networks, and the heterogeneity of devices used make many performance assessment applications ineffective. On one hand, the impact of virtualization on these technologies needs to be assessed. For instance, QoS monitoring applications need to be able to monitor virtual connections. On the other hand, these technologies need to cope with ever-changing contexts and trade-offs between the monitoring costs and the benefits involved. Here, virtualization of application component facilitates changes, making it necessary for monitoring applications to keep up with this dynamicity.

Solutions such as Ceilometer [1], a monitoring solution for OpenStack, provide efficient collection of metering data in terms of CPU and network costs. However, it is focused on creating a unique contact point for billing systems to acquire all of the measurements they need, and it is not oriented to perform any action to try to improve the metrics that it monitors. Furthermore, security issues are not considered.

StackTach [2] is another example oriented to billing issues that monitors performance and audits the OpenStack’s Nova component. Similarly, but not specifically oriented to billing collected [3] gathers system performance statistics and provides mechanisms to store the collected values.

A recent project from OPNFV\(^1\), named Doctor [4], focuses on the creation of a fault management and maintenance framework for high availability of network services on top of virtualized infrastructures. In terms of security, OpenStack provides a security guide [5] providing best practices determined by cloud operators when deploying their OpenStack solutions. Some tools go deeper in order to guarantee certain security aspects in OpenStack, for instance: Bandit [6] provides a framework for performing security analysis of Python source code; Consul [7] is a monitoring tool oriented to service discovery that also performs health checking to prevent routing requests to unhealthy hosts.

\(^1\) https://www.opnfv.org/
In the context of MUSA, we consider the monitoring of multi-cloud based application where each application component can be deployed in a different cloud service provider. This architecture brings more challenges to be able to fulfil an end-to-end security monitoring of the application execution and communication at runtime. To our knowledge, no security monitoring solution has been designed for such multi-cloud distributed systems. Before presenting the monitoring architecture adopted in MUSA, we discuss in the next section the state of art possible monitoring deployments and present their advantages and limits. Notice, that in MUSA, the monitoring solution includes the industrial monitoring agents of Montimage called MMT (stands for Montimage Monitoring Tool). More details about this tool are presented in section 2.3.

2.1.2 State of art on monitoring virtualized applications

To be able to assure end-to-end security in virtualized application components, a monitoring architecture needs to be defined and deployed. This will permits to measure and analyse the network/application flows at different observation points that could include any component of the system, such as physical and virtual machines. The choice of the observation point depends on the monitoring objective and also the monitoring administrator that can be one of the following actors:

- The Cloud Service Provider: Can deploy a monitoring tool (e.g., MMT solution) in his own cloud infrastructure including servers and routers. It has not the possibility to deploy its solution in any virtual machine or container but it can propose to his customers to deploy new VMs or containers from OS images that already integrate a monitoring solution.
- The application owner: Can deploy a monitoring tool (e.g., MMT solution) in each VM or container it deploys. A best practice is to have for each application component a monitoring agent or a set of monitoring agents to observe different behaviours at runtime and check security SLAs.

Setting up several observation points will help to better diagnose the problems detected. In cloud environments, it is possible to create network monitoring applications that collect information and make decisions based on a network-wide holistic view. This enables centralized event correlation on the network controller, and allows new ways of mitigating network faults.

The monitoring probes can be deployed in different points of the system. Let’s consider a single hardware entity that is controlled by a hypervisor that manages the virtual machines. A first approach consists of installing the monitoring solution (MMT) in the host system (hypervisor) that operates and administers the virtual machines (see Figure 1), in this way providing a global view of the whole system. This approach requires less processing power and memory to perform the monitoring operations, since the protection enforcement is located in a central point. In this way, network connections between the host and the virtual machines can be easily tracked allowing early detection of any security and performance issue. The main problem of this approach resides in the minimum visibility that the host machine has inside the virtual machines, not being able to access to key parameters such as the internal state, the intercommunication between virtual machines, or the memory content.

Figure 1. Network-based protection
Monitoring probes can also be located in a single privileged virtual machine that is responsible for inspection and monitoring of the rest (see Figure 2). This approach is called Virtual Machine Introspection (VMI) and offers good performance since the monitoring function is co-located on the same machine as the host it is monitoring and leverages a virtual machine monitor to isolate it from the monitored host [8]. In this way, the monitoring probes analyse the activity of the host through direct observation of the hardware state and thanks to inferences on software state based on a priori knowledge of software structure. VMI allows the monitoring function to maintain high levels of visibility, evasion resistance (even if host is compromised), and attack resistance (isolation), and even enables the manipulation of the state of virtual machines. Unfortunately, VMI based monitoring software is highly dependent on the particular deployment and requires privileged access that cloud providers need to authorize.

The approach that offers the best security performance is the deployment of the monitoring tools in every virtual machine. In this way robust protection can be achieved since the security software has a complete view of the internal state of every virtual machine, as well as the interactions with the host or any other virtual machine. Figure 3 shows how this approach can be deployed.

This third solution offers a good performance in terms of security even if loose visibility of hypervisor behaviour. Here, the processing power and memory required are distributed among the virtual machines. Furthermore, its deployment is simpler than other approaches since it can be included in the software image of the virtual machine, so it is automatically initiated when instantiating each virtual machine with no further configuration needed. On the other hand, the probes lose control over the physical resources and it is impossible to monitor what happens at hardware and hypervisor level. As an example, we can consider the case of one physical CPU shared among two virtual machines VM1 and VM2 assigned to two users U1 and U2. If the virtualization engine move CPU power from one VM1 to VM2, assigning two time slot to VM2 and 1 slot to VM1, VM1 will start going more slowly, but will always perceive 100% CPU. The only way of bypassing this, is to access and monitor the hypervisor or use their APIs if available.

Despite of the individual probes installed on each virtual machine, there is the need of a global monitoring coordinator that supervises the monitoring tasks of each probe installed on each virtual machine. For this, each probe must be able to directly interact with any other probe, as well as with the monitoring coordinator. Local decisions can be taken by the individual monitoring probes installed on
each virtual machine, and the monitoring coordinator can perform coordination, orchestration and complex event detection.

2.1.3 Distributed Monitoring Architecture for multi-cloud based applications

Considering the different monitoring deployments presented in the previous section, herein, a whole architecture integrating monitoring probes and coordinator is presented.

Figure 4 represents a possible deployment scenario for MMT in multi-cloud environment. As depicted, MMT probes capture performance and security meta-data from each virtual machine, and are able to perform countermeasures to mitigate attacks and security risks. MMT probes have the capacity of P2P communication, so they can share relevant information with the aim of increasing the efficiency of the security mechanisms and, thus, ensure the correct operation of the whole system. To perform coordination and orchestration of the whole monitoring system, a central MMT Operator (part of the MUSA security assurance platform) will receive information from the distributed MMT probes. The MMT Operator is also in charge of correlating events to create reports to inform network managers of the system activities, attacks avoided and countermeasures taken. Furthermore, it will be able to globally analyse the information provided by individual MMT probes with the ultimate objective of detecting complex situations that may compromise the system.

The architecture detailed in Figure 4 shows the deployment of MMT (MMT probes and MMT Operator) over a set of physical hardware platforms that can be part of one or several cloud service providers. The MMT Operator will be in charge of coordinating the diverse probes deployed in each virtual machine and provide a global view.

Differing monitoring architectures are required to monitor various elements of the cloud stack the MMT agents can monitor routers, firewalls etc. and these agents can only intercept data or monitor certain elements of the cloud infrastructure or indeed application.

Often CSPs and Application providers utilise different monitoring tools to analyse different components and aggregate the data and outputs to a single interface, however there is no intelligence behind this approach and it is generally very inefficient.

The multi-cloud paradigm and approach means managing, and monitoring distributed applications across heterogeneous cloud infrastructures is a challenge.

The best monitoring deployment looks at the various layers of which the MMT can intercept information to properly target different security levels. Cloud Service Providers however will only allow limited, if any access at all the infrastructure of which it operates. Therefore, buy in is needed both by the cloud consumer and the CSP to allow these different monitoring architectures to both co-exist and to allow the MMT to use this data to present it back to the consumer.
In the context of MUSA, we position ourselves from the application DevOps team and we consider that we can only control our application. For this reason, we only consider at this stage, the deployment of MMT in different VMs or containers where the application component is running. This solution can also introduce new vulnerabilities (since sensitive data is reported to an external tool) that will addressed in the final version of the deliverable D4.3.

2.2 Security monitoring techniques for cloud based applications

Threat detection systems in cloud based environment usually correspond to a hardware device or software application that monitors activity (e.g. network, host, user) for malicious policy violations. Several characteristics of detection systems are well defined in literature ([9]-[17]), among those, fault-tolerance, real time execution, self-monitoring, minimum operational, interoperability, self-adaptiveness, scalability, etc.

Zbakh, M. et al evaluated in [18] several IDS architectures through proposed multi-criteria decision technique, according to the above introduced requirement together with few others such as

- Performance,
- Availability (CSA inspired criteria)
- Service level expectations,
- Secured and encrypted communication channels,
- Accuracy, including the number of false positives (FP), false negatives (FN),
- Detection methods used, etc.

According to such literature, IDS architectures may vary if they are distributed, centralized, agent-based [19] or collaborative; also, the positioning of various observation points also defines various types of architectures (Section 2.1). In general, data collection and preparation are performed through a sensor or existing database which works as an input for the data analysis and detection. The latter engine corresponds to the module of the algorithms implemented to detect suspicious activities, detailed in the following sections and showed by Figure 5.

![Figure 5. Taxonomy of threat detection approaches](image)

2.2.1 Pattern-based approach and techniques

Also known as signature-based or knowledge-based or misuse-based techniques, this approach acts upon a set of rules which define a threat pattern. They have a high level of accuracy (i.e. low FN and FP), but are limited to only known attacks. Therefore, pattern-based techniques cannot detect variants of known or unknown attacks. On the other hand, keeping signatures’ databases updated may be a hard task.

Latest research focuses in facilitating cloud administrators the discovery of new attack patterns by updating signature databases more easily. To assess this automatic and offline analysis, Inductive
Logic Programming was proposed by Hamdi et al. [20], while Huang et al. [21] used Growing Hierarchical Self Organizing Maps (GHSOM) for the characterization of attack signatures. Other techniques are grouped as rule-based and state based techniques presented in the following.

### 2.2.1.1 Rule-based techniques

For known or variant of known attacks, context rule-based techniques have been considered in several research publications. In particular, Katz et al. [22] addresses data leakage threat, considering a context-based detection of confidential documents and sections of confidential information embedded in non-confidential documents. In detail, the corresponding rule-based techniques are:

- **Watermarking**: data breaches may occur in any of the data cycle and digital watermarking is a reviewed technique for detecting data tampering. Garkoti et al. [23] introduced spacial domain watermarking, encryption, and logging modules for clinical data.

- **Fingerprinting** was considered for malicious insider threat detection by Gupta et al. [24] through the analysis of well-known programs and the malicious modification of their system call sequences, as fingerprints, executed in the hypervisor. Provable Data Possession (PDP) is often related to data losses and preserving data integrity over a client prepossessing the data and then sending them to the cloud for storage while keeping a small amount of meta-data for later checking. The classical idea behind this technique can only be applied to static (or append-only) files. Hence, Erway et al. [25] presented a framework and construction for Dynamic Provable Data Possession, which extends the PDP approach with the support provable updates to stored data, using the new version of authenticated dictionaries based on rank information.

- **Sequence alignment**, while commonly used in bioinformatics, was proposed by Kholidy et al. [26] for account or service hijacking threats, in specific for masquerade attacks. They introduced Heuristic Semi-Global Alignment algorithm, which tests matching patterns of user’s session sequences (e.g., mouse movements, system calls, opened windows titles, written commands, opened file names) with the previous stored arrays.

### 2.2.1.2 State-based techniques

In relation with insider threats and further potential data related threats, Kumar et al. [27] considered a method relied in well-known Bell-LaPadula model which aims to determine the identity of the internal organization user who leaked data. The idea defines states, as the allowed access mode to any subject, with any object allowed, with respect to a defined security policy. This model is built on the concept of state machines with a set of allowable states. Various cryptographic and watermarking techniques can be applied, to identify the internal user involved in the leakage.
The described detection groups correspond to content-based techniques since they test ‘known or accepted’ action patterns. Therefore, data related threats are more deeply linked to these groups as they may have low occurrences, and other techniques more prone to volume based identification of misuses (e.g. statistical), may carry high FP rates. Additionally, hijacking threats such as masquerading attacks, are subjects of study. These links can be seen in Figure 6 where rule-based and state-based techniques share relations with the previous mentioned threats.

### 2.2.2 Behaviour-based approach and techniques

Also known as anomaly-based detection, this approach involves the collection of data in order to construct a model of normal behaviour and then test new observed behaviours against potential anomalies. An important aspect of the anomaly-based approach is the nature of the anomaly. Anomalies are classified as point, contextual and collective anomaly.

The first one (point anomaly) consists in a single data event deviating from normal pattern of the dataset. Contextual anomaly, on the other hand, exemplifies point anomaly within a particular known context. The latter (collective anomaly) happens when a collection of similar data events behave anomalously with respect to the rest of the dataset, where the group of data events defines a collective anomaly.

Techniques used for behaviour-based threat detection rely on the criteria adopted to model and identify behaviour classified as normal and behaviour classified as suspicious. Therefore, existing techniques can be grouped with respect to the way how such behaviours are analysed.

#### 2.2.2.1 Statistical techniques

Statistical approaches are in general predefined by a threshold, first order statistics or probabilities, in order to identify anomalies. One example is for a type of Denial of Service (EDoS) issued by [28], where the authors study user demands against thresholds over duration parameters as the maximum number of requests beyond when the auto-scaling feature is activated.

Non-parametric techniques also take place when the system observes the activity of subjects in terms of statistical distribution and creates profiles which represent their behaviours for later comparison.
(i.e. correlation). Shirazi et al. [29] presented a memory-less technique based in a non-parametric Cauchy function, recursively updated upon the signal at the previous step. Entropy based techniques focus on measuring the uncertainty or randomness associated with a random variable. For network flows, comparing the rate of entropy of some packet header fields with another sample of the same nature provides a mechanism for detecting changes in the randomness. For DDoS attack detection, Jeyanthi et al. [30] proposed a cross-layer implementation where first component analyzes incoming traffic rate and is handed to entropy profiler if acute. Sharma et al. [31] proposed a source and MAC address threshold based analysis for incoming packet rate. Same threats concern and approach was followed in [32] where the authors proposed to investigate VM status in real cloud environment based on OpenStack, arguing on the fact that the malicious VMs share similar attack patterns.

Shape analysis is utilized for detecting shared technologies threat, as seen in Figure 6. For example, in [33], the authors combined a two stage detection mode based on shape tests which are used to extract the attack features from the cache miss sequence, and regularity tests used to extract the attacks features from virtual CPU utilization sequence and virtual memory utilization sequence. Bates et al. [34] addressed covert side channel threats by detecting network flows watermarking. A Principal Component Analysis (PCA) was used by Marnerides et al. [35], not only to reduce datasets dimensionality but also to separate the normal data from anomalous.

Signal decomposition techniques such as Ensemble Empirical Mode Decomposition (E-EMD) were presented in [36]. The authors proposed a data-driven method for malware, motivated by the fact that the algorithm can sufficiently decompose and describe clouds’ non-linear and non-stationary traffic. Catastrophe theory explores that systems can respond to continuous changes in control variables by producing sudden, drastic and discontinuous changes from one equilibrium state to another (e.g. from normal state to anomalous). A catastrophe potential function was introduced by Xiong et al. [37], in order to describe the dynamic process in cloud network traffic.

### 2.2.2.2 Machine learning based technique

Machine learning techniques generally refer to the technique where we continuously analyse historical behaviours (of an application or system) in order to detect deviations from a specific baseline that we learned (can be static or dynamic base line). This technique permits to improve the detection capability by learning from previous results. In this subsection, we group these techniques with respect to their underlying models for detecting security threats in the clouds.

Decision trees are used in [38], addressing an unsupervised clustering algorithm for unlabelled data. After labelling, a decision tree based analyser with Incremental Tree Inducer model is trained, updating itself.

SVM technique was proposed by Watson et al. [39]. They studied an online novelty implementation of a supervised one-class SVM algorithm, an extension of traditional two-class SVM which outputs either a known class (VM normal behaviour) or unknown classes to the classifier, for each particular input vector.

Artificial Neural Networks expose their accuracy based on the configuration of their hidden layers and training phase. Based on different feed forward neural networks with back propagation algorithm, Pandeeswari et al. [40] evaluated the error of the different ANN, with a fuzzy aggregation module is used to combine the different neural networks results. A Synergetic Neural Network (SNN) was addressed by Xiong et al. [41], given the dynamicity of the network traffic. Their argument relies in that under some situations, the changing trend of the cloud-based network traffic is only determined by a few primary factors and less contribution of others.

### 2.2.2.3 Clustering techniques

Clustering is utilized under the underlying assumption that 'normal’ data instances lie distance-wise closer to a given centroid of a cluster, whereas anomalous data points are recognized due to their much longer distance.
K-means technique was followed by Marnerides et al. [42], while showing it is directly affected by live-migrations. In this testbed they detected DoS and netscan threats successfully when arose, but also achieved high scores when only migration and normal traffic occurred. The same clustering idea was used in the dimension reasoning technique (based on Local Outlier Factor) that was suggested for memory leakage, and malicious port scan, by Huang et al. [43]. GHSOM techniques were also addressed by Li et al. [44], by proposing a cluster system that identifies nmap$^2$ malicious behaviours in VMs through system call distributions and secondly, derives rules for SVM detection.

### 2.2.3 Hybrid-based approach

Depending on the architecture and a set of threats to be detected, the use of techniques in Cloud architecture can require a hybrid approach [45][46][47][48].

While signature-based approach is more rigorous in its detection, behaviour-based methodology is able to 'learn' new threats. Therefore, the combination of previously mentioned approaches in Sections 3.1 and 3.2 and may reach a more extensive and accurate detection. As an example, Modi et Patel [46], used SNORT$^3$ for signature-based detection, whereas for anomaly-based detection they focused in Bayesian, associative and decision tree classifiers.

Another way of analysing incoming IP packets was done by Shamsolmoali et al. [47] in a two stage detection: their system extracts the Time to Live value and computes the number of hops the packet has travelled, and then packets headers are analysed for anomaly in the normal trained database.

### 2.2.4 Data acquisition

#### 2.2.4.1 Monitoring design considerations

Depending on the type of threat, some of them may be visible depending on the network, host or hypervisor based monitoring positioning. As it relies on the type of threat intended to detect, the monitoring layers can be classified as follows:

- Network-based monitor activity of network traffic, mostly IP and transport layer;
- Host-based (HIDS) running on individual hosts or devices on the network;
- Hypervisor-based capturing virtual machine introspection in order to gather system-specific features (e.g. process list, threats count, number of open ports);
- Cross-layer based related the monitoring from any combination of the previously mentioned.

Deploying design of threat detection system devices in the cloud infrastructure is very relevant. As mentioned in [49], CSPs have to take into account if the used services are compromised, if hosts are used to attack other victims or if the detection system itself is under attack. Therefore, it is useful to design a detection component concerning each end-user and deploy a collaborative communication exchange between detection nodes. If the detection system is used to monitor a VM host on the host itself, it cannot be guaranteed that the system (e.g. HIDS) works properly when the host is compromised, as the attacker could have modified it not to send any reports. For data related threats at upper levels, HIDS may need to have separate configurations for each end-user client. Furthermore, a detector associated with a specific VM should be sufficiently lightweight to travel with the migrating VM.

By covering a wide range of threats numerous detection systems are deployed; in fact, these systems may have significant performance. In should be noticed that some studies have addressed this topic (see, for example [50]) by processing data in parallel with MapReduce techniques.

---

$^2$ https://nmap.org/
$^3$ https://www.snort.org/
2.2.4.2 Datasets

Evaluating the effectiveness of a given detection technique against specific threat (or their groups) is mainly performed through corresponding experimentation. That is the reason, why building a proper dataset that contains ‘normal’ and abnormal behaviours with respect to different threats and taking account various monitoring abilities, always remains a challenging problem.

Most commonly used datasets reviewed for cloud threat detection are: KDD Cup 99 [51][52][53][54], DARPA 2000 [51] and self-generated testbeds [55][56][57][58][59].

Due to threat analysis and multiple layer vulnerabilities concerned, it may be intuitive to conduct the use of joint datasets as mentioned in the previously cross-layer based system. Nevertheless, this decision should concern techniques used and performance when analysing. Having high dimension vectors may add higher complexity, and the correct election of the characterizing features is key. For instance, Watson et al. [59] studied that raw system level—per process—features (e.g number of threats, memory usage), are not useful if considering each sample as a single feature vector. They built statistical meta-features of each feature across all processes. They also used one class SVM technique with system and network data, while detection was no more effective than when analysed separately (only 10% improvement).

As for tools for collecting data, for network level existing software like Wireshark, MMT tool, Bro, CoralReef and tcpdump[^4] are considered. Hypervisor tools like libVMI and Volatility. System log tools in the other hand, are held by Ntrace and Strace [24], while OProfile, which uses event-based sampling [33].

2.3 MUSA monitoring agents

To be able to deeply analyse security, in MUSA we rely on different agents to be installed in different VMs or containers where application components are deployed. These agents collect data coming from network, from system and from application internals and send them to a global monitoring platform called MUSA assurance platform. More details about these agents are presented in the following subsections.

2.3.1 Network monitoring agent

The MUSA monitoring agent [60] is based on MMT monitoring solution (MMT stands for Montimage Monitoring Tool). It is a monitoring solution that combines a set of functionalities presented in the following list:

- Data capture, filtering and storage,
- Events extraction and statistics collection, and,
- Traffic analysis and reporting providing, network, application, flow and user level visibility.

Through its real-time and historical views, MMT network agent facilitates network performance monitoring and operation troubleshooting. With its advanced rules engine, MMT monitoring agent can correlate network events in order to detect performance, operational, and security incidents.

MMT monitoring agent is composed of a set of complementary, yet independent, modules as shown in the following Figure 7.

[^4]: www.tcpdump.org
**MMT-Capture**: allows the capture of network packets based on the libpcap library.

**MMT-Filter**: is a basic filtering capability provided by MMT monitoring agent that permits to focus on only some specific types of traffic depending on the usage of the network probe.

**MMT-DPI** is the core packet processing module, it is a C library that analyses network traffic using Deep Packet and Flow Inspection (DPI/DFI) techniques in order to extract hundreds of network and application based events, measure network and per-application QoS/QoE parameters and KPIs. The MMT-DPI library has a plugin architecture. It is possible to extend the extraction engine with new protocols and also add new security metrics. For this, a plugin needs to be created specifying the extraction to add. An MMT-DPI plugin will initialize a protocol structure that contains the required information regarding the protocol attributes, as well as the functions allowing extracting the data corresponding to these attributes. In the context of MUSA, a set a security metrics has been identified and integrated in MMT monitoring agent to be able to report them to the MUSA security assurance platform. These extracted metrics are important to be able to perform security analysis of the communication between different application components and detect potential security flaws.

**MMT-Security** is an advanced rule engine that analyses and correlates network and application events to detect performance, operational and security incidents. It is powered with self-learning capabilities to derive the baseline network for dynamic threshold based analysis and potential Deny of Service (DoS) attacks detection.

**MMT-QoS** is an interesting module in MMT monitoring agent that allows providing a certain visibility on the quality of the network in terms of different KPI like delays, jitter, response time etc.

### 2.3.1.1 MMT Features

MMT allows granular traffic analysis capabilities through the ability to extract a wide range of network and application based traffic parameters and events (RTT, jitter, loss, HTTP response time, etc.). It permits application classification making possible the detection of applications using non-standard port numbers.

Besides, MMT has a powerful rule engine: that allows the detection of the occurrence of complex sequence of events that conventional monitoring does not detect. This can be used for example to monitor the access control policy (authorized users are authenticated prior to using a critical business
application), for anomaly or attacks detection (excessive login attempts on the application server), advanced performance monitoring (identification of VoIP calls with QoS parameters exceeding acceptable quality thresholds), etc.

In the context of MMT monitoring, DPI (Deep Packet Inspection) and DFI (Deep Flow Inspection) are used to help detect and tackle harmful traffic and security threats as well as security SLA violation in Cloud based environments. We define a set of security properties for network traffic, at both control and data levels, to catch interesting events. Indeed, based on the defined security properties, we register the attributes to be extracted from the inspected packets and flows. These attributes are of three types:

- **Real attributes**: They can be directly extracted from the inspected packet. They correspond to a protocol field value.
- **Calculated attributes**: They are calculated within a flow. Packets from the same flow are grouped and security/performance metrics are calculated (e.g. delays, jitter, packet loss rate) and made available for the MUSA security assurance platform.
- **Meta attributes**: These attributes are linked to each packet to describe capture information. The time of capture of each packet (timestamp attribute) is the main meta attribute in the current version of MMT monitoring agent.

### 2.3.2 Application monitoring agent

The role of the MMT-Application agent is to deliver information about the internal state of the target system i.e. multi-cloud application component to the MUSA security assurance platform during its operation. It continuously checks and monitors application health. It notifies the MUSA security assurance platform about measurements of execution details and other internal conditions of the application component.

The application monitoring agent is a Java library built of two parts. The first part is an aspect to be weaved into the application code via pointcuts in order to send application-internal tracing information to the MUSA security assurance platform for analysis. It is composed of a set of functions that can be weaved in strategic application points to capture relevant internal data. The second part connects the aspect with the notification tool via a connector library and it provides a simple interface to send log data to the MUSA security assurance platform in a secure way. In other words, the application monitoring agent is responsible for extracting the information from the system, and the MMT connector is responsible for transferring it.

The application monitoring agent has two interfaces connecting it to the MUSA framework:

![Figure 8. Application monitoring agent interfaces](image)

- The application monitoring library is deployed as a Java AspectJ package. It is statically woven into the target application at compile time using AOP. It connects to the application monitoring agent based on annotations inserted into the source code base.
The output of the application monitoring agent, a peek into the internal state of the monitored system, is sent to the MUSA Security assurance platform. This platform is accessed using the MMT Connector library, which provides a one-way secure channel for sending log messages. It consumes a list of name-value pairs of textual data. The MMT Connector uses HTTPS for communication.

To use the application monitoring library, it must be woven into the target application, and the target source code must be annotated, using standard-like Java annotations (e.g., @Monitor). During operation of the target application component, the application monitoring library produces a stream of log messages. The developers can specify measurement points that generate data which is sent to the MUSA assurance platform. Measurement points can be attached to classes, methods and attributes and depends on security metrics to be monitored. More details are available following this link: https://github.com/MUSA-Project/MMT_Annotations

### 2.3.3 System monitoring agent

MMT-System agent monitors system resources which may be the cause of server performance degradation, like CPU, memory and disk accesses, and spots performance bottlenecks early on.

MMT-System agent relies on Linux “top” command which is used frequently by many system administrators to monitor Linux performance and it is available under many Linux/Unix like operating systems. The top command used to display all the running and active real-time processes in an ordered list and updates it regularly. It displays CPU usage, Memory usage, Swap Memory, Cache Size, Buffer Size, Process PID, User, Commands and much more. It also shows high memory and CPU utilization of all running processes. The top command is much useful for system administrator to monitor and take correct action when required.

![Linux top command for system resources monitoring](image)

**Figure 9. Linux top command for system resources monitoring**

### 2.4 Security metrics that can be collected

As mentioned in Deliverable 2.1, MUSA adopts and extends the Security Metric Catalogue developed in the SPECS project [61], which defined a model for metric representation and proposed a set of security-related metrics that were relevant for the project.
The initial set of security metrics available in the catalogue has been reviewed and enriched by MUSA by (i) taking into account the metrics that have been proposed by standardization groups (e.g., in the Special Publication “Performance Measurement Guide for Information Security” by NIST [62] and in the “Consensus Security Metrics v1.1” by the Center for Internet Security (CIS) [63]) and by other EU projects (such as Cumulus [64] and A4Cloud [65]), and by (ii) introducing specific metrics relevant to the MUSA case study applications.

The result of this collection and analysis process is the catalogue reported in Appendix B, which represents the current MUSA Security Metric Catalogue. Note that the final version of the catalogue will be available only at the end of the project and that this is still a draft version.

Collected metrics are heterogeneous in that they significantly differ one from the others based on different aspects.

Some of the identified metrics actually give information on the “level of security” of the system under observation by taking into account the detection of specific events of interest (e.g., number of detected attacks or of mitigated vulnerabilities). These metrics can be typically monitored with classical monitoring tools and give information on how well the system was configured or on how much the system is at risk.

Usually, these “technical” monitorable metrics may be used by a customer to specify his/her requirements in an SLO (e.g., the customer may require that the percentage of incidents reported within required time frame is 100%).

It should be noted, however, that it makes sense to use a metric of this kind in an SLO only if the customer is provided with the means to monitor it (directly or indirectly) and if the desired value (or range of values) set for the metric is somehow under the control of the provider (e.g., it makes no sense to set an SLO like “number of attack attempts<100” because the provider cannot control the behaviour of attackers but only limit the resulting damage).

A different class of metrics is the one of “management” metrics, which refer to the way an organization sets-up its security policies and procedures and handles security incidents (e.g., cost of incidents, notification of incidents, level of automation in the procedures adopted by the organization for recording the privacy training sessions of its employees, etc.). These metrics may be used in SLOs since they represent high-level requirements that target organization aspects and can be “measured” by simply checking how the system is configured. Also, they may be used to configure the target system, when possible, with the parameters defined by the customers.

This introduces the third type of metrics, namely the “enforceable” metrics: they are preferably used for the enforcement of specific security configurations (e.g., the “HTTPS_ON” metric identifies whether the automatic redirect from HTTP to HTTPS is implemented) and can be “measured” by simply checking the current system configuration.

Finally, technical metrics (both monitorable and enforceable) can refer to different application domains:

- Data protection: these metrics refer to the protection of data in transit or at rest (e.g., data availability);
- Communication protection: these metrics refer to the protection of (HTTP) communications (e.g., use of HTTPS in communications);
- SW component/system protection: these metrics refer to the protection of a SW component or a SW system (e.g., frequency of vulnerability scans).

Metrics have been classified also based on the Abstract Metrics of reference. An Abstract Metric is a metric that defines an abstract standard of measurement used to assess a property and therefore describes what the result of the measurement means (e.g., frequency of operation, cost, feature activation etc.). All the abstract metrics considered are reported in Appendix B.
3 Security enforcement and mitigation in multi-cloud applications

3.1 Existing security enforcement mechanisms for multi-cloud application

Prevention, monitoring, detection, and mitigation generally illustrate the defence life-cycle. Prevention involves the implementation of a set of defences, practices, and configurations prior to any kind of attack, with the aim of reducing the impact of such attack. These issues could be addressed by network security, data protection, virtualization and isolation of resources. Traditionally, well-known countermeasures have focused on dealing with threats through a variety of methods devised around questions such as where is the attack detected? How is the attack detected? What is the response mechanism? Where to apply the response mechanism? Where is the control (decision) center from which filtering rules are taken?

Previous studies have assessed the analysis of such mechanisms, for instance, Carlin et al. [66] studied vulnerabilities and countermeasures and proposed a flow chart showing the exiting DDoS cloud protection systems and comparing the implementation of different features in the proposed systems. Other methods utilized are profiling based techniques, in order to discriminate the misusability from users (i.e. trying to gain privileges); IDS, pattern matching in the search for specific confidential words trying to be breached, or queries in databases monitoring. We summarize the mechanisms in the following subsections.

3.1.1 Resource Isolation and Management

Resource isolation is a key security feature to protect VMs from malicious attacks. The isolation-based defence approaches can be split into those that isolate the running of VMs and those that focus on the isolation of shared resources [67]. The first approach may limit the ability of the system to schedule the work of legitimate VMs and keep the best practices for installation and configuration stages. To implement the second approach, a monitoring and remediation mechanism is needed to probe all resource requests and to allocate these requests to VMs. In addition, implementing strict policies needed to enforce isolation is difficult across a large-scale distributed system, such as cloud. As an example, Volokyta et al. [68] suggests a VM Monitor to monitor other VMs. The proposed system intercepts system calls and maintains a log file of system warnings. Following with this idea, Yu et al. [2] proposed a security-awareness VM Management scheme based on Chinese Wall policy. They assess side channel attacks by analyzing user constraints relations with the VMs they use. SNORT and OpenFlow are combined by Chung et al. [69] to produce an IDPS that can reconfigure a cloud network system in real time using Iptables. This decides if a response is necessary and chooses from options including, traffic redirection, traffic isolation, deep packet inspection, MAC address change, IP address change, block port or quarantine.

3.1.2 Authentication, Authorisation and Accounting

The term AAA usually refers to solutions aimed at providing identity authentication, authorisation to access computer resources or services, security policy enforcement or resource/service usage auditing. The AAA mechanisms give computer resources and services protection by ensuring that they can be accessed only by authorized parties (users, services or devices). These mechanisms are countermeasures for malicious attackers, data related threats (eavesdropping, stealing, etc.) and account, service & hijacking.

For identity management, the current trends involve federated ID and single sign-on (SSO) solutions [70]. Besides, there are currently a number of Identity-as-a-service (IDaaS) solutions managed by third-party service providers that can easily be integrated in applications. While facilitating a lot the
work in ID management, this type of solutions also bring concerns about service availability and identity data protection.

On authentication, while there are a number of authentication protocols it is recommended to use strong multi-factor authentication (at least two-facto) techniques where possible.

For stronger data protection, the best solution is to combine access control mechanisms both at design and operational phases. The AAA mechanisms should be accompanied by proactive monitoring to detect unauthorized activity at operation. At the forensics level the mechanisms should be able to profile and monitor host-based information such as database activity, file system access patterns, system calls and OS commands, and others.

### 3.1.3 Encryption and Key management

Encryption is usually the means used to protect data in transit and at rest and ensure data integrity, confidentiality and non-repudiation. It is the basis of authentication and authorisation mechanisms. The secure management of encryption keys is the basis for ensuring unbreakable encryption. This involves strong key generation, storage, backup and management, and destruction practices.

There are multitude encryption mechanisms that are usable in cloud, many of them being open source. A thorough comparison of cryptographic libraries can be found in [71].

As an example of existing encryption mechanisms for cloud is the End-to-End encryption mechanism offered in SPECS project [72]. It offers a browser plugin/extension which enable to share data among different users protecting them through ad-hoc encryption techniques that enable to identify any possible attempt to alter data.

It is currently under study the possibility of including similar enforcement mechanisms in the MUSA framework.

### 3.1.4 Denial of Service attack mitigation

The Denial of Service (DoS) or Distributed Denial of Service (DDoS) attack mitigation mechanisms aim at ensuring service availability. Protection against DoS or DDoS attacks should include both detection and mitigation/remediation of the attacks.

This is precisely what the DoS mitigation mechanism of SPECS platform offers [72].

The mOSAIC platform also included tools that protect applications from some types of intrusions and Denial of Service attacks [73][74] in cloud. The platform offered such mechanisms in form of a cloud SLA based Intrusion Tolerance as a Service.

It is currently under study the possibility of including similar enforcement mechanisms in the MUSA framework.

### 3.1.5 Network/Web-based protection

Network/Web-based protection is used in organizations faced with the task of keeping services operative and available. To handle the high volume of network traffic, researchers have attempted to create data-loss prevention systems, and used mechanisms such as:

- Filtering, which can comprehend the use and update of firewall rules, adding specific information to packet headers in order to identify legitimate packets [75] or analysing a packet header parameter (i.e. TTL IP field [76]),
- Rate limiting,
- Logical and physical migration of resources, along with stateful application management,
- Redundancy models or distributions,
- Overload protection such as auto-scaling or load balancing.
The following broad categories can be used to classify most DDoS prevention approaches. Shameli-Sendi et al. [77] proposed a taxonomy (shown in Figure 11), for DDoS followed by a detailed classification of latest publications according to: strategy models, mitigation tactics, etc. In Figure 12 one can see this schema categorized as collaborative or non-collaborative. Additionally, Osanaiye et al. [78] assesses resilience in for DDoS in cloud environments and proposes a mitigation framework and taxonomy. Wang et al. [79] study the impact of their proposed defence mechanism for DDoS. Their architecture named “DaMask”, shown in Figure 10, contains a DaMask-M module which has functions of countermeasure selection and log generation. When DaMask-M receives an alert, it tries to match the alert to a countermeasure.

![Figure 10. Workflow of DaMask](image)

As an example for countermeasures for Denial of Service, is preferentially assessed by the use of an intrusion detection system (IDS). The most common DDoS defence approaches combine elements located in the source-end and victim-end in to combine their advantages. Madhusudan et al. [80] proposed a system considering an IDS along with a countermeasures system. First stage uses attack graph analytical procedures and the latter employs a reconfigurable virtual networking approach once a VM is under inspection.

Cloud Service providers also work closely with their upstream providers to mitigate the impact DDOS and attempt to isolate the source and route traffic elsewhere. In the context of multi-cloud applications, generally speaking unless there has been an active-active configuration deployed of a service in another facility, the application’s functionality will be severely impacted by a CSP coming under DDOS.

Of course, often Global Tier 1 internet providers can be impacted directly by DDOS [81] and outages can affect cloud platforms across entire continents, so there is no guarantee that simply distributing cloud resources over large geographical distances will provide the uptime said application needs, especially when the network infrastructure is the same as other data centres. This is where deeper understanding of the network resilience of a data centre is required; however such investigative works are only cost effective for mission critical applications and services.
Web applications are also vulnerable to layer DoS attacks, through exploiting Representational State Transfer (REST) API in cloud. REST, an abstraction for distributed communication over a network, can be used as an alternative to SOAP. The vulnerability is due to its exposure, as services using REST do not require authentication. As an example, Michelin et al. [82] proposed a defence strategy, shown in that uses authentication token. This approach has two different modes, namely: monitoring and filtering as mentioned in the following Figure 13.
3.1.6 SVA Security and Control

Other type of security enforcement relies in providing the necessary Software Vulnerability Assessment (SVA) tools for the application to be protected. For example, the SPECS platform provides mechanisms for dynamically activating and configuring the SVA tools over a web container. The offered SVA tools are able to detect and upgrade the vulnerabilities or fix the misconfigurations in the specified software packages deployed in cloud.

It is currently under study the possibility of including a similar enforcement mechanism in the MUSA framework.

3.2 MUSA enforcement agents examples

In the following we provide a short description of the MUSA enforcement mechanisms and agents that are being developed in the project. A more detailed description will be included in deliverable D4.2 Initial MUSA security assurance platform.

It is expected that besides these two examples, the MUSA framework final version includes other enforcement mechanisms based on other existing security mechanisms such as the ones offered in SPECS project [72] (see above).

3.2.1 High availability framework

The High Availability framework (HA framework), designed and built by TUT, is a cloud-oriented open source HA solution that aims to provide enhanced service availability without source code modification. It is capable of scaling the service by deploying and controlling the required number of service instances, handling service failures by migrating the instances in case of failure, and it provides dynamic routing in the cloud environment to help services to find the requested endpoint even if the service itself does not have any HA capabilities. The framework can be deployed beside the component for IaaS nodes, or as a part of VM/Docker image in case of PaaS installation. In case of IaaS, framework provides Docker service to allow containerized service deployment.

In addition, the HA framework is designed to ensure secure communication (HTTPS) between components with the certificate verification. It allows to be sure that the request comes from the
authorized piece of infrastructure. In addition, role-based API access control is provided to filter unauthorized requests even before they will reach the service.

The HA framework is built using a number of open-source projects with a number of additional patches as well as configuration and security enforcement scripts. It includes Corosync [83]/Pacemaker [84] suite developed by ClusterLabs, which takes care of the resource management; Consul [7] directory to sync the real-time data between the nodes; Consul-template [85] configuration manager to reconfigure the services using the directory data; HAproxy [86] as a lookup proxy and load balancer; Nginx with custom Lua scripts as a security gateway for certificate and ACL checks; Docker [87] as a container agent; and Registrator [88] by GliderLabs [89] to make the deployment process easier. Single node deployment diagram is shown in Figure 14 below.

Figure 14. MUSA high availability framework

In terms of HA enforcement, the framework is capable of balancing resources on its own, this mechanism is launched automatically only in critical situations. This behaviour is configured by the DevOps team when the new service is introduced to the framework and can be changed later. During the normal operation, the framework respects the decision made by MUSA DST.

In terms of deployment, the HA framework relies on Chef cookbook, which are used by the MUSA Deployer. Main cookbook of the framework can be included as a dependency into any component cookbook with minimal effort.

3.2.2 Access control framework

The MUSA Access control framework, designed and developed by Tecnalia, is an open source solution aimed at guaranteeing: first, that only authorised end-users of the multi-cloud application
components can access them and second, that only authorised requester components can consume services in other components.

The MUSA Access control framework is developed in Node.js [90] and is composed as shown in Figure 15 by a MUSA API Gateway, a MUSA IDM and a MUSA enforcement agent.

**Figure 15. MUSA access control framework**

- The **MUSA API Gateway** is the gateway exposed to the multi-cloud application Clients and ensures security and relies on a reverse proxy pattern.

- The **MUSA IDM** provides the Authentication and Authorization service. It relies on OpenID Connect [91] for single sign-on and identity provision and OAuth 2.0 [92] for the identity tokens. All data between the Client (it can be a web application or a native application) and multi-cloud application components (backend services deployed in cloud) is transferred using JWT (Jason Web Tokens) [93]. The MUSA Token Exchange is the security token service that manages the life cycle of a JWT (create, sign, validate).

- A **MUSA Enforcement agent** that is deployed together with the component is the one that permits/denies access to the component services, based on XACML policies [94]. The agent includes the Policy Decision Point (PDP) and Policy Enforcement Point (PEP). The access policies would be defined in the Policy Administration Point (PAP) in Enforcement module within MUSA Security Assurance Platform.
4 Notification

4.1 Principle

The role of the notification module is to continuously report to DevOps team the status of the running multi-cloud application. This includes the notification of the potential alerts and violations of the security SLA contract detected by the monitoring mechanisms as well as the recommendation of security mitigation mechanisms that can be applied at different phases of multi-cloud application development (i.e. design phase, selection of the CSP, deployment). Thus, the notification module defines different mechanisms to suggest and elaborate responses to the detected issues and keeps the DevOps team informed on the potential risks and the different security operations applied on their sensitive data (access, migration, etc.).

Finally, we can conclude that the notification module is involved during the two phases presented in sections 2 and 3:

- To report the monitoring results of security SLAs (single and aggregated results).
- To present recommendation strategies in order to react against a security alert or violation.
- To inform the DevOps teams about the new security mechanisms enforced at runtime in automatic, semi-automatic or manual way.

4.2 MUSA Web based notification

In the context of MUSA, the notification module has been designed to inform the DevOps teams about the status of the running multi-cloud application from a security point of view by relying on a Web based report that is composed of a dashboard and several sub-reports to details information related to specific security metrics.

![Figure 16. Web based notification in MUSA Security Assurance Platform](image)

Two type of notifications has been defined during the SLA monitoring phase:

- Alerts: Alerts are notified when a security metric is about to be violated but not violated yet. This allows to the application administrator to react before a violation occurs.
- Violations: Violations occur when one of the SLOs (service level objectives) is not respected. In this case, a countermeasure is needed at different levels depending on the violation severity.

In the case of a potential alert or violation detection, a set of security enforcement mechanisms to counteract the security issue are presented to the DevOps team. These recommendations can be more or less easy to deploy depending of the severity of the detected security flaw and the multi-cloud development phase that is impacted.
Depending of the deployment strategy, the activation of the recommended security enforcement mechanism can be either:

- Automatic: In this case, the MUSA security assurance mechanism activates the security mechanism and informs the DevOps team about the activation status.
- Semi-automatic: In this case, several steps of security enforcement mechanism can be performed automatically but still the intervention of the DevOps team is required.
- Manual: In this case, which is the most common case, the DevOps team needs to manually perform the necessary step to activate the counter-measure.

Depending on the severity of the potential security flaw detected by the monitoring mechanism part of the MUSA security assurance SaaS, the reaction can impact either:

- The design phase: In this case, the application composition or/and the risk analysis to define the SLA templates need to be updated and the whole workflow of the MUSA framework needs to be redone.
- The reselection of a CSP: This can be related one or several application components. In this case and based on the new measurements performed by the monitoring tool, the CSP can recommend new CSP that better fulfils the security experts’ requirements. The redeployment of the application component and the migration of the data are required in this case.
- The deployment of new security enforcement mechanisms that will fix the security issue detected.
- The activation or reconfirmation of an already deployed security enforcement mechanism. This last option is the most likely to be automated but depends on the security control policy adopted to grant or deny the MUSA security assurance platform to remotely perform changes on configuration at running.

More technical details on the Web based notification modules are presented in the deliverable D4.2.
5 Conclusion

The document presents a state of art of existing security monitoring and enforcement mechanisms and provides first details about the monitoring agents to be integrated in the MUSA security assurance platform SaaS. It also presents two example of security enforcement agents integrated into this same platform: The high availability framework and the XACML based access control framework. These security mechanisms are integrated into the MUSA security assurance platform SaaS. This platform is an external entity that allows to monitor the multi-cloud application already deployment in different CSPs. It detects potential deviations from security SLAs and triggers counter-measures to enforce security during application runtime. The security health of the running application are continuously reported to the DevOps team thanks the notification module of MUSA. More details about the technical features of the MUSA security assurance platform SaaS are presented in D4.2 deliverable.
References


[61] The SPECS project web site. Available at: http://www.specs-project.eu/


[64] The CUMULUS project web site. Available at: http://www.cumulus-project.eu.

[65] The A4Cloud project web site. Available at: http://www.a4cloud.eu


Federated Identity solutions from Wikipedia. Available at: https://en.wikipedia.org/wiki/Federated_identity (Retrieved November 2016)

Comparison of cryptographic libraries from Wikipedia. Available at: https://en.wikipedia.org/wiki/Comparison_of_cryptography_libraries (Retrieved November 2016)

The SPECS project deliverable D4.2.2. Available at: http://www.specs-project.eu/publications/public-deliverables/d4-2-2/ (Retrieved November 2016)


Corosync. Available at: http://corosync.github.io (Retrieved November 2016)

Pacemaker. Available at: http://clusterlabs.org/ (Retrieved November 2016)

Consul_template. Available at: https://github.com/hashicorp/consul-template (Retrieved November 2016)

HAproxy. Available at: http://www.haproxy.org/ (Retrieved November 2016)
[87] Nginx. Available at: https://nginx.org/ (Retrieved November 2016)
[88] Docker. Available at: https://www.docker.com/ (Retrieved November 2016)
[91] The OpenID Connect 1.0. Available at: http://openid.net/connect/ (Retrieved November 2016)
[92] The OAuth 2.0 specification. Available at: https://oauth.net/2/ (Retrieved November 2016)
[93] JWT.IO for JSON Web Tokens. Available at: https://jwt.io/ (Retrieved November 2016)
Appendix A. MUSA motivation and background

The main goal of MUSA is to support the security-intelligent lifecycle management of distributed applications over heterogeneous cloud resources, through a security framework that includes: a) security-by-design mechanisms to allow application self-protection at runtime, and b) methods and tools for the integrated security assurance in both the engineering and operation of multi-cloud applications.

MUSA overall concept is depicted in the figure below.

**Figure A.1: MUSA overall concept**

MUSA framework combines 1) a preventive security approach, promoting Security by Design practices in the development and embedding security mechanisms in the application, and 2) a reactive security approach, monitoring application runtime to mitigate security incidents, so multi-cloud application providers can be informed and react to them without losing end-user trust in the multi-cloud application. An integrated coordination of all phases in the application lifecycle management is needed in order to ensure the preventive oriented security to be embedded and aligned with reactive security measures.
Appendix B. The MUSA Metric Catalogue

Table 1 reports the metrics collected so far, for which the following information is given:

- Metric ID: a unique metric identifier;
- Metric Name: an intuitive name for the metric;
- Abstract metric ID: the ID of the Abstract metric that the metric inherits from (see Table 2 for a description of the Abstract metrics considered);
- Applicability: the applicability domain of the metric (see Section 4.2);
- Description: a description of the metric;
- Value range: the type and range of values that the metric can assume;
- Unit: the unit of measure of the metric, if applicable. The unit can be a percentage (%), a level (in case of qualitative metrics that can assume a discrete range of values), a cost (computed in Euros or “Euros per event”), a time period (days, hours) or the number of occurrences of a given event/action/item.
- Value description: for qualitative metrics assuming only a discrete (level) value, the meaning of the different levels is described;
- Default: the (possible) default value of the metrics;
- Operator if used in SLO: the operator that may be used when adopting the metric to define and SLO in an SLA.

<table>
<thead>
<tr>
<th>Metric ID</th>
<th>Metric Name</th>
<th>Abstract Metric ID</th>
<th>Applicability</th>
<th>Description</th>
<th>Value Range</th>
<th>Unit</th>
<th>Value Description</th>
<th>Default</th>
<th>Operator if used in SLO</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_AV</td>
<td>Data availability</td>
<td>AV</td>
<td>Data protection</td>
<td>Percentage of time in which data access is available to data owners.</td>
<td>0 ≤ int ≤ 100</td>
<td>%</td>
<td>n/a</td>
<td>ge (≥)</td>
<td></td>
</tr>
<tr>
<td>SERVICE_AV</td>
<td>Service availability</td>
<td>AV</td>
<td>SW component/system protection</td>
<td>Percentage of time in which service access is available to users.</td>
<td>0 ≤ int ≤ 100</td>
<td>%</td>
<td>n/a</td>
<td>ge (≥)</td>
<td></td>
</tr>
<tr>
<td>COMPENSATORY_COST</td>
<td>Total expenses due to compensatory damages</td>
<td>COST</td>
<td>management</td>
<td>This metric indicates the total expenses incurred due to compensatory damages.</td>
<td>int &gt; 0</td>
<td>euro</td>
<td>n/a</td>
<td>leq(&lt;=)</td>
<td></td>
</tr>
<tr>
<td>AVG_COMPENSATORY_COST</td>
<td>Average expenses due to compensatory damages</td>
<td>COST</td>
<td>management</td>
<td>This metric indicates the average expenses due to compensatory damages per upheld complaint/incident</td>
<td>int &gt; 0</td>
<td>euro</td>
<td>n/a</td>
<td>leq(&lt;=)</td>
<td></td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
<td>Type</td>
<td>Units</td>
<td>Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
<td>-------------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INCIDENT S_COST</td>
<td>Cost of Incidents</td>
<td>COST</td>
<td>euro</td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVG_INCIDENT_COST</td>
<td>Mean Cost of Incidents</td>
<td>COST</td>
<td>int&gt;0</td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVG_INCIDENT_RECOV_COST</td>
<td>Mean Incident Recovery Cost</td>
<td>COST</td>
<td>n/a</td>
<td>leq(&lt;=)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVG_PATCH_COST</td>
<td>Mean Cost to Patch</td>
<td>COST</td>
<td>int&gt;0</td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INFR_DATALOCATION</td>
<td>Datacenter Location</td>
<td>DL</td>
<td>geographical zone</td>
<td>eq(=)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES_UTILIZATION_EFF</td>
<td>Resource Utilization Efficiency</td>
<td>EFF</td>
<td>0 ≤ int ≤ 100 %</td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA_WS_ON</td>
<td>Write-Serializability Activation</td>
<td>FA</td>
<td>yes / no</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA_RF_ON</td>
<td>Read-Freshness Activation</td>
<td>FA</td>
<td>yes / no</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FWSEC_ON</td>
<td>Forward Secrecy Activation</td>
<td>FA</td>
<td>yes / no</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metric</td>
<td>Category</td>
<td>Source</td>
<td>Value</td>
<td>Details</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>--------</td>
<td>-------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSTS_ON</td>
<td>HTTP Strict Transport Security Activation</td>
<td>FA</td>
<td>Communication protection</td>
<td>yes / no</td>
<td>n/a</td>
<td>yes</td>
<td>eq (=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTTPS_ON</td>
<td>HTTPS to HTTPS Redirect Activation</td>
<td>FA</td>
<td>Communication protection</td>
<td>yes / no</td>
<td>n/a</td>
<td>yes</td>
<td>eq (=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEC_COOKIES_ON</td>
<td>Secure Cookie Enforcement</td>
<td>FA</td>
<td>Communication protection</td>
<td>yes / no</td>
<td>n/a</td>
<td>yes</td>
<td>eq (=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CERT_PIN_ON</td>
<td>Certificate Pinning Activation</td>
<td>FA</td>
<td>Communication protection</td>
<td>yes / no</td>
<td>n/a</td>
<td>no</td>
<td>eq (=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VULN_SCAN_FREQ</td>
<td>Vulnerability Scanning Frequency</td>
<td>FOP</td>
<td>SW component/system protection</td>
<td>int &gt; 0 hours</td>
<td>24</td>
<td>eq (=)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VULN_LIST_UPD_FREQ</td>
<td>Vulnerability List Update Frequency</td>
<td>FOP</td>
<td>SW component/system protection</td>
<td>int &gt; 0 hours</td>
<td>24</td>
<td>eq (=)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW_UPDATE_CHECK_FREQ</td>
<td>SW Update Check Frequency</td>
<td>FOP</td>
<td>SW component/system protection</td>
<td>int &gt; 0 hours</td>
<td>24</td>
<td>eq (=)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUDIT_RECORD_FREQ</td>
<td>Audit Record Generation Frequency</td>
<td>FOP</td>
<td>SW component/system protection</td>
<td>int &gt; 0 days</td>
<td>7</td>
<td>ge (&gt;=)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metric</td>
<td>Property</td>
<td>Data Protection</td>
<td>Yes / No</td>
<td>N/A</td>
<td>Yes / No</td>
<td>Eq (=)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>----------</td>
<td>-----</td>
<td>----------</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>K_ANONIMITY_ON</strong></td>
<td>K-anonymity</td>
<td>This metric ensures that, given person-specific field-structured data, the individuals who are the subjects of the data cannot be re-identified while the data remain practically useful. Data is said to have the k-anonymity property if the information for each person contained in the release cannot be distinguished from at least k-1 individuals whose information also appear in the release.</td>
<td>yes</td>
<td>n/a</td>
<td>no</td>
<td>eq (=)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>L_DIVERSITY_ON</strong></td>
<td>L-diversity</td>
<td>This metric ensures l-diversity, which is a form of group based anonymization that is used to preserve privacy in data sets by reducing the granularity of a data representation. This reduction is a trade off that results in some loss of effectiveness of data management or mining algorithms in order to gain some privacy. The l-diversity model is an extension of the k-anonymity model which reduces the granularity of data representation using techniques including generalization and suppression such that any given record maps onto at least k other records in the data.</td>
<td>yes</td>
<td>n/a</td>
<td>no</td>
<td>eq (=)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>T_CLOSENESS_ON</strong></td>
<td>T-closeness</td>
<td>This metric ensures the t-closeness feature, which is a further refinement of l-diversity group based anonymization that is used to preserve privacy in data sets by reducing the granularity of a data representation. This reduction is a trade off that results in some loss of effectiveness of data management or mining algorithms in order to gain some privacy. The t-closeness model extends the l-diversity model by treating the values of an attribute distinctly by taking into account the distribution of data values for that attribute.</td>
<td>yes</td>
<td>n/a</td>
<td>no</td>
<td>eq (=)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONFIDENCE LEVEL</td>
<td>Level of confidentiality</td>
<td>LOC</td>
<td>Data protection</td>
<td>This metric indicates the level of confidentiality achieved by a system regarding client data independently of the means used to achieve this objective.</td>
<td>0 ≤ int level ≤ 4</td>
<td>(level)</td>
<td>Level 0 – Data confidentiality does not satisfy any of the next levels. Level 1 – Data may be accessible by the cloud provider personnel for regular operational purposes, under the control of an authentication, authorization and accounting (AAA) mechanism. Level 2 – Technical and organizational measures are in place so that data may only be accessible to privileged CSP personnel (administrators) for debugging or maintenance purposes, under the control of an AAA mechanism. Level 3 – Technical and organizational measures are in place so that data is only accessible to privileged CSP personnel to respond to law enforcement or extraordinary requests made by the client, under the control of an AAA mechanism.</td>
<td>0</td>
<td>ge (&gt;=)</td>
</tr>
<tr>
<td><strong>KEY EXPOSURE LEVEL</strong></td>
<td><strong>Key Exposure Level</strong></td>
<td><strong>LOC</strong></td>
<td><strong>Data protection</strong></td>
<td><strong>This metric indicator of key exposure to reflect the level of confidentiality afforded to cryptographic secrets, from a cloud client point of view.</strong></td>
<td><strong>0 ≤ int ≤ 4</strong></td>
<td><strong>(level)</strong></td>
<td><strong>0 ≤ L ≤ 4</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------</td>
<td>--------</td>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>------------</td>
<td>----------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level 0 – Access to decrypted data or cryptographic secrets is available to specific personnel of the CSP, for administrative or debugging purposes only.</td>
<td>Level 1 – Access to decrypted data or cryptographic secrets is available to specific personnel of the CSP, for administrative or debugging purposes only.</td>
<td>Level 2 – Access to decrypted data or cryptographic secrets is available to specific personnel of the CSP, for administrative or debugging purposes only. It is governed by the principle of dual control and split knowledge.</td>
<td>Level 3 – Access to decrypted data or cryptographic secrets is not available to any personnel of the CSP.</td>
<td><strong>0 ≤ int ≤ 4</strong></td>
<td><strong>Level 0</strong></td>
<td><strong>0 ≤ L ≤ 4</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
available to specific personnel of the CSP in exceptional circumstances only. It is governed by the principle of dual control and split knowledge, under the supervision of a hardware security module.

Level 4 – Cryptographic secrets needed to decrypt the data are known to the cloud client only.

<p>| <strong>ACC_TRAINING_LEVEL</strong> | <strong>Account of Privacy and Security Training</strong> | <strong>LOC management</strong> | <strong>This metric describes the quality of the accounts given with respect to the privacy training and awareness programs in place.</strong> | <strong>0 ≤ int ≤ 4</strong> | <strong>(level)</strong> | <strong>Level 0 – No records of training are maintained. Level 1 – Records of training sessions are maintained, but there is no evidence of individual attendance. Level 2 – Individual records of attendance are maintained. Level 3 – Individual evaluation of the training contents is performed and recorded. Level 4 – The training program includes automated procedures for recording attendance as well as for evaluating personnel individually.</strong> | <strong>0 ge (&gt;=)</strong> |</p>
<table>
<thead>
<tr>
<th><strong>DATA_ISO,</strong> <strong>TESTING</strong> <strong>LEVEL</strong></th>
<th><strong>LOC</strong></th>
<th><strong>DATA</strong> <strong>PROTECTION,</strong> <strong>MANAGEMENT</strong></th>
<th><strong>This metric describes the level of testing that has been done by the cloud provider to assess how well data isolation is implemented.</strong></th>
<th><strong>0 ≤ int ≤ 3 (level)</strong></th>
<th><strong>Level 0 – No data isolation testing has been performed. Level 1 – Read/write isolation has been tested. Level 2 – Secure deletion has been tested, in addition to read/write isolation. Level 3 – Absence of known side channel attacks has been tested, in addition to read/write and secure deletion.</strong></th>
<th><strong>0 ge (&gt;=)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CONSENT</strong> <strong>TYPE</strong></td>
<td><strong>Type of Consent</strong></td>
<td><strong>LOC</strong></td>
<td><strong>DATA PROTECTION,</strong> <strong>MANAGEMENT</strong></td>
<td><strong>This metric describes the type of consent obtained for collecting, using and sharing private data. The type of consent can be ranked in levels according to its preference.</strong></td>
<td><strong>0 ≤ int ≤ 3 (level)</strong></td>
<td><strong>Level 0 – No Consent: Consent is not obtained at or before collection of private data. Level 1 – Implied Consent: The consent is inferred from the behaviour of the data subject, or even from failing to explicitly object. No opt-out or opt-in mechanisms are offered. Level 2 – Opt-out Consent: Data subjects can take measures for prevent the collection of private data, but no opt-in mechanisms are offered. Level 3 – Opt-in Consent: Data subjects explicitly</strong></td>
</tr>
<tr>
<td>NOTICE_TYPE</td>
<td>Type of notice</td>
<td>LOC</td>
<td>Data protection, management</td>
<td>This metric describes the type of privacy notice provided by the collecting organization, depending on how the privacy notice is offered to the data subjects. Ideally, multi-layer notice should be provided so data subjects have the information necessary to make decisions at any point in time.</td>
<td>0 ≤ int ≤ 2 (level)</td>
<td>Level 1 - Single notice: The organization provides only a single document describing the privacy notice. Level 2 - Multi-layer notice: The organization provides different layers of notices. Each layer can present different degrees of information, as long as the union of all the layers is compliant with applicable privacy regulations.</td>
</tr>
</tbody>
</table>

| DATA_SUBJECT_ACCESS_PROCEDURES | Procedures for Data Subject Access Requests | LOC | Data protection, management | This metric describes the quality of the procedures in place for guaranteeing data subjects’ access to their personal information. | 0 ≤ int ≤ 3 (level) | Level 0 - No procedures are established for permitting data subject access to their personal information. Level 1 - Procedures for data subject access exist but are not documented or consistent. Level 2 - Documented and consistent processes for data subject access are established. Employees responsible of such procedures |

| | | | | | | | | |
are identified and trained on how to respond to requests. There also exist procedures for handling denial of access. Level 3 - Automated and self-service procedures for data subject access are in place, including the case of denied access.

<p>| PRI_NOTICE_READABILITY | Readability (Flesch Reading Ease Test) | LOC | manage ment | This metric describes quantitatively the level of readability of a given text, computed from the number of sentences, words and syllables. This is of interest for assessing readability of privacy notices and notifications, which should be written in a clear and concise way. This metric is known as the Flesch Reading Ease Test, and is widely utilized for evaluating readability. | 0 ≤ real ≤ 100 | n/a | 90.0–100.0: Very easy to read. Easily understood by an average 11-year-old student. 80.0–90.0: Easy to read. Conversational English for consumers. 70.0–80.0: Fairly easy to read. 60.0–70.0: Easily understood by 13- to 15-year-old students. 50.0–60.0: Fairly difficult to read. 30.0–50.0: Difficult to read. 0.0–30.0: Very difficult to read. Best understood by university graduates. | 70 | ge (&gt;=) |</p>
<table>
<thead>
<tr>
<th>PRI_RESP_LEVEL</th>
<th>Rank of Responsibility for Privacy</th>
<th>LOC</th>
<th>management</th>
<th>This metric describes numerically at what level within the organization hierarchy the person responsible for privacy is located.</th>
<th>int &gt; 0 (level)</th>
<th>n/a</th>
<th>eq (=)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG_INTEGRITY_LEVEL</td>
<td>Log Unalterability</td>
<td>LOC</td>
<td>SW component/system protection</td>
<td>This metric describes the level of protection of the log management systems against tampering.</td>
<td>0 ≤ int ≤ 2 (level)</td>
<td>Level 0 – No integrity mechanisms are in place. Level 1 – Log integrity is protected only by access control measures. Level 2 – Cryptographic mechanisms are in place for guaranteeing log unalterability or WORM (Write Once Read Many) devices are used.</td>
<td>0 ge (&gt;=)</td>
</tr>
<tr>
<td>AUTHENTICATION_LEVEL</td>
<td>Identity Assurance</td>
<td>LOC</td>
<td>SW component/system protection</td>
<td>This metric describes the quality of the authentication mechanisms in place.</td>
<td>0 ≤ int ≤ 4 (level)</td>
<td>Level 0 – No authentication mechanisms are in place. Level 1 – Simple challenge response mechanisms are allowed and no identity proofing is required. Level 2 – Single factor remote network authentication is required; in this case, authentication is successful if the claimant proves control of the authentication token through a secure</td>
<td>0 ge (&gt;=)</td>
</tr>
</tbody>
</table>
**D4.1: Initial security assurance mechanisms and tools v1.1**

| INCIDENT NOTIFICATION | Type of incident notification | LOC. management | This metric describes the quality of the notification procedures after a privacy incident or breach. | 0 ≤ int(lev e) ≤ 3 | Level 0 – No notification of privacy incidents is done, or it is done inconsistently. Level 1 – General notification, usually as a public notice. Affected users may not be aware of the incident. Level 2 – Individual notification to each affected user. Level 3 – Automated and self- | 0 ≤ ge(>=) | 53 |

- Level 3 – Multifactor authentication mechanisms are in place. Proofs of control of the authentication token are done through a cryptographic protocol.
- Level 4 – Multifactor authentication with a hardware cryptographic token is required. Strong cryptographic mechanisms are required along physical tokens with a FIPS 140-2 level greater than 2, and identity proofing is done in person.
| CRYPTOGRAPHIC STRENGTH | LOC | DATA PROTECTION, COMMUNICATION PROTECTION | It is a measure of the strength of a cryptosystem in terms of the expected number of operations required to defeat the underlying cryptographic mechanism. The values (level 1-8) are based on ECRYPT II recommendations 2012:  
<table>
<thead>
<tr>
<th>Metric</th>
<th>Abbreviation</th>
<th>Definition</th>
<th>Calculation</th>
<th>Units</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOR</td>
<td>Level of Redundancy</td>
<td>Represents the number of replicas of a software component that are set-up and kept active at the same time during system operation</td>
<td>int &gt; 0</td>
<td>n/a</td>
<td>1</td>
<td>ge (≥)</td>
</tr>
<tr>
<td>LOD</td>
<td>Level of Diversity</td>
<td>Represents the number of different software / hardware replicas of a software component that are set-up and kept active at the same time during system operation</td>
<td>int &gt; 0</td>
<td>n/a</td>
<td>1</td>
<td>ge (≥)</td>
</tr>
<tr>
<td>MTB INCIDENTS</td>
<td>Mean time between incidents</td>
<td>This metric measures the mean time between two subsequent incidents</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTT_REV_OKE_USERS</td>
<td>Mean time to revoke users</td>
<td>This attribute describes quantitatively how fast an organization revokes users’ access to data and organizationally-owned or managed (physical and virtual) applications, infrastructure systems, and network components, based on user’s change in status (e.g., termination of employment or other business relationship, job change or transfer).</td>
<td>int &gt; 0</td>
<td>hour(s)</td>
<td>n/a</td>
<td>le (≤)</td>
</tr>
<tr>
<td>MTT_RESPOND_COMPLAINTS</td>
<td>Mean time to respond to complaints</td>
<td>This metric indicates the average time that the organization takes for responding to complaints from stakeholders.</td>
<td>int &gt; 0</td>
<td>hour(s)</td>
<td>n/a</td>
<td>le (≤)</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
<td>Time to</td>
<td>MTTC</td>
<td>SW Component</td>
<td>System Protection</td>
<td>Data Protection</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>---------</td>
<td>------</td>
<td>---------------</td>
<td>-------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>MTT_REACT</td>
<td>Timeliness of reaction</td>
<td>MTTC</td>
<td>SW Component/System Protection</td>
<td>This metric indicates the average time to react to a security incident</td>
<td>int&gt;0</td>
<td>seconds</td>
</tr>
<tr>
<td>MTT_CRITICAL_PATCH</td>
<td>Mean Time to Deploy Critical Patch(es)</td>
<td>MTTC</td>
<td>SW Component/System Protection</td>
<td>The metric measures the average time taken to deploy a critical patch to the organization’s technologies</td>
<td>int&gt;0</td>
<td>seconds</td>
</tr>
<tr>
<td>MTT_DETECT_INCIDENT</td>
<td>Mean Time to Incident Discovery</td>
<td>MTTC</td>
<td>SW Component/System Protection</td>
<td>This metric measures the effectiveness of the organization in detecting security incidents, as it returns the average time to detect incidents</td>
<td>int&gt;0</td>
<td>seconds</td>
</tr>
<tr>
<td>MTT_RECOVER</td>
<td>Mean Time to Incident Recovery</td>
<td>MTTC</td>
<td>SW Component/System Protection</td>
<td>This metric measures the effectiveness of the organization to recovery from security incidents</td>
<td>int&gt;0</td>
<td>seconds</td>
</tr>
<tr>
<td>MTT_PAT</td>
<td>Mean Time to Patch</td>
<td>MTTC</td>
<td>SW Component/System Protection</td>
<td>This metric measures the average time taken to deploy a patch to the organization’s technologies</td>
<td>int&gt;0</td>
<td>seconds</td>
</tr>
<tr>
<td>MTT_COMPLETE_CHANGES</td>
<td>Mean Time to Complete Change</td>
<td>MTTC</td>
<td>SW Component/System Protection</td>
<td>The average time it takes to complete a configuration change request.</td>
<td>int&gt;0</td>
<td>seconds</td>
</tr>
<tr>
<td>Metric</td>
<td>Definition</td>
<td>Unit</td>
<td>Lower Bound</td>
<td>Upper Bound</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>------</td>
<td>-------------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MTT</strong> CONTAINMENT <strong>MTTC</strong></td>
<td>Mean Time from Discovery to Containment</td>
<td></td>
<td>int &gt; 0</td>
<td>seconds</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PRI_PROG_UPDTS</strong></td>
<td>Privacy Program Updates</td>
<td>NOE</td>
<td>int &gt; 0</td>
<td>(updates)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PERIODIC_PRI_ASSESSMENTS</strong></td>
<td>Periodicity of Privacy Impact Assessments for Information Systems</td>
<td>NOE</td>
<td>int &gt; 0</td>
<td>(assessments)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CERTIFICATIONS</strong></td>
<td>Frequency of certifications</td>
<td>NOE</td>
<td>int &gt; 0</td>
<td>(certifications)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RCVD_PRI_V_AUDITS</strong></td>
<td>Number of privacy audits received</td>
<td>NOE</td>
<td>int &gt; 0</td>
<td>n/a</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DATA_SU_ACCESS_RQSTS</strong></td>
<td>Number of Data Subject Access Requests received during a given period of time</td>
<td>NOE</td>
<td>int &gt; 0</td>
<td>(requests)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>COMPLAINTS_NUMB</strong></td>
<td>Number of complaints received during a given period of time.</td>
<td>NOE</td>
<td>int &gt; 0</td>
<td>(complaints)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
<td>Calculation</td>
<td>Unit</td>
<td>Notes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
<td>------</td>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRI_INCIDENT_NUMB</td>
<td>Number of privacy incidents</td>
<td>int &gt; 0 (incidents)</td>
<td>n/a</td>
<td>le (&lt;=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3RD-PARTY_PRIVINCIDENTS_NUMB</td>
<td>Privacy incidents caused by third parties</td>
<td>int &gt; 0 (incidents)</td>
<td>n/a</td>
<td>le (&lt;=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAMAGE_INCIDENTS</td>
<td>Incidents with damages</td>
<td>int &gt; 0 (incidents)</td>
<td>n/a</td>
<td>ge (&gt;=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INCIDENTS</td>
<td>Number of Incidents</td>
<td>int &gt;=0</td>
<td>n/a</td>
<td>le (&lt;=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VULNERABILITIES</td>
<td>Number of Vulnerabilities</td>
<td>int&gt;=0</td>
<td>n/a</td>
<td>le (&lt;=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TESTED_BCR_PLANS</td>
<td>Number of Business Continuity Resilience (BCR) plans tested</td>
<td>int &gt; 0 (plans)</td>
<td>n/a</td>
<td>le (&lt;=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SANCTIONS</td>
<td>Sanctions</td>
<td>int &gt; 0 (sanctions)</td>
<td>n/a</td>
<td>eq (=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metric Code</td>
<td>Description</td>
<td>Metric Type</td>
<td>Min Value</td>
<td>Max Value</td>
<td>Category</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>-----------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>DATA_ENCRYPTION_PERC</td>
<td>Data encryption percentage</td>
<td>PERC</td>
<td>0</td>
<td>100</td>
<td>≥ (&gt;=)</td>
<td></td>
</tr>
<tr>
<td>DATE_INDICATION_PERC</td>
<td>Record of Data Collection, Creation, and Update</td>
<td>PERC</td>
<td>0</td>
<td>100</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>DATA_CLASSIFICATION_PERC</td>
<td>Data classification</td>
<td>PERC</td>
<td>0</td>
<td>100</td>
<td>≥ (&gt;=)</td>
<td></td>
</tr>
<tr>
<td>PII_AUTHORIZ_PERC</td>
<td>Authorized collection of PII</td>
<td>PERC</td>
<td>0</td>
<td>100</td>
<td>≥ (&gt;=)</td>
<td></td>
</tr>
<tr>
<td>PRI_BUDGET_PERC</td>
<td>Privacy Program Budget</td>
<td>PERC</td>
<td>0</td>
<td>100</td>
<td>≥ (&gt;=)</td>
<td></td>
</tr>
<tr>
<td>PRI_TRAINING_PERC</td>
<td>Coverage of Privacy and Security Training</td>
<td>PERC</td>
<td>0</td>
<td>100</td>
<td>≤ (&lt;=)</td>
<td></td>
</tr>
<tr>
<td>NOTIFIED_PRI INCIDENTS_PERC</td>
<td>Coverage of incident notifications</td>
<td>PERC</td>
<td>0</td>
<td>100</td>
<td>≥ (&gt;=)</td>
<td></td>
</tr>
<tr>
<td>MITIGATED_HIGH_VULNS_PERC</td>
<td>Mitigation of High Vulner</td>
<td>PERC</td>
<td>0</td>
<td>100</td>
<td>≥ (&gt;=)</td>
<td></td>
</tr>
<tr>
<td>Measure</td>
<td>Description</td>
<td>Metric</td>
<td>Lower Bound</td>
<td>Upper Bound</td>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>Remote Access Control Measure</td>
<td>SW component/system protection, management</td>
<td>It measures the percentage (%) of remote access points used to gain unauthorized access.</td>
<td>$0 \leq \text{int} \leq 100$</td>
<td>%</td>
<td>n/a</td>
<td>le ($&lt;=$)</td>
</tr>
<tr>
<td>User Accounts Measure</td>
<td>SW component/system protection, management</td>
<td>It measures the percentage (%) of users with access to shared accounts.</td>
<td>$0 \leq \text{int} \leq 100$</td>
<td>%</td>
<td>n/a</td>
<td>le ($&lt;=$)</td>
</tr>
<tr>
<td>Incident Response Measure</td>
<td>SW component/system protection, management</td>
<td>It measures the percentage (%) of incidents reported within required time frame per applicable incident category.</td>
<td>$0 \leq \text{int} \leq 100$</td>
<td>%</td>
<td>n/a</td>
<td>ge ($=&gt;$)</td>
</tr>
<tr>
<td>Remediated Vulnerability Measure</td>
<td>SW component/system protection, management</td>
<td>It measures the percentage (%) of vulnerabilities remediated within organization-specified time frames.</td>
<td>$0 \leq \text{int} \leq 100$</td>
<td>%</td>
<td>n/a</td>
<td>ge ($=&gt;$)</td>
</tr>
<tr>
<td>Mitigated OS Vulnerabilities</td>
<td>SW component/system protection, management</td>
<td>It measures the percentage (%) of operating system vulnerabilities for which patches have been applied or that have been otherwise mitigated (over total number of applicable vulnerabilities found through scans)</td>
<td>$0 \leq \text{int} \leq 100$</td>
<td>%</td>
<td>n/a</td>
<td>ge ($=&gt;$)</td>
</tr>
<tr>
<td>Vulnerability Scanning Coverage</td>
<td>SW component/system protection, management</td>
<td>It measures the percentage of the organization’s systems under management that were checked for vulnerabilities during vulnerability scanning and identification processes. This metric is used to indicate the scope of vulnerability identification efforts.</td>
<td>$0 \leq \text{int} \leq 100$</td>
<td>%</td>
<td>n/a</td>
<td>ge ($=&gt;$)</td>
</tr>
<tr>
<td>Successful Audits received</td>
<td></td>
<td>This metric describes the percentage of independent reviews and assessments performed to the policies and procedures in place for complying with applicable contractual and regulatory obligations.</td>
<td>$0 \leq \text{int} \leq 100$</td>
<td>%</td>
<td>n/a</td>
<td>ge ($=&gt;$)</td>
</tr>
<tr>
<td>Metric Name</td>
<td>Category</td>
<td>Description</td>
<td>Range</td>
<td>Minimum</td>
<td>Maximum</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------</td>
<td>---------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td><strong>Data Subject Access Requests</strong></td>
<td>PERC</td>
<td>Responder data subject access requests</td>
<td>0 ≤ int ≤ 100%</td>
<td>n/a</td>
<td>ge (=&gt;)</td>
<td></td>
</tr>
<tr>
<td><strong>Data Protection</strong></td>
<td>PERC</td>
<td>This metric describes the percentage of data subject access requests that have been responded and for which a record of the request and the response exists.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Certification of Acceptance of Responsibility</strong></td>
<td>PERC</td>
<td>This metric describes the percentage of employees who have certified their acceptance of responsibilities for activities that involve handling of private data.</td>
<td>0 ≤ int ≤ 100%</td>
<td>n/a</td>
<td>ge (=&gt;)</td>
<td></td>
</tr>
<tr>
<td><strong>Reviewed Complaints</strong></td>
<td>PERC</td>
<td>This metric indicates the percentage of complaints that have been reviewed during a given period of time.</td>
<td>0 ≤ int ≤ 100%</td>
<td>n/a</td>
<td>le (&lt;=)</td>
<td></td>
</tr>
<tr>
<td><strong>Configuration Changes</strong></td>
<td>PERC</td>
<td>It measures the percentage (%) of components that undergo maintenance in accordance with formal maintenance schedules.</td>
<td>0 ≤ int ≤ 100%</td>
<td>n/a</td>
<td>ge (=&gt;)</td>
<td></td>
</tr>
<tr>
<td><strong>Media Sanitization</strong></td>
<td>PERC</td>
<td>It measures the percentage (%) of media that passes sanitization procedures testing for FIPS 199 high impact systems.</td>
<td>0 ≤ int ≤ 100%</td>
<td>n/a</td>
<td>ge (=&gt;)</td>
<td></td>
</tr>
<tr>
<td><strong>Physical Security Incidents</strong></td>
<td>PERC</td>
<td>It measures the percentage (%) of physical security incidents that were solved in a given period of time.</td>
<td>0 ≤ int ≤ 100%</td>
<td>n/a</td>
<td>le (&lt;=)</td>
<td></td>
</tr>
<tr>
<td><strong>Percentage of Timely Incident Resolution</strong></td>
<td>PERC</td>
<td>It measures the percentage (%) of security incidents that were solved in a given period of time.</td>
<td>0 ≤ int ≤ 100%</td>
<td>n/a</td>
<td>ge (=&gt;)</td>
<td></td>
</tr>
<tr>
<td><strong>Authorized Users</strong></td>
<td>PERC</td>
<td>It measures the percentage of employees who are authorized access to information systems only.</td>
<td>0 ≤ int ≤ 100%</td>
<td>n/a</td>
<td>ge (=&gt;)</td>
<td></td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
<td>Last Request Location</td>
<td>Most Frequent Request Location</td>
<td>MTPD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------------------------</td>
<td>--------------------------------</td>
<td>------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PERSONNEL SCREENING PERC</td>
<td>Personnel Security Screening Measure</td>
<td>PERC</td>
<td>manage</td>
<td>It measures the percentage (%) of individuals screened before being granted access to organizational information and information systems.</td>
<td>0 ≤ int ≤ 100 %</td>
<td>n/a</td>
</tr>
<tr>
<td>SECURITY - CONTRACTS PERC</td>
<td>Service Acquisition Contract Measure</td>
<td>PERC</td>
<td>manage</td>
<td>It measures the percentage (%) of system and service acquisition contracts that include security requirements and/or specifications.</td>
<td>0 ≤ int ≤ 100 %</td>
<td>n/a</td>
</tr>
<tr>
<td>DEVICE PROTECTION PERC</td>
<td>System and Communication Protection Measure</td>
<td>PERC</td>
<td>manage</td>
<td>It measures the percentage of mobile computers and devices that perform all cryptographic operations using FIPS 140-2 validated cryptographic modules operating in approved modes of operation. Strategic Goal: Accelerate the development and use of an electronic information infrastructure. Information Security Goal: Allocate sufficient resources to adequately protect electronic information infrastructure.</td>
<td>0 ≤ int ≤ 100 %</td>
<td>n/a</td>
</tr>
<tr>
<td>LAST_REQUEST_LOC</td>
<td>Most recent request location</td>
<td>RL</td>
<td>Data protection, SW component/system protection</td>
<td>The metric returns the last location source of the request made by a user to the cloud service</td>
<td>geographical zone</td>
<td>n/a</td>
</tr>
<tr>
<td>MOST_FREQ_REQUEST_LOC</td>
<td>Most frequent request location</td>
<td>RL</td>
<td>Data protection, SW component/system protection</td>
<td>The metric returns the most frequency location source of the request made by a user to the cloud service</td>
<td>geographical zone</td>
<td>n/a</td>
</tr>
<tr>
<td>MTPD</td>
<td>Maximum tolerable period for disruption (MTPD)</td>
<td>TIME</td>
<td>manage</td>
<td>This metric indicates the maximum tolerable period for disruption, as defined by the organizations’ BCR plans.</td>
<td>int &gt; 0 minutes</td>
<td>n/a</td>
</tr>
<tr>
<td>APP_RSP_TIME</td>
<td>TIME</td>
<td>SW component/system protection</td>
<td>The average time (in milliseconds) to answer a specific request (can be categorized by request)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
<td>--------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>int &gt; 0</td>
<td>milli seconds</td>
<td>1 le (&lt;=)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

APP_RSP_TIME: Application response time