Towards Transparent and Trustworthy Cloud

Marco Anisetti, Claudio A. Ardagna, and Ernesto Damiani, DI–Università degli Studi di Milano
Antonio Maña, University of Malaga
George Spanoudakis, City University of London

Despite its immense benefits, cloud computing raises several concerns due to lack of trust and transparency. The proposed process and infrastructure are applied to a case study on cloud certification showing their utility.

The cloud computing paradigm provides a vision of IT where resources are consumed and leased on demand and on a pay-per-use basis. The immense benefits of the cloud in terms of flexibility, resource consumption, and simplified management, make it the first choice of users and industries for deploying their IT architecture, and for service provisioning and procurement. However, the benefits of the cloud are not free. Cloud computing raises several concerns due to lack of trust and transparency, while customers need guarantees on cloud services nonfunctional properties (such as security, privacy).[1] Unfortunately, many cloud service providers still fail to provide complete transparency regarding the security and privacy compliance measures they have in place to protect their customers' sensitive information and intellectual property. Of course, cloud providers have deployed security controls to prevent attacks and unauthorized activities. Information about such controls' operation and their effectiveness is rarely made available to customers. Therefore, cloud users do not have access to all security intelligence and log information on potential threat vectors, which impairs their ability to estimate risks.[2]

Service providers are especially reluctant to take full responsibility over security and privacy breaches of their services, while cloud users increasingly ask for trustworthy evidence on the status of service security.

In the last few years, as the cloud has reached *functional maturity*, the research community has focused on protecting security and privacy of its users from different angles.[3] Several techniques have been proposed, including solutions to protect confidentiality and integrity of data and applications, maintain isolation among cloud tenants, and identify malicious and fraudulent activities. Despite the good intention behind such efforts, they have led to the proliferation of heterogeneous approaches and have increased user confusion. Indeed, the perception that the operation of different security and privacy controls may be affected in a different way by virtualization and cloud-specific factors is a main obstacle to cloud adoption in emerging domains, such as big data domains.[4]

It clearly emerges the need of increasing the cloud transparency by providing reliable evidence on the behavior of the cloud and its services. Traditional security verification techniques composed of static analysis approaches are not enough and must be extended to support continuous evidence collection from in-production cloud services at runtime. A proper evidence collection process provides the basis of a trustworthy cloud. On the contrary, the absence of nontransparent evidence hinders users' trust on the real cloud behavior and prevents users from taking full advantage of the cloud functionalities.

Assurance vs. Security

The definition of a transparent and trustworthy cloud is fundamental to widen its adoption in critical scenarios managing sensitive data and applications. Recently, security solutions have been considered as the key enablers towards increasing trust in the cloud. However, their roles fall far short of expectation due to the fact that involvement of customers and service providers in security management is reduced by lack of trustworthy evidence on their behavior. It has then become a mantra to consider the cloud as an "environment where magic happens", while at the same time an "untrusted environment".

To fill in this gap, security assurance techniques have been introduced to increase transparency in the working of cloud services and corresponding security techniques. This increases the confidence of customers that their data and the applications they use/own are treated according to their wishes.[3,5,6,7] Often security and security assurance are considered similar concepts and are used interchangeably. Though related, they are profoundly different as discussed in the following two definitions.

- *Cloud security* can be defined as a way to actively protect assets (data and applications) by internal and external threats and attacks, to provide an environment where customers interact in a secure way.
- *Cloud security assurance* can be defined as the way to gain justifiable confidence that infrastructure and/or applications will consistently demonstrate one or more security properties, and operationally behave as expected despite failures and attacks.[3,8]

The notion of security includes all those mechanisms (for example, encryption, access control, trusted computing) necessary to provide a given security property. The notion of assurance instead compliments the one of security by providing methodologies for collecting and analyzing evidence that can prove or refute security properties. Generally, there are different assurance techniques that can be readily deployed in the cloud such as audit, certification, and compliance techniques. Though intrinsically different, they share a common aspect that supports the implementation of a trust model based on each of them: they need to collect and validate reliable evidence to prove a given security property.

The research on assurance is therefore moving from the definition of new assurance techniques to the definition of reliable approaches to evidence collection at the basis of a trustworthy cloud. Reliable evidence collection is key for increasing the perceived trust of the users in the cloud. To be effective, evidence collection processes must follow some basic requirements, which are summarized below.

- R1. *Trusted authority.* An evidence collection process SHOULD be controlled and managed by a recognized, independent, trusted authority.
- R2. *Standardized evidence collection.* The evidence SHOULD be collected in a standardized way (independently by the considered system) and according to different collection mechanisms (for example testing, monitoring, trusted computing). Heterogeneous data schemas and formats should be managed according to common machine-readable documents providing guidance for agents implementing the collection process.
- R3. *Verifiable evidence.* Collected evidence MUST be verifiable and its integrity guaranteed according to standardized techniques.
- R4. *Integrity of collecting infrastructure.* The integrity of the infrastructure used for evidence collection MUST be protected and continuously verified.
- R5. *Continuous and real time.* To accommodate the dynamic and adaptive nature of the cloud, cloud service behavior MUST be continuously checked, through a continuous and real-time process for evidence collection.
- R6. *Multilayer and multitarget.* Evidence collection SHOULD cope with multilayer and multitarget cloud scenarios, where evidence is collected at all layers of the cloud stack.

These requirements represent the basis to implement a trustworthy assurance infrastructure for a cloud environment.

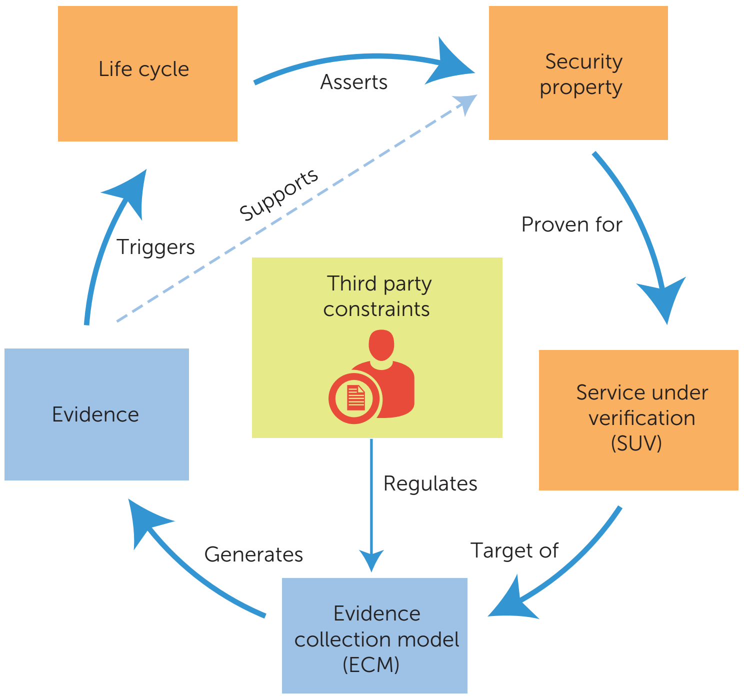


FIGURE 1. Conceptual representation of the methodology. ECM is the Evidence Collection Model and SUV is the Service Under Verification.

Evidence Collection Makes Cloud Transparent

An evidence collection process for the cloud works at all layers of the cloud stack and executes on the target cloud-based service. It implements a continuous and real-time process where heterogeneous agents (that is, probes) inspect the cloud behavior and produce evidence on a service status. To support the need of standardizing evidence collection, our process is driven by an Evidence Collection Model (ECM), a machine-readable document describing i) how evidence should be collected from the Service Under Verification (SUV) and ii) how the evidence should be aggregated to support a given property. ECM defines all the activities required for deploying and executing the evidence collection process. The probes implement such activities and execute them on the SUV for gathering the required evidence. To this aim, the SUV owner provides *cloud hooks*, that is, a description of cloud configurations (for example, access credentials, Infrastructure as a Service (IaaS) configurations) and service end-points available for evidence inspection. Figure 1 summarizes our methodology. A property is verified for a specific SUV, according to constraints posed by a trusted authority. Evidence is continuously collected by exercising the SUV according to ECM. It is then given as input to the corresponding security assurance technique that eventually triggers changes in the property verification results (e.g., the status of a certificate, the outcome of an audit process) following a predefined life cycle. The life cycle defines the states of the assurance process and how they can be reached depending on the results of the evidence collection and aggregation. (An example of life cycle is presented in the section "Case Study: Certification in the Cloud".)

Evidence Types

The evidence can be of three different types, *test-based evidence*, *monitoring-based evidence*, and *Trusted Computing-based evidence*, as follows.

Test-based evidence. Test-based evidence is a major source for software assurance and, recently, has been applied to assurance scenarios focusing on service and cloud-based systems.[5,9-11] It is suitable for testing of software/services in both lab and in-production environments. In a test-based evidence collection model, i) the evidence is represented as the set of test cases to be executed on the SUV, ii) the probes are scripts injecting test cases via the SUV's hooks and collecting corresponding results, iii) the hooks are a set of configurations and available end-points permitting probe execution. The definition of mandatory test cases is driven by constraints defined by the trusted authority in the ECM, which specify the least set of testing activities to be done on the SUV for a successful verification process supporting a given property.

Monitoring-based evidence. Monitoring-based evidence has been largely used in Intrusion Detection System/Intrusion Prevention System for evaluating complex system behavior. Recently, it has been proposed also for service and cloud verification.[12] Monitoring-based evidence is suitable for dynamic collection of evidence and/or for security properties that cannot be tested online (such as, distributed denial-of-service robustness or robustness to penetration attacks). In a monitoring-based evidence collection model: i) the evidence is the set of monitored events, ii) probes correspond to event captors that may intercept events sent to or by the system that is being monitored, or internal events of this system which are being audited through some event capturing mechanisms (e.g., audit facilities of web servers, databases), iii) the hooks are implemented as an event bus collecting all the required events from the SUV. ECM for monitoring defines the set of expected events, and the (temporal) patterns that they should comply with for the property under verification to hold.

Trusted Computing-based evidence. Trusted Computing-based evidence has been only recently applied to cloud assurance, with the aim of guaranteeing that assurance results actually refer to targeted services in the cloud.[13] Trusted Computing (TC) technology is a hardware-based approach for ensuring the integrity of IT systems. A secure hardware device called Trusted Platform Module is used to perform security measurements of a running system and to produce dynamic evidence of its state (https://www.trustedcomputinggroup.org/). Such evidence is included in assurance techniques for the cloud by i) specifying information to allow clients to verify the integrity and state of the running service, and ii) modifying the semantics so that only proofs for which the former verification succeeds are considered valid. TC-based evidence can be used to verify properties of either platforms or services running in the cloud, or to produce dynamic evidence based on the actual behavior of a system. TC-based evidence can also be used to verify the integrity of components used for the collection of testing- and monitoring-based evidence.

Evidence Collection Processes

To address the requirements introduced in the section "Assurance vs. Security", an evidence collection process based on the types of evidence in the section "Evidence Types" must support *hybrid evidence collection* and *incremental verification*. To this end, we analyze two evidence collection models, namely, *Hybrid ECM* and *Incremental ECM*.

Hybrid evidence collection model. It is used when SUV includes mechanisms requiring different types of evidence to be effectively verified.[14] Hybrid ECM can be enacted using two different models.

- *Full Hybrid model.* A single Hybrid ECM is specified describing an evidence collection process based on probes gathering different types of evidence. Probes run in parallel or in a sequence. The trusted authority then specifies how to correlate the multitype evidence according to the life cycle in hybrid ECM.
- *Hybrid by Combination model.* Multiple ECMs are specified to collect different types of evidence and drive different collection processes. One ECM is designated as the assurance process owner (i.e., Master ECM), while the others are acting as auxiliary processes used for providing evidence results when required (for example, Slave ECMs). The master gathers the results of the evidence collection process of the slaves, and integrates them within its own process on the basis of its life cycle.

The *Fully Hybrid* model requires a custom ECM designed for supporting hybrid evidence collection, while *Hybrid by Combination* requires independent auxiliary evidence collection processes.

As an example, a Hybrid by Combination model can collect test-based evidence of SUV behavior and TC-based evidence on the integrity of the testing probes collecting such evidence.

Incremental evidence collection model. It improves the *resilience* of assurance processes and corresponding results to cloud events. It minimizes the amount of activities to be done, when events at cloud, service, and assurance levels affecting existing processes are observed.[5] Incremental ECM extends the life cycle to manage scenarios where assurance results are invalidated. Its goal is to reduce as much as possible the window of time in which the assurance results are invalid, while minimizing the amount of additional activities needed to bring back the assurance results to a valid state.

Assurance results might become inconsistent due to events that temporarily or permanently invalidate part of the collected evidence. Such events can be classified as follows.

- *Service-level events*, where evidence is insufficient for a given period of time due to SUV evolution (for example, service versioning) and/or contextual changes (for example, service failures). Focusing on SUV evolution, let us consider property data-leakage-prevention based on 3DES encryption.[15] If a new version of the service is released without support for 3DES encryption, the property is no longer supported by the collected evidence. Otherwise, if contextual changes cause a failure in the mechanism implementing 3DES encryption, evidence becomes insufficient and the assurance results are invalidated. As soon as the mechanism returns available, the assurance results become again valid.
- *Cloud-level events*, where evidence is insufficient due to specific cloud events (for example, service migration). For instance, let us consider property data-leakage-prevention based on an encrypted storage.[15] If the service is temporarily moved to a different infrastructure providing a storage service with no support for encryption, the property is no longer proven by the collected evidence.

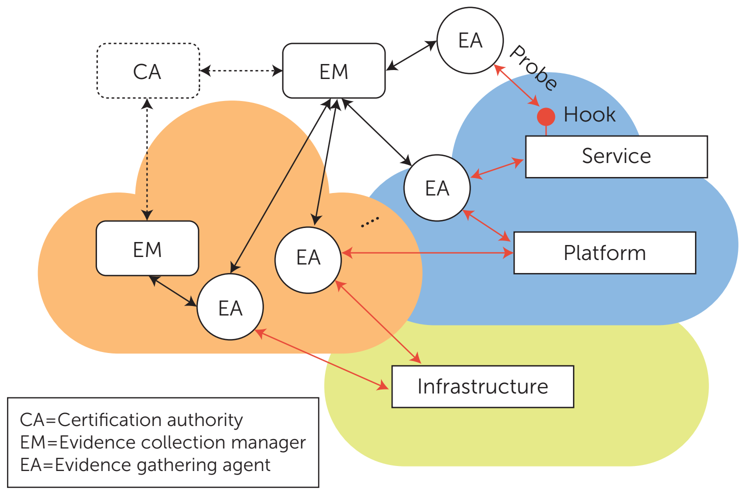

CA
EM
EA
Probe
Hook
Service
EA
EM
EA
...
Platform
EA
Infrastructure
CA=Certification authority
EM=Evidence collection manager
EA=Evidence gathering agent

FIGURE 2. Evidence collection architecture. EM is the Evidence Collection Manager, EA is the Evidence Gathering Agent, and CA is the Certification Authority

- *Assurance-level events*, where all or a part of the evidence becomes invalid/contradictory, because new conditions and constraints for the validity of the property for a specific SUV are introduced by the trusted authority. For instance, a new bug/vulnerability has been discovered for a specific mechanism (for example, Heartbleed openssl bug—http://heartbleed.com/) and needs to be verified on the SUV.

The above classes of events trigger an incremental ECM requiring one or more of the following activities.

- *Adaptation*. No evidence collection activities are requested. Incremental ECM only adapts assurance results.
- *Evidence collection reexecution*. Observed events require to reexecute (part or all) evidence collection activities. It is usually due to service- and cloud-level events (for example, different configurations or context), and requires the reexecution of probes on (different) hooks.
- *New evidence collection activities*. Observed events require executing new assurance activities. It is usually due to service-level events (e.g., a new version of the service), and requires the definition of new probes and hooks.
- *Reevaluation from scratch*. Observed events require a complete re-evaluation of the SUV. It can be due either to service-level events (such as, SUV evolution) or assurance-level events.

Evidence Collection Infrastructure

The European FP7 Certification infrastrUcture for MUlti-Layer cloUd Services (CUMULUS) project (http://www.cumulus-project.eu/), aiming to define models, processes, and tools for cloud certification, has implemented an assurance infrastructure based on certification. Here, we concentrate on the portion of the CUMULUS infrastructure managing the evidence collection processes in the sections “Evidence Types” and “Evidence Collection Processes”. It is implemented as a master-slave distributed architecture (see Figure 2), composed of two main components: *Evidence Collection Manager* (EM) and *Evidence Gathering Agent* (EA). The *Evidence Collection Manager*—the master of the evidence collection process—provides the interface used by the trusted authority (Certification Authority CA in CUMULUS) for defining and managing all evidence collection activities (dotted lines in Figure 2). It implements the following features: i) *fully automatic* set up of collection activities, including slave initialization, ii) aggregation of the results of the evidence collection activities, iii) life cycle evaluation, and iv) support for evidence collection processes. *Evidence Gathering Agent*—the slave of the evidence collection process—is responsible for SUV evaluation. It deploys and executes probes, collects evaluation results, and returns the results of the evidence collection activities to the *Evidence Collection Manager*. Agents require confidential information (hooks) of the cloud/service providers (e.g., internal configurations, authorization credentials) for connecting the probes to the target SUV, and can be deployed inside (internal cloud evaluation) or outside (interface-level evaluation) the target cloud, depending on the level of intrusiveness that might be tolerated by the cloud/service providers themselves.

Both deployment strategies build on a chain of trust where the trusted authority delegates (part of) the assurance and evidence collection processes. In a cloud environment, in fact, different from traditional assurance where the trust model is based on a trusted authority responsible for the whole assurance process, the trust is shared among the trusted authority and its accredited lab executing ECM by means of the Evidence Collection Architecture (ECA). More in detail, the chain of trust of a cloud assurance process relies on the trust the client has on i) the trusted authority (offline actor) signing the assurance results, responsible for the definition of *authority constraints*, and in charge of checking that these constraints are addressed by ECM, ii) the *accredited lab* (online actor) delegated by the trusted authority and responsible for the correct execution of

ECM and corresponding life cycle, iii) the integrity of the ECA, and its correct configuration and deployment by the accredited lab on the basis of ECM.

The evidence collection infrastructure accomplishes all requirements in the section "Assurance vs. Security". It is driven by a trusted authority requirements [R1], and provides a standardized evidence collection model based on machine-readable documents and common agents [R2]. It provides verifiable and reliable evidence through the definition of standardized hooks interfacing with the Service Under Verification (SUV) [R3]. It is based on trusted computing platforms to verify the integrity of evidence collection agents [R4]. It implements a continuous and real-time collection at all layers of the cloud stack [R5], [R6]. The infrastructure supports some additional requirements. First, it supports a distributed deployment over the cloud. Then, it provides auto-configuration of evidence collection agents, implementing a semi-automatic assurance process, and adapts to changing conditions in the cloud. Finally, it can be trusted to manage the assurance process as expected and in a correct way. An open source implementation of the CUMULUS evidence collection infrastructure, which supports a test-based evidence collection process, is available at http://goo.gl/98vuYr, where the code of the manager and the agent, as well as some examples of probes, are referenced.

Case Study: Certification in the Cloud

We present a case study on cloud certification, where the CUMULUS infrastructure is used to demonstrate the utility of the evidence collection processes in the section "Evidence Collection Makes Cloud Transparent". A certification process is driven by a pair composed of a non-functional property to be certified and the target of certification (ToC), that is, a specific instantiation of a generic SUV. Nonfunctional properties could span different domains of software and services including security, privacy, reliability, and performance, to name but a few. Examples of nonfunctional properties can be found in "D2.1: Security-Aware SLA Specification Language and Cloud Security Dependency Model" which defines among others security properties like *data leakage prevention* and *data exchange confidentiality*, and privacy properties like *timely access* and *deletion ability*.[15] The goal of a certification process is to guarantee a target level of assurance, by (semi) automatically collecting evidence supporting the fact that the ToC actually holds the desired property. Often, some constraints are given on the amount and composition of evidence that must be collected to guarantee an adequate level of assurance, and on

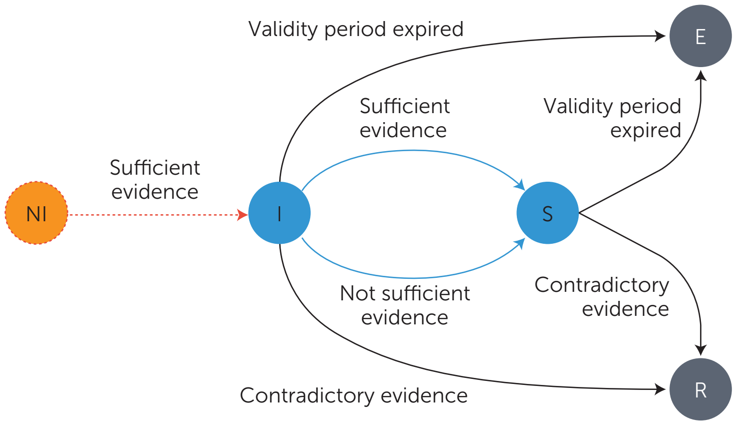


FIGURE 3. The Certificate Life Cycle (CLC). The CLC describes the evolution of the certificate status from when it is issued to when it is revoked, suspended, expired or renewed.

the tools used to collect them in-production or in a lab environment. If evidence meets certification constraints, a certificate is awarded to the ToC.

The case study considers a real-world industrial scenario consisting of an application for the remote management of public lighting (lighting application for short). The application provides real-time information system performance and infrastructure status. It is deployed on OpenStack, an open source IaaS solution that permits to manage and monitor infrastructure resources, stores application's credentials in a MySQL database, and provides a storage for lighting's information. We present two certification processes. The first is based on a hybrid evidence collection process and is aimed at certifying property *data alteration prevention*; the second is based on an incremental evidence collection process and is aimed at certifying property *confidentiality of exchanged data*. Evidence is collected according to *Certificate Life Cycle* (CLC), which describes the evolution of the certificate status from its issuing (i.e., evidence is sufficient to prove a property for the ToC) to its revocation, suspension, expiration, or renewing. CLC is modeled as an automaton (see Figure 3), where each transition between two certificate states is regulated by specific conditions on the collected evidence. Specifically, the transition between states *NI* and *I*, denoted *NI*→*I* (dashed arrow in Figure 3) is regulated by issuing conditions. When issuing conditions are verified, a certificate is issued. Subsequently, if collected evidence becomes insufficient to prove the property, then the certificate is suspended and moves to state *S* (*I*→*S*). The certificate returns to state *I* (*S*→*I*), if the evidence returns sufficient (*certificate renew*). State *R* (*certificate revocation*) is reached

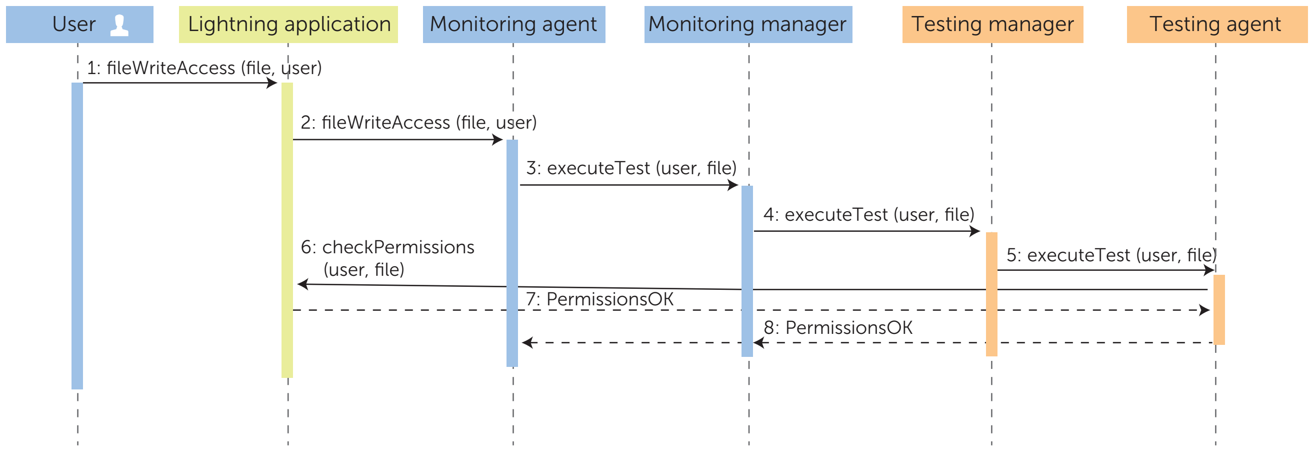

User
Lightning application
Monitoring agent
Monitoring manager
Testing manager
Testing agent
1: fileWriteAccess (file, user)
2: fileWriteAccess (file, user)
3: executeTest (user, file)
4: executeTest (user, file)
5: executeTest (user, file)
6: checkPermissions (user, file)
7: PermissionsOK
8: PermissionsOK

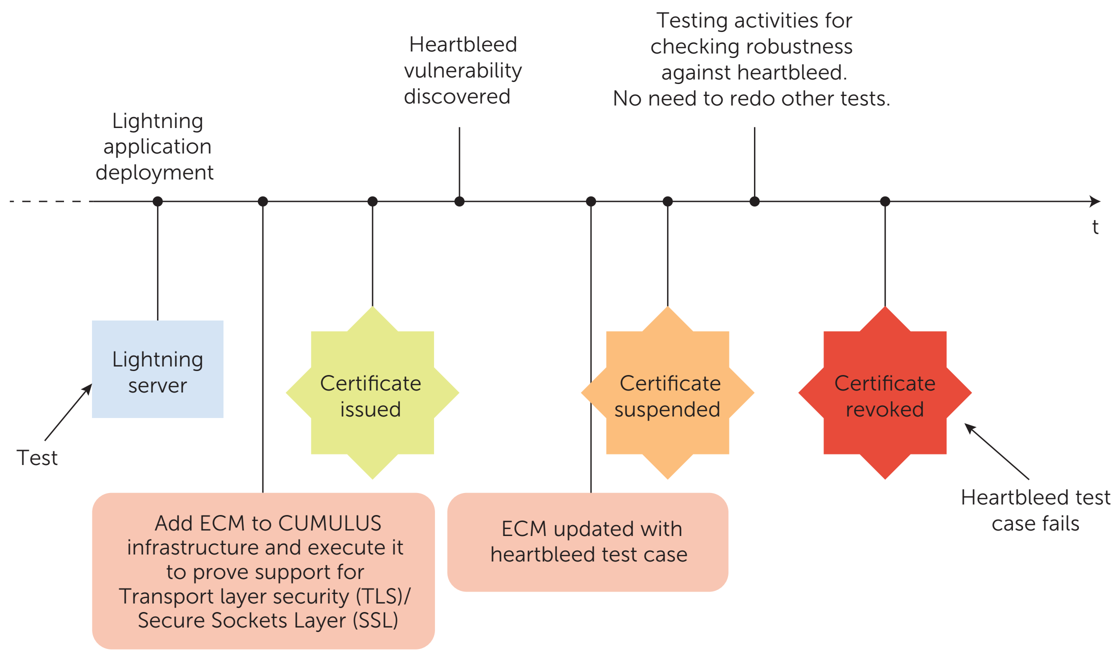
FIGURE 4. Certification of property data alteration prevention.

when the collected evidence is contradictory and does not prove the property (e.g., a successful attack is observed). State *E* (*certificate expiration*) is reached when the certificate validity expires. We note that states *R* and *E* are sink states and trigger recertification from scratch.

Property data alteration prevention. Figure 4 presents a certification process based on hybrid evidence for certifying lighting application against property data alteration prevention. The property requires that only authorized users can update a file. To certify it, the infrastructure uses a hybrid by combination model, where a master ECM based on monitoring drives a slave ECM based on testing. As depicted in Figure 4, after a user's request for updating a file (fileWriteAccess(file, user)) is received by the application (step 1), it is forwarded to the CUMULUS monitoring agent (step 2). The monitoring agent, which is not able to verify that the user has the rights to execute the request, asks the test manager via the monitoring manager to execute a test case (executeTest(user, file)). The test case needs to ensure that the user had the appropriate authorization rights (step 3). The test manager instructs the test agent to execute function checkPermission(user, file) (step 4) and the latter returns the result to the monitor agent (step 5). We note that testing/monitoring managers are the evidence collection managers and testing/monitoring agents the evidence gathering agents of the evidence collection architecture in Figure 2. The condition for certificate issuing in the lifecycle (($NI \rightarrow I$) in Figure 3) requires the user to have access rights to the file. If this is confirmed by the testing agent the certificate for property data alteration prevention is awarded.

Property confidentiality of exchanged data. Figure 5 presents a certification process based on incremental evidence for certifying lighting application against property *confidentiality of exchanged data*. The property requires the application to offer a confidential network channel for data exchanged with external parties. To certify it, the infrastructure uses a test-based incremental evidence collection model, checking the robustness of the HTTPS communication channel between the application and external parties. The condition for certificate issuing in the lifecycle (($NI \rightarrow I$) in Figure 3) requires lighting application to support Transport Layer Security (TLS)/Secure Sockets Layer (SSL) with strong ciphers. We use the customized Nmap script engine to collect information about TLS/SSL for the application communication channel, including the certificate time validity, the TLS/SSL version, and the supported ciphers. Upon a successful verification is performed and a certificate awarded for the application, we start a continuous certification process. For instance, let us assume that a new bug is discovered on the HTTPS implementation used by the application (e.g., Heartbleed Bug of openSSL). The certificate is suspended according to the lifecycle ($I \rightarrow S$) and new evidence is collected to verify robustness of the HTTPS against the bug. If the verification succeeds the certificate comes back to the issued state ($S \rightarrow I$), otherwise it is revoked ($S \rightarrow R$) according to the lifecycle. Figure 5 shows the whole verification chain leading to certificate revocation.

Cloud assurance can make the difference in the next evolution of cloud computing. Increasing the assurance in the cloud in fact could attract new



Lightning application deployment
Heartbleed vulnerability discovered
Testing activities for checking robustness against heartbleed. No need to redo other tests.
t
Lightning server
Test
Certificate issued
Certificate suspended
Certificate revoked
Heartbleed test case fails
Add ECM to CUMULUS infrastructure and execute it to prove support for Transport layer security (TLS)/ Secure Sockets Layer (SSL)
ECM updated with heartbleed test case

FIGURE 5. Certification of property confidentiality of exchanged data. TLS is the Transport Layer Security, SSL is the Secure Sockets Layer, and CUMULUS is the European FP7 Certification infrastrUcture for MUlti-Layer cloUd Services.

businesses, which operate in security-critical environments, and accelerate their movement to the cloud.

However, much work still needs to be done. Firstly, costs and risks must be carefully evaluated when an assurance architecture is integrated within the cloud infrastructure. Secondly, impact on cloud performance must be minimized in order to maintain high quality of service for the cloud. Finally, a proper assurance approach should i) maximize evidence reuse by composition, ii) maximize service quality by discovering and selecting the best services on the basis of the assurance results, iii) support an enhanced architecture able to share evidence results at runtime and on demand, and iv) take advantages by emerging big data analytics techniques.

Given the dynamics of the cloud and the need of managing evolving evidence, a sound trust model must be defined, departing from the assumption of having a trusted authority available anytime anywhere, as well as an approach to the dynamic generation of assurance results.

References

1. S. Pearson, "Toward Accountability in the Cloud," *IEEE Internet Computing*, vol. 15, no. 4, 2011, pp. 64–69.
2. V. Bellandi, S. Cimato, E. Damiani, G. Gianini, and A. Zilli, "Towards Economics-Aware Risk Assessment on the Cloud," *IEEE Security & Privacy*, vol. 13, no. 6, 2015, pp. 30–37.
3. C.A. Ardagna, R. Asal, E. Damiani, and Q.H. Vu, "From Security to Assurance in the Cloud: A Survey," *ACM Comp. Surveys*, vol. 48, no. 1, Aug. 2015, pp. 2:1–2:50.
4. D. Ruta, L. Cen, and E. Damiani, "Fast Summarization and Anonymization of Multivariate Big Time Series," *Proc. PSBD 2015*, Oct. 2015.
5. M. Anisetti, C.A. Ardagna, and E. Damiani, "A Certification-Based Trust Model for Autonomic Cloud Computing Systems," *Proc. ICCAC 2014*, Sept. 2014.
6. A. Sunyaev and S. Schneider, "Cloud Services Certification," *Comm. ACM*, vol. 56, no. 2, Feb. 2013, pp. 33–36.
7. Cloud Security Alliance (CSA), "CSA Security, Trust & Assurance Registry (STAR)," https://cloudsecurityalliance.org/star/.
8. T. Winograd, H.L. McKinley, L. Oh, M. Colon, T. McGibbon, E. Fedchak, and R. Vienneau, "Software Security Assurance: A State-of-the Art Report (SOAR)," Information Assurance Technology Analysis Center, 2007.
9. D.S. Herrmann, *Using the Common Criteria for IT Security Evaluation*, Auerbach Publications, 2002.
10. M. Anisetti, C.A. Ardagna, E. Damiani, and F.

Saonara, "A Test-Based Security Certification Scheme for Web Services," *ACM Trans. Web*, vol. 7, no. 2, May 2013, pp. 1–41.
11. D. Kourtesis, E. Ramollari, D. Dranidis, and I. Paraskakis, "Increased Reliability in SOA Environments Through Registry-Based Conformance Testing of Web Services," *Production Planning & Control*, vol. 21, no. 2, June 2010, pp. 130–144.
12. M. Krotsiani, G. Spanoudakis, and K. Mahbub , "Incremental Certification of Cloud Services," *Proc. SECURWARE 2013*, Aug. 2013.
13. A. Muñoz and A. Maña, "Software and Hardware Certification Techniques in a Combined Certification Model," *Proc. SECRYPT 2014*, Aug. 2014.
14. S. Katopodis and G. Spanoudakis , "Towards Hybrid Cloud Service Certification Models," *Proc. SCC 2014*, June–July 2014.
15. CUMULUS Consortium, "D2.1: Security-Aware SLA Specification Language and Cloud Security Dependency Model," http://cumulus-project.eu/index.php/public-deliverables.

MARCO ANISETTI *is an assistant professor at the Università degli Studi di Milano. He received the PhD degree in computer science from the Università degli Studi di Milano in 2009. His research interests are in the area of computational intelligence and its application to the design of complex systems and services. Recently, he has been investigating the adoption of computational intelligence techniques in the area of security mechanisms for distributed systems, with particular consideration of Cloud and SOA security and software/service certification. He is the recipient of the GIRPR 2010 Award for the Best PhD thesis in the area of pattern recognition. Contact him at marco.anisetti@unimi.it*

CLAUDIO A. ARDAGNA *is an associate professor at the Dipartimento di Informatica, Università degli Studi di Milano, Italy. His research interests are in the area of cloud security and certification. He is the recipient of the ERCIM STM WG 2009 Award for the Best PhD Thesis on Security and Trust Management. He coauthored the Springer book "Open Source Systems Security Certification". He has been a visiting researcher at George Mason University (2008–2010) and EBTIC-Khalifa University (2014). The URL for his web page is http://www.di.unimi.it/ardagna. Contact him at claudio.ardagna@unimi.it.*

ERNESTO DAMIANI *is a full professor at the Università degli Studi di Milano and the director of the Università degli Studi di Milano's PhD program in computer science. He has held visiting positions at a number of international institutions, including George Mason University in Virginia, LaTrobe University in Melbourne, Australia, and the University of Technology in Sydney, Australia. He has also done extensive research on advanced network infrastructure and protocols, taking part in the design and deployment of secure high performance networking environments, both as chief scientist and in management positions. His areas of interest include Web services security, processing of semi and unstructured information (e.g., XML), and semantics aware content engineering for multimedia. Also, he is interested in models and platforms supporting open source development. He has served and is serving in all capacities on many congress, conference, and workshop committees. He is a senior member of the IEEE and ACM distinguished scientist. The URL for his web page is http://www.di.unimi.it/damiani. Contact him at ernesto.damiani@unimi.it.*

ANTONIO MAÑA *received his PhD degree in Computer Engineering from the University of Malaga, where he is currently associate professor of software engineering in the Computer Science Department. His current research activities include security and software engineering, information and network security, ubiquitous computing and ambient intelligence, application of smart cards to digital content commerce, software protection, DRM, and mobile applications. The URL for his web page is http://www.lcc.uma.es/~amg/eng/.*

GEORGE SPANOUDAKIS *is a professor of computer science at City University London. He is also a member of the Council of the University of Piraeus in Greece, and has held visiting positions in several other universities in Europe. His research focuses on cloud computing and software systems security. In these areas, he has published extensively and attracted significant research funding from the EU, national research councils, and directly by industry. Currently, he is the technical coordinator of the F7 EU project CUMULUS (2012–15) that focuses on security certification of cloud services. Professor Spanoudakis has served in the program and organization committees of several international conferences and workshops, and the editorial boards of several journals. Beyond research, he has been providing consultancy advice to private companies, universities, public funding, and standardization bodies in the UK and overseas. The URL for his web page is http://www.city.ac.uk/people/academics/george-spanoudakis. Contact him at g.e.spanoudakis@city.ac.uk.*