Reducing implementation efforts in continuous auditing certification via an Audit API

1st Dorian Knoblauch
Fraunhofer FOKUS
Berlin, Germany
dorian.knoblauch@fokus.fraunhofer.de

2nd Christian Banse
Fraunhofer AISEC
Garching b. München, Germany
christian.banse@aisec.fraunhofer.de

***Abstract*—Continuous auditing reduces the frequency in which compliance is verified. This results in more trustworthiness for the cloud service and therefore lowers the barrier of adopting cloud for customers in high-risk sectors such as banking. However, implementing continuous auditing as of today is a tedious task and not standardized, which leaves the service providers implementing the whole audit process and the technical infrastructure. We are proposing a solution for this problem by defining a standardized way of establishing the continuous auditing process for an IT infrastructure as well as providing the necessary tools as a reference implementation. In this paper we present how complexity in setting up the technical requirements for continuous auditing can be highly reduced by providing an easy to implement Audit API and continuous auditing methodology.**

***Index Terms*—Cloud, auditing, continuous, api**

I. Introduction

National and international security standards, such as the ISO 27001 family[1] comprise a set of controls defining minimum security requirements for IT services. These are often high level and have to be individually operationalized for each implementing organization. The implementation of said standards as well as the assessment if the implementation is correct and effective is a potentially complex and cost-intensive tasks. External audits are the usual way to get a third party approval on the performed security efforts. The external auditor has to substantiate whether an operationalization of a high-level requirement was performed, i.e. the auditee must prove that certain controls are in place in an information security management system (ISMS) [5]. Additionally, an auditor determines if the execution is compliant with the current standard. The high-level approach is state-of-the-art in audit-based "point-in-time"-certifications, which usually have a validity of several years. It is, however, a huge limiting factor when thriving for a more continuous or even automated assessment of the compliance status and therefore a certification that is based on a more up to date assessment.

The current security certification landscape consists of certification schemes that mostly require annual or bi-annual audits to obtain or prolong a certification. In certain markets and areas, this does not reflect the rapid technological change or the high demanding regulatory requirements. Enforce a higher audit frequency will improve this situation by increasing the level of trust but will introduce more costs due to more audits that have to be performed. The challenge is to decrease the complexity of these assessment and therefore the audit process. A decreased complexity will also allow automating a significant part of the assessment. First steps towards these goals have also been proposed by the Cloud Security Alliance, which has announced STAR Continuous in beginning of 2019. Specifically the STAR Continuous Self Assessment, which provides a way for cloud service provider to regularly perform a self assessment based on the Cloud Control Matrix (CCM) [4] catalog of security controls. Furthermore, initiatives such as the European H2020 project EU-SEC[2] aim to addresses the challenge of reducing the complexity of the assessments and introduce automation in the audit process.

The foundation of our work lays in a methodology for breaking down high-level controls into atomic and easy-to-evaluate attributes as well as an automated and tool-supported human assessment that allows for a much higher audit frequency than traditional "point-in-time"-audits. Additionally, the definition of these attributes and metrics help developers in building tools that will perform the majority of the assessment. Our developed architecture facilitates these tools as well as the parts of the process that still have to be implemented by the service provider. We have defined precise interfaces which define the service providers obligation and the ones performed by auditing tools.

II. Related Work

Research into the certification of cloud services and especially its automation has been conducted for several years. In 2013, Cimato et al. introduced the concept and initial design proposal for cloud service certification [3]. Following up on this work, Anisetti et al. [2] [1] proposed a test-based scheme for the automation of certification. However, the introduction of automation results in a gap between what *has* to be checked and what *can* be checked.

Stephanow and Kunz [6] addressed this gap by redesigning the traditional certification process. This differs from our approach since we are not changing the certification process; in fact, we are trying to take traditional certification schemes

[1]https://www.iso.org/isoiec-27001-information-security.html

[2]https://www.sec-cert.eu

as a foundation. Stephanow and Kunz are adding suitable tooling to support continuous certification of cloud services, for example, further detailed in their service-specific work about security testing of web applications [8] as well as continuously assessing the location of IaaS components [9]. While our approach requires the same kind of tooling, we are trying to follow a more universal solution, instead of focusing on individual service and delivery models.

Lins, Schneider and, Sunyaev [7] are proposing a conceptual CA architecture that is partly similar to ours. We share similar views on the relation between cloud service and auditor entity, but we have extended this approach by defining a precise API for exchanging evidence and have also taken the certification authority into account. Our approach is functional and not conceptual.

III. Framework Architecture

Continuous auditing requires a specific suitable architecture that is capable of facilitating both automated and non-automated assessments. The nature of auditing and certification is trust via third party involvement. As different parts of the audit and certification processes are performed by different parties the architect has to reflect that modular aspect. From a high level view the procedures can be separated into a *preparation phase* and the four *execution phases* (see Fig. 1). The architecture for continuous auditing has to facilitate the data gathering and processing as well as the data flow itself. The elements of the preparation phase are the inputs for the execution phases and result in a state of compliance. The objectives, attributes, metrics, frequencies and the scope are utilized in the execution phases to process the infrastructure data and the manual assessments to a compliance statement. The actual execution has to be implemented by the tools that are used for continuous auditing.

A. Preparation Phase

Our continuous auditing approach is independent of the applied security standards and requirements. Which means that each control that is subject to a traditional "point-in-time"-certification and is checked in this context can also be assessed by continuous auditing. Usually, an auditor makes the assessment based on the control, which is present in a running text form. This makes continuous manual assessment harder as it requires constant interpretation of the text and it even makes automated assessment impossible. Our approach solves this problem by breaking down the controls into small easy to asses and even computable attributes. This breakdown model views security controls as a set of objectives, called Service Level Objectives (SLO) and Service Qualitative Objectives (SQO), similarly to what happens when defining Service Level Agreements. Objectives are essentially constraints defined on the basis of security attributes of an information system. To verify that a certain security control is in place a company should verify that the associated objectives are met.

Partly this task is also subject to a traditional point-in-time audit as it includes defining objectives, but in this case, it is focused on also making the assessment of objectives fully or partly automated. Taking a requirement like the need for encryption as an example, which is usually implemented in every aspect of the service where it is needed according to aspects like risk exposure and regulation, we end up with a set of objectives which define precisely what parts of the service require what kind of encryption. To assess the rightful implementation, i.e. if the objectives are fulfilled, an auditor would perform a code review, interview the developer or examine a penetration test report. This is the part where continuous auditing provides improvement in efficiency. Other than reassessing whole set of controls, continuous auditing only demands to do an assessment of atomic aspects of each control. For each aspect a suitable measurement will be defined and executed in a reasonable frequency.

One key element of continuous auditing is the definition of those measurable attributes and objectives that describe a security control, as show in Fig. 2 (Blue):

- Each control framework consists of multiple controls, which are designed to give assurance on the fulfillment of a requirement.
- When preparing for continuous auditing, each one of those controls has to be described via its characterizing objectives, namely SLOs and SQOs.
- Objectives are described as constraints on one or more security or privacy attributes; each attribute makes an aspect of the objective assessable. By assessing all those

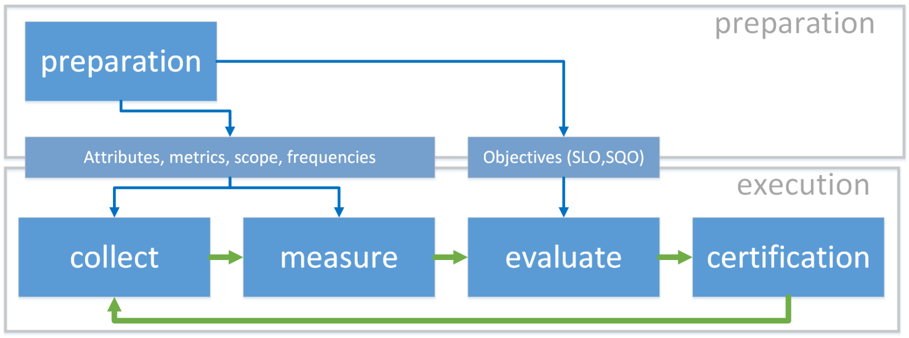

Fig. 1. Model of continuous auditing phases

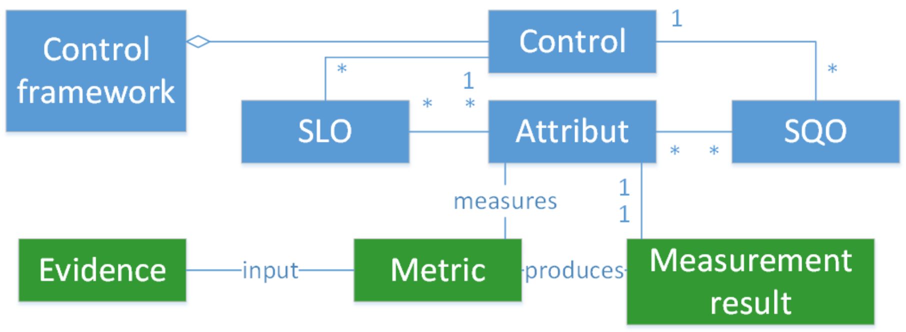

Fig. 2. Conceptual UML model for continuous auditing

attributes, we can provide an evaluation on the achievement of the objective.

It is also important to note that security controls are context specific. Their implementations vary depending on the specificity of the risk appetite and technological environment of the CSP. Moreover, some controls are meant to satisfy policy requirements, others to verify procedures such as incident management procedures and others are meant to verify specific technical implementations. Consequently, the frequency with which each control should be assessed varies greatly. An example for short frequency would be the control for an effective Identity Access Management where constant accesses demand a higher frequency.

A measurement (Green in Fig. 2) provides a qualification or quantification of an attribute. In this context, the measurement process consists of three elements:

- Evidence can be considered as the input in a measurement. Evidence can be as simple as a plain number or as complex as a large unstructured document. The kind of evidence often defines whether it is suitable for automated reasoning on an attribute or if its complexity requires a human interpretation. In an automated environment, the evidence is produced either via monitoring of already produced data or via a specific test. Those tests are often conducted by specific test suites, manually written scripts or enterprise-targeted security monitoring solutions. In the case of evidence that requires human interpretation the number of sources is much broader in a sense that even a screenshot or documentation can be considered as valid evidence.
- Metric is a standard for a measurement. It defines the function that transforms the evidence into the measurement result. By doing so it implicitly gives it a unit and, in most cases, it normalizes the output by returning a ratio or percentage value. Therefore, the metric requires a qualifiable or quantifiable measurable evidence to produce the result in an unambiguous manner.
- Measurement result refers to the output of the metric and does allow to reason on an attribute and ultimately on a control or objective.

Metrics provide knowledge about the characteristics and attributes of an IT infrastructure, through units, rules and the values from the analysis of the evidence. The evidence is processed into a measurement result via a metric.

Coming back to the encryption example, this means that each objective such as *encryption of data in transit* has to be defined by suitable measurements like an automated check if the correct TLS configuration is applied. For an objective that requires to store assets in an encrypted way, possible measurements could be to check if the hard drive where the data is stored is encrypted correctly or even to check if the data itself is encrypted on the application layer.

B. Process of Operationalization

In the preparation phase, the proper operationalization of the selected set of controls takes place. Key actions in this phase are the definition of the scope, the identification of the objectives (SQO, SLO) associated to each control, the determination of the frequencies at which each objective should be checked, the definition of attributes and metrics, as well as the identification of points where the measurements should be taken.

The scheme defines the following procedure for each control of a security framework:

1) According to the comprehensiveness and the coverage of a control a set of matching objectives has to be defined.
   a Objectives that describe a specific quantitative characteristic, where the value follows interval scale or ratio scale have to be defined as Service Level Objectives (SLO).
   b Objectives that describe a specific qualitative characteristic, where the value follows the nominal scale or ordinal scale have to be defined as Service Qualitative Objectives (SQO).
2) Each Objective is described by defining attributes. Those attributes are more specific than an objective and do reflect just one measurable aspect of an objective. They are determinable via either a qualitative or a quantitative aspect.
3) Each attribute has to be assigned to a measurement procedure, which will provide a measurement result that describes the state of the attribute. Metrics have to be

suitable for the infrastructure of the organization as well as the attributes characteristic.

a The evidence is obtained on the infrastructure by performing measurements and stored in the evidence store.
b The measurement is then performed according to the metric.
c The measurement result expresses a qualitative or quantitative assessment of an attribute.

Following the encryption example, the breakdown might look like this:

- Requirement: *Data has to be encrypted*
  - SQO: Data is encrypted at rest
    - Attribute 1: Encryption algorithm used for data
    - Attribute 2: Encryption used for hard drive
  - SQO: Data is encrypted in transit
    - Attribute 1: Encryption used for send data
    - Attribute 2: Configuration of TLS

More complex requirements often have to be described by more objectives and attributes. It is common that those attributes originating from complex objectives have to be assessed manually by a human. For instance, a requirement that demands the establishment and the maintenance of certain policies and procedures inside an organization might have one attribute per policy or procedure. A human auditor then determines if a certain policy or a procedure is rightfully established and maintained and by that, the attribute gets assessed. Our research has revealed that as of today we're able to asses just about a firth of the CCM completely automatically.

IV. Implementation

A. Collection and Measurement Phase with an Audit API

The collection phase is the first step of the *execution phase* of an ongoing continuous auditing process. It facilitates the collection of data for the automated assessment as well as for the non-automated assessment. Collection of data is driven by the metric that has been chosen to provide input about an attribute. In the context of continuous auditing, data is referred to as evidence. Depending on the type of assessment the tools used could be various. Automated assessment is mostly driven by monitoring tools like log analytics, network statistics, and monitoring, process statistics or resource utilization. While non-automated assessment requires humans to verify the existence and the effectiveness of certain processes and to read documents or examine records. In both cases, the frequency at which the evidence is collected is influenced by the objective and ultimately by the certification target. The evidence collected in this phase can originate from various sources and therefore are in formats or representations which make proper processing difficult, for instance, due to the unadjusted scale of two values or a log message that needs further processing.

The measurement phase describes the process that transforms the collected raw data into a usable measurement result. In the context of continuous auditing, a measurement result quantifies or qualifies an attribute. Attributes require the measurement result to be in a particular format or representation. This way of conducting the measurement and interpreting the raw data is usually defined in a metric. The measurement phase is about the actual execution of the operations that qualify or quantify an attribute. The result can be considered an evidence like the raw data itself.

We have defined an Audit API that normalizes all those different sources of evidence and gives the auditing entity the information needed for the assessment. As security efforts are mainly driven by securing the assets the Audit API is also asset-driven. Security implementations are driven by factors like chosen technologies, requirements and others. This means that there is no archetype for a secure implementation. The common element among most services is a multi-layered architecture, i.e. a web application running on top of a platform which then runs on top of an infrastructure. The Audit API addresses this by allowing multiple *scopes* for one service, referring to a single layer in the overall architecture. In the encryption example, this means that evidence on encryption is provided for the infrastructure level as well as the application level. The specification of the Audit API was made available as open source within the EU-SEC project[3]. It is reflecting a starting point for discussion and we welcome contributions to the specification from other researchers.

B. Evaluation and Certification Phase

In the evaluation phase, the compliance status with the certification goal is determined by evaluating the controls. Technically a control is a set of objectives which are described as compilations of attributes, which then, in turn, are evaluated by a measurement. In our case, evidences are retrieved via the Audit API and delivered to the auditing entity.

In our reference implementation, we have extended the Clouditor[4] tool to accept and process evidences in the format of our defined Audit API. Additionally, to test the implementation of Clouditor against a realistic service, we had two different cloud services. First, we launched a Cloud-based document sharing service on Amazon Web Services (AWS), based on a customized version of Nextcloud[5]. The second Service we used is Fabasoft Cloud[6] from Fabasoft. We implemented the service-provider specific part of the Audit API for both services.

Following a discussion with experts of a highly-regulated domain, the banking sector, we identified ten high-level security requirements, a financial institution might have on a document sharing service and mapped those to CCM controls. Each control was then broken down into SQO and SLO attributes. For each defined attribute, the evidence is retrieved and evaluated. This evaluation leads to the assessment of an

[3]https://github.com/eu-sec/continuous-auditing-api-spec

[4]https://www.aisec.fraunhofer.de/de/fields-of-expertise/projekte/Clouditor.html

[5]https://github.com/nextcloud

[6]https://www.fabasoft.com/en/products/fabasoft-cloud

objective is fulfilled or not and therefore if a requirement is full filled or not.

For the encryption example, the assessment follows this scheme:

- Clouditor requests all scopes of the service via the API call `/scopes/`
- For each identified `scope`, the tool requests needed evidence for each object using the `/{scope}/objects/` end-point.
- For each `objectId` on each `scope` the encryption information is gathered by the API call `/{scope}/persistence/{objectId}/encryption`.

It is the duty of the provider-specific implementation of the Audit API to retrieve the actual evidence from suitable sources within the service itself, such as log files or by issuing calls to the AWS infrastructure, whereas it is the duty of the auditing tool, such as Clouditor to evaluate the evidence in respect to the required attributes. In our AWS based example, we parse the log of nextloud to retrieve information like the encryption parameters of each shared file or when particular users had access to the application. Information like password requirements, if two-factor authentication is activated or where the files are located on a hard drive is retrieved by reading log files or querying the application database.

The comprehensive assessment of all evidences results in a compliance status. This status is communicated to a certification authority. Based on this compliance status a certification is issued, suspended or revoked.

V. Conclusion and Future Work

Up until now, a major part of audit assessments has to be done with human intervention, but this might change in the future. We have provided a solution that helps to reduce the problem of high implementation efforts for continuous auditing by defining a clear and simple API used in the auditing process. Rather than implementing the full mechanism that is performing the assessment, the service provider only has to define the objectives and implement its specific part of the interface. This approach works regardless of the underlying IT infrastructure since all possible layers of the cloud stack are addressed in the evidence gathering as well as the assessment.

We have extended the security auditing tool Clouditor with an adapter to process evidence generated by this Audit API and implemented the service-provider specific parts in a Cloud-based document sharing solution based on Nextcloud and the Fabasoft Cloud. This solution serves as part of a real-world scenario pilot in the financial sector of the EU-SEC project and we expect to share further results of the pilot in mid-2019.

For future work, there are multiple possible ways how technical improvements can improve on the control breakdown process, such as advancement in natural language processing or a machine readable security controls catalog which contains machine-readable objectives and even attributes. Furthermore, the current Audit API specification is just a starting point for further contributions.

Acknowledgement

This work was partly funded by the European Unions Horizon 2020 project EU-SEC, Grant No. 731845.

References

[1] Anisetti, M., Ardagna, C., Gaudenzi, F., Damiani, E.: A certification framework for cloud-based services. In: Proceedings of the 31st Annual Symposium on Applied Computing (SAC). pp. 440–447. ACM (2016)

[2] Anisetti, M., Ardagna, C.A., Damiani, E., Gaudenzi, F., Veca, R.: Toward Security and Performance Certification of OpenStack. In: 8th International Conference on Cloud Computing (CLOUD). IEEE (2015)

[3] Cimato, S., Damiani, E., Zavatarelli, F., Menicocci, R.: Towards the certification of cloud services. In: 9th World Congress on Services (SERVICES). pp. 92–97. IEEE (2013)

[4] Cloud Security Alliance (CSA): Cloud Controls Matrix v3.0.1 (11-12-18 Update). https://cloudsecurityalliance.org/artifacts/csa-ccm-v-3-0-1-11-12-2018-FINAL/ (2018)

[5] Humphreys, E.: Implementing the ISO/IEC 27001 Information Security Management System Standard. Artech House, Inc., Norwood, MA, USA, 1st edn. (2007)

[6] Kunz, I., Stephanow, P.: A process model to support continuous certification of cloud services. In: 31st IEEE International Conference on Advanced Information Networking and Applications, AINA 2017, Taipei, Taiwan, March 27-29, 2017. pp. 986–993 (2017). https://doi.org/10.1109/AINA.2017.106, https://doi.org/10.1109/AINA.2017.106

[7] Lins, S., Schneider, S., Sunyaev, A.: Trust is good, control is better: Creating secure clouds by continuous auditing. IEEE Trans. Cloud Computing **6**(3), 890–903 (2018). https://doi.org/10.1109/TCC.2016.2522411, https://doi.org/10.1109/TCC.2016.2522411

[8] Stephanow, P., Khajehmoogahi, K.: Towards continuous security certification of software-as-a-service applications using web application testing techniques. In: 31st IEEE International Conference on Advanced Information Networking and Applications, AINA 2017, Taipei, Taiwan, March 27-29, 2017. pp. 931–938 (2017). https://doi.org/10.1109/AINA.2017.107, https://doi.org/10.1109/AINA.2017.107

[9] Stephanow, P., Moein, M., Banse, C.: Continuous location validation of cloud service components. In: IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2017, Hong Kong, December 11-14, 2017. pp. 255–262 (2017). https://doi.org/10.1109/CloudCom.2017.29, https://doi.org/10.1109/CloudCom.2017.29