Towards Hybrid Cloud Service Certification Models

Spyros Katopodis, George Spanoudakis, Khaled Mahbub
City University London
London, UK
e-mail: {Spyros.Katopodis.1, G.E.Spanoudakis, K.Mahbub}@city.ac.uk

Abstract— **In this paper, we introduce a hybrid approach for certifying security properties of cloud services that combines monitoring and testing data. The paper argues about the need for hybrid certification and examines some basic characteristics of hybrid certification models.**

Keywords-cloud security, certification, hybrid models

I. INTRODUCTION

The certification of cloud service security has become a necessity due to on-going concerns about cloud security and the need to increase cloud trustworthiness through rigorous assessments of security by trusted third parties [1][2]. Unlike the certification of security in traditional software systems, which is based on static forms of security assessment (e.g., the Common Criteria model [11]), the certification of cloud service security requires continuous assessment [5]. This is because cloud services are provisioned through dynamic infrastructures operating under security controls and other configurations that may change dynamically introducing unforeseen vulnerabilities. Cloud service security can also be compromised because of attacks on co-tenant services.

Recent work on cloud service certification applies dynamic forms of security assessment, notably dynamic testing (e.g., [4]) or continuous monitoring (e.g., [6][10]). These overcome some of the limitations of traditional security certification and audits (e.g. they produce machine readable certificates incorporating dynamically collected evidence). However, there are cases where existing approaches cannot provide an adequate level of assurance. Testing, for instance, may be insufficient for transactional services, as it is normally performed through a special testing (as opposed to the operational) service interface. Monitoring based certification may also be insufficient if there is conflicting or inconclusive evidence in monitoring data; such data may, for example, not cover all traces of system events that should be seen to assess a property.

To overcome such problems, we are working on a hybrid approach for certifying cloud service security that could combine both monitoring and testing evidence. Our approach is based on the cloud certification framework of the CUMULUS project [1]. In this paper, we introduce the basic concepts of hybrid certification models and present examples of such models formalised using EC-Assertion (i.e., the monitoring language of the CUMULUS framework).

The rest of the paper is organized as follows: Sect. 2 reviews related work; Sect. 3 introduces hybrid certification models, giving formal examples of such models; and Sect. 4 presents concluding remarks and directions of future work.

II. RELATED WORK

Dynamic cloud security certification is the focus of some recent work. The Cloud Security Alliance has generated the STAR self-assessment certification framework [6]. STAR is limited to monitoring and, to the best of our knowledge, is not implemented yet. A configurable cloud certification framework allowing the definition and realisation of different monitoring based cloud certification models is described in [10]. Monitoring has also been combined with model checking techniques to assess properties of software cloud services in [7]. A test based security certification scheme for cloud services has been proposed in [12]. Providing security assurance through IT audits has been the norm in industrial practice. IT audits, however, focus on providing guidelines for inspection of security controls on IT and cloud infrastructures and they are not automated [3][8].

III. HYBRID CERTIFICATION MODELS

A. Background: CUMULUS Certification framework

Our work on hybrid security certification is part of the EU Project CUMULUS, which focuses on incremental, multiple evidence and multi-layer cloud service security certification. In CUMULUS, certification is a process that is carried out according to a *certification model* [13]. This model defines: (i) the *security property* to be certified, (ii) the cloud service that this property applies to (aka *target of certification* (TOC)), (iii) the *evidence* that should be used to assess the property, (iv) the conditions that determine the *sufficiency* of evidence for issuing a certificate for the property, and (v) ways to treat *conflicts* in the evidence.

CUMULUS offers a monitoring infrastructure for realising monitoring based certification models based on EVEREST [10]. EVEREST enables the monitoring of runtime events produced by distributed systems based on rules and assumptions expressed in an Event Calculus based language, called EC-Assertion [10]. Rules express conditions that must be satisfied at all times by runtime events, whilst assumptions express the ways of deducing information from such events (e.g. the state of the monitored system). Both rules and assumptions are defined in terms of *events* and *fluents*. An event is something that occurs at a specific instance of time and has instantaneous duration. Fluents represent system states and are initiated and terminated by events. The basic predicates used by EC-Assertion are:

- *Happens(e,t,[L,U])* – This predicate denotes that an event *e* of instantaneous duration occurs at some time point *t* within the time range *[L,U]*. An event *e* is specified as *e(_id,_snd,_rcv,TP,_sig,_src)* where *_id* is its unique id of it, *_snd* is its sender, *_rcv* is its receiver, *_sig* is its signature, and *_src* is the source where *e* was captured from. *TP* is the event's type. EC-Assertion offers three built-in event types: (a) captured operation calls (REQ), (b) captured operation responses (RES) and (c) forced operation execution events (EXC), i.e., operation executions triggered by the monitor itself.
- *Initiates(e,f,t)* – This predicate denotes that a fluent *f* is initiated by an event *e* at time *t*.
- *Terminates(e,f,t)* – This predicate denotes that a fluent *f* is terminated by an event *e* at time *t*.
- *HoldsAt(f,t)* – This is a derived predicate denoting that a fluent *f* holds at time *t. HoldsAt(f,t)* is true if *f* has been initiated by some event at some time point *t'* before *t* and has not been terminated by any event within *[t',t]*.

B. Hybrid certification models

The key concept underpinning a hybrid certification model is to cross-check evidence regarding a security property that has been gathered from testing and monitoring and, provided that there is no conflict within it, to combine it providing assurance for properties. Consider, for example, a scenario where the property to be certified is cloud service availability. If availability is measured as the percentage of the calls to service operations for which a response was produced with a given time period *d*, a monitoring check should verify exactly this condition. However, the trace of service calls that has been examined by the monitoring process might not cover all the operations in the service interface or the expected peak workload periods of the underlying infrastructure. In such cases, before issuing a certificate for service availability, it would be necessary to test any of the above service usage conditions that have not been covered yet. The combination of monitoring and testing can be attempted in two basic modes:

(1) The *dependent mode* – In this mode, a security property is assessed for a TOC by a *primary* form of assessment (monitoring or testing) which triggers the other (*subordinate*) form in order to confirm and/or complete the evidence required for the assessment.

(2) The *independent mode* – In this mode, a security property is assessed for a TOC by both monitoring and testing independently without any of these assessments being triggered by outcomes of the other. Then at specific points defined by the evidence sufficiency conditions of the certification model the two bodies of evidence are correlated and cross-checked to complete the hybrid assessment.

Beyond the elements of certification models that were overviewed in III.A, a hybrid certification model should also define: (a) the mode of hybrid certification; (b) the way of correlating monitoring and testing evidence; (c) conditions for characterising these types of evidence as conflicting, and (d) the way in which a final overall assessment of the property can be generated based on both types of evidence.

In the following, we give examples of hybrid certification models of both modes, formalise them in EC-Assertion and use this formalisation to examine generic relationships that exist in hybrid models.

C. Example 1: Hybrid, dependent mode models

Our first example shows the use of a hybrid approach in certifying *data integrity-at-rest*. As defined in [9], this property expresses the ability to detect and report any alteration of stored data in a target of certification (TOC).

To demonstrate the difference between monitoring and hybrid certification models, we first present the monitoring certification model for data integrity-at-rest, expressed by the EC_Assertion monitoring rule R1 that is listed below. The specification of this rule as well as all models in the paper, assumes the following agents and variables denoting them: service consumers (_sc), target of certification (_TOC), authentication infrastructure (_AI), certification authority (_CA).

```
Rule R1:
Happens(e(_e1,_sc,_TOC,REQ,_updOp(_cred,_data,_auth),_T
OC),t1,[t1,t1]) ^
Happens(e(_e2,_TOC,_AI,RES,_updOp(_cred,_data,_vCode),_
TOC),t2,[t1,t1+d1]) ^ (_vCode ≠ Nil) ⇒
Happens(e(_e3,_TOC,_A,REQ,_notifO(_cred,_data,_auth,_h)
,_TOC),t3,[t2,t2+d2])
```

According to R1 when a call of an update operation in a `_TOC` is detected at some time point t1 (see event `Happens(e(_e1,_sc,_TOC,REQ,_updOp(_cred,_data,_auth),_TOC),t1,[t1,t1])`) and a response to this call occurs after it (see event `Happens(e( _e2 ,_TOC, _AI, RES,_updOp(_cred, _data,_verCode),_TOC),t2,[t1,t2+d1])`) indicating that the request has been granted (see condition `(_vCode ≠ Nil)` in the rule), the monitor should also check for the existence of another event showing the call of an operation in some authorisation agent `_A` to notify the receipt and execution of the update request (see `Happens(e(_e3,_TOC,_CA, REQ,_notifO(_cred,_data,_auth,_h),_TOC),t3,[t2,t2+d2])`)[1]. The above model has two limitations in providing assurance for the integrity-at-rest property: (1) it cannot capture updates of data that might have been carried out without using the update interface assumed of `_TOC` (i.e., `_updOp(_cred,_data,_vCode)`), and (2) it cannot check that the operation `_updOp` has checked authorisation rights before updating data, and

A hybrid model could be used in this case to overcome partially the first of these limitations. More specifically, a hybrid model in this case could be based on periodic testing to detect if stored data have been modified and monitor the periods between the tests that revealed data modifications to check if appropriate notifications have also been sent. Data

[1] Note that the operation signatures used in the rule may change depending on `_TOC` without affecting the generality of the rule.

modifications could be detected by obtaining the hash value of the relevant data file in the TOC periodically. Then, if across the execution of two consecutive tests, the last retrieved hash value of the file is different from the previous hash value, a data modification action can be deduced. In parallel with the execution of this periodic test, the hybrid model will also monitor the execution of notification operations. Hence, when a data modification action is detected by two consecutive tests, the hybrid model could also check whether a correlated notification operation has been executed within the period between the tests.

This hybrid model model can be expressed using the following monitoring rule and assumption:

```
Rule R2: Happens(e(_e1,_CA,_TOC,EXC(Tper),
_getHash(_TOC,_file,_h1),_CA), t1, [t1,t1])  ^
HoldsAt(LastHash(_file,_h2,t2),t1) ^ (_h1 ≠ _h2) ⇒
Happens(e(_e3,_TOC,_CA,REQ,_notifO(_cred,_data,
_auth,_h1),_TOC),t3,[t2,t1])

Assumption A1: Happens(e(_e1,_CA,_TOC,REQ,
_getHash(_TOC,_file,_h1),_TOC),t1,[t1,t1])  ^
HoldsAt(LastHash(_file,_h2,t2),t1) ^ (_h1 ≠ _h2) ⇒
Terminates(_e1,LastHash(_file,_h2,t2),t1) ^
Initiates(_e1,LastHash(_file,_h1,t1),t1)
```

Rule R2 is "hybrid" as it includes normal monitoring events (i.e., `REQ` and `RES` events) and events that trigger the execution of tests (i.e., `EXC` events). R2 expresses a hybrid dependent mode model where evidence arising from testing triggers the acquisition of monitoring evidence. Hence, testing is the primary form of assessment. In particular, R2 forces the execution of the event `Happens(e(_e1, _CA, _TOC, EXC(`T_{per}`), _getHash(_TOC, _file,_h1),_TOC), t1,[t1,t1])` periodically every T_{per} time units to invoke the operation `_getHash` in the testing interface of `_TOC` and obtain the current hash value (`_h1`) of the data file (`_file`) of `_TOC`. If this value is different from the hash value recorded by a previous test at some t2 (i.e., the value recorded in the fluent `LastHash(_file,_h2,t2),t1`), rule R2 checks if an update notification has also occurred between t2 and t1, as expressed by the monitoring event `Happens(e(_e3,_TOC,_A,REQ,_notifO(_cred,_data,_auth,_h1),_TOC),t3,[t2,t1])`. The hybrid model uses also a monitoring assumption (i.e., A1). This assumption is used in the model to update the hash value recorded in the fluent `LastHash`, if a test retrieves a hash value that is different from the last recorded one.

Although the above model can capture data updates that have taken place without the invocation of the file updating interface, it cannot guarantee that it can capture all possible updates that might have taken place. In particular, it won't be able to detect if more than one updates have taken place between two consecutive executions of the periodic test. Hence, it addresses the first of the limitations of the monitoring problem (i.e., limitation (1)) only partially.

To address the second limitation of the monitoring model (i.e., limitation (2)), it is possible to construct a different hybrid model. This model could rely on testing to ensure that every time that an agent that requests a data alteration, it has the authorisation right to do the requested alteration. This model can be expressed by the monitoring rule below:

```
Rule R3: Happens(e(_e1,_sc,_TOC,REQ,_updOp(_cred,_data,
_auth),_TOC),t1,[t1,t1])  ^
Happens(e(_e2,_TOC,_AI,RES,_updOp(_cred,_data,_vCode1),
_TOC),t2,[t1,t1+d1]) ^ (_vCode1 ≠ Nil) ⇒
Happens(e(_e3,_CA,_AI,EXC,_authorO(_cred,_auth,_vCode2)
,_TOC),t3,[t2,t2+d2])^(_vCode2≠Nil)
```

Rule R3 monitors requests for updates of `_TOC` data through its normal updating interface. However, for every such request that is granted by `_TOC`, it requests the execution of a test to check if the entity that requested the update had indeed the authorisation to update data. This is expressed by the `EXC` event `Happens(e(_e3,_CA,_AI,EXC,_authorO(_cred,_auth,_verCode2),_TOC),t3,[t2,t2+d2]))` and the condition `_verCode2 ≠ Nil`. In R3, the monitoring evidence triggers the execution of tests. Hence, the rule expresses a dependent hybrid model where monitoring is the primary form of assessment. Rules R2 and R3 are examples of general time correlation structures that may arise in dependent hybrid certification model and which are shown in Figure 1.

Part (a) of the figure shows dependent hybrid certification models where testing is the dominant form of assessment. In such models, test plans each consisting of a series of tests (i.e., $\{Test_{n1},\ldots,Test_{nL}\}$) are executed according to some periodic schedule. Assuming that the execution of a test plan starts at $t_s^{(n)}$ and ends at $t_e^{(n)}$, the hybrid model may also check for monitoring events that occurred within the interval $[t_s^{(n)}–d_1,\ t_e^{(n)}+d_2]$ in order to provide an assessment of the security property of interest. Note that the length of the execution of each test plan and the monitoring events found within $[t_s^{(n)}–d_1,\ t_e^{(n)}+d_2]$ may vary.

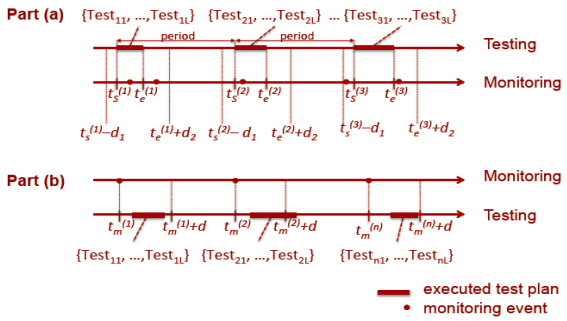

Figure 1. Dependent mode hybrid certification models

Part (b) of the figure shows the timelines of evidence collection in dependent hybrid certification models where monitoring is the dominant form of assessment. In such models following the collection of monitoring evidence (events), tests plans are executed to cross-check/complete it. The execution of these plans starts within the range $[t_m^{(n)},\ t_m^{(n)}+d]$ where $t_m^{(n)}$ is the time of occurrence of the last event in a pattern of events that should trigger the execution of the plan and *d* is a period set by the model. The length of the execution of each test plan may vary.

D. Example 2: Hybrid, independent mode models

Our second example shows the use of a hybrid approach in certifying cloud service availability. As defined in [9], this property expresses the ability of a TOC to produce a non-faulty response within a certain period of time and is measured by the percentage of calls that satisfy this condition over an assessment period. An independent hybrid model for the certification of TOC availability could be based on collecting evidence regarding the availability of a TOC through monitoring and testing independently (i.e., without any of these activities being triggered by outcomes of the other) and then correlating and cross-checking the collected pools of evidence to produce a hybrid assessment of the property. More specifically, the hybrid model could include monitoring formulas to record instances of invocation of TOC operators where TOC produced a response within the acceptable time limit and the instances where it did not, and keep a record of counters of these instances from which an overall availability measure could be drawn. The formulas that could be used to collect this monitoring evidence are as follows:

Assumption A2 (monitoring evidence):
Happens(e(_e1,_CA,_TOC,REQ,_OP(_data),_TOC),t1,[t1,t1])
^ **Happens**(e(_e2,_TOC,_CA,RES,_OP(_data),_TOC),
t2,[t1,t1+t_{av}]) ^ **HoldsAt**(MCounterA(_TOC,_MCA),t2)⇒
Terminates(_e1,MCounterA(_TOC, _MCA), t2) ^
Initiates(_e1,MCounterA(_TOC, _MCA+1), t2)^
Initiates(_e1,MAvail(_TOC,_OP(_data),t2-t1), t2)

Assumption A3 (monitoring evidence):
Happens(e(_e1,_CA,_OC,REQ,_OP(_data),_TOC),t1,[t1,t1])
^¬**Happens**(e(_e2,_TOC,_CA,RES,_OP(_data),_TOC),t2,[t1,t1
+t_{av}]))^ **HoldsAt**(MCounterU(_TOC,_MCU), t2) ⇒
Terminates(_e1,MCounterU(_TOC,_MCU), t2) ^
Initiates(_e1,MCounterU(_TOC, _MCU+1), t2) ^
Initiates(_e1,MUnav(_TOC,_OP(_data),t2-t1),t2)

The first of the above monitoring formulas (i.e., assumption A2) monitors calls to any operation in a _TOC and the responses to them (see events Happens(e(_e1, _CA, _TOC, REQ, _OP(_data),_TOC), t1, R(t1,t1)) and Happens(e(_e2, _CA, _TOC, RES, _OP(_data), _TOC), t2,[t1,t1+t_{av}])) and if a response is within the required period (t_{av}), it updates the counter of instances where _TOC was available and records the related call (in fluents MCounterA(_TOC,_MCA,t2) and MAvail(_TOC,_OP(_data),t2-t1), respectively). The second formula (i.e., assumption A3) monitors calls to _TOC operations that did not produce a response within the required time, and keeps an overall counter of unavailability and the related calls in fluents MCounterU(_TOC,_MCU,t2) and MUnav(_TOC,_OP(_data),t2-t1)).

The hybrid model for the certification of availability could also incorporate a test-based availability assessment sub-model. This sub-model can execute a randomly selected operation in the interface of _TOC periodically to check its availability, and keep a record of instances of test-triggered invocations of operations of TOC in which a response was produced within the required time period, and instances of test-triggered invocations where it was not.

This sub-model is expressed by following formulas for collecting testing evidence:

Assumption A4 (testing evidence):
Happens(e(_e1,_CA,_TOC, EXC(T_{per}),
_x=random(interface(_TOC)),_TOC),t1,[t1,t1]) ^
Happens(e(_e2,_TOC, _CA RES,_x,_TOC),t2,[t1,t1+t_{av}])^
HoldsAt(TCounterA(_TOC,_TCA),t2)
⇒ **Terminates**(_e1,TCounterA(_TOC,_TCA),t2) ^
Initiates(_e1,TCounterA(_TOC,_TCA+1),t2)^
Initiates(_e1,TAvail(_TOC,_x,t2-t1),t2)

Assumption A5 (testing evidence):
Happens(e(_e1,_CA,_TOC, EXC(T_{per}),
_x=random(interface(_TOC)),_TOC),t1,[t1,t1]) ^
¬**Happens**(e(_e2,, _TOC, _CA, RES,_x,_TOC),t2,
[t1,t1+t_{av}])^ **HoldsAt**(TCounterU(_TOC,_TCU),t2) ⇒
Terminates(_e1,TCounterU(_TOC,_TCU),t2) ^
Initiates(_e1,TCounterU(_TOC,_TCU+1),t2) ^
Initiates(_e1,TUnav(_TOC,_x, t2-t1),t2)

A4 and A5 are similar to assumptions A2 and A3 respectively except that, instead of monitoring real operation calls, they execute a randomly selected operation in the interface of _TOC periodically (see the event Happens(e(_e1,_CA,_TOC,EXC(T_{per}),_x=random(interface(_TOC)), _TOC),t1,[t1,t1])) to check its availability, and update fluents recording the overall counters of availability and unavailability of _TOC and the test executions that revealed them.

In the hybrid model, the assumption pairs (A2, A3), and (A4, A5) are used to collect evidence independently without any monitoring events triggering tests or vice versa. However, it might still be desirable to correlate the testing and monitoring evidence. For example:

(a) The overall availability measure may be computed on the basis of both test and monitoring evidence as A = (_MCA + _TCA)/(_MCA+ _TCA+_MCU+ _TCU).

(b) Testing events may be considered as valid evidence of TOC availability only if all real calls of TOC in the range [t_{mon}–d, t_{mon}+d] produced responses within t_{av}.

(c) Monitored calls to TOC that produced a response within the maximum allowed period t_{av} may be disregarded in computing TOC's availability if the relevant responses were marginally below t_{av} and all testing events for the same TOC in the range [t_{mon}–d,t_{mon}+d] (t_{mon} is the timestamp of a monitored call) produced responses after the maximum allowed period t_{av}.

(d) An availability measure based on testing (monitoring) evidence will be used for issuing a certificate only if the availability measure based on monitoring (testing) evidence over the same period is no more than 1% different from it.

Clearly, several other combinations of monitoring and testing evidence may be defined. In general, in an independent hybrid certification model:

- individual (or groups of) instances of monitoring and testing evidence may be cross-validated against each other before producing an overall assessment on the basis of any of these types of evidence (see (b) and (c) above);

- aggregate assessments based on each type of evidence may be validated against an aggregate assessment based on the other type before issuing a certificate (see (d) above); or
- an aggregate assessment may be formulated from both monitoring and testing evidence as in case (a) above.

Compared to non-hybrid models for certifying availability, the hybrid model introduced above can produce availability assessments of higher confidence as the monitoring and testing evidence can be cross-checked before being used in an assessment (and certificate) and can both be included in a certificate depending on the chosen validation checks. Hybrid models offer also a more extended pool of evidence and possibilities to decide which data are relevant and of sufficient quality so that they can be taken into consideration for issuing a hybrid certificate. Apart from increasing the confidence level of assessments, hybrid models are also more customisable than traditional certification models since they offer the choice of deciding how test and monitoring evidence should be correlated, cross-checked and used in assessments.

IV. COMPARISON BETWEEN HYBRID AND TRADITIONAL CERTIFICATION MODELS

In this section we present a comparison between hybrid and traditional certification of cloud services. Traditional approaches for certifying security properties rely on manual inspections and audits, proving to be static, inflexible, non-automated and unable to realise the economic dimension that the Cloud entails [14]. As already stated, CSA has generated the STAR self-assessment certification framework that allows cloud providers to submit self-assessment reports, when fully implemented [6]. At the same time, CSA has generated the Cloud Controls Matrix (CCM), which provides a framework of controls that gives detailed understanding of security concepts and principles that are aligned to the Cloud Security Alliance guidance in 13 domains [15]. CCM facilitates regulatory compliance and provides organizations with the needed structure, detail and clarity relating to information security tailored to the cloud industry. It is also specifically 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, integrating the ISO/IEC 27001 management systems standard **Error! Reference source not found.**. However, CCM is human-process centric requiring from the companies that adopt it to address the issues that they define critical concerning cloud security and to pre-assess how mature their systems are [16][20]. CCM enables the integration, monitoring and managing of cloud services through a framework that can take care of the elementary issues regarding cloud security [16], but it does not support certification as an automated service in the cloud [17].

Traditional certification models (i.e. ISO/IEC 27001, NIST) also require manual inspections and are unable to provide the required level of assurance in cloud computing and to fit the dynamic nature of the cloud, focusing on monolithic software components [4] and failing to address on-demand self-service, dynamic allocation of resources and multi-tenancy [14][18]. Additionally, traditional certification models lack in trust, transparency and accuracy, as they do not support the constant provision of information about the security of cloud services, unlike our hybrid approach that relies on incremental monitoring and automated testing and it is focused on cloud services.

IT audits have been widely used in providing security assurance. Security auditing approaches focus on providing guidelines and are not automated [8]. One more drawback is that they require that the consumer relies on third-party auditors for security assurance.

Common Criteria (CC) certification uses Evaluation Technical Reports [19]. CC has also a human-centric approach, unlike our model, which is not designed to support automated security certification, targeting static, monolithic systems and requiring a large investment of resources [4].

V. CONCLUSIONS

The certification of cloud service security properties often needs to be based on hybrid models combining testing and monitoring evidence that have been collected dynamically during service provision. In this paper, we have examined some of the basic characteristics of hybrid certification models based on the formal modelling of examples of such models. Hybrid certification models as introduced are based on continuous monitoring of security services and automated testing of these. Consequently, the new dynamic models will support automation of the certification process and enhanced customer reliability and trust to the cloud services compared to the previous traditional ways of certifying security properties on the cloud. We have shown that the combination of testing and monitoring evidence can happen in two basic models (independent and dependent) and examined temporal relationships between the testing and monitoring process under these two modes. The example models that we have presented are implementable in EVEREST, the monitoring infrastructure of the CUMULUS project. Our next steps are to expand the certification model specification language of CUMULUS to enable the definition of hybrid models, and evaluate our approach experimentally.

ACKNOWLEDGMENT

This work reported in this paper has been partially funded by the EU F7 project CUMULUS (grant no 318580).

VI. REFERENCES

[1] G. Spanoudakis, E. Damiani, A. Mana, "Certifying Services in Cloud: The Case for a Hybrid, Incremental and Multi-layer Approach," IEEE 14th Int. Symp. On High-Assurance Systems Engineering, 2012.

[2] D. Catteddu, G. Hogben, "Cloud Computing: Benefits, Risks and Recommendations for Information Security," ENISA, 2009.

[3] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing v2.1," http://www.cloudsecurityalliance.org/guidance/csaguide.v2.1.pdf .

[4] S. Cimato, E. Damiani, F. Zavatarelli, R. Menicocci, "Towards the Certification of Cloud Services," IEEE 9th World Congress on Services, 2013.

[5] NIST, Guide for Applying the Risk Management Framework to Federal Information Systems: A Security Life Cycle Approach, (NIST Special Publication 800-37), Feb 2010.

[6] J. Reavis, D. Catteddu "Open Certification Framework. Vision Statement," Cloud Security Alliance, Aug 2012.

[7] H. Foster, G. Spanoudakis, K. Mahbub, "Formal Certification and Compliance for Run-Time Service Environments," IEEE 9th Int. Conf. on Services Computing, 2012.

[8] Z. Chen, J. Yoon, "IT Auditing to assure a secure cloud computing," 2010 IEEE 6th World Congress on Services.

[9] CUMULUS consortium, "Security-aware SLA specification language and cloud security dependency model," Deliverable D2.1, Sep 2013.

[10] G. Spanoudakis, C. Kloukinas, K. Mahbub, "The SERENITY Runtime Monitoring Framework" in Security and Dependability for Ambient Intelligence, Springer, pp.213-237, 2009.

[11] D. S. Herrmann, Using the Common Criteria for IT Security Evaluation, CRC Press, Inc., Boca Raton, FL, USA, 2002.

[12] M. Anisetti, C. A. Ardagna, A..Damiani, A. Saonara, "A test-based security certification scheme for web services," ACM Trans. Web 7(2), 2013.

[13] CUMULUS consortium, "Certification models," Deliverable D2.2, Sep 2013.

[14] Windhorst I., Sunyaev A., "Dynamic Certification of Cloud Services," Eighth International Conference on Availability, Reliability and Security (ARES), 2013 ,412-417, 2013

[15] https://cloudsecurityalliance.org/star/self-assessment/

[16] https://cloudsecurityalliance.org/research/ccm/

[17] Saxena S., "Ensuring Cloud Security Using Cloud Control Matrix", International Journal of Information and Computation Technology, pp. 933-938, 2013.

[18] Kaliski B. Jr., Pauley W., "Toward Risk Assessment as a Service in Cloud Environments.", Proceeding HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, 2010.

[19] Kaluvuri S.P., Koshutanski H., Cerbo F.D., Mana A, "Security Assurance of Services through Digital Security Certificates". in Proc. of the 20th IEEE International Conference on Web Services (ICWS 2013). 2013.

[20] Reavis J.,Catteddu D,."Cloud Security Alliance - Open Certification Framework Vision Statement", Aug 2012