Harmonized Monitoring for High Assurance Clouds

Ani Bicaku[1,3], Silvia Balaban[2], Markus G. Tauber[1,3], Silvia Balaban[2], Aleksandar Hudic[3], Andreas Mauthe[4], and David Hutchison[4]

[1]University of Applied Science Burgenland, Eisenstadt, Austria
[2]Karlsruhe Institute of Technology, Center for Applied Legal Studies, Karlsruhe, Germany
[3]Austrian Institute of Technology, Vienna, Austria
[4]Lancaster University, Lancaster, UK

***Abstract*— Due to a lack of transparency in cloud based services well-defined security levels cannot be assured within current cloud infrastructures. Hence sectors with stringent security requirements hesitate to migrate their services to the cloud. This applies especially when considering services where high security requirements are combined with legal constraints. To tackle this challenge this paper presents an extension to our existing work on assurance methodologies in cloud based environments by investigating how current state of the art monitoring solutions can be used to support assurance throughout the entire infrastructure. A case study is used in which monitoring information representing a set of relevant security properties is being collected. As result, we propose that a combination of existing tools should be used to harmonize existing monitoring artifacts. We describe and evaluate an Evidence Gathering Mechanism (EGM) that provides this harmonization and show how this can support assurance. This can also underpin legal proceedings from an evidence law perspective.**

I. Introduction

The main motivation for adopting cloud technology in different sectors is to increase efficiency and minimize IT costs by utilizing new concepts such as elasticity, scalability and on-demand resource provisioning. These properties make cloud computing also attractive critical infrastructure (CI) operations (such as water, electricity, public transportation, health care and telecommunications).

Since through assurance guarantees can be given on the level of security a system provides, it can also be useful for ensuring legal compliance. A cloud provider and a user can agree on a well-defined security level that an assurance framework (AF) has to maintain. Any deviation from contractually agreed levels then lead to a breach of duty. As the AF permits independent, continuous and automatic monitoring it guarantees users of cloud systems that this level is being fulfilled. If it cannot be maintained (due to a fault, negligence or for other reasons under the control of a cloud service provider) a breach of duty would be given. The legal position of the cloud user in a respective lawsuit could therefore be strengthen considering the evidence available through the AF. For instance, it would allow to check if an agreed security level has been maintained when a possible fault occurred. Especially if a high level was originally agreed the probability that a fault occurs should be low. In the case of a violation (e.g. through negligent behavior or someone failing to exercise reasonable care) the cloud provider can be held responsible for not providing an adequately secured system. In order to accommodate this the AF permits continuous checking of contractual obligation. The results of this process can then be used as evidence in a possible lawsuit by establishing the level of security the system had when a fault occurred. In our previous work in the SECCRIT [1] project, we proposed a methodology for modeling high assurance, i.e. analyzing how high level security properties can be measured per component of a cloud service and define how continuous aggregation of measured information can be achieved [1], [2].

The aim of this work is to investigate how existing cloud monitoring solutions can offer a unified view on assurance related monitoring artifacts regardless of the tools in use for low level monitoring. An example of a security property in our existing approach [1], [2] is "*strong password*". It supports confidentiality by validating a set of characteristics (i.e. length, special characters, numbers or history) which are used for a password to assure a certain degree of password complexity, based on NIST [3]. This property can be monitored by performing checks on e.g. the PAM (Pluggable Authentication Modules) module, assuming a Linux based system.

In order to support such assurance methodologies we have to first analyze and identify the existing cloud-based monitoring solutions that are capable of supporting a set of representative security properties to understand their legal relevance. Therefore, we present here a derived set of technical properties that should be part of a monitoring tool for high assurance environments. We furthermore investigate how existing monitoring tools support the monitoring of individual properties. This work mainly looks at open-source monitoring solutions, since they can be customized for our needs with respect to assurance. We find that a combination of existing tools and individual modules can indeed be used to support assurance.

Therefore we propose an Evidence Gathering Mechanism (EGM), which is designed to gather and harmonize relevant monitoring artifacts in infrastructure, tenant and service layer using existing monitoring tools. We show that a binary indicators (0 or 1) can be derived from monitoring elements to harmonize individual results despite from different heterogeneous tools. This also allows to define security properties, link low level monitoring to the higher level assurance con-

[1]www.seccrit.eu

cepts, categorize the security properties (e.g. by using CIA), and record (respectively report) the numbers supporting the individual category.

This paper is organized as follows: Section II provides the detailed definitions and the sources of the representative set of security properties. Section III provides a survey of existing monitoring tools based on monitoring types. Also, it gives an overview of existing scientific work of cloud monitoring in context of high assurance. In section IV we present the evaluation of the monitoring tools for each security property associated with security classes and analysed at various levels. Section V presents the proposed architecture of the EGM and also gives examples how a security property can be monitored to provide high assurance in cloud environments. Finally, section VI concludes the paper and gives an outline of our future work.

II. Property Analysis and Evaluation

This section specifies a representative set of security properties. These have been derived from a study including major stakeholders as part of the project SECCRIT. Further, the evaluated properties have been adopted from NIST, ISO or other security standard catalogues, and are categorized in CIA (Confidentiality, Integrity and Availability) classes.

From an evidence law perspective the confidentiality class is specifically important for damage compensation cases. For instance, in cases where cloud users outsource their data and it is subsequently accessed by non-authorized persons causing damage to the cloud user, proof is required to substantiate a potential claim against the provider. Additionally, assuming a Cloud user is outsourcing the storage of personal data belonging to a third party. If this data is processed in a way that is outside the scope of the agreed purpose (i.e. the purpose for which the Cloud user has a legitimization), the Cloud users will have to provide evidence in case of litigation in order to disprove any potential allegation against them. The integrity class is relevant for establishing if a breach of duty has occurred when data has been modified and this has led to damage. In turn the availability class is important for SLA Compliance, specifically in order to check if the owed performance duty has been fulfilled properly.

Hence, for each particular property a generic description and a general method of monitoring has to be provided. A more technical example of monitoring is provided in section V.

A. Confidentiality Properties

- **Concurrent Session Control**, based on NIST [4] supports confidentiality by restricting the number of concurrent sessions per system account, account type, or both. We monitor this property by checking if the tools that monitor concurrent session information and control it are installed on the component of consideration.
- **Strong passwords**, based on NIST [3] supports confidentiality by assuring that a certain degree of complexity is required for passwords. This property can be monitored by checking whether tools or services, enforcing complex passwords are installed and actively being used on the component of consideration.
- **Encryption**, based on ISO [5] supports confidentiality by assuring that access to the encrypted data, remains restricted to those who are not in possession of the proper encryption/decryption key. We monitor this property by checking whether encryption and secure communication protocols are present on the component of consideration.

B. Integrity Properties

- **Data Error Correction** [6], supports integrity by assuring that data is resilient to errors. This can be measured by checking if the techniques that support data error correction are installed on the component of consideration.
- **Information (Data) Consistency**, based on ISO [5] supports integrity by assuring that at any point of time the transmitted or stored information is resilient to alteration. This can be monitored by checking if tools that support integrity are installed on the component of consideration.
- **Data Alteration Prevention**, based on ISO [5] supports integrity by assuring that there are present services which can prevent any kind of unauthorized modification of the data. This can be monitored by identifying if there are mechanisms like hashes or digital signatures applied to a particular component.

C. Availability Properties

- **Load Balancing**, based on NIST [7] supports availability by assuring the distribution of incoming requests across multiple servers. This can be monitored by assuring that there are services present that can perform load balancing techniques.
- **Resource Utilization**, based on ISO [8] supports availability by assuring that a particular service of a system could be spawned across physical or virtual nodes. This property can be monitored by checking if data replication techniques are present on the component of consideration.
- **Live Backup**, based on ISO [5] supports availability by assuring that a live backup can be conducted. This property can be monitored by assuring that there exist services to support backup on the entity of consideration and there is evidence that those means are in use.

III. Monitoring Overview and Related Work

This section gives an overview of existing state-of-the-art work in respect of cloud based monitoring solutions and concepts. First, we give an overview of existing monitoring tools related to high assurance in CI environment. In particular it is investigated if the monitoring tools are capable to provide (i) cross layer monitoring; (ii) continuous monitoring; (iii) interval or event based monitoring; (iv) multi-tenant monitoring. Second, we consider the scientific work and publications for cloud monitoring related to high assurance.

A. Cloud Monitoring Solutions

Existing open-source monitoring solutions are investigated and summarized related to high assurance of services hosted

in cloud environments. Also, the evaluated solutions are investigated if they are extendable and which of them already support the set of representative properties. Due to the fact that the monitoring solution may also offer unofficial plug-ins this work is mainly focused on an official set of features. This section is based on output from the SECCRIT project, specifically Deliverable 5.2 [2].

Nagios [2] is a well-known and provides a widely used monitoring solution capable of monitoring all devices using the TCP/IP protocol suite within interconnected networks, and network and infrastructure devices. This monitoring solution is capable of acquiring data from infrastructure layer and delivering them to the central aggregation module (i.e. a server used for data acquisition and data analysis). Nagios does not support device auto-discovery, which in large scale dynamic environments can be a significant drawback. Also, like a lot open-source solutions Nagios is based on complex text based configuration files which makes it challenging to administer. Moreover, Nagios does not have the ability to distinguish devices per types (e.g. servers, routers, or switches). Therefore, Nagios is not capable of addressing challenges such as cross layered monitoring. To define policies or interval based monitoring in multi-tenant environments it requires additional plugins giving the possibility of further extension.

Sensu [3] is a monitoring solution, also known as "Monitoring Router", a system which provides all the gathered data to event handlers to correlate and analyse the results. The architecture of Sensu is message-oriented using RabbitMQ and JSON payloads with a friendly user interface solution. Sensu uses RabbitMQ to establish client server communication. Hence, it is compatible with message oriented architectures. The main benefit of this tool is the ability to flexibly customize its acquisition modules and flexibly scale across infrastructures. Unfortunately, Sensu is not designed to monitor multi-tenant environments or provide policy and interval based monitoring.

Munin [4] is a monitoring solution written in Perl and designed to monitor infrastructure, network devices, and services. This monitoring solution analyses data by using RRD-Tool and graphing systems to present the output in graphs via the user interface. Unfortunately, Munin does not offer much flexibility when accessing monitored hosts and configuring the monitored network. Although Munin offers the ability to run various customisable scripts, it only provides a fixed interval-scan rate where these scripts could be initiated. One of the major drawbacks of Munin is its text file configuration model which can be quite complex.

Private Cloud MONitoring System **PCMONS** [9] is a monitoring system designed for private cloud infrastructures. It is a compound of infrastructure layer, integration layer and the view layer (based on Nagios) divided into eight components (Node information gatherer, Cluster Data Integrator, Monitoring Data Integrator, VM Monitor, Configuration Generator, Monitoring Tool Server, UI and Database). A main goal of PCMONS is extensibility of its services.

Zabbix [5] is an enterprise monitoring tool focused on network infrastructure monitoring. It is based on an agent-less monitoring concept without connecting to the host and therefore reducing the impact on the performance. Furthermore, Zabbix provides distributed monitoring in real-time with centralized web administration. It offers an overview of all hosts inside the network from a single point of entry. However, its capability is limited when it comes to automatic discovery of monitored devices .

Zenoss Core [6] is an IT monitoring software that offers an overview of the entire IT infrastructure from the physical infrastructure to the services running through its rich feature set (e.g. automatic discovery, performance analysis, sophisticated alerts, etc.). Furthermore, Zenoss offers extensibility and flexible customization of monitoring environments via its additions such as ZenPack.

Discussion

This short survey validates the monitoring issues with respect to assurance methodologies based on monitoring types.

TABLE I
MONITORING TYPE

	Nagios	Sensu	Munin	PCMONS	Zabbix	Zenoss
Cross layer monitoring		✓		✓		✓
Continuous monitoring	✓	✓	✓	✓	✓	
Interval based monitoring			✓			
Event based monitoring	✓	✓			✓	✓
Multi tenant monitoring				✓		

Table I shows that only Sensu, PCMONS and Zenoss can distribute a monitoring mechanism across all layers (infrastructure, tenant and service). Another important output of the survey is the identification of continuous monitoring capabilities. Continuous monitoring allows solution to adapt and to monitor cloud infrastructure changes without requiring manual interaction. So new nodes or VMs can be automatically registered and monitored. The result shows that most of them support continuous monitoring except Zennos. Event-based monitoring is based on information about the state of the system. Changes in the system state will generate an event for the monitoring software. Interval-based monitoring checks the system based on different time intervals according to the resources (e.g. CPU, memory, storage and VMs) that have to be monitored. The monitoring tools are divided between interval-based (i.e. Munin) and event-based monitoring (i.e. Nagios, Sensu, Zabbix and Zenoss). Another objective of this evaluation is multi-tenant support, i.e. the ability to control user permissions allowing users to see and control only resources that they are authorized to view and change. This property is supported only by PCMONS. The conclusion from table I is that a single tool does not fulfil all the specified properties. Therefore, we propose an EGM in section V

[2]www.nagios.org
[3]www.sensu.com
[4]www.munin-monitoring.org
[5]www.zabbix.com
[6]www.zenoss.com

through which all the required properties can be accessed in a unified and harmonised way.

B. Cloud Monitoring Scientific Work

At the moment to the best of our knowledge none of the existing surveys or taxonomies investigate the possibility of a harmonized solution for assurance in a cloud enivronment.

Aceto et al. [10] evaluated cloud monitoring solutions across the following three dimensions: (i) layers, according to CSA, (ii) abstraction levels, high and low monitoring level and (iii) test and metrics. In addition, the authors provide motivations, properties, basic concepts, open issues and discuss future directions essential for cloud monitoring platforms. However, with respect to our approach their analysis does not address high assurance for cloud environments.

Fatema et al. [11] identify practical capabilities of an ideal monitoring tool by evaluating a variety of monitoring tools. Based on these capabilities, the authors provide a detailed taxonomy where the monitoring tools have been classified according to their purpose: (i) general purpose (to determine how these tools can be extended), and (ii) cloud specific purpose. The authors also highlight the importance of assurance in cloud computing and the correlation with cloud monitoring, but their research focus is not on assurance for highly sensitive infrastructures and services.

Uriate and Westphall [12], present a novel cloud-based monitoring architecture. They validate the monitoring architecture through an experimental evaluation and analytical analysis within respect of scalability and invasiveness. However, the authors do not address any specific security properties for their evaluation, or address high assurance in their objectives.

Andreolini et al. [13] propose a novel monitoring architecture by combining a hierarchical approach with decentralized monitoring that addresses scalability and high availability. In comparison with their approach, we propose a combination of existing monitoring solutions to address assurance in the context of CIA.

Ward and Barker [14] perform a survey and taxonomy of cloud monitoring to evaluate how existing tools meet the challenges of cloud computing. They validate open-source and commercial monitoring tools based on two categories: (i) monitoring solutions developed only for cloud computing, (ii) tools which are not developed especially for cloud monitoring but have related goals and results. However, with respect to our analysis, they do not address assurance in CI or monitoring with respect to infrastructure, tenant or service layer.

To the best of our knowledge none of the existing work addresses assurance in the context of high security for protecting CI services. Also they do not evaluate any of the monitoring tools in the context of CIA. Therefore, we focus our work on variety of open-source monitoring tools based on the ability to monitor different security properties that are defined with respect to high assurance of CI [2], [15]. Furthermore, we propose a possible solution for the problem by harmonazing different monitoring tools and custom-based scripts. This solution is presented in a form of an Evidence Gathering Mechanism.

IV. Overview of Security/Assurance Support in Existing Monitoring Tools

The selected security properties are investigated in respect to individual monitoring solutions and the capability to monitor a specific property across different layers. The result of this investigation is shown in table II where the monitoring tools are evaluated based on their ability to monitor a specific property in infrastructure, tenant or service layer. The evaluation of the monitoring tools with respect to security properties identifies three different cases: (i) the monitoring tools that can monitor a property in each layer without the need of a plugin or additional scripts (e.g. the property *resource utilization* can be monitored by Munin in all three layers), (ii) the monitoring tools that can monitor only one layer and for the other layers a plugin or an additional script is needed (e.g the property *data error correction* in the infrastructure layer can be monitored by Zabbix, in the tenant layer by a Nagios plugin, but to monitor the service layer an additional script is needed), (iii) none of the monitoring tools have the ability to monitor the property, so a script is needed for each layer (e.g properties like *strong passwords*, *encryption* and *load balancer* completely lack the support of monitoring tools).
These outcomes lead us to harmonize different monitoring tools and customized scripts in an EGM to support all the properties in table II. In section V we describe our solution and give some implementation ideas for additional scripts needed to monitor a property which is not covered by these monitoring tools.

V. Evidence Gathering Mechanism Design

Based on the evaluation shown in table II and discussed in the previous sections it can concluded that there is not one single tool that can support the assurance methodologies. Motivated by this result, we propose an EGM (Evidence Gathering Mechanism) to harmonize monitoring solutions across the layers with respect to the assurance methodology. The

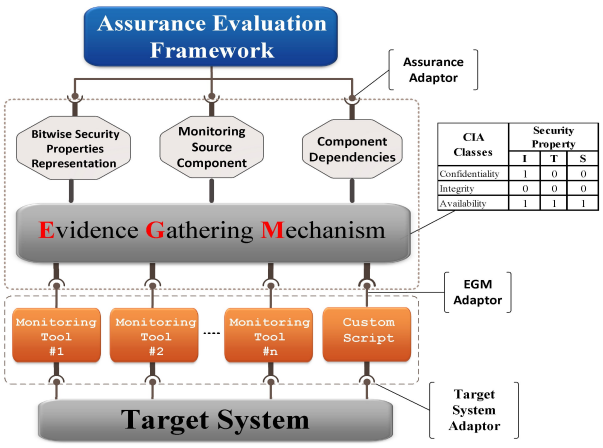

Fig. 1. Evidence Gathering Mechanism Framework Design

TABLE II
MONITORING TOOLS THAT SUPPORT EVALUATED SECURITY PROPERTIES

		Nagios			Sensu			Munin			PCMONS			Zabbix			Zenoss			Script		
		I	T	S	I	T	S	I	T	S	I	T	S	I	T	S	I	T	S	I	T	S
Confidentiality	Concurrent Session Control			∗					x											+		
	Strong Passwords																			+	+	+
	Encryption																			+	+	+
Integrity	Data Error Correnction	∗	∗											∗								+
	Information Data Consistency						∗							x	x							
	Data Alteration Prevention			∗																+	+	
Availability	Load Balancing																			+	+	+
	Resource Utilization				x	x		x	x	x	x	x		x	x		x	x				
	Live Backup			∗	∗															+		

Note: With "**x**" is marked the layer if the particular tool has the ability to monitor the property. With "∗" is marked the layer where an existing unofficial plug-in already exists to monitor the specific security property and with "**+**" when an additional script is needed to monitor the respective layer of the property.

EGM, shown in Fig 1 is designed to acquire, store and analyze security related evidence and support Assurance Evaluation Framework (AEF) [2]. The main purpose is to use simultaneously the information obtained by existing monitoring tools and customized scripts. The proposed architecture consists of the following components:

- **Target System** represents the infrastructure, tenant or service resources monitored by a monitoring tool or a scripts.
- **Monitoring Tool** represents the investigated open-source monitoring tools evaluated in section III.
- **Custom Script** represents pieces of code from scratch or customized existing plug-ins.
- **EGM** is designed to gather, store and analyze monitoring data from monitoring tools and custom scripts.
- **Bitwise Security Properties Representation** represents the evaluated properties in section II.
- **Monitoring Source Component** provides the source of each security property mapped in the corresponding security standard catalogue.
- **Component Dependencies** represents the way how components are linked with each other.
- The **Assurance Evaluation Framework** is an ongoing work for the project introduced in one of SECCRIT outputs [2].
- **Adaptors** are designed to make readable the data obtained from each component for further analysis.

Based on the result shown in table I and II it can be seen that 50% of the properties could be monitored by different existing tools. We propose EGM, a design that will allow to do that or alternatively to use customized scripts where need be. The harmonization of existing tools allows unified view via the bit mask (which is only an implementation detail for EGM), and aggregation via logical operations and policies.

A. Security Property Implementation

We give an example on how a security property (SP) can be measured based on the results obtained from the table II.

SP - Concurrent Session Control

- *General Definition*: Concurrent session control restricts users to log-in more than one session in different browser/machine or even a different browser/terminal within the same machine.
- *Engineering Definition*: This property defines the number of logged sessions for a user. If the user is limited to a session count equal or smaller than three (based on table III) concurrent session control is defined as strong.

TABLE III
CONCURRENT SESSION CONTROL LOGICAL STATE

Logical State	Infrastructure	Tenant	Service
0	$MaxSessions > 1$	$MaxSessions > 1$	$MaxSessions > 3$
1	$MaxSessions = 1$	$MaxSessions = 1$	MaxSessions=3

- *Definition Source*: NIST Special Publication 800-53 :IA-5 [5]

TABLE IV
CONCURRENT SESSION CONTROL IMPLEMENTATION

Name	File	Line
SSH configuration	/etc/ssh/sshd_config	˝UsePAM yes ˝
PAM configuration	/etc/security/limits.conf	˝*hard maxlogins 1 ˝

- *Implementation Possibility*: This property can be monitored by checking the number of logged in sessions. For assuring concurrent session control at the infrastructure, tenant and service layer a script can be developed that will check sshd_config or pam_limits module in the /etc directory as showed in table IV.

Based on table II, Munin can monitor this property at the tenant layer and Nagios at the service layer. The Nagios plugin check_weblogic_sessions will check the sessions of a web application in a specific time frame. Once the plugin is running the OpenSessionsCurrentCount will be checked to determine if it is greater than the maximum sessions specified. In case it is less or equal to the maximum concurrent session specified, it will return the message:
[OK] 2 concurrent sessions less than or equal to 3
In case it is greater than the maximum concurrent session specified, it will return a CRITICAL state with a message:
[CRITICAL] 4 concurrent sessions greater than 3

The described Nagios plugin can be customized to show the logical state 0 for [CRITICAL] and 1 for [OK]. Thus, to monitor concurrent session control we need to install Munin for tenant layer, adapt Nagios plugin for service layer and develop an additional script for infrastructure layer. This

outcome leads us to harmonize different monitoring tools and customized scripts in an EGM, due to the fact that one single tool can not cover all the layers of a specific security property.

B. Gathering Security Property Information

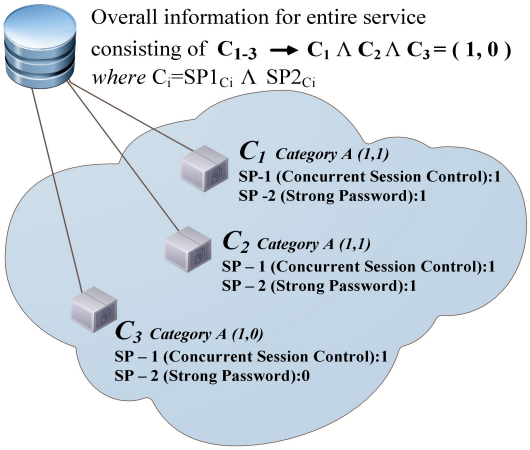

Fig. 2. A simplified use case for information harmonization

Fig. 2 shows three components (C_i) contributing to a cloud service. For each component C_i the binary state of a security property can be derived from the monitoring artifacts as illustrated earlier. This allows a multitude of ways to aggregate this information, e.g. overall properties of the same type; or all components; or via dependency trees – but this is subject to assurance policy development [1], [2] for which the harmonization presented in here serves as a foundation.

VI. Conclusion

This paper presents an analysis of the state of the art in cloud monitoring with respect to high assurance. It discusses existing cloud monitoring solutions, which are evaluated based on their ability to support high assurance clouds. Each solution is assessed based on monitoring type and the ability to monitor a specific property at a specific layer. Our analysis has shown that existing tools can be used but a harmonization between them is needed. We found that individual existing tools can monitor some but not all properties. In this paper the set of security properties associated with specific security classes (i.e. confidentiality, integrity and availability) are analyzed at various layers (i.e. infrastructure, tenant and service). Further, it is is investigated and measured to what extent a considered component of the cloud services fulfills the required assurance level. Considering the outcome of section IV, an Evidence Gathering Mechanism (EGM) is proposed. It is designed to gather, store and analyze monitoring data. The proposed architecture represents a practical approach and adopts open-source tools whenever feasible, e.g. an OpenStack platform and open-source monitoring tools to collect, store and analyze monitoring data. The architecture fulfills the requirements by providing an architecture that harmonizes monitoring tools and customized scripts to support assurance methodologies for cloud based services. The EGM architecture has been designed for a representative set of properties and shows that for high assurance environments appropriate harmonization is needed. Moreover, this work is promising as it gives a common viewpoint over heterogeneous and independent monitoring elements that can then be used for assurance across the infrastructure. Especially from a legal point of view the assurance framework seems very promising as it can ensure compliance with contractual duties and also provides evidence of possible negligent behavior.

As further work we are currently evaluating EGM by extending the approach to support additional properties and classes. EGM is used for monitoring at the infrastructure, tenant and service layer for the evaluated classes and properties, and also to support additional classes e.g auditability. EGM will not only be used for open-source but also for commercial cloud infrastructures in order to evaluate assurance.

ACKNOWLEDGEMENT

The research has been funded by the European Commission, in the context of the Seventh Framework Programme (FP7) project SECCRIT (Grant No.312758).

References

[1] A. Hudic, M. Tauber, T. Loruenser, M. Krotsiani, G. Spanoudakis, A. Mauthe, and E. R. Weippl, "A multi-layer and multi-tenant cloud assurance evaluation methodology," in *International Conference on Cloud Computing Technology and Science (CloudCom-2014)*, 2014.

[2] A. Hudic and M. Tauber, "D5.2 Cloud assurance profile and evaluation method," Tech. Rep., January 2015.

[3] R. Ross *et al.*, "NIST sp 800-53, revision 4," *Security and Privacy Controls for Federal Information Systems and Organizations*, 2013.

[4] R. Ross, S. Katzke, A. Johnson, M. Swanson, G. Stoneburner, G. Rogers, and A. Lee, "Recommended security controls for federal information systems," *NIST Special Publication*, vol. 800, p. 53, 2005.

[5] I. ISO and I. Std, "ISO 27002," *Information Technology-Security Techniques-Code of Practice for Information Security Management. ISO*, 2005.

[6] G. Sivathanu, C. P. Wright, and E. Zadok, "Ensuring data integrity in storage: Techniques and applications," in *Proceedings of the 2005 ACM workshop on Storage security and survivability*. ACM, 2005, pp. 26–36.

[7] S. NIST, "800-44 version 2," *Guidelines on Securing Public Web Servers*, 2007.

[8] I. ISO and I. Std, "ISO 15408-2: 1999," *Information Technology-Security Techniques-Evaluation Criteria for IT security. ISO*, 1999.

[9] S. A. De Chaves, R. B. Uriarte, and C. B. Westphall, "Toward an architecture for monitoring private clouds," *Communications Magazine, IEEE*, vol. 49, no. 12, pp. 130–137, 2011.

[10] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, pp. 2093–2115, 2013.

[11] K. Fatema, V. C. Emeakaroha, P. D. Healy, J. P. Morrison, and T. Lynn, "A survey of cloud monitoring tools: Taxonomy, capabilities and objectives," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2918–2933, 2014.

[12] R. B. Uriarte and C. B. Westphall, "Panoptes: A monitoring architecture and framework for supporting autonomic clouds," in *Network Operations and Management Symposium*. IEEE, 2014, pp. 1–5.

[13] M. Andreolini, M. Colajanni, and M. Pietri, "A scalable architecture for real-time monitoring of large information systems," in *Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on*. IEEE, 2012, pp. 143–150.

[14] J. S. Ward and A. Barker, "Observing the clouds: a survey and taxonomy of cloud monitoring," *Journal of Cloud Computing*, vol. 3, no. 1, pp. 1–30, 2014.

[15] A. Hudic, T. Hecht, M. Tauber, A. Mauthe, and S. C. Elvira, "Towards continuous cloud service assurance for critical infrastructure it," in *2014 2nd International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 2014, pp. 175–182.