Adopting an Agent and Event Driven Approach for Enabling Mutual Auditability and Security Transparency in Cloud based Services

Moussa Ouedraogo[1], Eric Dubois[1], Djamel Khadraoui[1], Sebastien Poggi[2] and Benoit Chenal[2]
[1]*Luxembourg Institute of Science and Technology, 5 Avenue des hauts Fourneaux, L4362 Esch/Alzette, Luxembourg, Luxembourg*
[2]*Victor Buck Services S.A, L-8308 Capellen, Luxembourg, Luxembourg*
{moussa.ouedraogo, eric.dubois, djamel.khadraoui}@list.lu, sebastien.poggi@victorbuckservices.com benoit.chenal@learch.lu

Keywords: Cloud, Security Transparency, Mutual Auditability, Monitoring, Event Specification and Detection.

Abstract: We propose an event-driven approach for the automated audit of cloud based services security. The proposed approach is a solution to two of the intrinsic security issues of cloud based services, notably the need of security transparency and mutual auditability amongst the stakeholders. We leverage a logic based event specification language to represent patterns of events which occurrence can be evidence of security anomaly or breach or simply a sign of a nefarious use of the cloud infrastructure by some of its users. The use of dedicated algorithms for the detection of composite events coalesced with the definition of primitive events structure based on XCCDF format ensures the reuse and interoperability with security audit tools based on the Security Content and Automation Protocol-SCAP. The implementation and application of the approach on a cloud service dealing with electronic archiving have demonstrated its feasibility and viability.

1 INTRODUCTION

For most businesses and individuals, Cloud based services are the alternative to achieving cost-efficiency in the provisioning and consumption of services. However companies dealing with security and/or privacy critical data, have often shown some reluctance to fully embrace the trend, even if there is evidence that the trend is starting to sift at least for the banking and financial sector (http://www.businesscloudnews.com/2014/06/02/cloud-in-financial-services-what-is-it-not-good-for/). Several factors could explain such an attitude towards the cloud: In the cloud, the data and the mechanisms necessary for its processing may reside in the provider's premises. This leads to some devolution of security matters about such data and processes to the cloud provider whose capability and/or due diligence to deal with the security issues may be mistrusted or simply feeble. The uncertainty on the actual location of the data is also exacerbated by the complexity of the chain of provider-consumer. In fact, although a CSP may be registered in a given country, the chain of provider-consumer may be such that the actual data centre used by the CSP is located elsewhere. Given the stored information may be subject to the legislation of the country where it is stored physically, this may also pose serious privacy management challenges. In fact there may be ambiguity in understanding which regulation applies for a data about a third country citizen (which should normally be subjected to national regulation) but stored in another country, where regulation towards privacy may be well different. The multi-tenancy aspect that is most often used to characterize cloud computing also introduces a new risk unique to cloud services, the possibility of attacks from other consumers, who may be competitors or simply hackers, co-located on the same infrastructure, e.g., servers, hard disks, virtual machines. This is well exemplified by "Amazon Zeus botnet" incident involving Amazon EC2's infrastructure (McAfee and Guardian Analitics, 2012), whereby cybercriminals, by initially hacking into a service hosted by Amazon cloud infrastructure, were able to install command-and-controls infrastructure with the aim to infect client computers and steal their banking credentials. This incident is a reminder that the security of the cloud service is only as good as at its weakest link given that a vulnerability at a tenant application may result in the jeopardy of the whole service. This status quo calls for techniques that help to foster more security assurance in the cloud realm. Security assurance being the ground for confidence that security deployed and/or managed by a third party is correctly implemented and also effective against the

risks (Ouedraogo et al., 2012). In third party services such as the cloud, security assurance can be practically met by probing the security of the CSP through audits and by gaining more visibility on its security policy and operation through security transparency mechanism (Winkler, 2011). Consequently, achieving a wider adoption of cloud based services would depend on how effective issues related to mutual auditability and security transparency can be addressed (Chen, 2010; Ouedraogo et al., 2013; Nuñez et al., 2013; Sunyaev and Schneider, 2013). Technics and approaches tailored in that vein of idea should enable the Cloud Service Consumer (CSC), provided the existence of contractual clauses with the Cloud Service Provider (CSP), to gather evidence that corroborate or challenge compliance, performance and security claim made by the CSP, while at the same time enabling the latter to monitor the activity and traffic of the users to ensure no abuse and nefarious use of the cloud is made.

This paper's contribution can be summarized as an effort to leverage Event-driven computing (Luckham, 2005; Etzion and Niblett, 2010) and Multi agent systems-MAS- (Ganzha and Paprzycki, 2014) to foster more security transparency and enable mutual-auditability in a cloud setting. To achieve this, we resort to a tree based specification of security events of interest by the CSC and CSP while within the infrastructure, software agents are generated for capturing such events in case they materialize. We amalgamate real time security related event detection and logic-based rules for empowering both the CSC and CSP, with effective means of depicting and promptly detecting anomalies and security or QoS breaches. An event is here considered as a happening of interest (related to security or quality of service procurement) to the CSP or CSC.

The paper is organized as follows: Section 2 analyses the related work. Section 3 provides a description of the adopted architecture. In Section 4 we specify monitor-able event using a logic based language. Section 5 shows how audits and transparency are enforced, while Section 6 presents an application case. Section 7 provides some concluding remarks.

2 RELATED WORKS

Initiatives purporting to address the issue of mutual trust and transparency in the cloud have mainly revolved around the topic of audit, Virtual machine introspection and Service level agreement. Audits standards including SSAE16 (www.ssae16.com), and its international version ISAE3402 (http://isae3402.com/) rely in a large part on the words and assessment of the CSP, information that cannot be guaranteed to be immune from bias. Recent efforts in cloud audits have leaned towards automation. For instance, the CSA cloud-audit (http://cloudaudit.org/CloudAudit/Home.html) pur-ports the automation of standard audit and related assurance and compliance effort by providing a controlled set of interfaces to allow CSCs or their representatives to assess their services. Dolitszcher et al. (2013) propose a cloud audit methodology based on the usage of MAS for conducting the audit of virtual machines dynamically allocated to clients to account for changes within the cloud infrastructure.

Rak et al. (2011) adopts APIs derived from the mOSAIC project (http://www.mosaic-project.eu/) to build up an SLA-oriented cloud application that enables the management of security features related to user authentication and authorization. An extension of the work of Rak et al. can be found through the EU FP7 project Specs aiming to deliver a platform for providing a security services based on SLA management. The SLA monitoring in SLA@SOI relies on EVEREST+ (Lorenzoli and Spanoudakis, 2010), which is a general-purpose engine for monitoring the behavioural and quality properties of distributed systems based on events captured from them during the operation of these systems at runtime. The major problem with the adoption of SLA management as a means to enhance security transparency is primarily on its practicality. Indeed the academic notion of SLA appears to be far more extensive than it is in reality. In the context of this work, the authors have approached a number of CSPs in Luxembourg with the aim to get a glimpse on the set of items that were part of their SLA. Most often, such documents were restricted to the sole aspects of allocated bandwidth, storage capacity, etc..; while the only security aspect included was related to service availability. Clearly, the items included in those specifications were those the companies were confident they could deliver on. Their argument on the most pressing and challenging issues such as security was that stringent mechanisms were in place for its guarantee as evidenced by their certifications.

Unlike the existing initiatives, our approach leverages events processing to enable mutual audit between the CSC and CSP. While existing commercial and open source solutions for event analytics such as Splunk (Carasso, 2012), Arcsight (http://www.arcsight.net/) and Graylog2 (https://www.graylog.org/graylog2-v0-92/) are based on log analysis, our initiative is based on near-real time

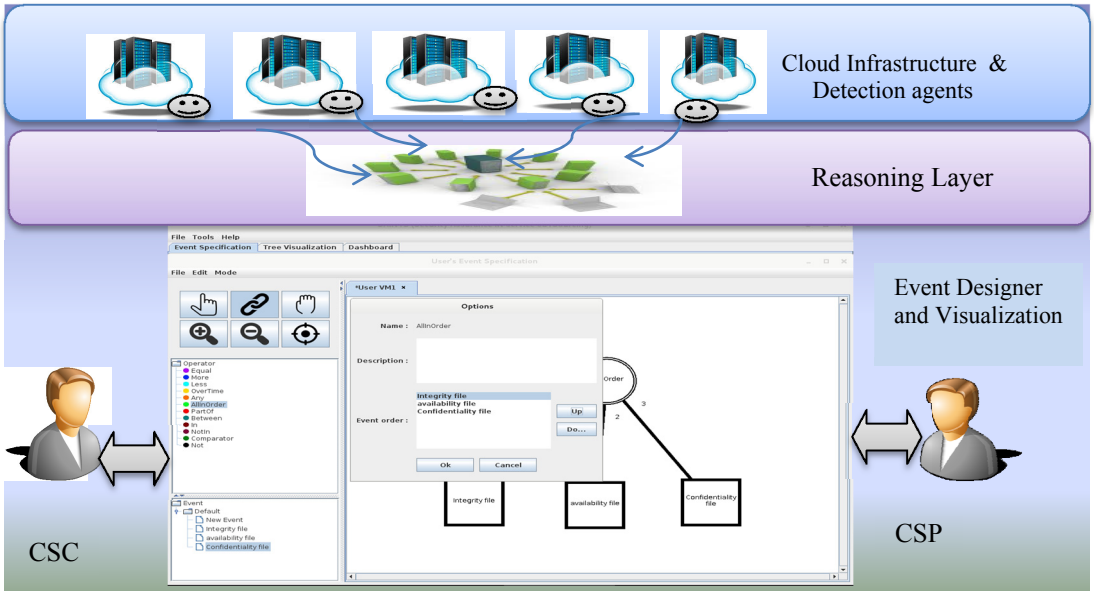

Figure 1: Architecture of the event specification monitoring for transparency and mutual auditability.

detection of primitive events followed by a reasoning on whether the specified composite event has materialized. Thus, allowing the concerned CSC/CSP to promptly take mitigating actions. Also, provided the existence of a contractual agreement, both the CSP and CSC can specify and launch a monitoring of the security of the other.

Additionally our approach is that the event driven specifications provides a good expressivity for capturing events of interest emanating from an SLA, an internal policy or external regulations while allowing the reuse of existing security management tools.

3 HIGH LEVEL ARCHITECTURE OF THE APPROACH

Our event driven approach and tool is made of three main components as depicted in Figure 1.

The first component is a case tool or *Event designer* that offers a graphical design interface to the cloud stakeholders, for specifying patterns of events that are of interest. Upon the design of the events composites that could result from combining different patterns to monitor, the stakeholder provides technical details for each event using an event specification format of choice. In the context of our work, the Extensible Configuration Checklist Description (XCCDF) format (Waltermire et al., 2011) is adopted as further elaborated in Section 4.1. The case tool also serves as a dashboard for the visualization of the events status once the monitoring is triggered.

The second component relates to the elaboration of a multi-agent system embedded within the cloud infrastructure (CI) with the purpose of detecting each primitive events specified within the Event Designer console. The definition and management of the detection agents are performed using JADE platform (Bellifemine et al., 2008). The peculiarity of the agent structure and organization is adopted from the work of Ouedraogo et al. (2014).

The third component is the event processing layer. Individual atomic events captured from the cloud infrastructure are sent by specialized agents to the reasoning layer where dedicated algorithms detailed in Section 5 will be resorted for informing the stakeholder when a specified pattern has materialized.

In the following we further elaborate on how those three components play a role in practically delivering mutual audit and help foster better transparency.

4 SPECIFICATION OF MONITORABLE EVENTS

The decision to adopt an event driven approach to audit and monitor the security in the cloud is underscore by the argument that events provide a powerful construct to capture current state of a system and deviations from expectation and to predict future security or QoS related issues (Luckham, 2005; Etzion and Niblett, 2010). Additionally a well-defined architecture can support event based moni-

toring in ensuring the prompt dissemination of its occurrence to the interested parties who would make judgment on the course of action to adopt. Amongst others, it may be a way to hold cloud providers accountable for a security breach that may has stemmed from a lax in their security; a breach of SLA or other escrows between the two parties. The set of patterns and the detection algorithms associated could also constitute a powerful tool for a cloud provider concerned with activities of its clients.

A prerequisite for effective event patterns detection is the definition of a clear event structure coupled with the adoption of a pattern specification language that is expressive enough to capture the realm of events of interest and their propensity. Only after that one can begin to implement the required strategy for the ensuing detection. This section provides an insight into our event based approach.

4.1 A Primitive Event Structure to Ensure Re-Use of Existing Security Tools

Commonly an event is defined as an occurrence of interest within a system or domain (Etzion and Niblett, 2010). Subsequently, dealing with events could purport the monitoring of a system or process with the intent to flag exceptional or anomalous behaviors. Alternatively, event analysis could be the baseline for (i) predicting a major event before they actual takes place as in fraud detection application, financial market trends and natural disaster; (ii) diagnosing a problem based on deductive reasoning after the observation of symptomatic events. While most event structure includes header information that provides meta-information about the event (identification number, occurrence time, description, category, etc...), attributes related to the event payload is intrinsically linked to the intent sought for their processing. For instance, for the description of an event structure pertinent to a credit card fraud, the geographical locations where individual purchase takes place and the amount of money involved are very salient information to capture. Owing to the fact that this audit emphasis on anomalies related to security, the idea was then to adopt an event structure that could allow the re-use of existing security audits tools given the area of network and system audits is already beaming with a plethora of tools. The adoption of an Extensible Configuration Checklist Description Format (XCCDF) like format as a baseline for the primitive events structure was thus to ensure reuse and interoperability with Security Content and Automation Protocol (SCAP) tools. XCCDF is an XML based format used to specify security checklists and benchmarks amongst others. The overarching purpose of the format is to provide a uniform expression of security checklists, benchmarks, and other configuration guidance, and thereby foster more widespread application of good security practices. With analogy to the XCCDF format, we specify an event with the following attributes:

```
<Name>..<\Name>
<Identification>..<\Identification>
<Description>..<\Description>
<Category>.. <\Category>
<Time stamp>.. <\Time stamp>
<Value>..<\Value>
<Frequency>..<\Frequency>
<Rule>..<\Rule>
<Probe>..<\Probe>
```

In the context of this work, the most relevant attributes associated to a primitive event include the rule attribute which is the underlining policy based on which the associated probe (either a SCAP tool or an in-house program) could interpret and carry out the specificities of the rule, leading to the detection of the primitive event. Furthermore, the rule attribute encompasses and provides reasoning about the expected and exceptional behaviours and states. The field associated to a rule take as input a path leading to a file, thus allowing one to define a comprehensive set of policy that should drive the detection of primitive events of interest. A rule could be specified in a logical language such as Etalis (Anicic et al., 2012) and Drools fusion (http://www.drools.org/) or programing language including Python as in the case of our implementation. The category attribute is a field we added, for allowing the user to systematically classify events based on their typology and/or interest for the stakeholders. Secondary event attributes such as name, identification, description, timestamp, frequency, denote respectively, the given name unique identifier of the event, a succinct description of it, the time at which the event was created, the frequency at which the tool should be probing the event.

4.2 Event Patterns Specification

We hereafter use the term *event pattern* to refer to any composite event whereby primitive events (leaves of the tree) are associated through a logical operator or connector. An event tree can thus be a simple event patterns or a combination of patterns of different semantic leading to a much complex event specification. Adopting a tree based representation also allows leveraging relevant graph theories for the efficient processing of the event nodes. Our event-based cloud audit and monitoring is based on event

patterns specification using the Yet Another Language for Event Specification or YALES logic (Zhang and Unger, 1996). The choice of YALES lies on its expressiveness but also to the fact that it allows to capture synchronous and asynchronous events and also provides event counter and calendar constructs. YALES distinguishes three types of events: temporal events denoting the explicit time instants in real life and can be deemed relative temporal event when the offset is equal to span. Calendar events express the periodical activities that occur regularly or irregularly many times in terms of real life time measurements and primitive events. A primitive event is considered as an explicit event taking place at a discrete or during a time span.
More precisely, our framework and tool allows both the cloud client and provider to declare and detect the following event patterns or event composite (note that this list of pattern is not exhaustive and new one can be added for detection purpose):

Disjunction of Events, E1 ∨ E2 ∨... ∨ En or any Pattern: This pattern of events, occurs when one or more of primitive events E1,..,En occurs. The detection algorithm for the ANY pattern is later provided.

Conjunction of Events, A (m, {E1, E2 ... En})| Part of: This pattern of event occurs if all the constituent events occur. With a slight extension and with the notation of A (m, {E1, E2... En}) where m < n, we can express the idea that if at least m out of n events occur, the event pattern has happened.

Sequences of Events, E1; E2; ..; En | All in Order: A sequence of events pattern expresses the requirement that occurrence of composite events be strictly in order in time, i.e. no adjacent events that occur at the same time are counted.

Event Counter, C (E, n^ | n | n+ | n-): | Overtime, Equals, More, Less: A way to tell how many times an event has occurred is useful. The event counter as an event pattern is designed to provide this kind of mechanism. Event counter, C (E, n^), will occur upon every nth occurrence of E. In this case C (E, n^) may be triggered more than once; C (E, n) is validated at when at the nth occurrence of E. Thus C (E, n) is only triggered once. C (E, n+) is validated when E has occurred not less than n times; while C (E, n-) will be flagged when E has occurred less than n times but at least once.

Moving Window, W (n, E, span): A moving window uses a moving interval with a fixed span to provide aggregate event information. Here, W (n, E, span) is used to mean that if there are more than n of E occurrences during a time period, then the composite event should be raised.

Put the Occurrence of an Event in Context of a Sequence of Other Events E: C | [E1, E2] and E IN C | [E1, E2]: Events in periods or intervening events are those that occur in a period marked by two reference events E1and E2. Two alternative cases can be considered: The first supposes a left-closed and right-open interval that we refer to as *BETWEEN event pattern;* and the second one considering a left-closed and right-closed interval referred to as *IN event pattern.* While both detection patterns *BETWEEN* and *IN* would require the event of interest to occur after the initiating event (E1), the IN pattern is detected only after the right-end event has occurred as opposed to the BETWEEN event.

5 FROM EVENTS SPECIFICATION TO AUDIT

The previous section provided the foundation for specifying the patterns of events relevant to the security audit. In this section, we depict how in practice patterns are used and analyzed to support mutual auditability and increase cloud transparency. The mutual audit process and the ensuing increase in transparency between the CSC and the CSP is supported by two main steps once the composite events have been specified: the generation and triggering of software agents for conducting detecting the primitive events and the reasoning on whether an overall event pattern of interest has taken place. This will ultimately lead to the generation of reports and alerts on the dashboard of the cloud stakeholder of interest.

5.1 Generation and Triggering of Agents

To conduct the audits, an organization of software agents is used. The first type of agent or *probe agents*, purport to conduct the detection of the primitive events within the pattern specified by the cloud stakeholder. The second type of agent is a single agent in some cases, referred to as *Event receiver* which role is to filter and aggregate the set of inputs that arrive from the probe agents before they are passed to the reasoning engine. In order to ensure the intrinsic link between a primitive event and a probe agent, we adopt the following reference format for agent during their generation: *Agent<EventName>@<Domain>,* where domain refers to the name given to the audit platform; and eventName- a unique name given to the event in the event structure provided in Section 3.1. In practice,

the probes agents actually carrying out the instruction within the *Rule* field of the primitive event, thus triggering an existing security tool or launching a pseudo code before collecting the results of the check. The creation and management of the agents can be done through a MAS platform such as JADE.

5.2 Reasoning and Detecting a Pattern

Primitive events alone may not always be significant to portray the emergency of a situation. In contrast, considering a pool of events from the same or different sources within the infrastructure of the CSC/CSP may reveal a pattern that relate to an anomaly or impending risk for the service stakeholder. As mentioned in the previous section, since we have adopted a tree based representation and validation of event patterns, efficiently detecting a pattern of events depends on how the tree is processed, and individual events are handled upon their detection. With this in mind, we have tailored for each of the event pattern specified in Section 3.2, a dedicated algorithm that is resorted by the reasoning engine to determine whether the composite event has materialized. Owing to page limitation, we only provide a description of some amongst them:

Any: this operator has a list of events and is validated if one of those events occurs. If the operator above this operator is a count operator, then this operator will reset only the event that validated it. This ensures that any other subsequent event in the list that was partially validated, will still be accounted for.

```
Algorithm:
boolean valid = false
for each event in list {
   if event is validated {
      add event to happenedList
      valid = true
   }
}
return valid
```

Part Of: This operator has a list of events and a trigger number. It is validated when the number of events validated in the list is equal or above to the trigger number. If the operator above this operator is a count operator, this operator will reset only the events of the list that are validated. Therefore if another event in the list was partially validated, it will not lose its current state.

```
Algorithm :
clearhappenedList
for each event in list {
   if event is valdiated {
      add event to happenedList
   }
}
if happenedList size >= trigger {
   return true
}
return false
```

5.3 Audit Reporting

Upon the processing of the events by the reasoning engine, it sends the result to the Report Generator which subsequently displays it at the HMI or Dashboard. Given the adoption of a tree based specification of the composite events to monitor for, the Event designer could also serve the purpose of visual dashboard whereby events detected by the agents get automatically highlighted in a different colour. For further details information on the detected events, another dashboard could be added, giving details on the event's name, primary identifier , date of creation, the probe agent tasked with detecting it, and its status (for instance *Occur or not happened*). Such information could then allow the CSP and CSC to respectively, to detect any nefarious and malicious use of its service by a CSC, detect any security anomaly within their virtual machines and/or held the CSP accountable for a breach of contractual agreement related to security and quality of service.

6 APPLICATION TO A REAL SCENARIO

The scenario described thereafter is a real use case we have worked on though the names of the companies involved have been altered.

L-BANK which is actually affiliated to an international Bank, has decided to use the service of D.CLOUD owing to the fact that the local branch was closing down. As information related to the bank customers could not be moved beyond the boundary of the country, it was therefore imperative to found a reliable service that could carry out the archiving in accordance with the legal framework. D.CLOUD was chosen primarily for the fact that archiving was the core of its business and also because of the security reliability (supported by certifications). This means a considerable amount of archives (bank customers' information) were ready to be transferred to D.CLOUD through SFTP connection mode for fast integration. L-BANK was then given an access to D.CLOUD's standard web portal. D-CLOUD could create new archives, search, restore, and consult individual archives. This portal is accessible directly over Internet (HTTPS) or through a VPN. At the end of the retention period, the

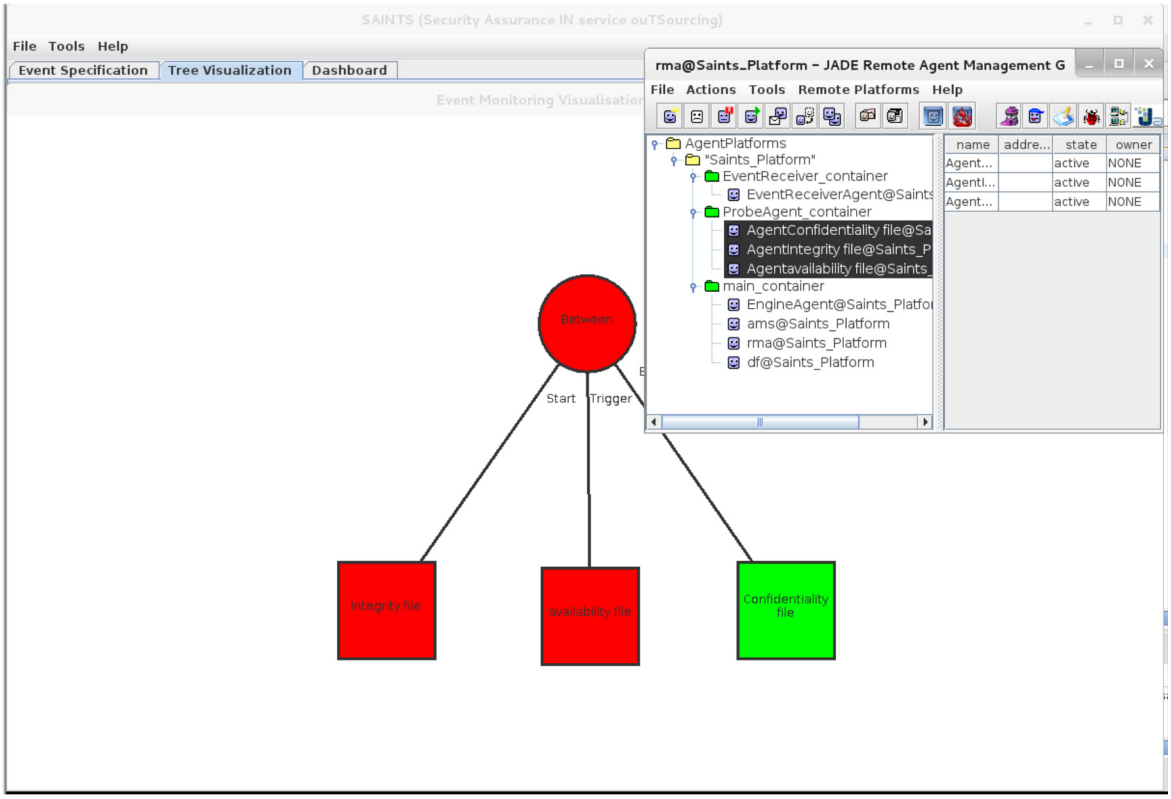

Figure 2: Launch of a JADE based agents and visualization of archive's related events.

documents are available for deletion or archiving. From L-BANK's view point the following events were identified as being relevant for monitoring:

-Confidentiality of the ANY (C1, C2) where

C1: An attacker or the cloud provider staff may access. the archive stored on the cloud provider server without authorisation.

C2: An error in storage or a bad virtual separation may cause a breach of confidentiality.

- Integrity of the Archives ANY (I1, I2, I3)

I1: Unauthorised modification by hackers may occur when the archive is stored on the CSP's server.

I2: A loss of integrity of the archive due to a corruption of data may occur when the cloud client consults the archive through the archive portal.

I3: In the documents management configuration, when the client consults the archive, an error may cause a loss of integrity or a deletion of the archive.

-Availability of the Archives ANY (A1, A2) where

A1:The platform may be unavailable due to a traffic overloading or a denial of service attack.

A2: Transmission and communication errors may occur when the archive is send to the cloud client.

The D-CLOUD in turn was concerned with unwarranted modification and activities from L-BANK with consequences on its services:

ANY (M1, M2) where:

M1:The cloud client may intentionally or not upload a malicious file as a archive.

M2: an error in the system or archive maintenance may result in unwanted changes.

Noteworthy, the specifcation of composite events combining events from the three groups is possible as it was the case during the application to the use case (Figure 1-2). Similarly most of the events above listed can be further specified as composite events of their own, but due to page restriction such an exrcicse will not be conducted in this paper.

7 CONCLUDING REMARKS

This paper has reported on an initiative for addressing security transparency and allowing mutual auditability within a cloud realm. Given that the use of raw logic based language may prove a challenge for some stakeholders wishing to engage in the systematic audit and monitoring of security and QoS related parameters, the choice of a tree based representation was made. Another key aspect of the approach is the adoption of the XCCDF format, which enables the reuse of existing security management tools.

The initial prototype based on the concepts and algorithms presented has been validate on an electronic archiving platform with the event specification and detection console allocated to a dedicated virtual machine, while the multi agent system platform JADE has been adopted for the specification and management of agent entrusted with the role of detecting primitive events as can be seen in Figure 2. NAGIOS plugins (Pervilä, 2007) along with other tailored programs where developed for the detection of primitive events within the Infrastructure.

The initial results were very encouraging as most of the security events of concerns provided by the SaaS provider and consumer and specified using the Event designer were detecting, by simulating alterations and attacks targeting the archived files. Furthermore, the capacity of the VM required for hosting the whole application (Event Designer and multi-agent detection platform) was confine to a 2 Go of RAM and in single CPU. Nonetheless, further applications are envisaged for better appraising the effect of deploying simultaneously a multitude of agents for detecting and reporting events of interest.

AKNOWLEDGEMENTS

This work has been conducted in the context of the SAINTS project, financed by the national fund of research of the Grand Duchy of Luxembourg (FNR) under grant number C12/IS/3988336. The authors also thanks Maimouna Seck and Charles Hubert Duthilleux for their work on implementing the tool.

REFERENCES

Anicic D., Rudolph S., Fodor P., Stojanovic N.: Stream reasoning and complex event processing in ETALIS. *Semantic Web 3(4*): 397-407 (2012).

Bellifemine F., Caire G, Poggi A., Rimassa G. 2008 JADE: A software framework for developing multi-agent applications. Lessons learned. *Information & Software Technology* 50(1-2): 10-21.

Carasso D. (2012) Exploring Splunk, CITO Research, New York.

Chen Y, Paxson V, Katz RH (2010) What's New About Cloud Computing Security? Report EECS Department, University of California, Berkeley,
http://www.eecs.berkeley.edu/Pubs/TechRpts /2010/EECS-2010-5.html.

Dölitzscher F., Knahl M., Reich C., Clarke N.L. 2013 Anomaly Detection in IaaS Clouds*. In proceedings of CloudCom* (1) 387-394.

Etzion O., Niblett P. 2010. Event Processing in Action. Manning Publications Company 2010, ISBN 978-1-935182-21-4, pp. I-XXIV, 1-360.

Lorenzoli D., Spanoudakis G. 2010 EVEREST+: Runtime SLA Violations Prediction: *In: Proceedings of the 5th Middleware for Service-oriented Computing Workshop*, ACM.

Luckham D. C. (2005) The power of events - an introduction to complex event processing in distributed enterprise systems. ACM 2005, ISBN 978-0-201-72789-0, pp. I-XIX, 1-376.

Ganzha M, Paprzycki M. (2014): Agent-oriented computing for distributed systems and networks. J. Network and Computer Applications 37: 45-46 (2014). McAfee and Guardian Analytics. 2012. Dissecting. Operation High Roller. Accessed 10 December 2014. From: http://www.mcafee.com/us/resources/reports/rp.operation-high-roller.pdf.

Nuñez D., Fernandez – Gago C., Pearson S., Felici M. 2013 A Metamodel for Measuring Accountability Attributes in the Cloud. In: *Proceedings of the 2013 IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2013),* IEEE.

Ouedraogo M., Khadraoui D., Mouratidis, H. and Dubois E. (2012): Appraisal and reporting of security assurance at operational systems level. *Journal of Systems and Software* 85(1): 193-208 (2012).

Ouedraogo M, Mouratidis M (2013) Selecting a cloud service provider in the age of cybercrime, Computers & Security, vol.38, pp.3-13 Special issue on Cybercrime in the Digital Economy, Elsevier.

Ouedraogo M., Kuo C.T, Tjoa S., Preston D, Dubois E., Simões P., Cruz T.: Keeping an Eye on Your Security Through Assurance Indicators. In proceedings of SECRYPT 2014: 476-483.

Pervilä, M.A., 2007. Using Nagios to monitor faults in a self-healing environment. *In:Seminar on Self-Healing Systems.* University of Helsinki.

Rak M, Liccardo L, Aversa R 2011. A SLA-based interface for security management in cloud and GRID integrations. I*n: Proceedings of the 7th International. Conference on Information Assurance and Security (IAS)*, pp.378-383, IEEE.

Robert J. Zhang, Elizabeth A. Unger (1996) Event Specification and Detection Technical report TR CS-96-8, 1996, *Kansas State University.*

Sunyaev A., Schneider S. 2013. Cloud services. certification *Communication of the ACM* 56(2): 33-36, ACM digital Library.

Waltermire D., Schmidt, C., Scarfone K.

Winkler V. (2011) Securing the cloud- cloud computer. security techniques and tactics. Syngress.

Ziring N. 2012. Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2*, NIST Interagency Report 7275Revision 4,* National Institute of Standards and Technology Gaithersburg, MD 20899-89.