# A certification framework for Cloud security properties: the monitoring path

Marina Egea, Khaled Mahbub, George Spanoudakis, Maria Rosa Vieira

Department of Computer Science, City University London, London, United Kingdom
{K.Mahbub, G.E.Spanoudakis}@city.ac.uk,
Atos, Research & Innovation, Madrid, Spain
{marina.egea,maria.vieira}@atos.net

**Abstract.** In this paper we describe the structure and functionality of a certification integrated framework aimed to support the certification of security properties of a Cloud infrastructure (IaaS), a platform (PaaS), or the software layer (SaaS). Such framework will bring service users, service providers and cloud suppliers to work together with certification authorities in order to ensure security properties and certificates validity in the continuously evolving cloud environment. For this purpose, the framework relies on multiple types of evidence gathering with respect to security, e.g., testing services, monitoring agents or trusted computing proofs. In this paper we will focus only on the monitoring case and will illustrate its use. Yet, this framework is designed to be able to follow models for hybrid, incremental and multi-layer security certification since cloud security has to build upon the entire cloud stack.

## 1 Introduction

The very nature of Cloud computing makes the systems deployed in such environment more exposed than ever before. The surface of attacks targeting applications and data has expanded from Web into mobile and cloud systems. As a consequence, the rapid adoption of detection and protection mechanisms for companies' assets along with the assurance of the resources they consume or provide in Cloud has turned out in one of highest priorities for companies using cloud support. Yet, this situation has also made other companies reluctant of migrating their systems to cloud due to the feel of losing control of their systems and becoming exposed.

But, not only the increased level of exposure but also the complexity in understanding the underlying infrastructure, platform or services delivering the required functionality to the consumers have made different organizations to classify cloud security properties or risks to both guide cloud vendors in their road to meet security, and to assist prospective cloud customers in assessing the overall security risk of a cloud provider. These taxonomies usually reference the security requirements of the global ISO/IEC 27001. For instance, the Cloud Control Matrix [1] that is designed to provide fundamental security principles to guide cloud vendors and to assist prospective cloud customers in assessing the overall security risk of a cloud provider.

However, how companies or individual customers get assurance that an IT product meets its security objectives is a further question. According to Common Criteria [2],

assurance can be derived from reference to sources such as unsubstantiated assertions, prior relevant experience, specific experience, or active investigation. The framework described in this paper supports the philosophy of providing (continuous) independent assurance by active investigation. Namely, it offers support for a (constant) evaluation of an IT resource in order to determine whether certain security property hold from a given point in time on, in a particular context. The security certification scheme is so conceived to create transparency in the industry, helping business to evaluate the security risks they may accept when working with a particular cloud service provider.

*Organization.* We start by briefly summarizing off-the-shelf monitoring tools in section 2. In section 3 we explain the infrastructure and functionality of a certification framework for Cloud security. In section 4 we explain a monitoring based certification process. We illustrate this process in section 5 using an example from the e-Health domain. Taking into account the state of the art of monitoring tools, we outline future work in section 6. Finally, we draw conclusions in section 7.

## 2 State of the art of cloud monitoring tools

To the best of our knowledge, there is no other certification infrastructure similar to the framework that we describe in section 3, i.e., there is no other framework able to manage monitoring based certification, testing based certification and, in an immediate future, also TPM based and hybrid certification (to combine both monitoring and testing approaches).

Regarding cloud infrastructure monitoring, there are a number of open source versions of industrial strength cloud monitoring tools that could be used by the CUMULUS framework as external evidence collectors (once that we have also certified their reliability). First of all, we note that most of these off-the-shelf tools are dealing with performance, availability, or resource consumption. This fact leaves most of the security properties listed in [3] as not addressed by state of the art tools. Later, in section 6 we report on security properties coverage (w.r.t. the list in [3]) that seems to be reached with the tools described here. This report is based on a preliminary suitability desktop analysis of these tools to monitor evidences related to the security properties.

*Zenoss Core* [8] is an enterprise network and systems management application written in Python that can monitor availability, performance, events and configuration across layers and platforms. It has an open source version, Zen Pack, that makes possible to monitor the capacity and performance of applications running on a Cloud Foundry platform. This pack allows monitoring metrics like memory or disk consumption or utilization, CPU average usage across instances, total and running application instances, etc..

*Nagios Core* [9] is an open source system and network monitoring application originally designed to run under Linux. It watches hosts (processor load, disk usage, ...), applications, services, OS, network protocols and is able to provide notifications. Nagios allows defining macros that can be instantiated as monitoring metrics for particular

services. E.g., a time-stamp in time format indicating the time at which the service was last detected as being in an 'OK' state. Moreover, it provides a powerful API allowing easy monitoring of custom applications, services, and systems since it has a simple plugin design that allows users to easily develop their own service checks.

*Hyperic* [14] provides proactive performance management with constant visibility into applications and infrastructure, and a notification service. The key monitoring and management capabilities of Hyperic are resource discovery, metric collection, and event tracking. Regarding resource discovery, the Hyperic agent managing a platform automatically discovers the resources, e.g., architecture, RAM, CPU speed, IP address, domain name, etc., and software on the platform, e.g., web servers, application servers, database servers, etc.. Regarding metric collection, the Hyperic agent can collect metrics for the resource targeting availability, performance, utilization, and throughput for each supported resource type. Regarding event tracking, Hyperic can monitor log and configuration files and record events of interest for most server types. E.g., user logins, windows registry key changes, error logs, etc..

*New Relic* [12] delivers software as a service which monitors web and mobile applications in real-time that run in cloud, on-premise, or hybrid environments. The New Relic Platform allows user to write plugin agents using Java or Ruby that can be run anywhere to collect metrics from any available system and report them to New Relic for customized dashboard visualization. A distinctive feature of New relic is that it provides metrics from end-user monitoring, e.g. apdex score; application monitoring, e.g., reponse times, performance of external services, most time consuming transactions; database monitoring, e.g., time spent in database calls; infrastructure monitoring, e.g., server resources monitoring, analysis of CPU, disk, memory, etc.; availability and error monitoring, e.g., application availability monitoring and alerting, incident or error detection, alerting and analysis.

*Ganglia* [10] is a scalable distributed monitoring system for high-performance computing systems such as clusters and grids. Ganglia is an open-source project, that allows monitoring the following metrics: system load averages that indicate the average number of processes running on the systems; memory usage averages that indicate the average usage of memory, in the form of user process or shared memory areas, system cache, buffer or swap areas; % CPU utilization across all processes on all systems; and network bandwidth usage averages that indicates the average use of network bandwidth across the nodes.

Less oriented towards performance or availability monitoring, *OSSEC* [13] is an Open Source Host-based Intrusion Detection System that performs log analysis, file integrity checking, policy monitoring, rootkit detection, real-time alerting and active response. It has a centralized manager that receives information from agents which may be running simultaneously for different operating systems, e.g., Linux, MacOS, Solaris, HP-UX, AIX and Windows.

In contrast with the monitoring tools that we have described before, the *EVEREST* monitoring framework [7] that is the one that we are currently employing to monitor

evidences in the CUMULUS project, supports a *formal approach* for the specification and constant checking of a wide range of monitoring rules with precise time constraints, and can deal with events that may be captured and notified from distributed sources and through different communication channels. EVEREST can also support the monitoring of conditions at various levels (e.g. network and application levels).

## 3 Infrastructure of a Cloud Security Certification framework

The certification infrastructure presented here is the core layer of the CUMULUS framework [4] since it is responsible for issuing, maintaining, or revoking certificates according to the different types of certification models that it can be provided with, e.g. testing-based, monitoring-based certification models or, as a refinement, trusted computing monitoring-based certification models. More details about the use of these types of models in the framework can be found in [3,5,4]. The framework components that are described next are in charge of managing the evidence gathered to assure a certain resource's security property, and accordingly the certificates that are issued, maintain or revoked based on the collected proofs. The security properties that we aim to address with the framework are those contained in the security property vocabulary specified in CUMULUS deliverable D2.1 [3, Annex A]. Each subsection of this Annex A represents one of the control domains of the Cloud Control Matrix [1].[1]

### 3.1 Components

Next, we describe the components of the certification framework that are depicted in Figure 1.

- `Certification Manager`. This component communicates with a CA through a Management API so as the CA can upload or delete new certification models. It is also connected with the Certification and Security Models databases and it delegates the management to the testing or monitoring manager according to the type of model the CA selects through a corresponding API. Moreover, it provides information stored in Certification Models to the 'Certification Generator/Attestation' component through the 'Generation API'.
- `Certificate Generator / Attestation`. This component issues certificates when enough and appropriate evidences are collected, according to the information of the Certification Models and based on the evidences stored in the 'Evidence Database (DB)'. It also stores the issued Certificates into the 'Certificates DB'.
- `Certification Communicator`. This component provides the 'Retrieval API' that allows the actors to retrieve certificates. This component also provides a 'Notification API' to notify actors about the status of certificates. Moreover, from the 'Certificates Registry DB,' the certification communicator collects all information about the requested certificates.

[1] The list of security properties contained by deliverable D2.1 [3, Annex A] was specified by the Cloud Security Alliance that is part of the CUMULUS project consortium

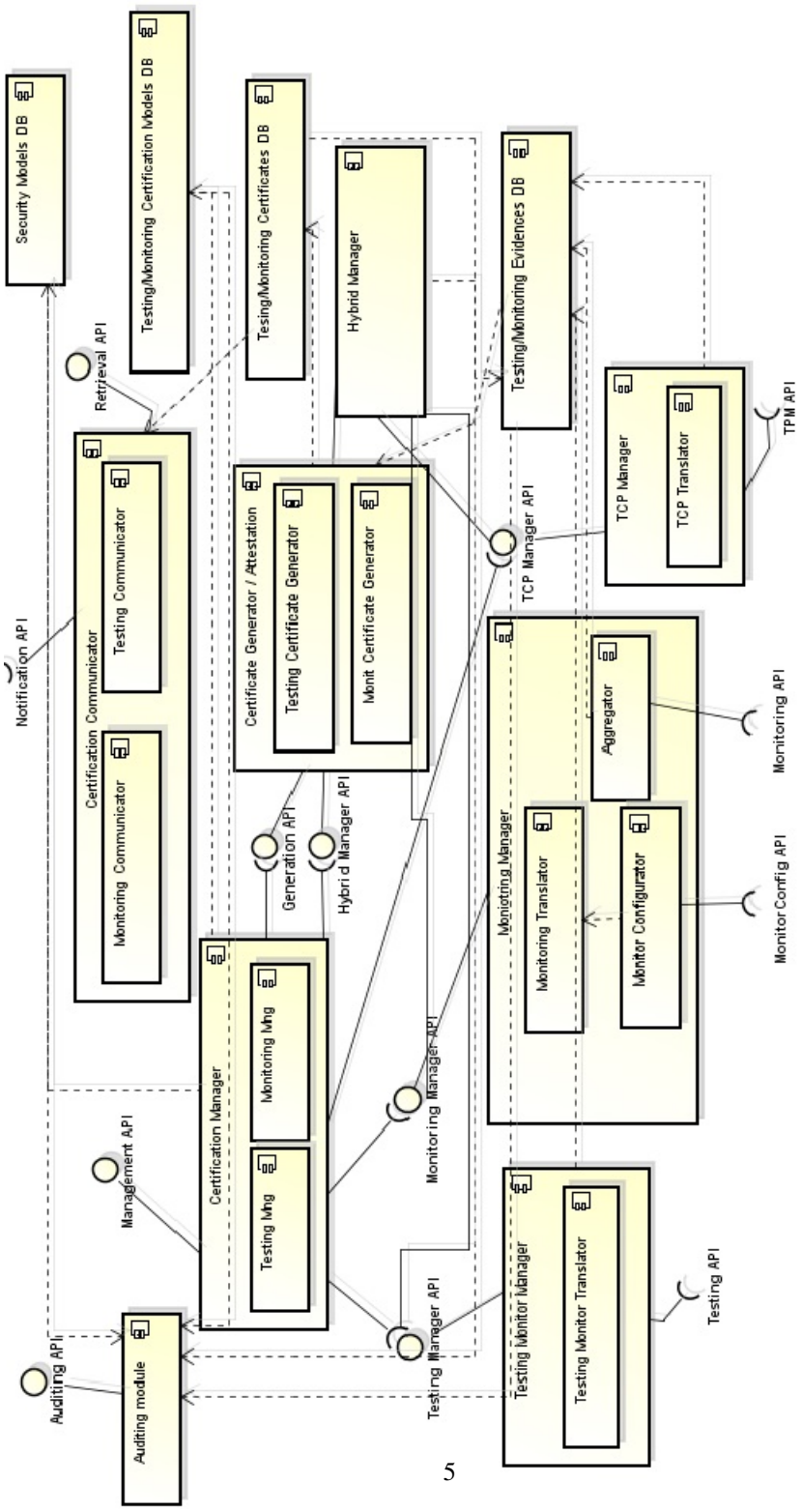


**Fig. 1.** Cumulus framework architecture

- `Certificates DB`. This is the database that stores the certificates: it acts as a blackboard architecture for the 'Certification Communicator' and the 'Certificate Generator/ Attestation' components. This repository is connected and properly related to the 'Certified Services DB.'
- `Evidence DB`. This database stores the aggregated evidence collected that sustain that a given security property holds for a resource. It plays the role of a blackboard architecture for the four types of Managers and the 'Certificate Generator/ Attestation' components.
- `Security Models DB`. This database stores the the security models defined by security experts.
- `Certification Models DB`. This database stores the certification models defined by security experts.
- `Monitoring Manager`. This component is responsible for planning and setting up the monitoring infrastructure when the 'Certification Manager' calls it through the 'Monitoring Manager API' according to a request from a Certification Model.
- `Testing Manager`. This module is responsible for planning and setting up the testing infrastructure when the Certification Manager calls it through the 'Testing Manager API', according to a request from a Certification Model.

### 3.2 Interactions and Usage

The information exchange with the certification infrastructure is mainly performed through the Retrieval API, and the Management API that are described next.

- `Management API`. This is a provided interface that enables to carry out framework management tasks, like adding and updating certification models in the 'Certification models DB', as well as requesting to issue a certificate for a specific service.
- `Retrieval API`. This is a provided interface that enables the communication with the framework: sending requests for certified components, checking the validity of certificates and getting runtime-related information.

There are other APIs, i.e., Notification API, and Auditing API that are not yet implemented but planned for future development. The `Notification API` will be an interface that deals with subscriptions and notifications for receiving certificates that fulfill the specified requirements at runtime. The `Audit API` will be an interface that allows cloud auditors to review the certification process supported by the framework.

In Figure 2 we illustrate at a high level how the process of certificate creation is supported by the interaction of the framework components, and driven through the Management API. Note that a CA directly interacts with a dashboard that is not part of the framework but allows user friendly interaction with it. The certificate creation process starts when a CA selects a resource (namely, a target of certification-TOC), e.g. a service, and a security property to be certified for this cloud resource, e.g. it stores data encrypted. For this combination, the CA also has to select a certification model (from those retrieved from the certification models database). Once these parameters have been instantiated, a certificate is requested to the certification manager that passes this request into the testing or the monitoring manager (depending on the certification

model that was selected). Either of these managers uses external but trusted cloud testing or monitoring modules to gather and store evidences that can sustain the security property chosen for the selected resource. When these evidences are sufficient, a certificate is generated by the Certification Generator component that, in addition, stores the new certificate that it has generated based on the evidences. In the following sections, we will provide more details of the certification process, placing the focus on the monitoring case.

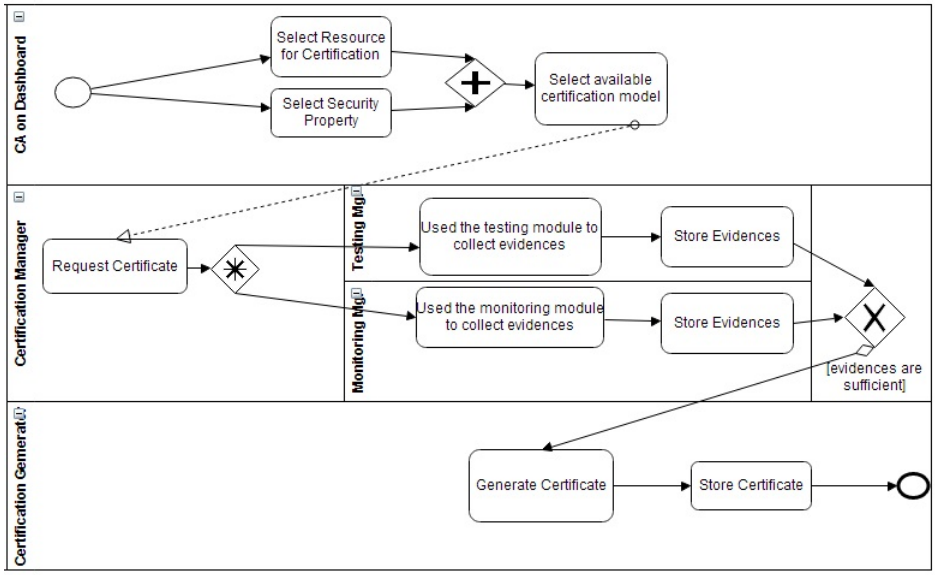


**Fig. 2.** Certificate creation process

We describe next the stakeholders that were identified for the CUMULUS framework [4]. Ultimately, all of them have to interact with the infrastructure explained in this section. *Cloud Service Consumers* use the framework either to develop an application fulfilling some security requirements, to retrieve a certified service, or to check the validity of a certificate; *Certification Authorities (CA)* use the framework either to issue a certificate for cloud services, or to define or validate a certification model (that can be used afterwards to certify various services); *Cloud Service Providers* use the framework either to specify a service to be certified, to issue a certificate for a service, or to define or check the validity of a certification model; *Cloud Auditors* use the framework to check the security compliance of a given application based on the documentation provided by the framework; and *Security Experts* that use the framework to define security properties and related certification requirements for a certain domain.

## 4 Monitoring based certificate creation process

The certification management process in the case of monitoring based certificates is handled by the monitoring manager. The whole process is driven by the monitoring based certification model (MBCM) that is passed to the manager. This model is defined in XML according to the schema shown in Figure 3. A MBCM specifies:

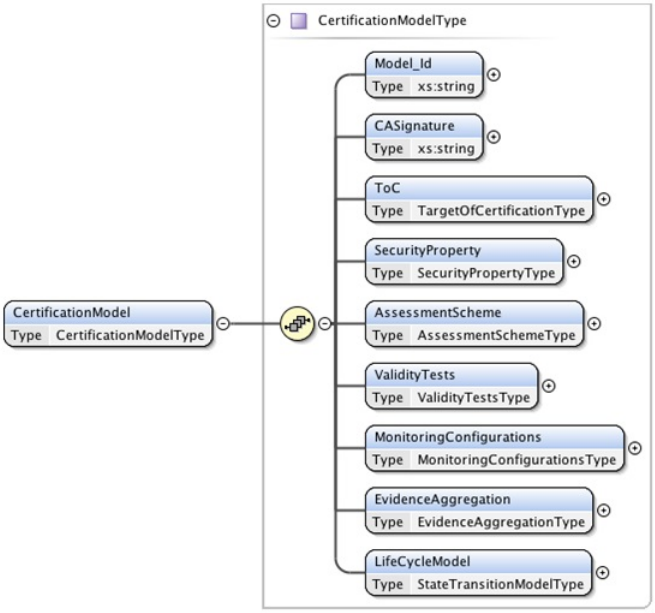


**Fig. 3.** MBCM Overall schema

1. the cloud service to be certified (i.e., a Target of Certification (ToC));
2. the security property to be certified for ToC;
3. the certification authority who will sign the certificates generated by the model;
4. an assessment scheme defining general conditions regarding the evidence that must be collected for being able to issue a certificate;
5. validity tests regarding the configuration of the cloud provider and the CUMULUS framework itself that should be satisfied before issuing certificates;
6. the monitoring configurations that will be used in order to collect the evidence required for generating certificates;
7. the way in which the collected evidence will be aggregated in certificates (evidence aggregation); and
8. a life cycle model that defines the overall process of issuing certificates.

The assessment scheme in an MBCM (see Figure 4) defines general conditions regarding the evidence that must be collected in order to be able to issue and maintain a certificate according to the particular MBCM. These conditions are related to:

i. the sufficiency of the collected evidence collection (e.g., minimum period over which a target of certification must be monitored before a certificate for the particular property of it can be issued) and are specified by evidence sufficiency condition elements,
ii. the absence of conflicting evidence regarding the property to be certified, i.e., evidence that the security property of interest would have been violated if the assessment were to be made over a different time period (specified as conflict elements), and
iii. the absence of any anomalies, i.e., indications of potential attacks or operating conditions which despite of not having affected yet the security property that is assessed, may do so in the future (specified by anomalies elements).

These conditions must be satisfied, in addition to the guarantee states that are part of the assertion definition of the property, for the certificate to be issued. [2]

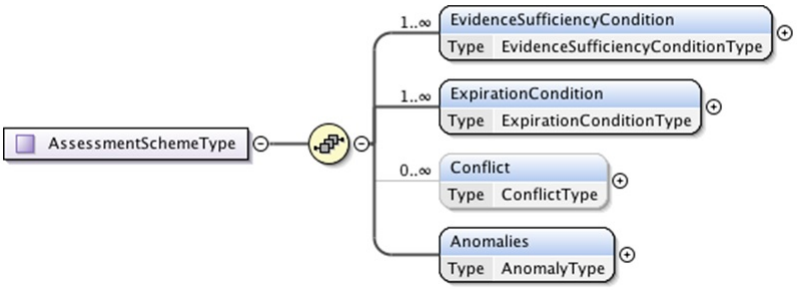


**Fig. 4.** MBCM Assessment Scheme

Once it receives an MBCM, the monitoring manager checks whether the security property that is to be certified can be monitored for the given TOC. This is performed by parsing the property, creating an abstract syntax tree (AST) for it and using AST to establish whether the TOC is placed on a cloud infrastructure that can provide the raw monitoring events required in order to check the security property. This check is performed by checking the monitoring capabilities of the cloud where TOC is placed [11,6]. If this check is successful, the monitoring manager creates a concrete monitoring configuration as part of the MBCM, i.e., a description of the event captors and the monitor that will be used to monitor the property and the subscriptions that will be required in order to enable the communication of events from the former to the latter. The transmission of events from event captors to the monitor of the CUMULUS framework takes place through an event bus (events are encrypted). Depending on the validity tests specified in MBCM, the monitoring manager may also require that the event captors and the event bus used for the capture and transmission of monitoring events run also

[2] A full description of the schema for specifying MBCMs is available in [5].

on infrastructures that are enabled by a trusted platform module (TPM) and that their implementations at the outset have integrity. Subsequently, the monitoring manager initiates the monitoring process by: (a) activating the event captors, and (b) translating the security property that is to be certified into monitorable assertions and passing them to the monitor to start the runtime checking. Following the above, steps the monitoring manager polls the monitor at regular intervals to collect any monitoring results, which are relevant to the security property that is being assessed. These results are stored in the evidence DB of the framework.

When the collected evidence is sufficient for making an assessment about the property, i.e., it satisfies the evidence sufficiency conditions of the MBCM, two further checks are performed to establish if a certificate can be issued for the TOC:

1. A check of whether the collected evidence shows any violations of the monitored security property;
2. A check of whether any additional validity conditions specified in the MBCM are satisfied. Such conditions may, for example, require a TPM-enabled confirmation of the integrity of the components used to collect and analyse the monitoring evidence, as well as the components of the CUMULUS framework itself throughout the monitoring process.
3. A check of whether any detected anomalies and conflicts are acceptable to the certification authority that will sign of the certificate.

These checks are performed by the certificate generator and if 1 and 2 are successful, the generator aggregates the accumulated monitoring evidence and creates a certificate of the security property for the TOC incorporating the aggregated evidence with it. The aggregation of monitoring evidence takes place as described by the evidence aggregation element of MBCM. Once generated, a certificate is stored in the certificates DB and can be retrieved by any external party that has (read) access rights to it. The retrieval of certificates is supported by the certification communicator component of the CUMULUS framework.

In the current implementation, we are using EVEREST (EVEnt REaSoning Toolkit [7]) as a key component of the certificate generator to perform the monitoring required during the certification process. EVEREST is an open-source monitoring framework developed by the authors of this paper that are working for City University to support the monitoring of service-based systems. EVEREST supports the monitoring of properties expressed in a first order temporal logic language based on Event Calculus.

## 5 An example: Certificate issuing in an e-Health scenario

An example of security property that could be certified through a monitoring based certificate in the e-Health scenario is an instance of an integrity requiring that a TOC should protect that data that it stores from unauthorized alterations. This property corresponds to `AIS:integrity:data-alteration-prevention` in the catalogue of properties that the Cloud Security Alliance has defined [3]. In the CUMULUS framework this property is specified as an assertion in SecureSLA* as shown in Figure 5. The description of SecureSLA* is beyond the scope of this paper and can be found in [3]. The above

definition of the integrity property uses the variable autop which indicates an invocation of the authorisation operation of `authorisation::interface::authorise` to check if `cns` has appropriate authorisation rights (successful authorization is indicated by requiring the output result of the authorization operation to be true). The property of protected (i.e., authorised) alteration is then specified by a guarantee state which requires that for each invocation of a data alteration operation, the authorization operation has been invoked prior to it and it has responded with an output that indicates the authorization of the particular consumer. The temporal sequence of the two operations is indicated by the condition less_than (request (autop), request (altop)) and less_than (reply(autop), request(altop)). The correlation of the requester of the alteration operation and the agent whose credentials are checked by the authorisation operation is ensured by using the same value for the requester parameter of the data alteration operation and the agent parameter of the authorisation operation (i.e., credentials). In the case of the above property the CUMULUS framework would first check if the cloud infrastructure where the specific TOC whose integrity is to be certified can provide primitive events capturing the calls of the operations `dataalteration::interface::deletepatient` and `authorisation::interface::authorize` and the responses to these calls, i.e., events matching the terms request(autop), request(altop), reply(autop), and request(altop) in the above assertion. If such event captors exist the monitoring process can start. The evidence sufficiency conditions in this case could, for example, require that the considered event log should include at least 10,000 invocations of each of the operations `dataalteration::interface::deletepatient` and `authorisation::interface::authorize`, gathered over a period of at least 1 year. Based on these conditions, if after collecting evidence satisfying the above conditions, there are no instances of violation of the assertion in Figure 5, a certificate for the relevant TOC and the property can be generated.

## 6 Future Work

In this section we report on a preliminary suitability desktop analysis of the state of the art Cloud monitoring tools to address security properties. Namely, we have performed an initial prospective analysis regarding which of these *off-the-shelf* monitoring tools seem to support evidence gathering in order to validate that any of the properties in the catalog defined in [3] are present. In [3] the interested reader can find more details about property definition. For our analysis, we have inspected the metrics that these monitoring tools could collect and judged whether their reported values could be used as evidence for the security properties in [3]. The result of this analysis is summarized in Table 1. Notice that we omit completely in the discussion those categories from [3] that do not seem to be supported by any monitoring tool. This result will be our starting point for a *hands-on* evaluation of some of these Cloud monitoring tools both regarding evidence gathering and how difficult or easy it is to use them as external monitoring tools that can gather evidence for our infrastructure in order to issue or maintain certificates.

Next, we will name the category, the number of its properties breakdown between parenthesis, and which monitoring tools seem able to gather evidences at any of the

```
assertion {
    sla_ template { ...
    altop is invocation [ invoke {
        endpoint = enpoint::id::a
        operation = dataalteration::interface::deletepatient
        param { name = requester value = credentials } } ]
   autop is invocation [ invoke {
        endpoint = endpoint::id::b
        operation = authorisation::interface::authorise
        param { name = agent value = credentials }
        param { name = result value = true } } ]

/* —AGREEMENT TERM —- */

    agreement_ term { id = agreement::term::1 guaranteedstate
        { id = guaranteedstate1
        ( less_ than (request(autop),(request(altop)) ) and
        (less_ than (reply(autop),request(altop)) ) } }
        }
}
```

**Fig. 5.** Assertion for data alteration prevention

cloud levels in relation to one property in that category. In short, based on the number of security properties covered, Nagios, Hyperic and New Relic are those tools that are able to gather evidences that could help CUMULUS to certify a major number of security properties. For these tools, we will perform a *hands on* analysis in the immediate future. This analysis will also consider how easy or hard is to achieve a proper interplay of these tools with the CUMULUS monitoring infrastructure.

We explain next what is shown briefly in Table 1. This table shows which monitoring tools seem to support which security properties from [3, Annex A] according to their provided descriptions:

- *A.1.AIS: Application & Interface Security* (18). OSSEC seems suitable to check both `A.1.2 Software alteration detection` and `A.1.4 Data alteration detection`. Nagios, Hyperic and New Relic could help to validate the property `A.1.9 network authenticated server access`. Also, New Relic could help to validate the property `A.1.17 user traceability`.
- *A.2 IVS: Infrastructure & Virtualization Security* (2). Nagios seems useful to validate `A.2.1 Tenant isolation level`.
- *A.4 SEF: Security Incident Management, E-Discovery & Cloud Forensics* (3). Nagios, Hyperic, New Relic and OSSEC seem to offer support for validating the properties in this category.
- *A14. Business Continuity Management & Operational Resilience* (11). Nagios, Hyperic and New Relic seem suitable to address this category.

| | Zenoss Core | Nagios Core | Hyperic | New Relic | Ganglia | OSSEC |
|---|---|---|---|---|---|---|
| A.1 (18) | | | | | | |
| A.1.2 | | | | | | ✓ |
| A.1.4 | | | | | | ✓ |
| A.1.9 | | ✓ | ✓ | ✓ | | |
| A.1.17 | | | | ✓ | | |
| A.2 (2) | | | | | | |
| A.2.1 | | ✓ | | | | |
| A.4 (3) | | ✓ | ✓ | ✓ | | ✓ |
| A.9 (5) | | | | | | |
| A.9.1 | | | | ✓ | | |
| A.11 (7) | | | | | | |
| A.11.3 | | | | | | ✓ |
| A.11.6 | | | | ✓ | | |
| A.14 (11) | | ✓ | ✓ | ✓ | | |
| A.15 (4) | | | | | | |
| A.15.3 | ✓ | ✓ | ✓ | ✓ | ✓ | |
| A.15.4 | ✓ | ✓ | ✓ | ✓ | ✓ | |

**Table 1.** Security properties coverage by SotA Cloud monitoring tools (Desktop analysis)

- *A9. Legal & Standards Compliance* (5). New Relic seems suitable to address in particular, the property `A.9.1 Country level anchoring`.
- *A.11 DSI (Data Security & Information Life cycle Management)* (7). New Relic seems suitable to address the property `A.11.6 Storage retrievability`. Also, OSSEC seems suitable for `A.11.3 Data leakage detection`.
- *A15. Change Control & Configuration Management* (4). Zenos, Hyperic, Nagios, New Relic, and Ganglia seem specifically adequate to validate some properties in this category. In particular, the properties `A.15.3 Percentage of timely configuration change notifications` and `A.15.4 Configuration change reporting capability`.

## 7 Conclusions

In this paper we have presented a certification infrastructure that is able to manage monitoring based certification, testing based certification and, in an immediate future, as we explain below, also TPM based and hybrid certification (to combine both monitoring and testing approaches). We have focused on the monitoring based certification and have illustrated our approach to address it with an example from the e-Health domain. The insfrastructure described in this paper is intended to offer support for a (constant) evaluation of an IT resource in order to determine whether certain security property hold from a given point in time on, in a particular context. In this manner we try to reinforce transparency in the industry, helping business to evaluate the security risks they may accept when working with a particular cloud service provider.

We have also reported on off-the-shelf tools for cloud infrastructure monitoring, and have identified a number of open source versions of industrial strength tools that could be used by the CUMULUS infrastructure as external evidence collectors. Finally, we have reported on the results of a preliminary suitability desktop analysis to address the catalog of the security properties in [3]. This analysis was based on the inspection of the metrics collected by these tools, and their potential use as evidences supporting these security properties.

We conclude noting that, in addition to the infrastructure described in section 3, there are some components that are not currently part of the framework but are planned to be implemented in the future. For example, an `Auditing Module` will allow cloud auditors to analyze the entire certification process, including the certifications models, the processed evidences and the issued certificates. Also, a `Trusted Computing manager` component will be integrated to provide authentication and measurement functions based on the Trusted Computing platform for both the Monitoring and Testing Monitor Managers. Finally, an `Hybrid Manager` component will be designed and implemented so as it can be responsible to handle hybrid certificates (e.g., certificates based upon both types of evidences gathered by monitoring and testing or TC) when the Certification Manager calls it through the 'Hybrid Manager API.' It should probably communicate with the other managers, in order to collect all required information for hybrid certification.

*Acknowledgment.* The work presented in this paper has been partially funded by the EU FP7 project CUMULUS (grant no 318580).

## References

1. Cloud Security Alliance. Cloud control matrix v.3.0.1, 2014. https://cloudsecurityalliance.org/research/ccm/.
2. CCRA. Common criteria. http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R4.pdf.
3. CUMULUS. Deliverable D2.1 'Security Aware SLA Specification Language and Cloud Security Dependency Model', May 2013. Available from http://www.cumulus-project.eu/.
4. CUMULUS. Deliverable D5.1 'CUMULUS framework architecture', June 2013. Available from http://www.cumulus-project.eu/.
5. CUMULUS. Deliverable D2.3 'Certification Models v2', May 2014. Available from http://www.cumulus-project.eu/.
6. Howard Foster and George Spanoudakis. Advanced service monitoring configurations with sla decomposition and selection. In William C. Chu, W. Eric Wong, Mathew J. Palakal, and Chih-Cheng Hung, editors, *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC), TaiChung, Taiwan, March 21 - 24, 2011*, pages 1582–1589. ACM, 2011.
7. C. Kloukinas G. Spanoudakis and K. Mahbub. The SERENITY runtime monitoring framework. In *Security and Dependability for Ambient Intelligence*, pages 213–238. Springer, 2009.
8. Zenoss Inc. Cloud Foundry Zen Pack, 2014. http://www.zenoss.com/solution/awareness.

9. Nagios Enterprises LCC. Nagios core, GNU License, 2014. http://nagios.sourceforge.net/docs/nagioscore/4/en/about.html#whatis.
10. BSD license open source software. Ganglia Monitoring System, 2014. http://ganglia.info/.
11. G. Spanoudakis M.Krotsiani and K. Mahbub. Incremental certification of cloud services. In *7th Int. Conf. on Emerging Security Information, Systems and Technologies (SECURE-WARE'13)*. http://openaccess.city.ac.uk/3236/, 2013.
12. New Relic. New Relic-Server, Browser, APM, 2014. http://newrelic.com/products.
13. OSSEC tem. OSSEC-Open source security, 2014. http://www.ossec.net/.
14. VMWare. Hyperic HQ-open source edition, 2014. http://www.hyperic.com.