Fraunhofer
AISEC

FRAUNHOFER INSTITUTE FOR APPLIED AND INTEGRATED SECURITY

CLOUDITOR

CONTINUOUS CLOUD ASSURANCE

Clouditor

Continuous Cloud Assurance

Philipp Stephanow, Christian Banse

clouditor@aisec.fraunhofer.de

Fraunhofer AISEC

Executive Summary

Using cloud services has become main stream and many companies use cloud resources either to provide cloud-based applications or to outsource parts of their internal infrastructure. Yet, using these external services poses critical questions regarding a company's compliance with internal and regulatory requirements.

Naturally, the obvious remedy lies in conducting necessary audits of cloud-based applications to check whether they comply with critical requirements. However, especially with complex cloud-based applications, such audits are time-consuming and costly. Furthermore, when considering dynamic changes inherent to cloud environments, frequently auditing those applications is inevitable, making compliance checks even harder and more expensive.

So the central question is: How to leverage the benefits of cloud resources while ensuring compliance with critical requirements in an efficient way? The answer is provided by *continuous cloud assurance*, an approach to automatically and repeatedly check whether a cloud-based service behaves as expected.

The *Clouditor* enables continuous cloud assurance. It is a set of tools which supports the design and deployment of continuous assurance techniques. The core features of the Clouditor are:

Continuous assurance It continuously, i.e. automatically and repeatedly tests, if a cloud-based application complies with critical requirements.

Minimal invasiveness Conducting continuous compliance checks requires no changes to the structure of the service.

Adaptiveness Dynamic reconfiguration according to changes of the cloud-based applications.

High accuracy Prior to deployment, alternative assurance techniques and their respective configuration are evaluated and compared to select the most suitable ones.

Low overhead Continuous tests are executed in a way that avoids incurring unnecessary overhead while retaining required accuracy of results.

The Clouditor prototype is currently developed and deployed within the cloud lab environment of Fraunhofer AISEC. Several exemplary scenarios have been implemented to continuously validate the compliance of cloud-based applications according to requirements, such as:

- Cloud resource location and deployment (e.g. only use cloud resources within EU),
- cloud resource availability and provisioning (e.g. of virtual machines),
- security configurations (e.g. firewalls, security groups, and user management), and
- non-existence of known security vulnerability (e.g. in web application components).

Contents

1 Introduction

Using and providing cloud-based services entails certain risks. The most prominent ones are security-related [1] but using cloud services involves further risks, such as legal risks, privacy risks, and risks of violating defined business processes. According to an empirical study conducted by the Cloud Security Alliance (CSA) [9], the top three threats to cloud services are:

- Insecure interfaces and APIs,
- data loss and leakage, and
- hardware failure.

This leads to the question how to control these risks, that is, how to unfold potential risks of using cloud services while ensuring that a cloud-based application complies with individual requirements?

Assurance techniques provide answers to this question: These techniques check if a service adheres to a specific set of requirements, thus ensuring that it behaves as expected. However, attributes of a complex application or service may change over time and these changes may be hard to predict or detect. Examples for such changes are:

- Configuration changes,
- patches and upgrades applied to individual components, and
- location changes, i.e. the data center used by a cloud provider for service delivery may vary over time.

Applying assurance techniques to cloud services therefore requires an approach capable of *continuously*, i.e. automatically and repeatedly detecting ongoing changes and assessing their impact on requirements. Bridging this gap, we present *Clouditor*, a dynamic test tool to enable continuous assurance of cloud resources and cloud-based applications.

The remainder of this document is structured as follows: After having provided an overview of the Clouditor toolbox (Section 2), main concepts of the test-based assurance techniques are detailed in Section 3. Thereafter, it is outlined how – prior to deployment – performance of test-based assurance techniques can be evaluated and alternative techniques can be compared. Finally, exemplary scenarios are presented on how the Clouditor can be used to continuously check whether a cloud-based application adheres to a specific set of requirements.

2 Clouditor Toolbox

The Clouditor toolbox consists of five main components which are shown in Figure 2.1. The *Engine* and the *Explorer* are responsible for continuously executing and adapting assurance techniques. The *Simulator* and the *Evaluator* are used prior to deployment; they serve to select techniques and respective configurations which are most suitable to check if a cloud service complies with a particular set of requirements. Lastly, the components can be viewed and configured from a *Dashboard*. Each component is designed as a micro-service and can be deployed in an individual container.

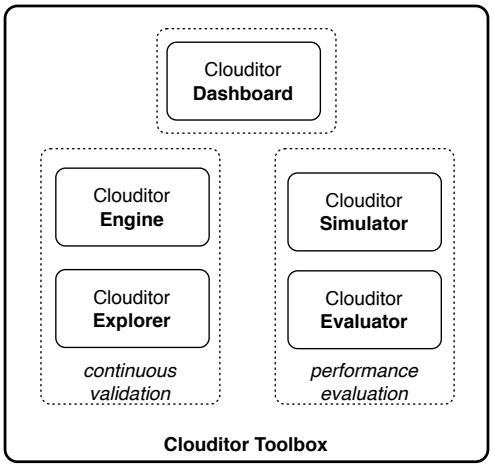

Figure 2.1: Tools of the Clouditor Ecosystem

Clouditor Engine The *Engine* continuously executes a defined set of tests to check whether a cloud service complies with a set of requirements and reports its results. Controlled input is provided to the cloud service and the returned output is evaluated, e.g. calling a cloud service's RESTful API and comparing responses with expected results. Parameters of the tests and their execution, i.e. the interval between tests, are configurable.

Clouditor Explorer The *Explorer* is responsible for automatically discovering a cloud service's composition, e.g. via an exposed API that can be utilized for continuous assurance. Furthermore, cloud services are subject to ongoing changes which may affect operation of continuous assurance techniques. The explorer detects changes of the cloud service and adapts configurations of continuous assurance techniques accordingly.

Trust in results of continuous assurance techniques hinges on their accuracy: How close are produced results to their true values? Answering this question, the Clouditor provides suitable tooling to evaluate how well a particular continuous assurance techniques performs. This evaluation is

conducted prior to productive deployment of assurance techniques. By evaluating and comparing alternative assurance techniques as well as their configurations, the most suitable ones can be selected. More details on this pre-deployment phase can be found in Section 4.

Clouditor Simulator The *Simulator* manipulates a cloud-based application to mock dissatisfying requirements, e.g. publicly expose sensitive interfaces to mimic violations of security configurations, which a specific assurance technique aims to detect. Simulation happens prior to productive deployment of assurance techniques, for example, during integration testing or staging of the cloud-based application. The Simulator thus establishes the ground truth to which results produced by a specific continuous assurance technique are compared.

Clouditor Evaluator The *Evaluator* compares simulated, dissatisfied requirements with the results of the assurance techniques. This allows to reason about the performance of a particular assurance technique and to compare it to alternative techniques. Moreover, using methods of inferential statistics, general statements about the performance of a particular assurance technique can be derived.

Lastly, the visualization of results is an important step in detecting and forecasting potential violations of compliance requirements.

Clouditor Dashboard The *Dashboard* uses information gathered by the other components to visualize the results of continuous testing. Depending on the executed tests and their associated metrics, different visualization components, such as time series graphs, burn-up charts or maps are used.

3 Continuous Validation

Requirements derived from CSA's Cloud Control Matrix (CCM) [5] or NIST SP 800-53 [6] are generic and often inherently ambiguous, making automatic validation infeasible. Thus supporting continuously checking compliance of a cloud-based application requires to extract underlying properties which can be automatically tested, thereby bridging the *semantic gap*.

Reasoning about properties of a cloud service requires collecting and evaluating *evidence*, i.e. observable information of the service, e.g. monitoring data, log files or source code. *Test-based assurance techniques* produce evidence by controlling some input to the cloud-based resource and evaluating the output.

The following section outlines the main elements of the *Clouditor Engine* which supports test-based, continuous cloud assurance. Thereafter, the *Clouditor Explorer* is described which is responsible for service discovery and automatic (re-)configuration (see Section 3.2).

3.1 Clouditor Engine

The *Clouditor Engine* implements and deploys test-based assurance techniques. It consists of *test suites* which comprise *test cases*, *workflows* which model dependencies between test suites, and *metrics* which are used to reason about the results of test suites. Figure 3.1 shows a high level architecture of the Clouditor Engine's components, including data and control flow.

Test cases Test cases form the primitive of any test; they implement any steps executed during the test, e.g. *first establish an SSH connection to a virtual machine, then execute a command to download and install a package on the VM*. A test case possesses a set of initialization parameters: For example, connecting to a VM via SSH may require username, hostname, and a path to a keyfile. Further, each test case possesses assert parameters specifying expected results, e.g. the returned values of the test case have to equal a particular string. Multiple test cases can be executed concurrently or successively.

Test suites Test suites combine test cases, in which each suite must contain at least one test case. A test suite only passes if all contained test cases pass. Execution of a test suite can be triggered multiple times, possibly set to infinity. The current iteration of a test suite has to be completed, i.e. all test cases bound to the test suite have to be completed, in order for the following iteration to start. The interval between consecutive iterations of a test suite can be fixed, e.g. ten minutes after the previous test suite execution has completed, or the interval can serve as a window from which the start of its next execution is selected randomly.

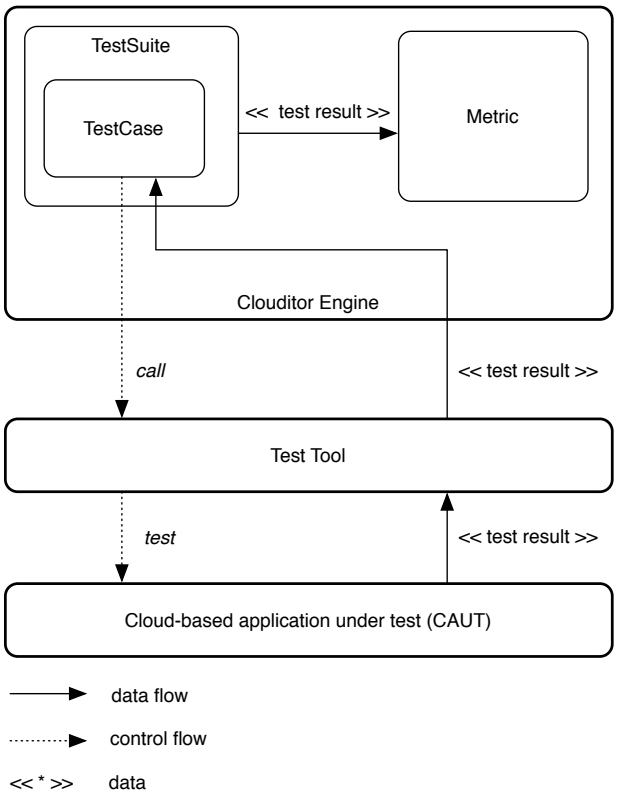

Figure 3.1: Overview of Clouditor Engine main components (with external test tool)

Workflows A workflow represents dependencies between iterations of different test suites. To that end, a workflow controls executions of test suites based on their results. As a basic example, suppose that after having successfully completed a number of iteration, a test suite run fails. The workflow defines how to handle this failure, e.g. whether to continue running the test suite for the remaining iterations, to terminate the test or start another test suite.

Test metrics Automatically evaluating statements over cloud services properties, e.g. *the availability of the service needs to be higher than 99.999% per year*, requires one final construct: Metrics. A metric takes the results of test suite runs as input, performs a specified computation and returns the result. To that end, a metric can use any information available from the result of a test suite run, e.g. at what time the test suite run was triggered, when it finished, and further information contained in the results of test case runs bound to the test suite run.

Preconditions Naively executing tests is prone to false positives, e.g. testing a webserver's TLS configuration may fail, not because of a vulnerable configuration but because the webserver cannot be reached. Computing metrics based on such a test suite will further increase their error. Thus assumptions made about the environment of the cloud-based application under test, i.e. preconditions, need to be tested as well.

3.2 Clouditor Explorer

Discovering interfaces of cloud-based applications and configuring the selected assurance technique is the task of the *Clouditor Explorer*.

Service Discovery In order to determine which assurance techniques can be utilized to check compliance of a specific cloud-based application, it is necessary to discover the application's composition and interfaces. To that end, designated service description APIs, inventory management systems as well as test-based discovery techniques can be used.

Naturally, the level of detail of the obtained service description depends on the privilege level granted to service discovery. If, for example, the assurance techniques will have identical access privileges as a regular user of the application, then only publicly available interface will be accessible to service discovery.

Configuration Generation Based on the service description which is provided by service discovery, candidate assurance techniques are proposed. Depending on the required compliance checks, that is, the requirement set which has to be validated continuously, corresponding assurance techniques are selected. For each selected technique, configurations are generated which are used by the *Clouditor Engine* to deploy test-based assurance techniques accordingly.

Configuration Adaption A cloud-based application may change during deployment, e.g. additional instances are launched due to increased load. Such changes can be detected by the *Clouditor Explorer* and deployed assurance techniques are reconfigured accordingly.

4 Performance Evaluation

The accuracy of results which are produced by test-based assurance techniques depends on various factors, e.g. test implementation, test environment, usage of external tools, etc. Without experimental evaluation, it is thus hard to make a statement about how well a specific continuous assurance technique detects compliance requirements violations.

Figure 4.1 shows the main components involved in performance evaluation. In order to evaluate test-based assurance techniques, it is assumed that correct results and errors of the assurance technique follow some unknown distributions. We take samples from these unknown distributions by running experiments which simulate violations of compliance requirements. By applying methods of statistical inference to these experimental results, it is possible to make statements about the general performance of the test-based technique under evaluation.

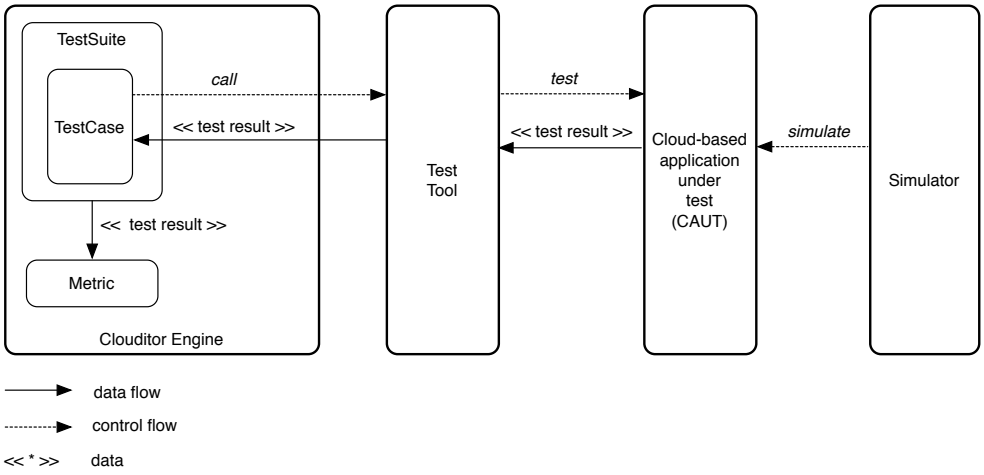

Figure 4.1: Overview of performance evaluation using the Clouditor toolbox

4.1 Clouditor Simulator

A *simulation* manipulates a cloud service under test to mock violations of compliance requirements which a specific test-based assurance technique aims to detect. Thus simulations are essential to establish the ground truth to which results produced by assurance techniques are compared.

The design of a simulation is driven by the requirements that should be tested. For example, a simulation may start and stop virtual machines to simulate violations of availability requirements, publicly expose sensitive interfaces to mimic violations of secure configuration requirements, or limiting bandwidth to simulate violations of quality of service requirements.

4.2 Clouditor Evaluator

The *Evaluator* compares simulated violations of compliance requirements with results produced by assurance techniques under evaluation. It computes different performance measures which allow to reason about how well a specific test-based technique works in detecting violations of compliance requirements.

Selecting suitable performance measures to evaluate and compare alternative test-based assurance techniques depends on the compliance requirement that the assurance technique aims to validate. Consider for example the performance of continuously testing a requirement that *a cloud-based application should not to be accessible through some blacklisted ports*. Here, the accuracy of the test-based technique can be described by simply counting correct results, given that the test-based technique correctly indicates a violation of the requirement. As another example, consider the requirement *The average time to fix critical security vulnerabilities should not exceed eight hours*. In this case, the performance of the test-based techniques depends on its ability to approximate singular intervals during which the requirement is violated.

5 Exemplary Scenarios

This section presents five exemplary scenarios on how the Clouditor can be used to check if a cloud-based application complies with a specific set of requirements. Within all the scenarios described hereafter the Clouditor's deployment is *minimally invasive*, that is, the Clouditor is deployed externally to the infrastructure of the cloud-based application; thus requiring no changes to the application's components.

5.1 Resource Location

The Clouditor can be used to continuously validate the location of cloud resources. This allows to check requirements which define that used cloud resources shall only be located within certain geographical boundaries. Such requirements may stem from

- national privacy regulations,
- a company's internal data protection guidelines, or
- individual contractual obligations.

The test-based technique firstly collects network information through interaction with a cloud resource used by the application. Machine learning models are used to learn the characteristics of a particular cloud resource. Using these models, the test-based technique can continuously validate a cloud resource's location (*LocationTest*). The test will fail in case the cloud resource has migrated to a different data center, located in a different geographical area.

Based on the results of *LocationTest*, metrics can be computed which allow to for instance count how many times a cloud resource's location was invalid and how long this compliance requirement was valid.

5.2 Service Availability

This scenario describes how the Clouditor can be used to continuously validate compliance requirements related to service availability and provisioning, for example, *the application should be available at least 99.999% per year*. Such requirements can, for instance be derived from

- Control *SC-6 Resource Availability* of NIST SP 800-53 [6],
- *IVS-04* of the Cloud Control Matrix (CCM) upon which the CSA certificate is based [4], or
- *Section 6.3.7 Resource Provisioning* of ENISA IAF [2].

Consider the following, alternative test-based techniques: The first possibility is *PingTest* which simply sends ICMP messages to publicly reachable interfaces and verifies that the returned round trip time (*RRT*) does not exceed a given threshold on average and standard deviation. A second possible technique uses TCP packets to determine whether the application is available (*TCPTest*). Similar to *PingTest*, for a test to pass, thresholds for the maximum average response time and the maximum response time of probes need to be defined which are not to be exceeded. The third possibility is *SSHTest* which tries to connect to a component of the application via SSH and then test the session.

Each of the exemplary test-based techniques described above can be used individually or combined. In the latter case, *PingTest*, *TCPTest*, and *SSHTest* are executed concurrently at each iteration, and only if all pass, the test passes. Based on the result of these test-based techniques, different metrics can be computed. These metrics allow to reason about the availability of a cloud resource.

5.3 Security Configuration

In this scenario, requirements related to secure communication and configuration are continuously validated. Such requirements may stem from

- *Section 6.4.5 Encryption* of ENISA IAF [2],
- *SC-8 Transmission Confidentiality and Integrity* of NIST SP 800-53 [6], or
- *IVS-04: Infrastructure & Virtualization Security Information System Documentation* of the CCM [5].

Consider the following two exemplary test-based techniques: One of them tests if data transferred to the cloud-based application is vulnerable during transit. The other one tests if the cloud-based application exposes vulnerable interfaces. Regarding secure communications, the TLS configuration of the cloud-based application is analyzed to identify weak cipher suites (*TLSTest*). To detect vulnerable interfaces, reachable network ports of the application (*PortTest*) are discovered.

Using the results of TLSTest and PortTest, different metrics can be derived, e.g. how many times configurations of interfaces and communication were vulnerable and how long it took to fix these vulnerabilities.

5.4 Input Validation

In the last example scenario, the Clouditor checks compliance with security requirements derived from, e.g.,

- *SI-10 Information Input Validation*, and *RA-5 Vulnerability Scanning* of NIST SP 800-53 [6],
- *Section 6.3.1. Software Assurance*, and *6.3.6. SAAS – Application Security* of ENISA IAF [2],
- controls *A.9.4.1: Information access restriction* and *A.12.6.1 Management of technical vulnerabilities* of ISO/IEC 27001:2013 [7], or

- *AIS-01: Application Security* and *TVM-02: Vulnerability & Patch Management* of the Cloud Control Matrix (CCM) [5].

Cloud-based applications following a Software-as-a-Service (SaaS) model usually make heavy use of web application technologies. The Open Web Application Security Project (OWASP) has classified the top 10 categories of vulnerabilities found in web applications based on their occurrence. Within that classification, *injection* attacks are on top of the list, a broad term which refers to different types of attacks such as SQL, OS commands and LDAP injection. Among all types of injection, SQL injection (SQLI) is the most common type of vulnerabilities existent in today's web applications.

The test-based technique continuously tests for SQL Injection (SQLI) vulnerabilities of SaaS applications. A test passes if no SQLI vulnerabilities have been found, otherwise it fails. Based on the test results, metrics are computed which, e.g., indicate the severity of a detected vulnerability as well as how long a vulnerability persisted.

5.5 Code Quality Gate Satisfaction

In this scenario, the Clouditor continuously validates whether one or multiple software projects meet defined code quality requirements. This can support validation of requirements derived from, e.g.,

- The chapter *Security Testing* of the Software Assurance Maturity Model (openSAMM) [3], requiring to establish quality gates which have to be passed for an application to be released, or
- Best Practice *SDL Practice #3: Create Quality Gates/Bug Bars* of Microsoft's SDL [8] requiring to define thresholds – so-called *bug bars* – which are *not* to be exceeded for the application to be released.

A necessary pre-requisite is the existence of a suitable code quality tool, such as SonarQube[1] and its integration within the build process. The Clouditor is then able to extract code quality information, i.e. the passing and failing of a quality gate, through defined REST APIs. While the current prototype implementation only supports the retrieval of such information from the aforementioned SonarQube tool, its metrics are agnostic to the actual deployed tool.

[1] https://sonarqube.com

Bibliography

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010. 5

[2] D. Catteddu, G. Hogben, et al. Cloud computing information assurance framework. *European Network and Information Security Agency (ENISA)*, 2009. 13, 14

[3] P. Chandra et al. Software Assurance Maturity Model (SAMM): A guide to building security into software development. http://www.opensamm.org/downloads/SAMM-1.0.pdf. Retrieved 11-2016. 15

[4] Cloud Security Alliance (CSA). Security, Trust and Assurance Registry (STAR). https://cloudsecurityalliance.org/star/certification/. 13

[5] Cloud Security Alliance (CSA). Cloud Control Matrix: Seurity Controls Framework for Cloud Providers & Consumers. https://cloudsecurityalliance.org/research/ccm/, 2013. 8, 14, 15

[6] J. T. FORCE and T. INITIATIVE. Security and privacy controls for federal information systems and organizations. *NIST Special Publication*, 800:53, 2013. 8, 13, 14

[7] International Organization for Standardization (ISO). ISO/IEC 27001:2013 Information technology – Security techniques – Information security management systems – Requirements. 14

[8] Microsoft. The Security Development Lifecycle (SDL). https://www.microsoft.com/en-us/sdl/. Retrieved 11-2016. 15

[9] S. S. G. L. Ryan Ko. Cloud Computing Vulnerability Incidents: A Statistical Overview, March 2013. 5

Fraunhofer AISEC

The Fraunhofer Institute for Applied and Integrated Security AISEC under the responsibility of Prof. Dr. Claudia Eckert is one of the leading research institutions in Europe. Fraunhofer AISEC is focused on development of application-oriented security solutions and their precise and tailored integration into existing systems. Core competences of over 90 scientific and technical members of staff lie in the areas of hardware security and the security of embedded systems, product and intellectual property protection, network security, and security in cloud- and service-oriented computing. Fraunhofer AISEC's clients operate in a variety of industrial sectors, such as the chip card industry, telecommunications, the automotive industry, and mechanical engineering, as well as the software and healthcare industries. The main goal is to support and improve the competitiveness of our clients and partners in the manufacturing and service sectors as well as those in the public sector.

How to collaborate with us

Methodology and tooling of the Clouditor is currently developed and deployed in the cloud lab environment located at Fraunhofer AISEC. As part of comprehensive research experiments, we have already implemented and deployed various test-based techniques, some of which are described as part of this document.

We are looking for:

- Providers and customers of cloud services with whom we evaluate the Clouditor toolbox in a real world setting,
- industry partners which seek to continuously validate compliance of cloud resources and cloud-based applications, and
- cloud auditors to further discover requirements suited for continuous assurance.

We are offering:

- Planning and deployment of continuous assurance techniques to check compliance of cloud-based applications,
- research-backed insights into continuous cloud assurance,
- deep knowledge about functional and non-functional test design and deployment, especially security testing, as well as
- analysis and design of security architectures of cloud-based applications.

The Authors

Philipp Stephanow is a senior researcher who started working for Fraunhofer AISEC in 2010. He focuses on cloud security and assurance, mobile security, and IoT security assurance. Philipp contributed to several public funded research projects targeting cloud security challenges. Furthermore, he was involved in numerous industry projects where he conducted threat and risk assessments of productively deployed services, analysed and derived security requirements, as well as proposed suitable security models. Currently, Philipp is the technical manager of the *Next Generation Certification (NGCert)*[2] project where he is responsible for the development and implementation of a cloud service certification system. Lastly, Philipp authored numerous scientific publications in the area of cloud security, in particular focusing on security aspects of continuous certification, and is currently finishing his doctoral thesis on trustworthy continuous cloud service certification.

Christian Banse is the deputy head of the department *Service and Application Security* of Fraunhofer AISEC and has been employed at the institute since 2011. Within the department he is responsible for the topics of network and communication security, especially in cloud and Software-Defined Networking (SDN) environments. Christian is also in charge of the Network and Cloud Security Lab of Fraunhofer AISEC, a test environment which allows the simulation of different network and cloud deployments and is extensively used in the evaluation of proof-of-concept implementations. His main research focus lies within the field of SDN security but also extends to other cloud and service-related areas, such as cloud service certification. He is the author of several publications in the field of communication and network security.

[2] www.ngcert.de

Fraunhofer AISEC

Fraunhofer Institute for Applied
and Integrated Security

Parkring 4
85748 Garching, Germany

www.aisec.fraunhofer.de
Phone: +49 89 322 99 86 119
E-Mail: clouditor@aisec.fraunhofer.de