# Sun Behind Clouds - On Automatic Cloud Security Audits and a Cloud Audit Policy Language

Frank Doelitzscher, Thomas Ruebsamen, Tina Karbe, Martin Knahl, Christoph Reich
Cloud Research Lab
Furtwangen University, Furtwangen, Germany
{Frank.Doelitzscher, Thomas.Ruebsamen, Tina.Karbe, Martin.Knahl, Christoph.Reich}@hs-furtwangen.de

Nathan Clarke
Centre for Security, Communications & Network Research
Plymouth University
Plymouth PL4 8AA, United Kingdom
N.Clarke@plymouth.ac.uk

***Abstract*—Studies show that when it comes to an integration of Cloud computing into enterprises, chief information officers and management still see some dark Clouds on the horizon. The biggest one is the lack of security, which results in distrust and skepticism against the technology, mainly originating from an intransparency of Cloud environments. To increase this transparency, the Cloud Research Lab at Furtwangen University develops the Security Audit as a Service (SAaaS) architecture for Infrastructure as a Service Cloud environments. It is targeted to ensure that a desired security level is reached and maintained within a frequently changing Cloud infrastructure. Despite a traditional security audit, which includes a comprehensive and therefore time-consuming security check of a whole infrastructure, a Cloud security audit needs to be lightweight enough to be executed right after an infrastructure change occurred, and precisely target-oriented to perform an audit of the specific infrastructure components affected by this change. This is called a concurrent security audit. In this paper, a Cloud audit policy language for the SAaaS architecture gets presented. First, the design and implementation of the automated audit system of virtual machine images, which ensures legal and company policies, is described. Second, on-demand deployed software audit agents that maintain and validate the security compliance of running Cloud services, are discussed.**

***Keywords*—Cloud computing, security policies, Cloud audits, agents**

What concerns were expressed during the decision-making process to migrate to the Cloud?

| Only asked of respondents whose company has hosted or Cloud-based services in use | Total | No. employees Fewer than 20 | 20 - 200 | More than 200 | Public | Private |
|---|---|---|---|---|---|---|
| Data security | 82% | 77% | 85% | 82% | 89% | 78% |
| Data privacy | 69% | 79% | 63% | 68% | 70% | 69% |
| Dependency upon internet access (availability and bandwidth) | 51% | 53% | 52% | 48% | 49% | 52% |
| Fear of loss of control/manageability | 46% | 35% | 48% | 54% | 53% | 43% |
| Confidence in the reliability of the vendors | 36% | 35% | 33% | 39% | 25% | 42% |
| Data sovereignty/jurisdiction | 33% | 37% | 31% | 30% | 30% | 34% |
| Cost of change/migration | 31% | 33% | 33% | 29% | 26% | 34% |
| Contract lock-in | 29% | 33% | 26% | 30% | 32% | 28% |
| Lack of clarity of impact of Cloud Services on business processes | 27% | 16% | 28% | 34% | 32% | 24% |
| Regulatory constraints | 26% | 16% | 30% | 30% | 28% | 25% |
| Contractual liability for services if SaI's are missed | 20% | 12% | 17% | 29% | 9% | 25% |
| Confidence in the vendors business capability | 19% | 19% | 19% | 20% | 17% | 20% |
| Confidence in knowing who to choose to supply service | 18% | 26% | 20% | 11% | 19% | 18% |
| Confidence in the clarity of charges (i.e. will they be cheaper than on-premise) | 18% | 16% | 13% | 23% | 19% | 17% |
| Lack of confidence in the business case to need Cloud Services | 11% | 9% | 11% | 13% | 11% | 11% |
| Lack of clarity in most appropriate Cloud deployment model | 10% | 7% | 13% | 9% | 8% | 11% |
| Lack of any advice from within the company to adopt | 8% | 9% | 7% | 9% | 13% | 6% |
| Lack of clarity in most effective service delivery model | 7% | 5% | 7% | 9% | 8% | 7% |
| Lack of any promotion or awareness by the people we buy IT from | 2% | 0% | 0% | 5% | 4% | 1% |
| Other | 1% | 2% | 2% | 0% | 2% | 1% |
| Base | 153 | 43 | 54 | 56 | 53 | 100 |

Fig. 1: What concerns were expressed during the decision-making process to migrate to the Cloud? [6]

## I. Introduction

This paper is the successor of the conference paper "Incident Detection for Cloud Environments" [1] presented at EMERGING 2011. In addition to the presentation of the Security Audit as a Service (SAaaS) architecture, the comprehensive definition of an automated virtual machine (VM) image audit system and the definition and presentation of a Cloud audit policy language, form a novel contribution for this extended journal paper.

Cloud vendors promise "infinite scalability and resources" combined with on-demand access from everywhere. This lets Cloud users quickly forget that there is still a real IT infrastructure behind a Cloud, and due to virtualization and multi-tenancy the complexity of these infrastructures is even increased compared to traditional data centers. This makes management of service provisioning, monitoring, backup, disaster recovery, security, etc. more complicated and, therefore, there is still a lack of trust in Cloud infrastructures. Enterprise analysts and researchers have identified Cloud specific security problems as the major research area in Cloud computing [2], [3], [4], [5]. The survey "Cloud Adoption and Trends for 2013" which was done amongst 250 CIOs and other IT executives at UK companies and public sector organizations states that security concerns are still the major issue, which hinders a broad industry acceptance of actually utilizing Cloud technologies (see Fig. 1). In fact, even for private Cloud solutions, where an enterprise runs its own Cloud IT infrastructure, security concerns are named as the major obstacles, which proves that there is a general lack of trust in Cloud computing security.

Security is a considerable challenge for Cloud environments due to its characteristics: seamless scalability, shared resources, multi-tenancy, access from everywhere, on-demand availability and third party hosting. Although existing industry recommendations (ITIL), standards (ISO 20000, ISO 27001:5, CobiT) and laws (e.g., Germanys Federal Data Protection Act) provide well established security and privacy rule sets for data center providers, research has shown that additional

regulations have to be defined for Cloud environments [2], [7]. The following examples of Cloud security incidents illustrate the need for Cloud computing security improvements:

- Hackers stole credentials of Salesforce.com's customers via phishing attacks (2007)
- T-Mobile customers lost data due to the "Sidekick disaster" of Microsoft Cloud (2009)
- Botnet incident at Amazon EC2 infected customer's computers and compromised their privacy (2009)
- Hotmail accounts were hacked due to technical flaws in Microsoft software (2010)
- Amazon customer services were unavailable for multiple days and data was lost due to a logical flaw in the Cloud storage design (2011)

Traditionally, IT infrastructure security audits are used to document a data-center's compliance to security best practices and laws. But, the major shortcoming of a traditional security audit is that it only provides a snapshot of an environments' security state at the performed audit time. This is adequate since classic IT infrastructures don't change that frequently. But because of the mentioned Cloud characteristics above, it is not sufficient for auditing a Cloud environment [1].

Beside the frequently changing infrastructure inside a Cloud, there is also a new dynamic in administrative tasks. The number of administrators of a traditional data centre is limited and they all are working under the same company security policy, while installing and maintaining machines. This can be completely different in a Cloud infrastructure. Public marketplaces for exchanging Cloud appliances such as, OpenNebula Marketplace [8], Amazon Web Services EC2 Management Console or the Amazon Web Services Marketplace [9] provide Cloud customers with an easy and efficient way of finding the right virtual machine image. But they also allow users to be administrators of their virtual machines, or upload and share their custom made VM images with other users. Although Cloud providers provide security guidelines [10] on how to prepare an image before releasing it to a marketplace, current research by Balduzzi [11], Bugiel [12] and Meer [13] shows that marketplace images are highly insecure due to old software versions or "forgotten" or restorable security credentials, such as SSH private keys. Users, uploading appliances are usually more or less anonymous. There is no way to easily determine whether a custom appliance is legit or maliciously manipulated. Images could contain rootkits, which are performing passive eavesdropping attacks such as traffic analysis, keylogging or transmission of user's data to external systems for industrial spying [11].

European and German data protection laws increase the necessity for users to re-validate the security status of virtual machines originating from preconfigured images by clearly putting the user who runs the image into responsibility (§3.7 German Data Protection Law, Art. 2d 4, European Guideline 95/46/EG) when data is processed in the Cloud. The Cloud user has to re-validate technical and organizational security measures taken by the Cloud provider initially at the beginning of a Cloud usage and periodically over time (§11 II - 4 German Data Protection Law).

To mitigate these problems, this work proposes the Security Audit as a Service architecture for Infrastructure as a Service (IaaS) Clouds. The contribution of this paper is structured in two parts: A) Description of an automatic Cloud audit architecture based on agents, which react on changes in a Cloud infrastructure. B) A Cloud audit policy language to describe security targets and enable automatic Cloud audits. It is shown that automatic audits of VMs and VM images can increase transparency and therefore security in Cloud computing environments.

In the remainder of this article, Section II - Use Cases of the SAaaS System, introduces the target scenarios, which this work aims to solve. Section III then gives an inside view into the process of automatic virtual machine image audits. Following, Section IV - A Cloud Audit Policy Language presents a comparison of existing security policy languages and introduces CAPL - a policy extension for the Cloud Infrastructure Management Interface (CIMI). Then, the Security Audit as a Service architecture is presented in Section V, which introduces the concept of using distributed agents to perform Cloud audits. An integration of the developed Cloud audit policy language is also shown. How the presented work increases security and transparency in Cloud environments is elaborated in Section VI - Evaluation. Section VII - Related Work, discusses related work on the topic of Cloud specific security problems, Cloud audits and other research similar to Security Audit as a Service, before Section VIII - Conclusion & Outlook wraps-up the paper and gives an outlook into future work.

## II. Use Cases of the SAaaS System

The Security Audit as a Service system presented in this work covers three use cases:

**1. Automated security audit:** In this use case the SAaaS architecture gets used as a Software as a Service (SaaS) solution. It enables users to plan and perform security audits of their IT infrastructure on a regular basis. An audit can consist of regular vulnerability scans of a user's internet exposed systems (not necessarily Cloud instances). Results get automatically evaluated, post-processed and submitted as a security report in a standardized format to the user. Additionally, to simplify black box scans it is imaginable to deposit an entry credential (e.g., a ssh key pair) in the service so that the service can log in and perform internal security scans. While such systems already exist as appliances (e.g., Nessus appliance), especially small SMEs can profit from this service running in a Cloud since they only need to pay per scan. For the Cloud provider this service is valuable since computing resources are only allocated for the duration of a scan. Afterwards, the compute resources are released and made available for different tasks.

**2. Monitoring and Audit of Cloud Instances:** User VMs running in a Cloud infrastructure are equipped with a SAaaS agent. The user creates security policies defining the behaviour of this VM to be considered "normal", which VM components are to be monitored and how to alert the customer in case of system deviation from the defined manner. The status gets conditioned in a user-friendly format in a Web portal - the SAaaS security dashboard. This continuous

monitoring creates transparency about the security status of a user's Cloud VMs hence increasing the user's trust into the cloud environment.

**3. Cloud Infrastructure Monitoring and Audit:** The security status of the entire Cloud environment, especially the Cloud management system, access to customer data and data paths are monitored. Usage and communication behaviour profiles are created automatically and continuously analyzed for substantial changes. This way monitoring across different customers is used by the Cloud provider as well as a 3rd party, like a security service provider to ensure the overall cloud security status. Standardized interfaces enable security audits of the Cloud infrastructure, which can lead to a cloud security certification.

## III. Auditing VMs of a Cloud

As previously mentioned, using third party appliance images from public marketplaces can pose a significant security risk. Therefore, not only running virtual machines need to be audited in a Cloud environment, but also virtual appliance images, from which virtual machines are created. This section describes requirements (see Section III-A), roles (see Section III-B), audit categories (see Section III-C) and a system architecture (see Section III-D) involved in solving the aforementioned issues. Virtual machine image auditing is regarded as a part of SAaaS use case 2 - Monitoring and Audit of Cloud Instances.

### A. VM Auditing Requirements

To be able to automatically audit VM images, it is essential to describe the security and privacy requirements, in a machine understandable way. This is commonly achieved by the definition of security policies, transferring a requirement into a checklist of one or multiple testable conditions. To respect Cloud user's and provider's security requirements, both parties need to be able to create policies. A key factor for the success of such a system is the detailed and distinct definition of security policies. However, this is contrary to a short VM deployment process a Cloud user expects. Therefore, we propose to create a very easily operable, security policy generator, where Cloud users can define security policies in a human way of thinking, such as: "*The VM must be checked for malware*". Such simple policies could be supported by a graphical Web interface with templates utilizing check boxes or drop-down lists. This needs then to be translated into a machine understandable format, which results in the audit checks to be performed. The output of these checks needs to be translated back into a human understandable format, which will form the audit report submitted to the image creator, Cloud provider and image user. In summary, the following important audit requirements can be identified:

- VM images need to be audited in an automatic manner, to provide short response times to an image creator who wants to publish its image.
- The system needs to respect different security requirements from the image creator as well as the Cloud provider.
- The system needs to produce a human understandable output in case an image did not pass the security check, providing the image creator with information about what prohibited the image release so that he is able to fix it.
- Security policies need to be described in a machine understandable way.

### B. VM Audit Roles

When it comes to auditing virtual machine appliances, there are a couple of different roles, which need to be considered: *appliance user*, *appliance creator*, *Cloud provider*, *audit service provider* and *audit tool provider*. These roles will be described in more detail in the following.

The **appliance user** is a customer of the Cloud provider obtaining the virtual machine images via the appliance store. The main concern of the appliance user is to make sure, that the VM complies with the company's IT security policy when using third party appliances. Such a security policy may include the necessity of malware checks (e.g., viruses, trojans, spyware, rootkits etc.), checks for undesirable software (e.g., games, file sharing software) but also a more detailed view on the operating system and services configuration of an appliance may be checked. For example, if there exist unprivileged system user accounts for running a Web server or if there are any leftover default passwords, which the appliance creator may have overlooked. The auditing's goal, from an appliance user's point of view, is to make sure a virtual appliance complies to his company's security policy, before the appliance is started and integrated in the company's IT infrastructure.

The **appliance creator** can be the Cloud provider himself or a customer of the Cloud provider. He creates individual VMs and shares them with other Cloud customers using an appliance store. Before publishing virtual appliances, the appliance creator has to make sure that there is no private data, which could compromise privacy (e.g., logs, browser cache, user information like names and addresses), left on the image. Another, often overlooked, aspect are non-securely deleted files on the image's file system. It is often possible to recover such files with little effort using file carving tools, like extundelete [14] or winundelete [15]. The auditing's goal from the appliance creator's point of view is therefore mostly to make sure policies, which prevent the disclosure of sensitive data, are used during auditing of the appliance before it gets published.

The **Cloud provider** provides the technical infrastructure for running the virtual machine image and also runs the appliance store. Providers usually have little or no interest in restricting the creation and publication of virtual machine appliances, as long as there is no violation of laws or terms of use. Such violations may include the intentional distribution of malware, intentionally misconfigured services or any form of illegal content, such as pirated software.

The two remaining roles **audit service provider** and **audit tool provider** have no immediate interest in auditing virtual machine appliances. They merely complete the auditing process by providing additional services and tools. The audit tool provider designs, develops and provides programs and services

TABLE I: Virtual Appliance Audit Categories

| | | Appliance User | Appliance Creator | Cloud Provider |
|---|---|---|---|---|
| Security | Malware | x | | x |
| | Undesirable Software | x | | |
| | Account Requirements | x | | |
| | Login Requirements | x | | |
| | Password Strength | x | | |
| | Access Rights | x | | x |
| | Service Misconfiguration | x | | x |
| | Unwanted Service Combination | x | | |
| Privacy | Browser Caches | | x | |
| | Log Files | | x | |
| | History Files | | x | |
| | Insecurely Deleted Files | | x | |
| Legal | Software Licenses | x | x | |
| | Illegal Content | x | x | x |
| | Customer Specific Requirements | x | | |

for auditing virtual machine images. The audit service provider is a specialist in auditing IT infrastructure and therefore has extensive knowledge about auditing procedures and methodologies, which he offers to the Cloud provider. He also provides the Cloud provider with work-flows, recommendations about the tools to use, knowledge about currently emerging threats to security and privacy as well as any additional auditing know-how.

### C. Auditing Categories

The auditing categories identified by the authors are **security**, **privacy** and **legal concerns**. Multiple audit cases from these categories can be arbitrarily combined to form an auditing policy. Each of the previously described roles has a different view on the requirements the audit process has to fulfill. Table I illustrates this circumstance. The security category includes requirements regarding the absence of malware and otherwise undesirable software in the virtual appliance. Also, the preconfiguration of the appliance's services, like access rights on a file system level, the combination of services (e.g., mail daemons and network attached storage (NAS) service on the same appliance), insecure default service configurations, the use of insecure default passwords and login requirements (allowing remote administrator access with passwords) are the most common concerns. The privacy category includes mostly requirements, which should help preventing unintentional data loss. This includes leftovers from the appliance setup process, like log files, command line history or insecurely deleted files. Additionally, this category also includes data generated by end-user applications (e.g., browser caches). The legal category includes all sorts of compliance requirements, for example illegal content stored on the image.

The appliance user obviously has much more interest in secure and compliant appliances. One could argue that data loss while using the appliance because of misconfigured services or backdoor programs could be assigned to the privacy category, but this data loss arises because of security problems.

The appliance creator has mostly privacy concerns, when publishing virtual appliances. Checks for the previously described problems need to be thoroughly executed before publishing a virtual appliance. Also, not publishing preinstalled software licenses is a legal problem, which needs to be checked for.

The Cloud provider's main concern is protecting his own infrastructure. Therefore, the categories, which apply are security and legal. Checking virtual appliance images for malware may reduce the risk of malware spreading and hackers using the appliance store as a basis for their attacks. Checking for illegal content may also be necessary because the Cloud provider stores virtual appliances.

### D. Automatic Auditing System Overview

Figure 2 shows an overview of our proposed auditing system for virtual appliances.

The *appliance store* is enhanced by an *Automatic Audit System* (AAS) for image auditing. The cloud infrastructure as well as the appliance store are run and maintained by the Cloud provider.

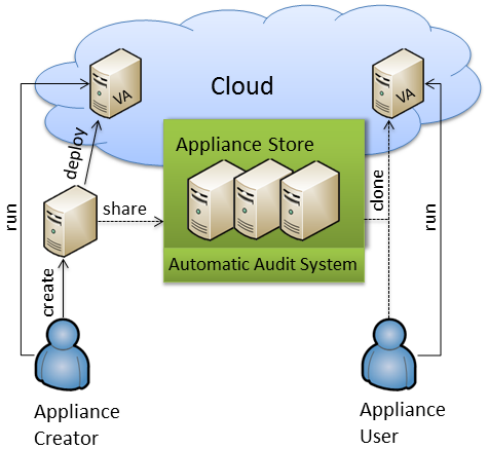


Fig. 2: Appliance Store with Automatic Audit System

A typical process of creating, auditing and sharing virtual appliances is described in the following:

1) The *appliance creator* creates a virtual appliance (VA) (e.g., setting up the operating system and services provided by the appliance). Optionally, the appliance creator can upload a policy document, which describes his requirements for publication of the VA. This policy is evaluated by the AAS. If any policy violations are detected (e.g., there are insecurely deleted files or log files in the VA), the publication process is immediately stopped.
2) The *AAS* audits the virtual appliance using the Cloud provider's policies (see III-C for audit case examples). This step is performed to make sure, the Cloud provider's requirements are met (e.g., protecting the Cloud infrastructure from malicious VAs).

3) On successfully passing the audit process the appliance is deployed in the Cloud and the image is published using the appliance store.
4) The *appliance user* now decides to use the virtual appliance. He therefore uploads his policy to the AAS, which in turn audits the appliance according to the audit cases defined in this policy.
5) If the audit process is passed successfully, the virtual appliance image is cloned by the appliance store and deployed to the cloud.

While this is a rather high level design of an automatic system more details of the technology used is described in the following sections. It becomes clear, that as an enabler a security policy language for virtual machines is needed.

## IV. A Cloud Audit Policy Language

This section first describes the requirements established for a Cloud audit policy language. Beneath generic attributes six specific policy scenarios get introduced, which have to be describable with the target language. Afterwards, several already existing security policy languages have been evaluated using the requirements and it is shown why none of them fulfills these requirements. Therefore, a new Cloud Audit Policy Language CAPL will be introduced, which is based on the Cloud Infrastructure Management Interface (CIMI) [16] specification.

### A. Security Policy Language Requirements

Apparently, there is a huge number of policy definition languages available (ASL, LaSCO, PDL, XACML, SPARQLE, SSPL, OVAL, etc.) all aiming to model security policies. Furthermore, business process languages (BPEL, WADE, YAWL, ADEPT, etc.) could be applicable as well, which increases the size of languages to choose from. To be able to evaluate, which language fits best a bottom up approach was taken by first defining the policy scenarios, which need to be describable by the security policy definition language.

**1.) Policy Scenario Modelling Support** The most important criterion in our evaluation process is the ability to model security requirements of Cloud components, such as VMs and their interaction with each other. To have a basis for the evaluation of this criterion a number of important example policy scenarios have been identified:

*P1 - Malware:* Since malware affects availability, integrity and confidentiality every VM image needs to be checked for viruses and rootkits before being started within a cloud. Running VMs must be checked on a regular basis. The resulting policy could be: “The VM is free of malware”.

*P2 - Filesystem changes:* Malicious attacks often result in change of the file systems’ content, such as modification of config files or installation of malicious software. Therefore, a Cloud audit policy should allow to define: a) A certain file (or folder) may not be changed at all. Every single change should raise an action. b) Validation of a certain file containing a specific content. Latter is most important in config files, which affect security relevant configurations. The resulting policy could be: “File X may not be changed”.

TABLE II: Policy Example Scenarios

| No. | Policy |
|---|---|
| P1 | The VM is free of malware |
| P2 | File X has not been changes |
| P3 | Upscaling of VMs in VM cluster “WWW-Server” is only permitted if average requests per second $\geq$ Y |
| P4 | Port Z is open, allowed protocols: HTTP |
| P5 | Software X must (not) be installed |
| P6 | The VM does not contain any personal information |

*P3 - VM scalability:* One attribute of Cloud environments is flexibility and on-demand availability of resources. Depending on a currently existing demand additional VMs can be added to a certain service cluster (VM upscale) and when demand lowers, VMs can be decommissioned again (VM downscale). But this could also be misused by attackers to compromise the availability of a customer’s Cloud based infrastructure, by downscaling VMs during a high demand period. Contrary, unnecessary upscaling of VMs increases the running costs of a customer. The resulting policy could be: “Upscaling of VMs in VM cluster “WWW-Servers” is only permitted, if average requests per second $\geq$ N”.

*P4 - Technical attribute modelling:* Security is expressed if the infrastructure complies to certain technical attributes. A very simple rule defines the state of a network port. A port can be closed or open. Same can be used for allowed protocols. The resulting policies could be: “Port 80 is allowed to be open”, “Allowed network protocols: HTTP”

*P5 - VM content:* A VM contains software, such as an operating system and certain application software. To increase security by banning certain software products or specific versions of a software, to prevent data leakage or just to be compliant to existing software licenses it can be necessary to restrict the existence of software on a VM. The resulting policies could be: “Software X must (not) be installed”.

*P6 - Data traces:* In case of VM marketplaces, users prepare VM images and offer them at the marketplace (role: appliance creator). It is important, that these images don’t contain any personal information of the VM image creator, such as private key files or passwords, which could lead to a security breach. It is to validate that files are cleared e.g., history file, and critical information is securely wiped (and not be restorable anymore even with file carving tools). The resulting policies could be: “The VM does not contain any personal information”. Table II summarizes the elaborated policy scenarios.

This is just a brief description of the policy language requirements. A detailed description can be found in the bachelor thesis “Design and Development of a Security Policy Language for Automatic Cloud Audits” [17].

In addition to the specific functional requirements, there are several additional generic criteria to be considered when choosing a policy language. To model the scenarios described above the following features must be supported by a security policy language:

- *Monitored Objects* are any kind of entities in the Cloud infrastructure, which shall be monitored (e.g., hosts, virtual machines, files).
- *Logical Policy Operations* can be used to create more complex policies by combining them with logical operators such as *AND* and *OR*.
- *Policy Scoping* By grouping virtual machines or policies the process of creating and managing policies becomes easier. Also, incorporating the ability to define provider-only policies or policies, which can only be used by the provider, may prove to be beneficial.

**2.) Technological Support** Policies can be described using textual as well as graphical methods. However, the focus of SAaaS will be on a textual description of policies. All language candidates will be analyzed with regards to their technological basis, especially whether they build upon established standards such as XML and JSON or they introduce completely new language formats. Using widely accepted technologies may be beneficial because there already exist a lot of tools such as parsers, interpreters and comprehensive documentation. Custom language formats however can be tailored to the problem domain and might improve flexibility and readability. To ensure a fast adoption by developers and leverage the large amount of tools already available, XML should be the preferred language base.

**3. Development Activity Estimation** Release cycles of tools, the size of the developing community and the adoption of a language by other projects indicates a high development activity and is an indicator for a future proof implementation.

**4. Documentation Quality** A comprehensive documentation is essential for understanding and evaluating a policy language. The quality of the documentation is hereby defined by factors such as the logical structure, accessibility, profoundness and consistency.

**5. Complexity & Integratability in SAaaS** The target language should be complex enough the fulfill all requirements but also generalisable up to a certain point. Too much complexity will affect the ease of learning by cloud administrators and therefore indirectly and negatively influence the utilization of the language, which affects its overall success. Furthermore, it is essential to evaluate if the language can be integrated into the SAaaS architecture, presented later in this paper in Section V.

### B. Evaluation of Existing Policy Languages

In this subsection, policy language candidates are evaluated and compared for their suitability as a security policy language in SAaaS.

*REI* is an OWL based language, developed by Lalana Kagal in 2005. REI allows the definition of management, security, privacy and conversation policies [18]. These policies define the optimal behavior in a problem domain. A policy is hereby defined by the prohibition, permission or the obligation to perform an action on a target. The focus on semantic technology is not needed for SAaaS and introduces needless complexity. Additionally, REI has no practical relevance nor has it spread beyond a PhD thesis, which it was developed for.

*Common Information Model (CIM)* is a model to describe elements and the relationships between them (such as policies). It addresses most of the SAaaS requirements and would have been a suitable candidate. However, an implementation according to the CIM standard would have gone way beyond the requirements of the SAaaS project. Therefore, CAPL (Section IV-C) only uses parts of CIM for its implementation.

*Ponder* is a policy specification language, which already features tools and services for policy enforcement and evaluation. One of the main concepts behind Ponder is the general-purpose object management system and message passing paradigm [19]. Here the language is meant to be implemented in a way that the actual decision making process (deciding whether a policy is fulfilled or not) needs to be as close as possible implemented to its data source. In a Cloud scenario this would mean, that the decision engine needs to be implemented on each single machine. In addition to the proprietary language base Ponder Talk, this is a knock-out criteria for the usage of Ponder for our scenario.

*LaSCO* follows a graph based approach to define policies. Despite this being a rather interesting approach, it introduces a lot of unnecessary complexity. Additionally, the problem of conflict management is not addressed [20] and similarly to REI LaSCO has not spread beyond academic boundaries (one dissertation in [21]), which makes this language unsuitable for SAaaS.

*Evaluation Overview:* Besides the aforementioned languages WS-Trust, IDMEF, SSPL, PAX PDL, CADF and KAoS have also been evaluated. However, none of those languages have proven to be useful for the SAaaS approach, which is why we omit going into further detail. All language criteria are listed in the evaluation summary, depicted in Table III. All knock-out criteria (which do not fulfill our requirements introduced in Section IV-A) are displayed in bold red. It is shown, that none of the evaluated security languages fulfills the established requirements. As a result an own Cloud audit security policy language needed to be developed.

### C. Cloud Audit Policy Language (CAPL)

CAPL is an extension of the Cloud Infrastructure Management Interface. Core features like the object model, the protocol and a simplified variant of CIMI classes (e.g., Machine, MachineConfiguration, MachineImage) are inherited by CAPL. However, due to the different focus of CIMI on managing Cloud infrastructures some parts of CIMI have not been adopted in CAPL because they are not required for the SAaaS scenario. A detailed description of CAPL features is provided in the following.

*1) User Roles:* CAPL uses slightly simplified definitions of the CIMI roles *Cloud Provider* and *Cloud Consumer*. The Cloud Provider manages and provisions Cloud services and possesses full access rights. The Cloud Consumer uses cloud services as well as the service for auditing his virtual machines. The Cloud Consumer has a limited set of access rights, which are required to define policies and triggering audits.

*2) Service Interface:* CIMI uses a REST based protocol for communication. CAPL adopts the CIMI service interface.

TABLE III: Comparison of existing security policy languages

| **Evaluation Property** | **REI** Support | Details | **CIM** Support | Details | **Ponder** Support | Details | **LaSCO** Support | Details |
|---|---|---|---|---|---|---|---|---|
| Technological Base | ⊖ | OWL Lite | ⊕ | MOF & UML, XML through WBEM | ⊖ | **PonderTalk (SmallTalk)** | ⊖ | Directed graphs |
| Definition of monitored objects | ⊕ | Targets, user | ⊕ | PolicyInSystem | ⊕ | Managed Object in Subject, Action, Target Syntax | ⊕ | Knots |
| Combination of policies | ⊕ | Denotic objects | ⊕ | Conditions, PolicyRule, PolicySet | ⊕ | Obligation policy | ⊕ | Conjunctions |
| Area of validity | ⊕ | Constraints, groups of objects, a single assignment seems difficult | ○ | Unclear | ⊕ | Self managed cell | ⊕ | Domains |
| Conflict management | ⊕ | Priorities | ⊕ | Priorities | ⊕ | Yes | ⊖ | **Not implemented** |
| Last version | ○ | Updated 2005 | ⊕ | Currently revised (version 2013) | ⊕ | 2011 | ○ | 2000 |
| Acceptance | ⊖ | **Just a PhD thesis work, cited in different paper, no practical application** | ⊕ | Windows Management Instrumentation[22], SBLIM project[23], IBM[24] | ○ | Cited in multiple papers | ⊖ | **Only used in PhD dissertation [21] and paper [20] of LaSCO author** |
| Community support | ⊖ | **None** | ⊕ | None for CIM, but for specific implementations | ⊖ | **None** | ⊖ | **None** |
| Documentation | ○ | Rough description of classes[18], paper, presentations, Examples[25] | ⊕ | UML Diagram[26], Policy profile[27] | ○ | Good examples, but for old Ponder version | ○ | Only one PhD dissertation |
| Complexity | ⊖ | Long training period, complex & nested architecture | ⊖ | High, due to inconsistencies of different versions | ○ | Policies are human friendly readable, but developing own is difficult | ⊖ | Very complex due to its graph based origin |
| Support of SAaaS policy scenarios | ⊕ | Yes | ⊕ | Yes | ⊕ | Yes | ⊕ | Yes |
| Implementation effort | ○ | Unclear | ⊖ | **Complete CIM and WBEM implementation necessary** | ⊖ | Very high, since it brings its own agents | ⊖ | **Completely different base layer** |
| Integratability in SAaaS architecture | ○ | **Semantic of OWL not necessary** | ⊕ | Yes | ⊖ | **No, own agents necessary, different philosophy of policy evaluation** | ⊖ | **No, due to graph based nature** |

*3) Language Basics:* CAPL enhances CIMI by adding several new classes:

- Machine
  The Machine class represents a machine, which shall be audited. CIMI uses Machines only for virtual machines. However, CAPL enhances the scope of Machines and includes host machines running virtual machines because those might be as well targets for audits.
- MachineTemplate
  The MachineTemplate defines the initial configuration of a VM.
- Policy
  Defines a policy rule (e.g., "a virtual machine must not contain malware"), which can be assigned to a machine or a group.
- PolicySet
  A PolicySet contains multiple Policies. Only if all contained conditions of the rules are fulfilled, the PolicySet evaluates to success. A PolicySet may be used like a poliy and attached to machines or groups. Rules contained in a PolicySet may be linked disjunctive or conjunctive (using *AND*/*OR*). This behavior originates from the CIM policy model [28].
- Group
  Groups are used to manage related objects like multiple rules and machines. In such a case all rules of the group apply to all machines.
- RuleType
  RuleType describes what a policy is supposed to check and defines attributes and configurations, which the policy has to set.

An example, which depicts the key features of CAPL, is shown in listing 1. This Policy describes the conditions under which upscaling of Web server VMs is allowed. In this case, it is measured whether upscaling is allowed or not.

Listing 1: CAPL Example

```
<Group xmnls="http://research.cloud.hs-furtwangen.de/capl/">
<id>https://research.cloud.hs-furtwangen.de/CAPLPrototyp/groups/wwwCluster1</id>
<name>Cluster at Loadbalancer1</name>
<refName>WWWCluster1</refName>
<description>Group of web servers at loadbalancer1
</description>
<created>2013-03-03</created>
<updated>2013-03-03</updated>
<enabled>true</enabled>
<machines>
  <machine href="https://research.cloud.hs-furtwangen.de/CAPLPrototyp/rest/machines/6" />
  <machine href="https://research.cloud.hs-furtwangen.de/CAPLPrototyp/rest/machines/7" />
</machines>
</Group>

<Policy xmnls="http://research.cloud.hs-furtwangen.de/capl/">
<id>https://research.cloud.hs-furtwangen.de/CAPLPrototyp/policies/upscale</id>
<name>Upscale is Allowed</name>
<refName>upscaleCluster1</refName>
<description>Upscale is only allowed when requests on machines is higher than 10</description>
<created>2013-03-04</created>
<updated>2013-03-04</updated>
<enabled>true</enabled>
<ruleType href="https://research.cloud.hs-furtwangen.de/CAPLPrototyp/rest/ruleTypes/upscale"/>
<intervalType>no</intervalType>
<targetResource href="https://research.cloud.hs-furtwangen.de/CAPLPrototyp/rest/machines/5"/>
<attribute key="metric">requests per second</attribute>
<attribute key="threshold">10</attribute>
<attribute key="cluster">https://research.cloud.hs-furtwangen.de/CAPLPrototyp/groups/wwwCluster1
</attribute>
</Policy>
```

## V. SAaaS ARCHITECTURE

To support the presented use cases of a concurrent audit system for Cloud environments (see Section II), an agent system to monitor Cloud environments is proposed. Before explaining the SAaaS architecture and advantages of agents in detail, we briefly want to explain the whole Security as a Service event processing sequence. To support this, consider the following example.

### A. Scenario Used to Explain the SAaaS Architecture

A typical Web application architecture consisting of one or multiple Webserver(s) and a database backend is deployed at VMs in a Cloud. The VMs are logically grouped together to *WWW-Cluster1*. Initially, the Cloud customer's administrator installs the VMs with the necessary software, e.g., Apache web server, MySQL database. After the functional configuration security policies are modelled to describe the target infrastructure state, such as the policies P1 - P3 introduced in Table II. This can be:

A) Technical rules like allowed network protocols and connections between VMs, or that the web server configuration is finished and a notification should be sent if changes to its config files are detected. As a result of these policies software agents called `VM agents` are configured with the necessary tools to monitor these requirements and automatically deployed to the VMs.

B) Business flow related security policies can be created as well, such as a simple scalability policy: "If the cloud management systems gets an upscale event request for components of *WWW-Cluster1*, first the actual load of all web servers needs to be checked. If the average load over all web servers is not higher than a certain threshold, e.g., 100 http connections / Web server, the upscaling gets denied and an alarm gets raised since the event must originate from a systems failure or a successful hacker attack at the Cloud management system. The same scenario works for downscaling in an inverse manner: If a downscaling event for *WWW-Cluster1* gets detected, but the actual load is above a downscale threshold, an alarm gets raised. Here, software agents are additionally located at the Cloud management system and the scaling service.

### B. Cloud Audits Using Agents

To generalize this scenario: in the SAaaS architecture a modelled security state of certain components gets monitored by agents, which are deployed at the specific resource, e.g., the Cloud management system (CMS), a Cloud host or a VM.

An agent can be defined as [29]:
"... a software entity, which functions continuously and autonomously in a particular environment ... able to carry out activities in a flexible and intelligent manner that is responsive to changes in the environment ... Ideally, an agent that functions continuously ... would be able to learn from its experience. In addition, we expect an agent that inhabits an environment with other agents and processes to be able to communicate and cooperate with them ..."

Since the agents in the SAaaS architecture are running independently, not necessarily connected to a certain central instance agents can receive data from other instances (e.g., the policy module) and send information to other instances like other SAaaS agents or the SAaaS' event processing system. The "central" event processing system gets itself implemented as an agent, which can be scaled and distributed over multiple VMs.

### C. Type of Agents

Agents collecting data are called `Sensor Agents`. If the location of an agent needs to be expressed, they are also titled as `VM Agent` (agent running at a VM), `Host Agent` (agent running on a Cloud host monitoring, the hypervisor) or `CMS Agent` (agent monitoring the Cloud management system). Specific targeted security audits perform specific checks of systems, which are affected by a change within the Cloud environment. These checks are performed automatically by so-called `Audit Agents`. The threshold values to be checked are defined as metrics and get checked in case of an event by `Metric Agents`. Changes to the Cloud infrastructure get detected by sensor agents before sent out the central event processing unit preprocessed and aggregated by a `Event Aggregator Agent`, which also runs on same location as the sensor agent, e.g., a VM. This is important to reduce the overall messages sent to the global Cloud event processing

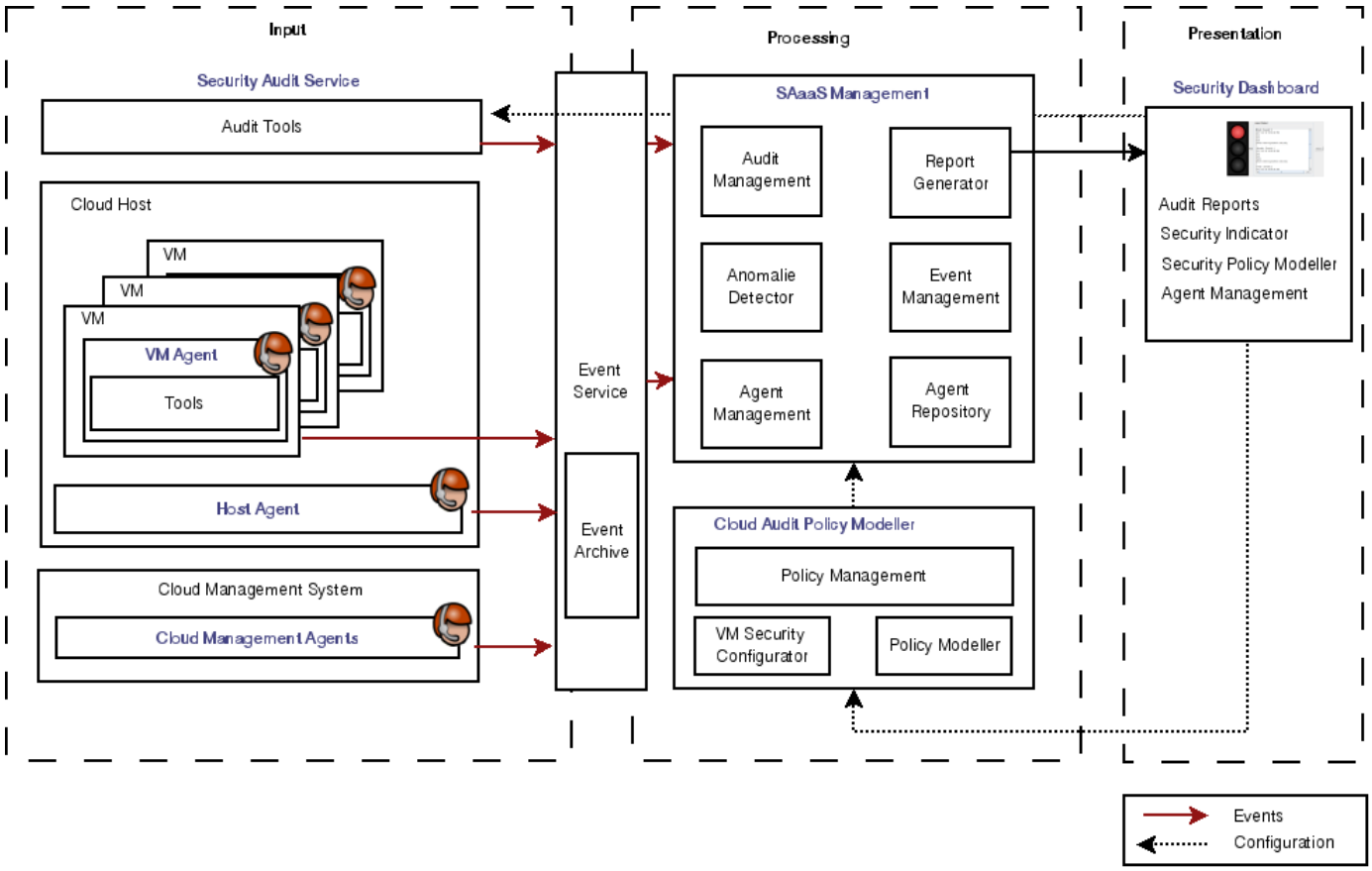


Fig. 3: SAaaS event processing sequence

system especially in large Cloud computing environments. The `Event Aggregator Agent` filters out possible VM dependent events like a started Web application session from *IP 1.2.3.4*. A more abstracted event gets send to the Cloud event processing system to detect (possible) user overlapping security incidences. This could be a message containing the number of not completed Web shop transactions by IP *IP 1.2. 3. 4* to pre-detect a Denial of Service attack.

### D. SAaaS Event Processing

Figure 3 gives a high level overview how events are generated, preprocessed, combined and forwarded within the SAaaS architecture. It can be divided into three logical layers: *Input*, *Processing* and *Presentation*.

*Input:* The SAaaS architecture gets its monitoring information from distributed agents, which are positioned at key points of the cloud's infrastructure to detect abnormal activities in a Cloud environment. Possible key points are: running VMs of Cloud users, the VM hosting systems (Cloud hosts), data storage, network transition points like virtual switches, hardware switches, firewalls, and especially the Cloud management system. A `VM agent` integrates several monitor and policy enforcing tools. Therefore, it loads necessary VM agent plug-ins to interact with stand-alone `tools` like process monitor, intrusion detection system or anti virus scanner. It gets installed on a VM likewise on a Cloud host. A logging component is recording the chronological sequence of occurrences building `audit trails`.

*Processing:* Each SAaaS agent receives security policies from the security policy modeller component. Through security policies each agent gets a rule set (its intelligence) specifying actions in case of a specific occurrence (e.g., modification of a frozen config file). Thus, every occurrence gets first preprocessed by an agent, which reduces communication between VM agents and Cloud management agent. The `Cloud Audit Policy Modeller` consists of a policy editor and a VM security configurator. An example of a Cloud specific security policy could be: "In case of a successfully detected rootkit attack on a VM running on the same Cloud as a users VM, the user VM gets moved to a different host to diminish the risk of further damage." whereas a security configuration could state: "In case a modification attempt of a file within / etc/php5/ gets detected, deny it and send an email to the Cloud administrator." Cloud audit policies get send from the Policy Management to the Agent Management to configure the corresponding agents. By using the monitoring information of the distributed agents in combination with the security policies a `Cloud behaviour model` is built up for every Cloud user. Cloud audit policies are also used as input for the Cloud management agent to detect user overlapping audit events. Forwarded higher level events are processed by an event processing engine. It is also fed with the modelled security flows from the `Security Policy`

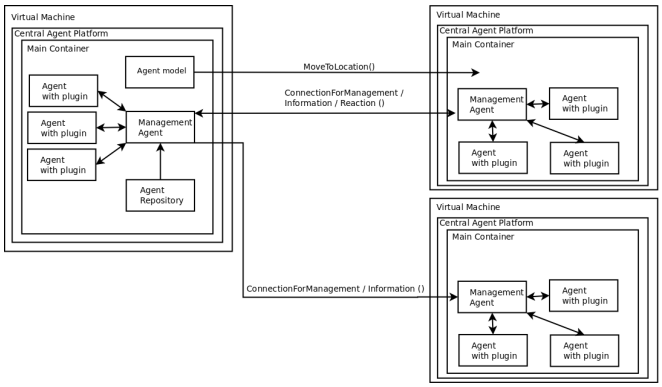


Fig. 4: Basic SAaaS agent design

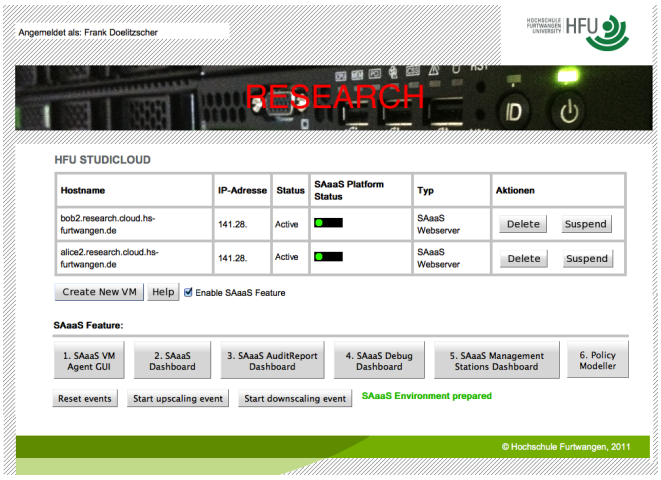


Fig. 5: HFU Cloud management interface

`Modeller` to aggregate information and detect behaviour anomalies. Countermeasures can then be applied to early detect and prohibit security or privacy breaches. The `Report Generator` conditions events, corresponding security status as well as audit report results in a human friendly presentation.

*Presentation:* As a single interaction point to Cloud users the `Security Dashboard` provides usage profiles, trends, anomalies and Cloud instances' security status (e.g., patch level). Information are organized in different granular hierarchies depending on the information detail necessary. At the highest level a simple three colour indicator informs about a users Cloud services overall status. It also provides a graphical user interface to deploy agents to Cloud instances. Figure 5 shows a part of the security dashboard prototype, which gets described in more detail in the next (but one) Section.

Communication between the distributed agents and the security dashboard is handled by an `Event Service`. Events will use the Agent Communication Language Format (ACL) [30] and are exchanged using a FIPA[31] compliant HTTPS Message Transfer Protocol (HTTPS-MTP) [32]. Events are also stored in an `Event Archive`.

### *E. How agents can improve incident detection*

Incident detection in Cloud environments is a non trivial task due to its characteristics as discussed in Section I. Therefore, it is important to have a high number of sensors capturing simple events. Preprocessed and combined complex events can be generated reducing the possibility of "event storms". Combined with knowledge about business process flows (specified in security policies), it will be possible to detect security incidents while keeping the network load low. The usage of agents delivers this possibility because agents are *independent units* that can be added, removed or reconfigured during runtime without altering other components. Thus, the amount of monitoring entities (e.g., network connections of a VM, running processes, storage access, etc.) of a Cloud instance can be changed without restarting the incident detection system. Simultaneously using agents can *save computing resources* since the underlying business process flow can be taken into account.

While single `sensor agents` can monitor simple events (e.g., user login on VM) and share them with other agents *complex events* can be detected. Given the scenario of a successful unauthorized login of an attacker at a virtual machine `VM2`, misusing a web server's directory to deposit malicious content for instance a trojan. Agent `A1` monitors the user login, agent `A2` detects the change of a directory content and agent `A3` detects a download of a not known file (the trojan). Instead of sending those three simple messages to a central event processing unit a `VM agent` can collect them, conditioning one higher level event message that `VM2` was hijacked. This can result in a predefined action by the `Cloud Management Agent` e.g., moving a hijacked VM into a quarantine environment, alerting the user and simultaneously starting a fresh instance of `VM2` based on its VM image.

By ordering agents in a hierarchical structure and preprocessing of detected events reduces network load originated from the incidents detection system. Furthermore, this makes the system more scalable by reducing data sent to upper system layers. This concept is introduced and successfully used in [33]. Combining events from system deployed agents (e.g., VM agents, host agent) and infrastructure monitoring agents (network agent, firewall agent) incident detection is not limited to either host or network based sensors, which is especially important for the characteristics of Cloud environments. Using agents has advantages in case of a system failure. Agents can monitor the existence of co-located agents. If an agent stops for whatever reasons this stays not undetected. Concepts of asymmetric cryptography or Trusted Platform Module (TPM) technology can be used to guarantee the integrity of a (re-)started agent. If an agent stops the damage is restricted to this single agent or a small subset of connected agents, which are requiring information from this agent.

### *F. SAaaS Agent Architecture Implementation*

For the SAaaS architecture we evaluated existing agent frameworks with the following requirements:

- Agents can be deployed, moved, updated during runtime
- Agent performance
- Open Source, documentation, community support

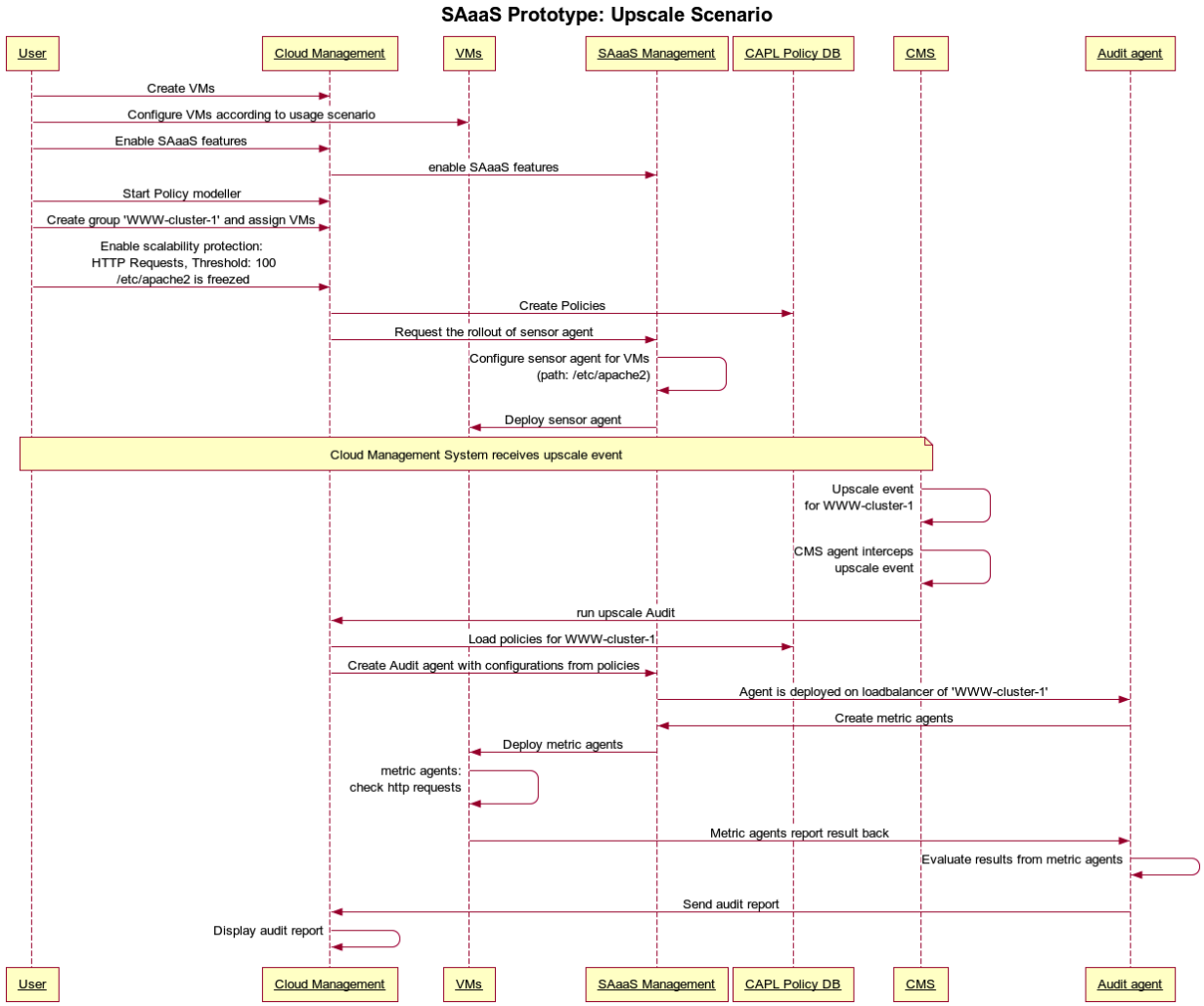


Fig. 6: Automatic security audit in case of upscale event

Since our Cloud environment at HFU's Cloud Research Lab CloudIA [34] is built around the Cloud management system Open Nebula another requirement was the agent programming language: Java. As a result we choose the Java Agent DEvelopment Platform (JADE), which enables the implementation of multi-agent systems and complies to FIPA specifications. Figure 4 illustrates a basic agent architecture. It shows three VM Sensor Agents. Agents live in an agent platform, which provides them with basic services such as message delivery. A platform is composed of one or more Containers. Containers can be executed on different hosts thus achieving a distributed platform. Each container can contain zero or more agents [35]. To provide monitoring functionality a VM agents interacts through agent plugins with stand-alone tools like process monitor, intrusion detection system or anti virus scanner, as depicted in Figure 4. To harness the potential of Cloud computing an agent can be deployed to a VM on-demand according to the policies a user defines. Different agents based on modelled business processes are stored within an agent repository. To be able to move a JADE agent to a running Cloud instance the Inter Platform Mobility Service (IPMS) by Cucurull et al. [36] was integrated. This supports the presented advantage of deploying agents on-demand if a designed business process flow was started (as described in Section V-E).

### G. CAPL Integration & SAaaS Prototype

The described Cloud audit policy language, presented in Section IV is seamlessly integrated into the SAaaS architecture. To show how SAaaS can increase transparency in Cloud environments the following prototype scenario was implemented. It is also depicted in Figure 6 and describes a whole SAaaS life cycle:

1) A Cloud user creates three new VMs on the Web based Cloud management interface, depicted in Figure 5. The VMs get configured as a typical Web application installation: two Web servers, which are delivering content from one database server.
2) After VM configuration is finished the user enables the SAaaS features, starts the security policy modeller and groups the VMs into group *"WWW-Cluster1"*.

Then he activates a scalability monitoring policy with a metric of HTTP requests per second and a threshold of 100 for *WWW-Cluster1*. Furthermore, he creates a policy saying that the *"/etc/apache2"* config directory should be considered frozen and therefore be monitored for changes.

3) As a result from enabling the SAaaS features and the policy creation, a sensor agent for filesystem monitoring gets deployed to the VMs of *"WWW-Cluster1"*. It utilizes the linux tool *inotify* [37] to watch the *"/etc/apache2"* directory.
Now let's assume there is a lot of load on the web servers due to a product launch of the user's company. Therefore, the Cloud management system gets an upscale event for *WWW-Cluster1*, which gets intercepted by a SAaaS agent monitoring the CMS.
4) The event provokes the policy and configures a scalability audit agent with a scalability check and deploys it on the SAaas management VM (in the case of a filesystem change event within a web server VM the audit agent would have been deployed to that particular VM).
5) The agent creates new metric agents, which get deployed to the web server VMs of *WWW-Cluster1* to check the current load of HTTP requests. They report the result back to the audit agent. The audit agent evaluates the result and decides, dependent on the average load reported by the metric agents, if the upscale event is okay or not.
6) The results get conditioned into an audit report.

As a first prototype, a two layered agent platform was developed, consisting of a `sensor agent` running inside a VM and a `Cloud management system agent`. Audit reports get displayed in a `Security Dashboard`. Since all Cloud VMs in CloudIA are Linux based, only Open Source Linux tools were considered during our research. Two notification mechanisms were implemented:

a) The tool sends agent compatible events directly to the agent plugin.

b) The tool writes events in a proprietary format into a logfile, which gets parsed by an agent plugin.

As for mechanism a) the filesystem changes monitoring tool *inotify* was used, whereas for mechanism b) *fail2ban* [15], an intrusion prevention framework was chosen. For demo purposes a simple Web frontend was written, which offers to launch several attack scenarios on a VM agents equipped VM in CloudIA. Before/after tests were performed to validate that an attack was detected and (depending on the plugin's configuration) prohibited. A prototype version of the security dashboard, depicted in Figure 7 showing a signal light indicator informed about occurring events. When started, it shows a green light. After launching an attack, the security dashboard indicator light changes its colour to yellow or red. The impact of a monitored event is defined by a severity matrix, shown in Table IV. Each severity value out of the Web server log file gets associated with a certain score. This score gets summed up for all events. Then the quotient gets calculated which is directly connected to the resulting colour.

TABLE IV: Severity matrix for security indicator light

| Severity | Value of message | Quotient | Colour |
|---|---|---|---|
| info | 0 | $< 1.4$ | green |
| low | 1 | $<= 1.4$ | green |
| middle | 2 | $> 1.4 < 2.5$ | yellow |
| high | 7 | $>= 2.6$ | red |

### H. VM Automatic Audit Integration & SAaaS Prototype

As described in Section III especially for a public Cloud it is necessary to audit VMs. Every time a VM is uploaded to the Cloud an image audit agent is checking the image according to the enterprise policies defined in CAPL.

## VI. Evaluation

This section evaluates the presented automatic audit system approach of the SAaaS system. First, it is discussed how SAaaS enhances the auditing of non-running virtual machine images. Therefore, an evaluation scenario from a Cloud customer's perspective is presented. We describe necessary user effort utilizing the presented automatic audit system and if he would take a manual approach. Second, it is evaluated how the presented Cloud audit policy language supports the established requirements, introduced in Section IV-A. Finally, it is elaborated how the presented concurrent audits of the SAaaS system address Cloud specific security issues and therefore enhance Cloud computing security and transparency.

### A. Auditing of VM Images

To evaluate the automatic audit system the following scenario is considered: A Cloud customer chooses an online shop virtual appliance (containing a web server and a database) out of a cloud's store. Before transferring actual data to the image the following security policies need to be evaluated: P1: The image must not contain any malware. P2: The image must not run any other software than the web server and the database. P3: The web server and database connection must be configured properly.

*1) Customer uses Automatic Audit System Approach:* When using the automatic audit system, the appliance user or appliance creator first describes his security policies. Since malware checks are usually a default policy, provided by the Cloud provider, there is no need to model those. Additionally, to simplify black box scans of the proper interaction between web server and database it is imaginable to deposit a predefined default start page, which could be browsed. The automatic audit system will parse the security policies and identifies the necessary audit cases, which are fetched from the database. The audit cases get sorted, dependent on how the checks can be executed. There are two kinds of security audit modes. *Offline VM audits* mount the VM's image and perform audit tasks on it, whereas *online VM audits* launch the VM in a quarantine environment of the cloud. There, audit tasks, which can be only performed on the running VM are executed, such as an analysis of open ports. Again, the results of the single audit cases will be submitted to the parser and saved as mini reports in the audit system's database.

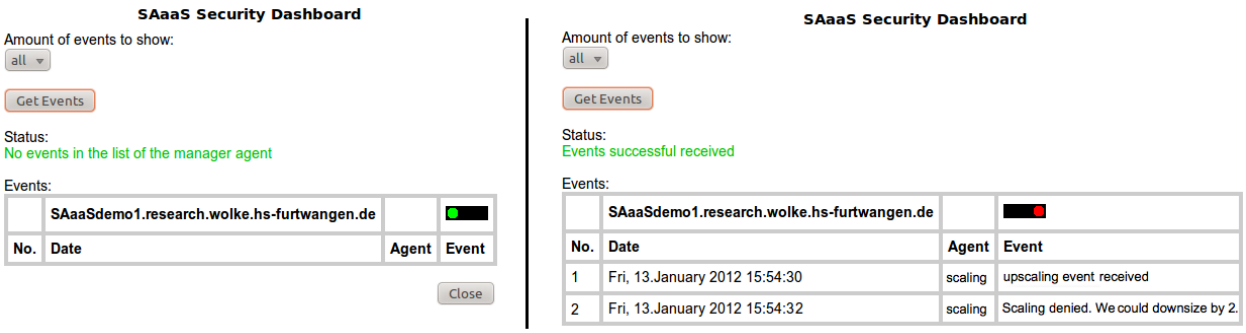


Fig. 7: Cloud security dashboard prototype

At last, the report generator conditions the results of all mini reports. The cloud management system is informed if the overall audit result is *"passed"* or *"failed"*. If the status is passed, the image can be added to the store, otherwise the release will be denied. Nevertheless of the result, the complete audit report will be sent to the appliance creator, to inform about necessary problems to be fixed.

*2) Manual Approach:* In contrast to the automatic audit system an offline audit of the appliance's image is not immediately possible. This is due to the fact, that the appliance user does not have direct access to images stored in the appliance store. The only two approaches possible are downloading the appliance's image, which enables offline auditing or to limit the audit process to online auditing. This is done by an administrator, who must have sufficient expertise in virtualization technologies, auditing methods and must be an audit tool expert. Additionally, for the sake of reproducibility and documentation, the appliance user has to follow a very well defined auditing process (assuming such a process exists). Downloading VM images and evaluating them offline imposes a significant network overhead on the appliance user as well as the Cloud provider. Manual online auditing can be performed, when a virtual appliance image is already started. The appliance user has to log in to the appliance in order to execute auditing tools and scripts. Additionally, the virtual appliance also has to be checked externally to determine which services are activated, for example by port scanning. Performing port scans on virtual machines executed in the cloud may trigger the Cloud provider's intrusion detection systems or may even be prohibited entirely by the Cloud provider's terms of use.

This demonstrates the overall complexity of a manual virtual appliance auditing process. The automatic audit system delivers the following advantages for appliance creator/user and Cloud provider: Improved security when using 3rd party virtual appliances (appliance user), well documented and formalized audit process (all), customizable, machine-readable audit policies (all) and additional revenue by offering audits as a service (Cloud provider).

## B. CAPL Evaluation

Because none of the evaluated languages fulfilled the requirements of SAaaS on a policy language (e.g., missing conflict management, combination of policies), while retaining a reasonable complexity, CAPL was developed, which specifically addresses all those requirements. CAPL is based on XML and extends the Cloud Infrastructure Management Interface CIMI by a definition of security policies. Cloud providers as well as Cloud users are enabled to define policy rules for virtual machines. Table V evaluates CAPL against the requirements, established in Section IV-A. By staying closely to the CIMI standard, it will be possible to define security policy for any CIMI compatible cloud infrastructure. This also increases the compatibility of the proposed SAaaS system. Another advantage of CAPL is its simplicity, since it is tailored to the SAaaS target scenario. However, developing a new policy language also has some negative aspects. Our results are used in the SAaaS project only, which leads to a rather poor acceptance. Also, besides the SAaaS project members there is no community surrounding and developing this language.

## C. Cloud Specific Security Issues Addressed by SAaaS

The German Federal Office for Information Security publishes the IT baseline protection catalogues enabling enterprises to achieve an appropriate security level for all types of information. In a comprehensive study [38] on all IT baseline protection catalogues as well as current scientific literature available [2][39][40][4][5], we identified the following Cloud specific security issues as solvable by the presented SAaaS system:

*Abuse of Cloud resources*
Cloud computing advantages are also used by hackers, enabling them to have a big amount of computing power for a relatively decent price, startable in no time. Cloud infrastructure gets used to crack WPA, and PGP keys as well as to host malware, trojans, software exploits used by phishing attacks or to build botnets like the Zeus botnet. The problem of malicious insiders also exists in classical IT-Outsourcing but gets amplified in Cloud computing through the lack of transparency into provider process and procedure. This issue affects authorisation, integrity, non-repudiation and privacy. Strong monitoring of user activities on all Cloud infrastructure components is necessary to increase transparency. The presented SAaaS use case A) Monitoring and audit of Cloud instances addresses this problem.

*Missing security monitoring in Cloud infrastructure*
Security incidents in Cloud environments occur and (normally) get fixed by the Cloud provider. But to our best knowledge no Cloud provider so far provides a system, which informs user

TABLE V: Evaluation of CAPL

| Requirement | CAPL Supp. | Details |
|---|---|---|
| Technological Base | ⊕ | XML |
| Definition of monitored objects | ⊕ | Tailored to problem domain |
| Combination of policies | ⊕ | Groups |
| Area of validity | ⊕ | CIMI compatible infrastructures |
| Conflict management | ⊕ | Included |
| Last version | ○ | Under active but internal development |
| Acceptance | ⊖ | Not spread beyond SAaaS |
| Community support | ⊖ | Only SAaaS |
| Documentation | ○ | CAPL documentation [17] |
| Complexity | ⊕ | Tailored to problem domain |
| Support of SAaaS policy scenarios | ⊕ | Full |
| Implementation effort | ○ | Own development |
| Integratability in SAaaS architecture | ⊕ | Fully integratable |

promptly if the Cloud infrastructure gets attacked, enabling them to evaluate the risk of keeping their Cloud services productive during the attack. Thereby the customer must not necessarily be a victim of the attack, but still might be informed to decide about the continuity of his running Cloud service. Furthermore, no Cloud provider so far shares information about possible security issues caused by software running directly on Cloud host machines. In an event of a possible 0-day exploit in software running on Cloud hosts (e.g., hypervisor, OS kernel) Cloud customers blindly depend on a working patch management of the Cloud provider. The presented SAaaS use case B) Cloud infrastructure monitoring and audit addresses this problem.

*Defective isolation of shared resources*

In Cloud computing isolation in-depth is not easily achievable due to usage of rather complex virtualization technology like VMware, Xen or KVM. Persistent storage is shared between customers as well. Cloud providers advertise implemented reliability measures to pretend data loss like replicating data up to six times. In contrast, customers have no possibility to prove all these copies get securely erased in case they quit with the provider and this storage gets newly assigned to a different customer. While the presented SAaaS architecture does not directly increase isolation in-depth it adds to the detection of security breaches helping contain its damage by the presented actions.

## VII. State of the art - Related Work

To put the presented work described in this paper into perspective, this section first discusses related research work on Cloud security issues, followed by other Cloud security research projects in contrast to SAaaS and the usage of agents to increase security. It then discusses related work regarding security of virtual appliance images. Afterwards, work on security policy languages is elaborated, which are an important part of the proposed work in this paper, too.

A rather high-level, but comprehensive view on the whole topic of Cloud computing security is given by Mather et al in the book "Cloud Security and Privacy" [41]. It provides a very good introduction especially by laying out the necessary groundwork. The most comprehensive survey about current literature addressing Cloud security issues is given by Vaquero et al. in [4]. It categorizes the most widely accepted Cloud security issues into three different domains of the Infrastructure as a Service model: machine virtualization, network virtualization and physical domain. It also proposes prevention frameworks on several architectural levels to address the identified issues.

Pearson [42] proposes several software design guidelines for delivering Cloud services taking privacy into account, such as using a privacy impact assessment, allowing user choice and providing feedback. While Chen et al. state in [5] that many IaaS-related Cloud security problems are problems of traditional computing solved by presented technology frameworks it also demands an architecture that enables "mutual trust" for the Cloud user and Cloud provider. Both papers confirm and complete the list of Cloud specific security issues identified by previous members of our research group, presented in [43]. Furthermore, they identified a demand for a two-way trust enabling architecture for Cloud infrastructures and the ability of "choosable security primitives with well considered defaults" [5]. The SAaaS architecture, proposed in this paper is targeted to provide this mutual trust. SAaaS' security audit policy language enables the user to define its own security policy and to choose from a spectrum of security subsystems as demanded by [5].

Zamboni et al. present in [44] show traditional Intrusion Detection Systems (IDS) can be enhanced by using autonomous agents. They confirm the advantages of using autonomous agents in regards to scalability and system overlapping security event detection. In contrast to our SAaaS architecture their research is focusing on the detection of intrusions into a relatively closed environment whereas our work applies to an open (cloud) environment where incidents like abuse of resources needs to be detected. Mo et al. introduce in [45] an IDS based on distributed agents using the mobile technology. They show how mobile agents can support anomaly detection thereby overcoming the flaws of traditional intrusion detection in accuracy and performance. The paradigm of cooperating distributed autonomous agents and its corresponding advantages for IDS' is shown by Sengupta et al. in [46]. The presented advantages apply for our SAaaS agents as well [1].

Amazon's Cloud platform Elastic Compute Cloud (EC2) allows users to create and share virtual machine images. Balduzzi et al. [11] analyzed the security risks of running third party images. The work gives a good insight about the current risk, which comes from pre-configured Cloud appliances. After the investigation of over 5000 Amazon Amazon Machine Images (AMIs), they found that 98% of Windows AMIs and 58% of Linux AMIs contain software with critical vulnerabilities [11]. Furthermore, two VM images were infected with malware, two were configured to write logs to an external machine, 21.8% contained leftover credentials that would allow a third party to remotely log into a machine [11].

Their approach is to start AMIs on EC2 and then scan them for security problems and privacy risks. However, their system is not intended to be used as an auditing service. They execute a predefined set of security and privacy checks and provide no way of customizing policies, which will be supported by the work proposed in this paper.

Wei et al. have worked on the problem of securing virtual machine images [47], and propose the *Mirage Image Management System* as a mitigation solution. Mirage is run by the Cloud provider to secure virtual machine images in his repositories. Mirage incorporates an access control framework and image filters, which remove unwanted information automatically from an image on publish/retrieve time. It incorporates a change tracking system and repository maintenance services, like periodic virus scanning. However, providing the appliance user and creator with a facility for easily creating custom policy definitions is not part of their work.

Schwartzkopf et al. present an "Update Checker", which investigates the up-to-dateness of installed software within VM images before they get launched in a Cloud environment [48]. They provide a graphical alerting method within the Cloud management system to inform the user about outdated software within a certain image. However, their approach only focuses on Linux images so far, which are being mounted and only checks for outdated software. The automatic audit system proposed in this work supports a wider spectrum of security or privacy checks through the concept of flexible security policies and offline and online investigation of VMs. However, the work proposed in [48] could be used to optimize the offline checks proposed in this paper.

Al Morsy and Faheem identified the need for automated policy enforcement systems [49]. Although, a lot of different security policy definition languages exist (e.g., LaSCO, XACML, SPARQL, etc.), it is shown that each of those has different limitations in terms of policy constraints. Therefore, Morsy and Faheem propose a policy automation framework including a new language called Standard Security Policy Language (SSPL), which tries to simplify the process of creating machine-readable security policies. The results of their policiy language analysis will be confirmed by the security policy language evaluation presented in this work. We further show why the developed Standard Security Policy Language does not meet the requirements of our work.

## VIII. Conclusion & Outlook

In this paper, we introduced the Security Audit as a Service architecture to mitigate the shortcomings traditional audit systems suffer to audit Cloud computing environments. It was shown that SAaaS provides automatic auditing of virtual machine images according to custom user-defined policies. The results are reduced security and privacy risks, a well defined and reproducible audit process, as well as good documentation of results, when using 3rd party virtual machine appliances, while keeping the required technical understanding of the audit process to a reasonable minimum. The description of audit requirements in a Cloud audit policy language, allows abstraction of the audit requirements to a level, where even non-technical experts should be able to transfer company security and privacy policies into audit policy documents. A prototype was presented where user enable an automatic evaluation of an upscaling event. Furthermore, the advantages of using agents as a source for sensor information were shown. By utilizing lightweight, on-demand deployable agents it is possible to perform specific targeted audits every time a change within a user's Cloud infrastructure is performed. The current status of the work implements this on a user basis, but the system will be even more valuable when customer-overspanning events will be evaluated.

Therefore, as for future work an anomaly detection module will be developed, which is targeted to learn "normal" usage behaviour of Cloud instances by their users. As a result, this will enable the SAaaS system to detect anomalies within a Cloud infrastructure. For the presented Cloud audit policy language CAPL the extension of the CAPL schema by CIMI required objects is planned. This will enable the SAaaS system to audit any CIMI compatible infrastructures and the SAaaS system will be Cloud provider interoperable. Also, the implementation of a graphical user interface for the security policy modeller is planned. Thus, user will be able to easily define policies. Furthermore, it is planned to extend the CAPL class `MetaResource`, which provides a functions catalog of CAPL policies. User then could access this catalog to get automatically information about necessary parameters of a policy they like to define. At last, a security evaluation of the SAaaS system is planned to prove if it imposes new security risks to a Cloud environment.

## Acknowledgment

This research is supported by the German Federal Ministry of Education and Research (BMBF) through the research grant number 01BY1116.

## References

[1] F. Doelitzscher, C. Reich, M. Knahl, and N. Clarke, "Incident Detection for Cloud Environments," in *Proceedings of the Third International Conference on Emerging Network Intelligence (EMERGING 2011)*, no. 978-1-61208-174-8, 2011, pp. 100–105.

[2] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing v2.1," 12 2009.

[3] European Network and Information Security Agency, "Cloud Computing Security Risk Assessment," Tech. Rep., 2009.

[4] L. Vaquero, L. Rodero-Merino, and D. Moran, "Locking the Sky: A Survey on IaaS Cloud Security," *Computing*, vol. 91, pp. 93–118, 2010.

[5] Y. Chen, V. Paxson, and R. H. Katz, "What's New About Cloud Computing Security?" EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-5, 01 2010.

[6] Cloud Industry Forum, "Cloud Adoption and Trends for 2013," vol. 08, 2013.

[7] F. Doelitzscher, C. Reich, and A. Sulistio, "Designing Cloud Services Adhering to Government Privacy Laws," in *Proceedings of 10th IEEE International Conference on Computer and Information Technology (CIT 2010)*, 2010, pp. 930–935.

[8] OpenNebula. (2012, July) OpenNebula Marketplace. [Online]. Available: http://marketplace.c12g.com/appliance-Accessed:10.06.2013

[9] Amazon. (2012, July) AWS Marketplace. [Online]. Available: http://aws.amazon.com/marketplace-Accessed:10.06.2013

[10] Amazon Web Services. (2012, July) How To Share and Use Public AMIs in A Secure Manner. [Online]. Available: http://aws.amazon.com/articles/0155828273219400-Accessed:10.06.2013

[11] M. Balduzzi, J. Zaddach, D. Balzarotti, E. Kirda, and S. Loureiro, "A Security Analysis of Amazon's Elastic Compute Cloud Service," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, ser. SAC '12. New York, NY, USA: ACM, 2012, pp. 1427–1434.

[12] S. Bugiel, S. Nürnberger, T. Pöppelmann, A.-R. Sadeghi, and T. Schneider, "Amazonia: When elasticity snaps back," in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 389–400.

[13] N. A. Haroon Meer. (2009) Clobbering the Cloud, part 4 of 5. [Online]. Available: http://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-sensepost-clobbering_the_cloud.pdf-Accessed:10.06.2013

[14] N E Case. (2012, July) Extundelete. [Online]. Available: http://extundelete.sourceforge.net-Accessed:10.06.2013

[15] WinRecovery. (2012, July) Winundelete. [Online]. Available: http://www.winundelete.com-Accessed:10.06.2013

[16] Distributed Management Taskforce Inc. (DMTF). Cloud Infrastructure Management Interface (CIMI). Accessed: 10.06.2013. [Online]. Available: http://www.dmtf.org/standards/cloud-Accessed:10.06.2013

[17] T. Karbe, "Design and Development of an Audit Policy Language for Cloud Computing Environments," Cloud Research Lab - University of Apllied Sciences Furtwangen, Tech. Rep., 2013. [Online]. Available: http://wolke.hs-furtwangen.de/publications/theses

[18] L. Kagal. (2004) Rei Ontology Specifications. [Online]. Available: http://www.csee.umbc.edu/~lkagal1/rei/-Accessed:10.06.2013

[19] Imperial College London. (2013, March) Ponder2. [Online]. Available: http://www.ponder2.net-Accessed:10.06.2013

[20] J. A. Hoagland, R. Pandey, R. P, and K. N. Levitt. A Graph-based Language for Specifying Security Policies.

[21] James A. Hoagland, "Specifying and Implementing Security Policies Using LaSCO, the Language for Security Constraints on Objects," Ph.D. dissertation, University of California Davis, 2000. [Online]. Available: http://seclab.cs.ucdavis.edu/projects/arpa/LaSCO/dis/dissertation.pdf

[22] MS TechNet. (2004, 09) Windows Management Instrumentation. [Online]. Available: http://www.microsoft.com/germany/technet/datenbank/articles/600682.mspx-Accessed:10.06.2013

[23] SBLIM. (2009) SBLIM Project Wiki. [Online]. Available: http://sourceforge.net/apps/mediawiki/sblim/index.php?title=MainPage-Accessed:10.06.2013

[24] IBM. IBM Director. [Online]. Available: http://publib.boulder.ibm.com/infocenter/eserver/v1r2/index.jsp?topic=%2Fdiricinfo%2Ffqm0ccommoninfomodel-Accessed:10.06.2013

[25] L. Kagal. (2004) Rei Examples. [Online]. Available: http://www.csee.umbc.edu/~lkagal1/rei/examples/univ/-Accessed:10.06.2013

[26] DMTF Policy Working Group. CIM Schema Final Documentation. [Online]. Available: http://dmtf.org/sites/default/files/cim/cim_schema_v2340/cim_schema_2.34.0Final-Doc.zip-Accessed:10.06.2013

[27] Distributed Management Taskforce Inc. (DMTF). (2007, 02) Policy Profile. [Online]. Available: http://www.dmtf.org/sites/default/files/standards/documents/DSP1003.pdf-Accessed:10.06.2013

[28] ——. Cim schema - policy model. [Online]. Available: http://www.wbemsolutions.com/tutorials/CIM/cim-model-policy.html-Accessed:10.06.2013

[29] J. M. Bradshaw, *An Introduction to Software Agents*. Cambridge, MA, USA: MIT Press, 1997, pp. 3–46.

[30] Foundation for Intelligent Agents. (2002) Fipa acl message structure specification. [Online]. Available: http://www.fipa.org/specs/fipa00061/SC00061G.html-Accessed:10.06.2013

[31] IEEE Computer Society standards organization. Founddation for Intelligent Physical Agents - FIPA. http://www.fipa.org/. [Online]. Available: http://www.fipa.org/-Accessed:10.06.2013

[32] JADE Tutorials. HTTP MTP for Inter-Platform Communication. [Online]. Available: http://jade.tilab.com/doc/tutorials/JADEAdmin/HttpMtpTutorial.html-Accessed:10.06.2013

[33] S. Staniford-chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagl, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "GrIDS - A Graph Based Intrusion Detection System For Large Networks," in *In Proceedings of the 19th National Information Systems Security Conference*, 1996, pp. 361–370.

[34] A. Sulistio, C. Reich, and F. Doelitzscher, "Cloud Infrastructure & Applications - CloudIA," in *Proceedings of the 1st International Conference on Cloud Computing (CloudCom'09)*, Beijing, China, 2009.

[35] David Grimshaw. JADE Administration Tutorial. [Online]. Available: http://jade.tilab.com/doc/tutorials/JADEAdmin-Accessed:10.06.2013

[36] J. Cucurull, R. Marti, G. Navarro-Arribas, S. Robles, B. Overeinder, and J. Borrell, "Agent mobility architecture based on IEEE-FIPA standards," *Computer Communications*, vol. 32, no. 4, pp. 712 – 729, 2009.

[37] inotify. - monitoring file system events. [Online]. Available: http://linux.die.net/man/7/inotify-Accessed:10.06.2013

[38] F. Doelitzscher and M. Ardelt and M. Knahl and C. Reich, "Sicherheitsprobleme für IT Outsourcing basierend auf Cloud Computing," *HMD - Praxis der Wirtschaftsinformatik*, vol. 281, 10 2011.

[39] Cloud Security Alliance, "Top Threats to Cloud Computing V1.0," 2010, https://cloudsecurityalliance.org/topthreats.html, 06.09.2011.

[40] European Network and Information Security Agency, "Cloud Computing Security Risk Assessment," Tech. Rep., 11 2009.

[41] T. Mather, S. Kumaraswamy, and S. Latif, *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. O'Reilly Media, Inc., 2009.

[42] S. Pearson, "Taking Account of Privacy when Designing Cloud Computing Services," in *Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing*, Vancouver, Canada, May 23 2009.

[43] F. Doelitzscher, C. Reich, M. Knahl, A. Passfall, and N. Clarke, "An Agent Based Business Aware Incident Detection System for Cloud Environments," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1, no. 1, p. 9, 2012.

[44] J. Balasubramaniyan, J. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni, "An Architecture for Intrusion Detection Using Autonomous Agents," in *Computer Security Applications Conference, 1998, Proceedings., 14th Annual*, dec 1998, pp. 13 –24.

[45] Y. Mo, Y. Ma, and L. Xu, "Design and Implementation of Intrusion Detection Based on Mobile Agents," in *IT in Medicine and Education, 2008. ITME 2008. IEEE International Symposium on*, dec. 2008, pp. 278 –281.

[46] J. Sen, I. Sengupta, and P. Chowdhury, "An Architecture of a Distributed Intrusion Detection System Using Cooperating Agents," in *Computing Informatics, 2006. ICOCI '06. International Conference on*, june 2006, pp. 1 –6.

[47] J. Wei, X. Zhang, G. Ammons, V. Bala, and P. Ning, "Managing Security of Virtual Machine Images in a Cloud Environment," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, ser. CCSW '09. New York, NY, USA: ACM, 2009, pp. 91–96.

[48] R. Schwarzkopf, M. Schmidt, C. Strack, and B. Freisleben, "Checking Running and Dormant Virtual Machines for the Necessity of Security Updates in Cloud Environments," in *Cloud Computing Technology and Science, 2011 IEEE Third International Conference on*, 2011, pp. 239 –246.

[49] M. Al-Morsy and H. Faheem, "A new standard security policy language," *Potentials, IEEE*, vol. 28, no. 2, pp. 19 –26, march-april 2009.