Journal of Cloud Computing
a SpringerOpen Journal

RESEARCH Open Access

An agent based business aware incident detection system for cloud environments

Frank Doelitzscher[1*], Christoph Reich[1], Martin Knahl[1], Alexander Passfall[1] and Nathan Clarke[2]

Abstract

Classic intrusion detection mechanisms are not flexible enough to cope with cloud specific characteristics such as frequent infrastructure changes. This makes them unable to address new cloud specific security issues. In this paper we introduce the cloud incident detection system Security Audit as a Service (SAaaS). It is built upon intelligent autonomous agents, which are aware of underlying business driven intercommunication of cloud services. This enables the presented SAaaS architecture to be flexible and to supported cross customer event monitoring within a cloud infrastructure. A contribution of this paper it to provide a high-level design of the SAaaS architecture, an introduction into the proposed Security Business Flow Language (SBFL), a first prototype of an autonomous agent and an evaluation about, which cloud specific security problems are addressed by the presented architecture. It is shown that autonomous agents and behaviour analysis are fertile approaches to detect cloud specific security problems and can create a cloud audit system.

Keywords: Cloud security, Cloud audit, Agents

Introduction

Enterprise analysts and research have identified cloud specific security problems as the major research area in cloud computing [1-4]. Since security is still a competitive challenge for classic IT environments it is even more for cloud environments due to its characteristics, like seamless scalability, shared resources, multi-tenancy, access from everywhere, on-demand availability and 3rd party hosting [5]. Pushed by cloud commercials promising "infinite scalability and resources" combined with on-demand access from everywhere, cloud user quickly forget that there is still a real IT infrastructure behind a cloud, where the architecture complexity is actually increased compared to traditional data centers. This also introduces security and availability issues as recent incidents at the major public cloud provider Amazon Web Services (AWS) show. After an infrastructure outage in April 2011, Amazon's Compute Cloud EC2 was not available, causing popular services like Reddit to be unable to serve its customers [6]. While such an outage violates EC2's quality of service level agreements of 99.95% availability: (≙ 1,825 outage days/year), Amazon's support handling had a strong impact on trust in current cloud provider:

(1) Amazon data centers are divided into several availability zones to distribute impact of (hardware) failures. For resilience reasons users distribute their data over different availability zones. As a result of the outage EC2 customers permanently lost data, although services were hosted on different EC2 availability zones. A company offering webservice usage monitoring lost 11 hours of historical data [7].
(2) During the crash an EC2 customer running a monitoring service of cardiac patients tried to reach Amazon's support unsuccessfully. Neither information about the expected downtime nor moving the unreachable instances to a different EC2 data center was offered.
(3) Since hardware sovereignty is given away in cloud computing security, health and monitoring information is critical to cloud users to build there services in an appropriate way regardless which cloud model (public, hybrid, private cloud) is used. This is already known from traditional IT outsourcing and providers try to establish trust to

*Correspondence: frank.Doelitzscher@hs-furtwangen.de
[1]Furtwangen University, Cloud Research Lab, Robert-Gerwig-Platz 1, 78120 Furtwangen, Germany
Full list of author information is available at the end of the article

customers by proving their compliance to IT security standards like ISO27001 or ISO9001. Amazon AWS so far seams to follow a contrary approach: although AWS provides status information about the cloud infrastructure at the Amazon Service Health Dashboard [8] users can only see a service' history over the last five weeks. Amazons problems from April 2011 were not visible anymore to users by the end of May 2011, as depicted in Figure 1.

Although existing recommendations (ITIL), standards (ISO 27001:5, CobiT, PCIDSS) and laws (e.g., Germanys Federal Data Protection Act) provide well-established security and privacy rulesets for data center providers, research has shown that additional regulations have to be defined for cloud environments [1,9]. In classic IT infrastructures security audits and penetration tests are used to document a data center's compliance to security best practices or laws. But, the major shortcoming of a traditional security audit is that it only provides a snapshot of an environments' security state at a given time (time of the audit was performed). This is adequate since classic IT infrastructures don't change that frequently. But, because of the mentioned cloud characteristics above, this is not sufficient for auditing a cloud environment. A cloud audit needs to consider the point of time when the infrastructure changes and the ability to decide if this change is not causing a security gap or an infrastructure misuse. Knowledge of underlying business processes is needed, for example, to decide if a cloud service scales up because of a higher demand of valid business requests or a hacker misuse. As a first approach to a continuous auditing system for cloud environments we are presenting a cloud monitoring environment in this paper.

While monitoring the security of large IT infrastructures with distributed sensors as input feeds is common for Intrusion Detection Systems (IDS) this approach breaks down for cloud infrastructures. Mainly, because of the complexity and frequently changing environment driven by the user. Traditional IDS setups are built around a single monolithic entity, which is not adaptive enough to do data collection and processing in an efficient and meaningful way [10]. To mitigate this we propose an incident detection system for cloud computing: Security Audit as a Service (SAaaS), which is build upon intelligent, autonomous agents collecting data directly at the source, analyze and aggregate information and distribute it considering the underlying business process. This data interpretation is achieved using a Security Business Flow modelling engine that allows to model security monitoring events which are aware of an underlying business flow. The usage of autonomous agents enables a behaviour anomaly detection of cloud components while maintaining the cloud's specific flexibility. Our system respects the following cloud specific attributes:

- A high number and complexity of distributed systems
- An often changing infrastructure (e.g., service scalability or user driven)
- A business process driven intercommunication of cloud service
- Cloudwide incident detection

In the remainder of this paper, we first describe related work (Section Related work). Continuing with selected cloud specific security issues (Section Cloud specific security issues in focus), which are targeted by our approach. (Section SAaaS architecture) then introduces the Security Audit as a Service (SAaaS) architecture. The concept of using a distributed agent framework is introduced and a first agent prototype is shown. (Section Framework evaluation) evaluates why the paradigm of autonomous agents is valuable for incident detection in cloud environments and how the presented SAaaS architecture addresses the

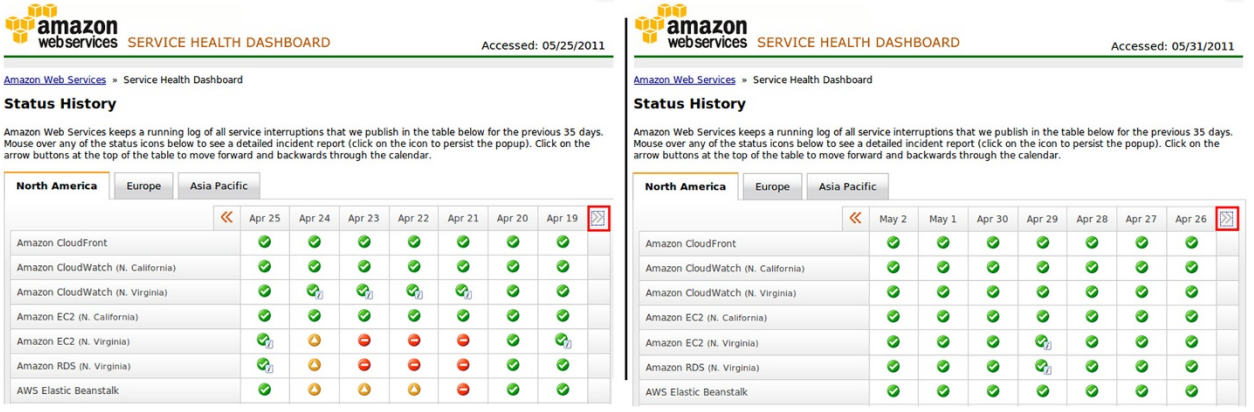

Figure 1 Amazon Health Dashboard accessed at 05/25/2011 and 05/31/2011.

presented cloud security issues. (Section Conclusion and future work) concludes the paper and informs about future work.

Related work

First, current literature identifying cloud security issues is shown, followed by a presentation of other cloud audit projects. Then some related work about using behaviour detection to perform anomaly detection gets presented. This section then continues with a discussion about other cloud security research projects in contrast to SAaaS. It ends with an overview of selected other research regarding the usage of autonomous agents to overcome traditional monitoring system limitations.

Cloud security literature

The most comprehensive survey about current literature addressing cloud security issues is given by Vaquero et al. in [3]. It categorizes the most widely accepted cloud security issues into three different domains of the Infrastructure as a Service (IaaS) model: machine virtualization, network virtualization and physical domain. It also proposes prevention frameworks on several architectural levels to address the identified issues. While Chen et al. state in [4] 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 cloud user and cloud provider. Both papers confirm and complete the cloud specific security issues identified by our research. Furthermore, they identified a demand of a two-way trust enabling architecture for cloud infrastructures and the ability of "selectable security primitives with well considered defaults" [4]. The SAaaS architecture will provide this mutual trust. SAaaS' Security Business Flow Language enables the user to define its own security levels and to choose from a spectrum of security subsystems as demanded by [4].

Cloud audit research projects

Wang et al. present in [11] a system to audit integrity and security of public data cloud storage. Their solution allows a third party auditor to be able to efficiently audit the cloud data storage without demanding the local copy of data, and introduce no additional on-line burden to the cloud user. Therefore they combine the public key based homomorphic authenticator with random masking to achieve a privacy-preserving public cloud data auditing system.

A "Dynamic Audit Services for Outsourced Storages in Clouds" is presented by Zhu et al in [12]. The approach uses fragment structures, random sampling and index-hash tables, supporting provable updates to outsourced data and timely anomaly detection.

Massonett et al. discuss in [13] the problem for IT security audits if federated cloud infrastructures are spanned across different countries. They introduce an existing federated Cloud monitoring infrastructure to monitor in which country data is actually saved without compromising Cloud isolation. In the presented approach collaboration is required between the cloud infrastructure provider and the user of the cloud, the service provider. The proposed architecture is validated by an e-Government case study with legal data location constraints.

Cloud audit standardization projects

To address the lack of existing standards regarding cloud specific security problems, a couple of open projects have been created. One working group of the Cloud Security Alliance (CSA) is the umbrella project *Governance, Risk Management and Compliance (CSR)* [14]. It includes the sub projects *CloudAudit*, *Cloud Controls Matrix (CCM)*, *Consensus Assessments Initiative Questionnaire (CAIQ)* and *Cloud Trust Protocol (CTP)*. The Cloud Audit A6 - Automated Audit, Assertion, Assessment and Assurance API [15] has the goal to provide a common interface and namespace for cloud computing providers to automate the audit, assertion, assessment, and assurance of their cloud environments. The Cloud Controls Matrix "is specifically designed to provide fundamental security principles to guide cloud vendors and to assist prospective cloud customers in assessing the overall security risk of a cloud provider." [14]. A questionnaire about which security controls exist in IaaS, PaaS and SaaS offers is provided by the Consensus Assessments Initiative Questionnaire. Finally, the CTP is an initiative to create a machine and human readable protocol which provides Transparency as a Service for cloud users about compliance of cloud provider [16].

The *EuroCloud Star Audit* [17] is a German certificate for a SaaS cloud provider. The audit aims to establish a high level of security and transparency for users and providers alike. The audit starts with the provider's general profile, carries on with contract and compliance including data privacy protection, general security, operation and infrastructure, operation processes and goes as far as application and implementation.

Behaviour analysis & anomaly detection

Nascimento and Correia [18] describe a system for finding anomalies in HTTP requests which can also be applied to HTTP services running as SaaS. They compare different algorithms of which most are based on analyzing string tuples of different lengths.

A good introduction to anomaly detection is given by Banerjee et al. in [19]. Starting with anomaly detection in general and its applications, they later go into detail about various algorithms. The work is finished with a case

study of anomaly detection in network intrusion detection where the authors present a system in use at University of Minnesota which in the past successfully detected various network anomalies undetected by SNORT like new worms or very slow network scans.

Lazarevic et al. [20], Garca-Teodoro et al. [21] and Patcha and Park [22] provide overviews over different anomaly detection techniques for intrusion detection. They all discuss advantages and drawbacks of various approaches, each focusing on a different subset. Lazarevic et al. compare different outlier detection techniques using distance-based methods like local outlier factor, nearest neighbor of unsupervised support vector machines. The algorithms are benchmarked using the 1998 DARPA Intrusion Detection Evaluation Data where the former scores the best results.

Roschke et al. [23] propose an intrusion detection system for cloud computing where each layer (SaaS, PaaS, IaaS) has its own detection system. The System also includes an anomaly detection part which is, however, not the focus of the paper and has to be detailed in future work.

Another IDS for cloud computing is described by Vieira et al. [24].

Similar cloud security research to SAaaS

Raj et al. [25] introduce a virtualization service implemented as Xen Virtual Machine (VM) extensions, which provides role based access control based on a trust value of a VM. This trust is based upon a VM's attributes, like number of open network connections. Access to different cloud services, like file access is given on a VM's trust value. The presented implementation methods are following the same idea as the SAaaS architecture: trust generation via behavioural monitoring to build a "normal" cloud usage profile. The implementation presented is mainly based on Xen tools. Since SAaaS is build upon the CloudIA infrastructure, which uses KVM, corresponding tools need to be identified/implemented. The presented SAaaS architecture will extend this trust model by the ability to evaluate business process driven cloud infrastructure changes due to the presented Security Business Flow Language.

In [26] Wei et al. present a VM image management system, which controls VM image access and tracks image provenance to address the issue of security patches for VM base images. It provides a prototype image scanner to scan software versions installed within a VM image and filter for user to exclude images with unpatched software stacks. To provide secure cloud hosts running VM images the authors of [3,27] propose the usage of TPM/TCPA crypt chips for secure OS installation. The SAaaS architecture can utilize all of the above introduced techniques to establish a trusted computing base for cloud environments and extends them to provide a cross customer trust.

The applicability of using complex event processing (CEP) in cloud environments is demonstrated by Schaaf et al. in [28]. The authors present a predictive cloud broker which reacts to changes in business processes and reflects them in up- or downscaling of cloud resources (VM)s. As a decision base distributed monitoring instances in relational database systems are used to feed the CEP engine. While this approach is an initial step in our direction the work remains quite high-level and in particular focuses on business process changes whereas SAaaS targets the detection of security incidents. Despite these different application areas, principles of this work can be applied to the SAaaS architecture. It extends them by the consideration of business process flows, and a reduction of event storms.

Distributed agents research

Intensive fundamental research on the applicability of autonomous agents and multi-agent systems at different areas of research, such as game theory, e-business, semantic web and distributed computing is done by M. Wooldridge [29]. Lots of his work is based on agents following the beliefs, desires, and intentions (BDI) model [30], which results in the intelligence an agent has. The agent framework proposed in this paper supports the BDI model and integrates its benefits. The advantages of using agents to overcome the challenges of monitoring a frequently changing infrastructure is discussed in research especially in the area of Intrusion Detection & Prevention Systems (IDS, IPS) and demonstrated in [31-34]. Lui et al. demonstrate in [31] that anomaly detection can enhance the detection of new, unknown network attacks in an IDS. They show that certain data mining algorithms are more suited to detected certain network attacks than others. Therefore multiple agents are proposed implementing different data mining algorithms to lead to a higher, more precise detection rate of unknown network attacks. This supports the idea presented in this paper that agents can enhance anomaly detection of unknown attacks. While the agents presented by Lui et al. are very static in their perimeter, the proposed SAaaS agents in our work can be extended during runtime to adapt better to the current infrastructure state. For example, if a cloud user adds a software to its VM, an existing file system monitoring agent configuration can be updated to additionally monitor this specific path as well. If this case includes the installation of a second agent on the VM, a local Aggregator agent will be updated, to accept / collect events from this new agent as well.

Zamboni et al. present in [35] how 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. While Chirumamilla et al. show in [32] that agents can enhance the security of wireless networks they don't really benefit from typical agent characteristics like deployment on demand as the SAaaS agents presented in this paper.

Mo et al. introduce in [36] an IDS based on distributed agents using 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 also shown by Sengupta et al. in [37]. The presented advantages apply for our SAaaS agents as well.

Krügel and Toth show in [34] that processing limitations of IDS can be enhanced by using mobile agents. Instead of moving sensor information to a central processing unit they use agents to correlate the monitored event data, thus increasing fault tolerance and scalability of the intrusion detection system. This idea is partially supported by the event preprocessing of the presented SAaaS agents in this work. This approach will be further investigated during the SAaaS project lifetime.

Cloud specific security issues in focus

This section describes cloud specific problems which we target by our approach. A detailed analysis of the actual security impact is not easy to find, because very often security problems are declared as cloud security problems, although they already exist in traditional IT-outsourcing scenarios and are merely exacerbated in cloud environments. The German Federal Office for Information Security publishes the IT baseline protection catalogs enabling enterprises to achieve an appropriate security level for all types of information. The catalogs were extended by a special module covering virtualization in 2010. In a comprehensive study on all IT baseline protection catalogs as well as current scientific literature available ([1-4,38]) we made a comparison between classic IT-Housing, IT-Outsourcing and cloud computing. The study is available at [39]. The following cloud specific security issues were identified as auditable by the SAaaS system. Furthermore, each identified issue is categorized by the affected core principles of information security, such as availability, confidentiality, integrity, etc., to provide a starting point for a risk management analysis.

1) **Detection of cloud resource abuse** 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 main reason lies in a 'frictionless' registration process where only a credit card number is used to 'authenticate' a customer. 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 the following core principles of information security: *authorization*, *integrity*, *non-repudiation* and *privacy*. Strong monitoring of user activities on all cloud infrastructure components is necessary to increase transparency.
2) **Cloud security transparency** 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 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 zero day exploit in software running on cloud hosts (e.g., hypervisor, OS kernel) cloud customer blindly depend on a working patch management of the cloud provider. Cloud providers usually do not disclose information on this processes to customers. By this issue the information security core principles textit integrity, *availability* and *non-repudiation* are affected.
3) **Recognition of defective shared resources isolation** 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 provider advertise implemented reliability measures to prevent data loss, like replicating data up to six times. In contrast customers have no possibility to prove if 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 to minimize its damage by the presented actions. Affected information security core principles are *integrity*,

authenticity, *confidentiality*, non-*repudiation*, *availability* and possibly *data privacy*.

SAaaS architecture

In this section, we first give an agent definition and show how an autonomous agent architecture can improve incident detection in cloud environments. We then introduce the SAaaS architecture components and elaborate the concept of Security Business Flow rules. Following, a first SAaaS agent prototype is presented.

Agent definition

An agent can be defined as [40]: "... 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, are self-defending and self-acting, we term them *autonomous*. Agents can receive and send data from or to other instances, like agents or SAaaS' event processing system. The "central" event processing system gets itself implemented as an agent, which can be scaled and distributed over multiple VMs.

How agents can improve incident detection

Incident detection in cloud environments is a non trivial task due to a cloud's characteristics:

Reduction of events Especially the frequently changing infrastructure poses a big challenge to define something, like normal behaviour and detect anomalous behaviour. Therefore it is important to have a high number of sensors capturing simple events. Simple events need to be pre-processed and abstracted to complex events, reducing the possibility "of event storms".

Fast adaption Combined with knowledge about business process flows it will be possible to detect security incidents in a frequently changing infrastructure while keeping the network load low. The usage of autonomous acting agents delivers this possibility.

Flexibility Agents can also be added, removed or reconfigurated 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.

Increased durability Furthermore, using autonomous 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 damage is restricted to this single agent or a small subset of agents which are requiring information from this agent.

In the following examples are presented to support the identified advantages. Imagine a business process of a web application `P1` (depicted in Figure 2) where user Bob adds a new user to a user database by filling out a web

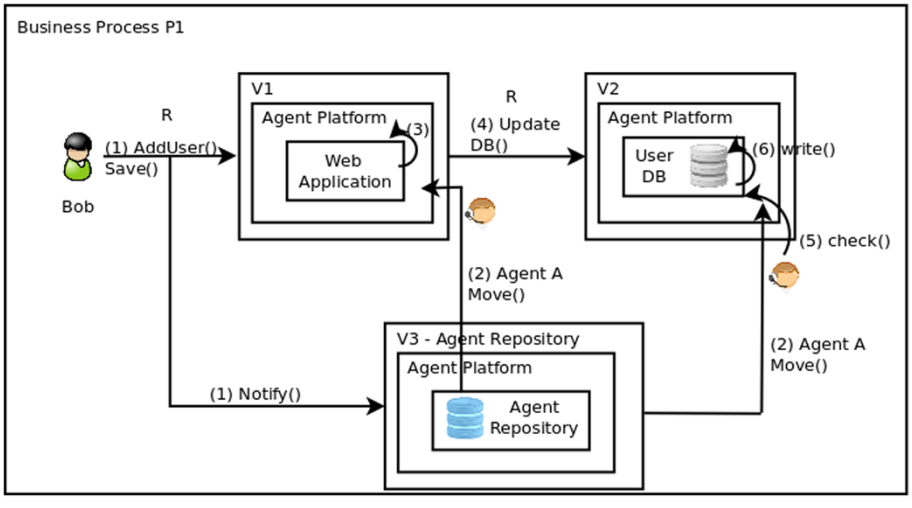

Figure 2 Business flow dependend agent deployment.

form. By pressing the "Save" button a legal request R (1) gets executed as part of business process P1. Agent A that monitors the database access gets moved (2) at the process start R to the request-executing VM(s) V1,V2, monitoring the data access (3),(5) during process time. After the data process has been comleted (4), (6) they delete themselves from V1, V2 after P1 has been finished. This results in increased flexibility and fast adaption to a changing infrastructure.

Furthermore, agents can be updated to new versions (depending their interface remains unchanged) without restarting the whole incident detection system or other SAaaS agents running at a VM.

SAaaS agents are enabled to be aware of underlying business flows (see Section Modelling of business flows). Therefore several simple agents are logically belonging together forming an *agent group* where every agent knows about the other agent. While single agents monitor simple events (e.g., user login on a VM) and share them with the agent group abstract, complex events can be created. 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 unknown file (the trojan). Instead of sending those three simple messages to a central event processing unit an Aggregator agent can collect them, conditioning one abstract 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. Thus a reduction of events can be achieved.

By ordering agents in a hierarchical structure (multiple simple agents can exists on the same platform e.g., inside a VM), preprocessing of detected simple events and information sharing between logical associated agents leads to a reduction of network load [41]. Furthermore, this makes the system more scalable by reducing data sent to upper system layers. This is introduced and used in [42]. Combining events from system agents (VM agent, host agent - see Figure 3) 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. This leads to an increased durability of the overall incident detection system.

Agent intelligence

An agent's intelligence is provided by the application of the Monitor, Analyze, Plan, Execute, Knowledge (MAPE-K) paradigm developed by IBM [43], which is a commonly used and accepted design for autonomous systems. Knowledge comes in the SAaaS approach from a rule based knowledge, provided by the SBF rules. Monitoring is done by simple sensor agents, like the introduced iNotify agent. Analysis, planning and execution is done via the Aggregator agent at this current phase of the research project. Due to the modular design this can be realized by several, single agents as well. Especially, the execution of actions dependent on the analysis result will be implemented as an extra agent, thus can be used independently of the underlying event detection (e.g. file system change, infrastructure change, etc). One advantage, of this agent's intelligence is a reduction of manual user interaction. When deploying a new VM, dependend on the VM template a user has chosen, the SAaaS Agent Management knows which agents need to be deployed to this VM. In case a new image gets used, first a software investigation agent gets deployed to the VM, analyzing which software packages are installed on this particular VM. Dependend on the results, the SAaaS agent management knows which agents should be deployed to this VM. The software analysis part is currently done by checking the local software package repository on the VM. Furthermore, in the case where agents are already deployed and actively monitoring, intelligence is applied by a special Security Business Flow (SBF) rule. In the SAaaS enabled cloud a user has to define one or multiple VM Management stations. These are basically IP devices, which are allowed to establish a management connection (e.g. SSH) to a VM. This prevents the SAaaS agents from sending false-positive events, like in the following scenario: Given a SAaaS enabled VM with an iNotify-agent already deployed monitoring file system changes. When the cloud user logs into this VM to change some configuration files, this gets immediately detected by the iNotify-agent. It sends these events to the local Aggregator-agent. Instead of just forwarding the events to the SAaaS event management system, the Aggregator-agent first checks, if an allowed management connection is established to this VM. Is this the case, the iNotify events were classified as "allowed" not resulting in an alarm. Only a "configuration changed" message gets forwarded. When the management connection is closed again, the Aggregator-agent notifies the SAaaS Agent Management to automatically deploy a new Audit agent to validate the security state of this VM after this configuration change. Thus, agents act autonomously.

SAaaS architecture components

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 output layer.

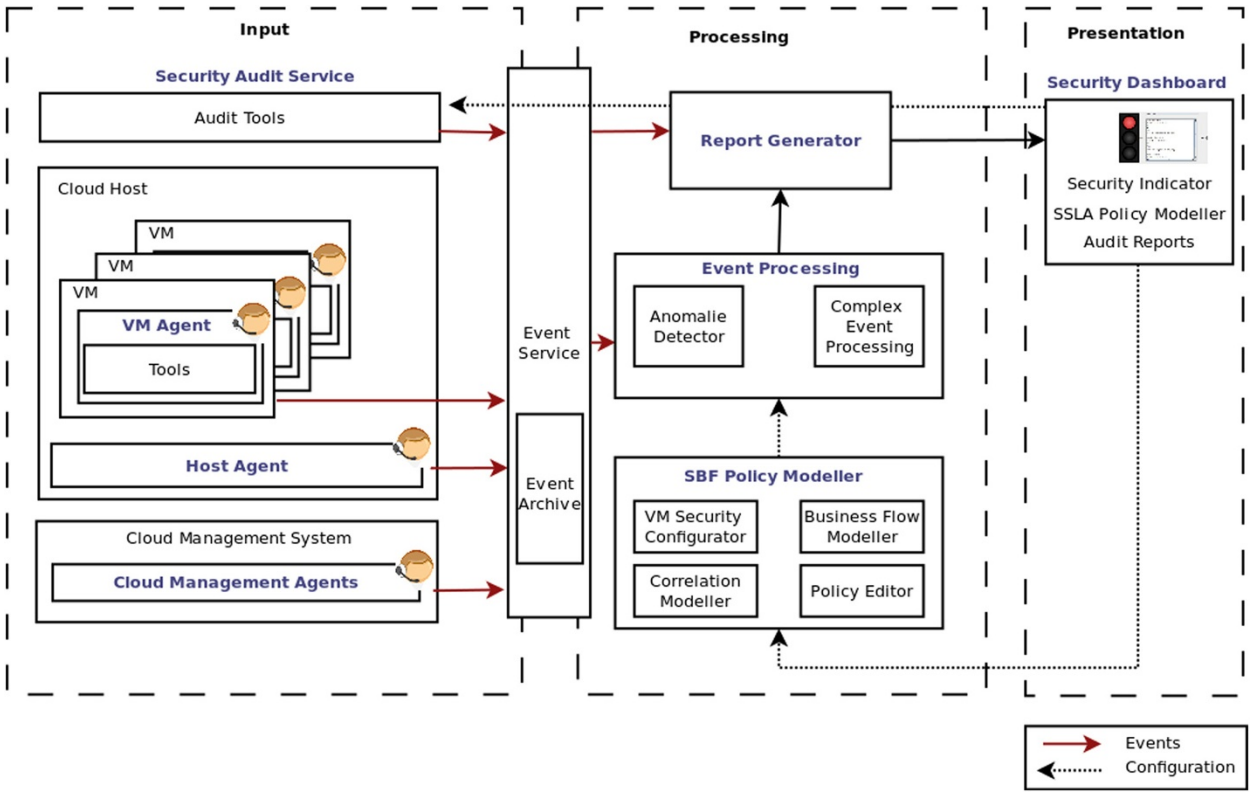

Figure 3 SAaaS architecture.

Input layer

The SAaaS architecture gets its monitoring information from distributed agents which are positioned at key points of the cloud's infrastructure to detect anomalous activities in a cloud environment. Possible key points are: running VMs of cloud users, the VM hosting systems, data storage, network transition points like virtual switches, hardware switches, firewalls, and especially the cloud management system. A *VM agent* integrates several monitoring and policy enforcing tools. Therefore it loads the necessary VM agent plugins 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 layer

Each SAaaS agent receives security policies from the Security Business Flow (SBF) policy modeler 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 freezed config file). Thus, every occurrence gets first preprocessed by an agent, which reduces communication between the VM agent and the Cloud Management Agent. Self learning algorithms will be evaluated to improve an agents' intelligence. The *Security Business Flow policy modeler* consists of a policy editor, a VM security configurator and a semantic correlation modeler to enable cloud user to design SBF and security policies. An example Security Business Flow aware rule (SBF rule) 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 minish the risk of further damage.*

whereas a security policy 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.*

Security policies get sent from the Security Audit Service to the corresponding agents. Using the monitoring information of the distributes agents in combination with the SBF rules a *cloud behaviour model* is built up for every cloud user. SBF rules are also used as input for the Cloud Management Agent to detect user overlapping audit events. Forwarded higher level events are processed by a *complex event processing (CEP) engine*. It is also fed

with the model led business flows from the *Business Flow Modeler* 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 layer

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 is 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.

Communication between the distributed agents and the Security Dashboard is handled by an *Event Service*. Events will use a standardized message format which is not defined yet. Our first prototype implements the Intrusion Detection Message Exchange Format (IDMEF) to support IT forensics. Events are also stored in an *Event Archive*.

Modelling of business flows

A customers cloud instances always serves a certain business case. One major advantage of the SAaaS architecture will be that its agents are aware of underlying business process flows. To achieve this a business process flow model language will be developed. Consider the following example: Given a typical web application system consisting of a webserver and a database backend deployed at two VMs in a cloud as depicted in Figure 4. All VMs are equipped with SAaaS agents. The user's administrator installs each VM with the necessary software, e.g., Apache webserver and MySQL database.

After the functional configuration of the installed software is finished the monitoring configuration gets designed in form of SBF rules. This can be technical rules, like allowed user logins, allowed network protocols and connections between VMs, or that the webserver configuration is finished and an alarm should be raised if changes to its config files are detected. Furthermore, the Security Business Flow Language allows to design rules considering the system's business flow. For example: if a request (using the allowed protocols) to the load balancer or database VM without a preceding service request to the web application is detected this is rated as an anomalous behaviour which does not occur in a valid business process flow. Therefore a monitoring event should be generated. SBF rules need to be modeled by a cloud user who is aware of its cloud instances and the underlying business process. Hence a formal modelling description for cloud environments needs to be developed. A first high level example of modeled SBF rules is shown in Listing 1. Line 4 - 19 describe possible technical rules, while line 21 - 26 models a business flow rule. In this case a request to the webserver is only valid if a preceding request was sent by the load balancer. Line 24 names the SAaaS agents, which needs to be contacted to resolve this constraint. Line 31 defines which action to take in case of a detected monitoring event.

Listing 1
SBF modelling example

```
1  <system>
2      <id>webserver_1</id>
3
4      <running_processes_1>
5          <process_name>
               /usr/sbin/apache
           </process_name>
6          <allowed_protocol_1>
7              <prot_name>
                   tcp
               </prot_name>
8              <src_port>80</src_port>
9              <src_system>
                   IP_of_load_balancer
               </src_system>
10             ...
11         </allowed_protocol_1>
12         ...
13     </running_processes_1>
14
15     <freezed_config_dir_1>
16         <path>/etc/apache2</path>
17         <allowed_access_type>
               read_only
           </allowed_access_type>
18         <allowed_accesser>
               running_processes_1
           </allowed_accesser>
19     </freezed_config_dir_1>
20
21     <request_1>
22         <name>
               Webapplication request
           </name>
23         <preceding constraint>
               loadbalancer_sent_request
           </preceding constraint>
24         <constraint_validator>
               SAaaS_loadbalancer_agent
           <constraint_validator>
25         ...
26     </request_1>
```

```
27
28          <incident_alarm_1>
29                  <name>
                            Webserverconfig_changed
                    <name>
30                  <origin>
                            freezed_config_dir_1
                    </origin>
31                  <action>email_1</action>
32                  ...
33          </incident_alarm_1>
34  ...
35  </system>
```

This is a first example - applicability of existing process languages like the Business Process Execution Language or definition of a new Security Business Flow Language is in future work. To reduce complexity a graphical policy modeler needs to be developed. For typical cloud usage components e.g., a webserver, profiles will be prepared, which models necessary dependencies, like config directory, associated processes or used network protocols.

SAaaS agent prototype

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 software platform
- Documentation & community support

Since our cloud environment at HFU's Cloud Research Lab CloudIA [44] is build around the cloud management system OpenNebula 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[a] specifications. Furthermore, it already provides a user interface, which alleviates agents creation, deployment and testing.

Figure 4 illustrates a basic agent architecture we already assumed in the SAaaS Use Case presented in (Section Modelling of business flows). It shows three SAaaS VM 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 [41]. To provide monitoring functionality a VM agent interacts through agent plugins with stand-alone tools, like process monitor, intrusion

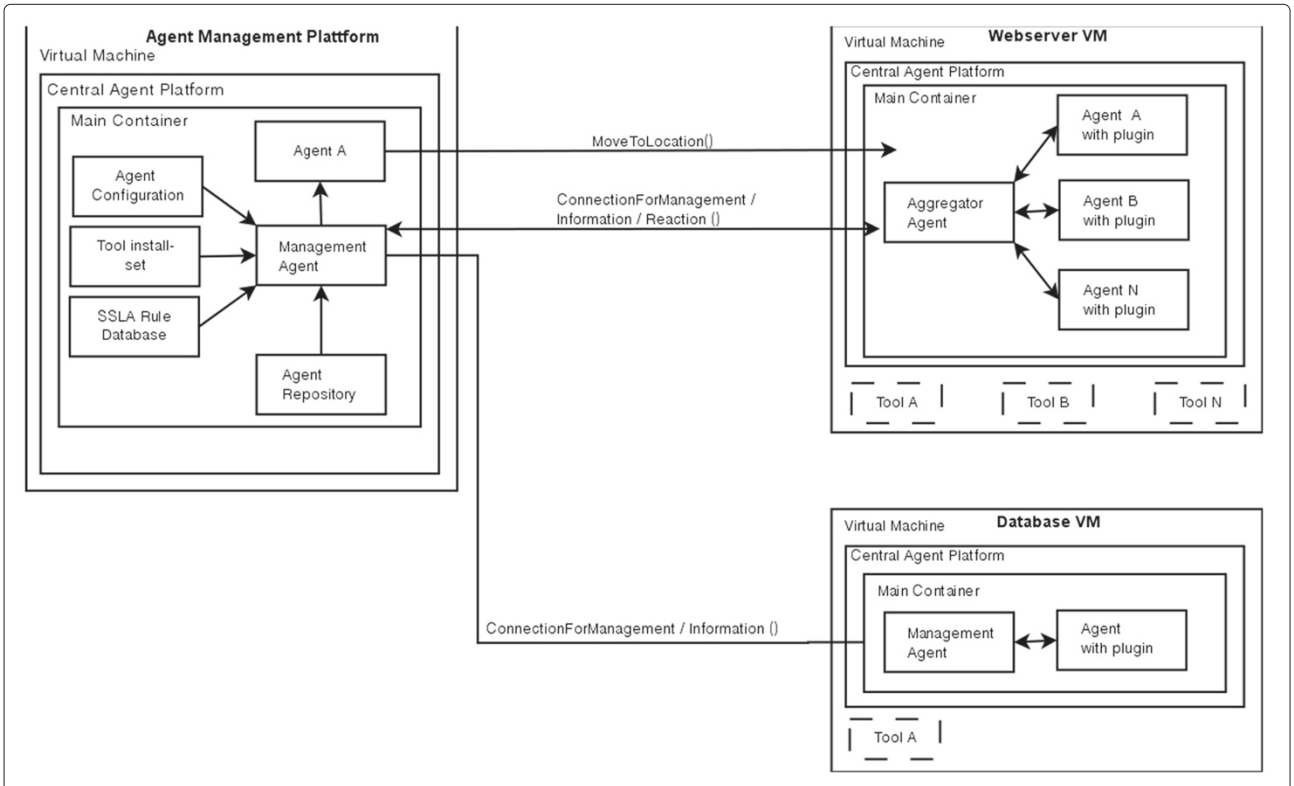

Figure 4 Basic SAaaS agent design.

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 SBF rules 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. [45] was integrated. This supports the presented advantage of deploying agents on-demand if a designed business process flow was started (as described in (Section How agents can improve incident detection). Though this implementation is up to future work.

As a first prototype, a two layered agent platform was developed, capable to deploy two *VM agents* which will be deployed and run inside a target VM: 1) iNotify-agent, b) webserver-agent. A *Cloud Management Agent* running as a service at a dedicated VM feeding information to a *Security Dashboard*. Regarding the tools feeding simple events to the two agents only Open Source Linux tools were considered during our research, since all VMs in CloudIA are Linux based. For communication between tool and agent 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 log file, which gets parsed by an agent plugin. As for mechanism a) the file system monitoring tool *iNotify* was used, whereas for mechanism b) parsing the Apache access log file (/var/log/apache2/access.log) was used.

For demo purposes a simple web frontend was written, which offers to launch several attack scenarios on a VM agent equipped VM in CloudIA.

First SAaaS demo scenario

In our first SAaaS demo we are showing the functionality of the agent framework, its successful integration in the CloudIA cloud infrastructure, a first version of simple SBF rules combined with a simple message reduction method. The following attack scenario is addressed: An attacker is able to successfully login to a target VM via ssh. Then a malicious file (e.g. a trojan) gets uploaded to the web server's directory. This file then gets downloaded by a possible future victim. The chronological sequence of the demo is shown in Figure 5. It contains three major phases:

1. VM preparation - Deploy of a webserver VM and assignment of a SBF "webserver" template
2. Attack & detection - VM gets attacked, agents generate multiple simple events
3. Evaluation - Attack is reported to SAaaS dashboard

VM preparation

First a cloud user deploys a new VM and configures it as a webserver (Figure 5, (1)). 5, (1)). After finishing the configuration he assigns a predefined SBF template in Figure 6. This automatically prepares (3) and deploys (4) an iNotify-agent and a webserver-agent to the running VM. The iNotify-agent is monitoring changes to the web servers' webroot directory, whereas the webserver-agents

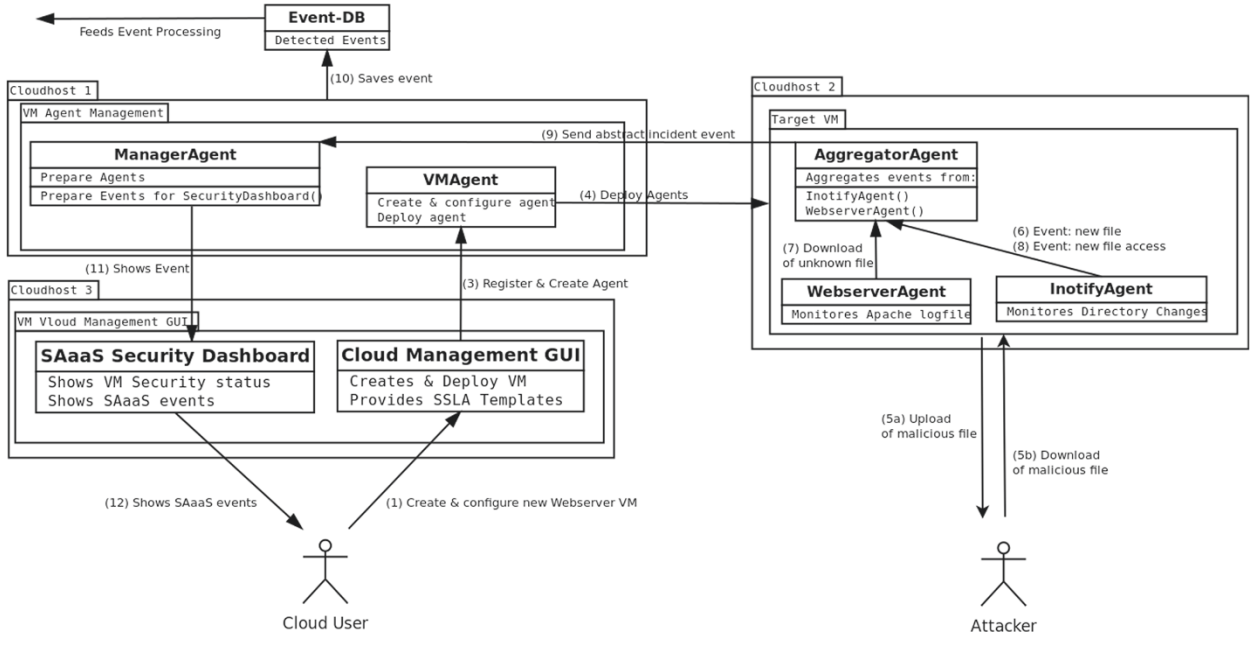

Figure 5 Chronological sequence of SAaaS demo1.

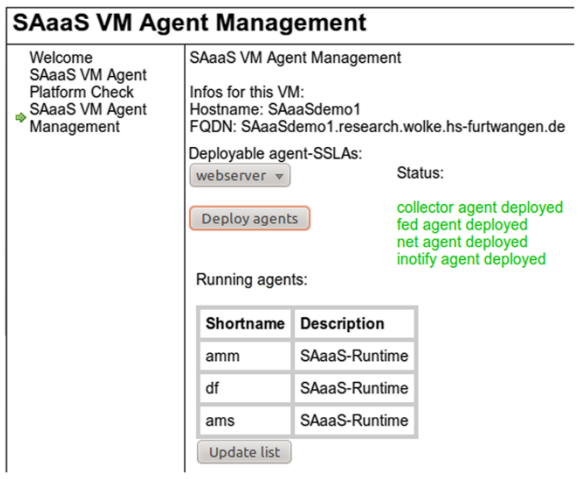

Figure 6 Assignment of SBF webserver template to VM.

compares http-get requests with a list of allowed html files of the webroot directory. It also includes a simple message reduction method for the collector agent stating: "All simple events occurring within one minute by these two agents get combined into one "Webserver attacked - unknown file detected" message".

Attack detection

An attacker logs in to the VM, uploads a malicious file to the VMs webserver root (5a). This then gets downloaded (5b). This results in three simple events generated by the corresponding agents:

iNotify-agent: creation of new file in webserver-root directory (6), read access of new file (8)
webserver-agent: download of unknown file (7).

These three simple events get combined to one abstracted "Webserver attacked - unknown file detected" event by the collector agents due to the simple message reduction method mentioned above, and sent out (9) to the Security Dashboard.

User notification

All events get saved into an event archive and forwarded to SAaaS CEP engine (10). The user gets informed (11) via a first prototype version of the Security Dashboard, depicted in Figure 7. It shows the VM's state before an attack. After launching an attack, the Security Dashboard indicator light changes its colour and provides a short information about the monitored event (Figure 7). The impact of an monitored event is defined in a simple severity matrix, as described in Table 1. Each severity value out of the webserver log file gets associated with a certain score. This score gets added for all messages. Then the quotient gets calculated which is directly connected to the resulting colour.

Security relevant anomaly detection through behavior analysis

To detect unforseen events, the system's behavior is constantly monitored and analyzed. Therefore it processes the events from each of the agents for enabling anomalous behavior detection. As described by Lui et. al in [31] agents with different anomaly detection algorithms are used to achieve a better detection rate.

To determine, which anomaly detection algorithms are suited best, an evaluation has to be done. The algorithms considered include machine learning approaches like neural networks or Markov models, statistical methods, outlier detection and some specialised algorithms [20-22]. This evaluation is part of current work therefore no final result is given here.

Generally, two methods to train a system exist: supervised learning where data is manually tagged as 'normal' or 'anomalous' and unsupervised learning where the detection system itself has to evaluate which data is normal and which is not. The latter is achieved for example by looking for extreme values in different key indicators.

To achieve an effective anomaly detection Winter et al. claim [46] that it is crucial that the types of events to be detected are defined in advance and the system is

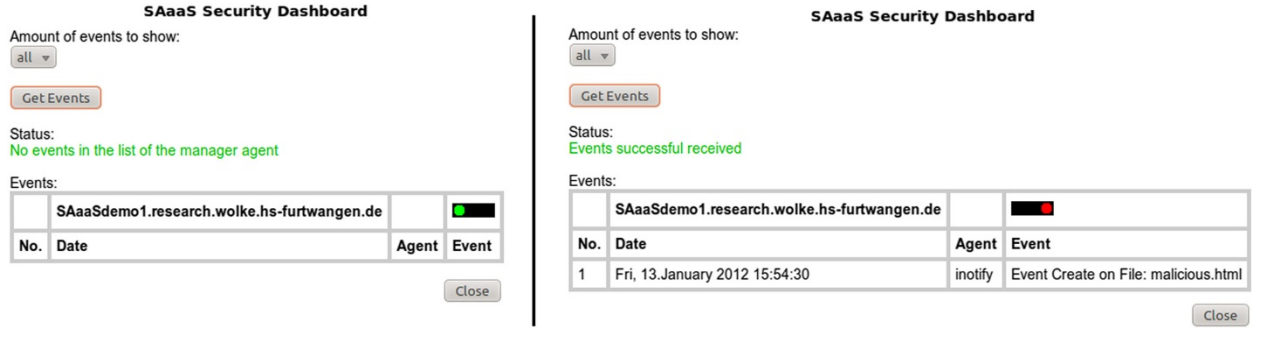

Figure 7 Cloud security dashboard prototype before and after detected attack.

Table 1 Severity matrix for simple 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

designed and configured accordingly. Therefore, we first determine which cloud specific security threats can be detected by anomaly detection. After that, we analyse which key factors have to be processed.

At the current state, we have identified the anomaly types to be detected. Some of these issues have already briefly been described in (Section Cloud specific security issues in focus) but are now presented in a more detailed manner.

Detection of account misuse

Account misuse is a well known problem from the area of web application and thus also affects the cloud's management web interface. An attacker can gain access to a victim's account by different means like hacking weak user credentials or exploiting security flaws present in the application [47].

In cloud computing however access to an account does not only grant access to the victim's data but also to the data hosted on services which run under the victim's account and thereby potentially to data of many service users. Additionally the attacker could spawn multiple new cloud instances, using them for his own malicious intents and thus financially damaging the victim as he has to pay for the instances. By bringing down the victim's services an attacker could potentially cause even more damage.

Monitoring the user's behaviour for anomalies can help detecting account misuse. For example heavy activity on a rather silent account would cause suspicion, especially if the login sourced from another country or continent. Additionally to the VM count and geographical login source, the weekday, daytime as well as usage of available VM types gets monitored.

Detection of distributed denial of service attacks

A cloud infrastructure can run a huge number of systems, maintained by different cloud users resulting in a heterogeneous level of security configuration. This is very attractive for misuse. A traditional SSH bruteforce attack tried numerous combinations of username:password couples on one target IP. Detection is fairly simple since this behaviour results in massive "login denied" messages in the SSH server's log file. Therefore modern SSH bruteforce attacks are carried out by compromised computers of a botnet (bots) which just try one or two username:password combinations at a specific target and then move on to a next one, depicted in Figure 8. Especially huge cloud infrastructures are very attractive targets to this attack, since all systems normally are within one IP range. If one vulnerable system was found (malicious ssh login was successful) this can compromise the whole cloud security state. This attack attempt mostly stays undetected in traditional host based monitoring systems. By monitoring the login attempts cloud wide over multiple cloud customer instances, a successful ssh login with preceding unsuccessful ssh login attempts at different cloud instances can be defined as anomalous behaviour and thus detected.

Detection of VM breakout

Several hypervisor vulnerabilities have been found in the past that allow an attacker to gain access to a cloud host from inside a VM (e.g. [48,49]) As cloud computing and especially IaaS relies heavily on virtualization technologies this poses a thread to every cloud provider.

By accessing the underlying host, an attacker not only gains access to real hardware but also to all other VMs running on this machine. It is therefore essential that VM escaping attacks get detected and appropriate counter measures are taken. This includes monitoring process activity on all cloud hosts for unusual activity, for example

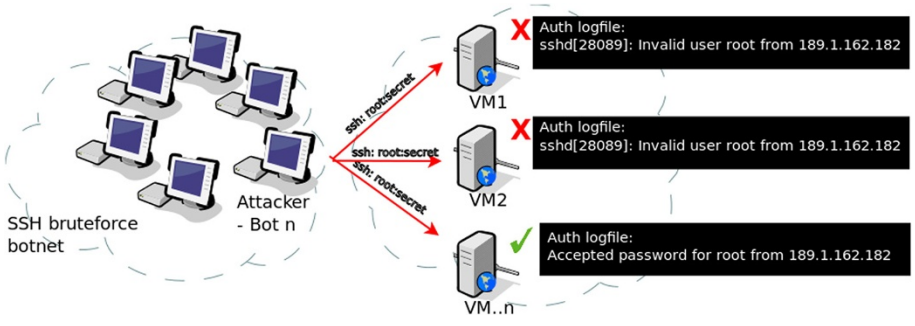

Figure 8 SSH bruteforce by botnet.

the commands that are being executed or syscalls as described by Hofmeyr et al. in [50].

Detection of cloud resource misusage

Scenario A - The possibility to quickly aggregate massive loads of computing power is very attractive. This is also true for criminals who can use this power for example to crack hashed passwords or decode CAPTCHAs. Technically there is however no non-intrusive way to decide for what exactly cloud computing power is used. Even if there was, data protection laws and credibility concerns prevent such monitoring.

One possibility to encounter misuse is to thoroughly check credit card or other payment data as most criminals wouldn't want to pay for themselves if other means such as stolen credit cards exist.

Scenario B - The cloud's computing power can also be used to carry out attacks on other targets. One possibility is to aggregate many VMs and use them to DDoS a single target and thereby preventing others to use its services. To detect such attacks, the network has to be monitored for anomalous activity especially from the inside. Due to the distributed nature of cloud computing information about network flow has to be collected at many different physical locations. To get the whole picture however, this data has to be analysed in the overall context.

An example scenario is depicted in Figure 9:

An attacker creates only a few number of instances in each of the cloud's data centers. Then he uses all of these world wide distributed instances to DDoS a victim's host. This host can be anywhere on the internet. If only the data in each data center was analysed for its own, maybe no attack would be detected as only a few number of hosts take part in it. If however, send the data to a centralized/distributed detection system, we can detect the attack at its whole extent.

Attacks on cloud services

To detect attacks on a user's services, behaviour profiles are generated for his systems. The SBF rules described above provide a basis for such a profile. However, additional properties like the VMs' CPU and memory usage can also be taken into account. These indicators are also combined as for example an attack could cause heavy CPU load but minor network load whereas for normal requests these values increase together. To detect the introduced anomalies the following key indicators have been identified:

- Account activity: VM count, VM type usage, geographical login source, weekday and daytime.
- Host monitoring: running processes, syscalls, user authentication
- Network monitoring: source and destination address and port, number of active connections per source/destination, duration of these connections, number of new connections in the last x seconds per source/destination, number of packets for each source/destination, number of bytes transferred per source/destination

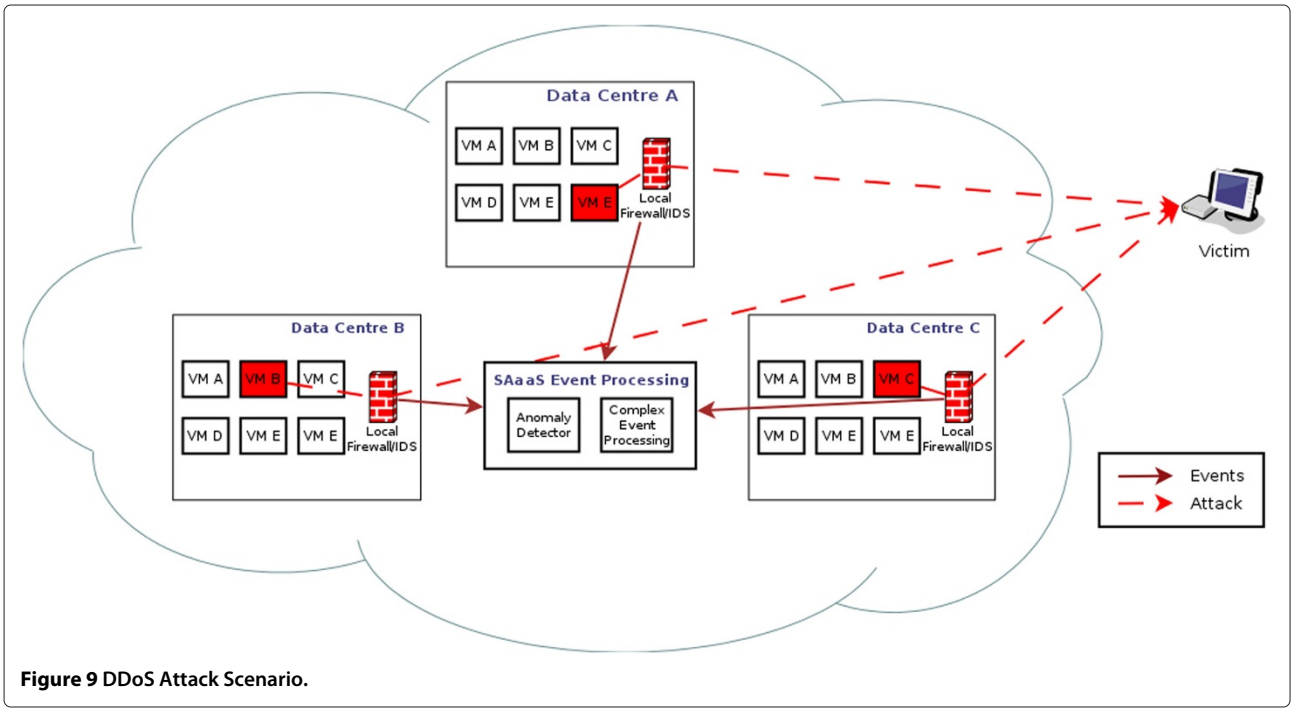

Figure 9 DDoS Attack Scenario.

- VM-Monitoring: CPU load, memory usage, hypercalls, VM scaling behaviour

The next step is to define these indicators more precisely and to find the best algorithms for each of the presented scenarios.

Automated security business role generation

Furthermore, behaviour analysis is to be used to support the definition of Security Business Flow rules and therefore the deployment of agents to a customers VM. By detecting which VM templates a user chooses and which software gets installed into a VM, the underlying business case gets detected. This could be a typical three tier architecture in a distributed environment, such as a web server VM (presentation tier), load balancer VM (logic tier) and database VM (data tier). This detection then results in the automated generation of SBF rules, such as allowed communication paths, validity of scalability commands or technical rules, like firewall rules or allowed protocols. A validity of scalability could be that, a high demand at the presentation tier, e.g. online shop fronted is necessary to qualify a scalability command of database VMs. The automated generation of SBF rules will support the presented SAaaS architecture and use cases. It is necessary for an automated detection, of which agents are necessary on a customers' cloud resources. This saves a huge amount of human work in the definition of security rules, which is a drawback of current incident detection systems.

Framework evaluation

In this section we first evaluate the usage of autonomous agents for the SAaaS architecture. We then discuss scenarios to show how the SAaaS architecture addresses the introduced cloud security issues presented in (Section Cloud specific security issues in focus).

Evaluation of the SAaaS architecture

Agent performance

It is essential for the SAaaS architecture that the agents are very efficient not causing a high offset of resource consumption. For our first prototype we want to show the overhead introduced to the cloud VMs by the JADE platform, how fast agents can be deployed to a VM in runtime and second how fast the agent communication is. All tests were done at the university's research cloud infrastructure CloudIA. Hardware of machines hosting the VMs was: 8x CPU: Intel(R) Xeon(R) CPU E5504 @ 2.00GHz 64-bit architecture, 12 GB of memory and 1 Gigabit Ethernet. Each VM was assigned with 512 MB RAM, 274 MB Swap, 1 CPU and 4GB HDD local storage. Each test was performed at least 10 times, average values are presented in the following.

Overhead JADE platform

First, the introduced overhead by the JADE platform was tested. E.Cortese et al. show in [51] that running the platform is not introducing a high CPU overhead. Additionally we measured how the boot process of a VM gets delayed because the JADE platform needs to be started. BootChart [52] was used to analyze overhead and it can be confirmed, that in worse case the JADE platform adds 2s to the total boot time of a VM of 15s. For the presented usage scenario this is totally acceptable.

Agent deploy time

Second, the time was tested it takes to create, configure and deploy an agent to a new platform. Therefore we introduced 5 measure times: t0 - agent gets created, t1 - agent gets started at the Agent Management VM, t2 - agent loaded its configuration, t3 - agent got transferred to target VM, t4 - agent starts working on target VM. The test was done first using no connection security for transferring the agents (MTP HTTP). Second a secure and authenticated channel between platforms was used (MTP HTTPS) to see how much impact secure communication introduces. Figure 10a confirms that in average it only takes 180ms for a full deploy of an agent using MTP HTTP and 350ms using MTP HTTPS. This proves the applicability of the JADE agent platform to support the presented SAaaS use case. Though further investigation needs to be done to clarify the MTP HTTPS overhead in detail.

Agent message time

Third, the agent message performance was tested. The roundtrip time of a variable amount of message with a variable payload (stringlength of event message) between two agents was measured. As a constraint a maximum roundtrip time of 1s was set. Figure 10b shows, that with the JADE agents we are able to send more than 500 messages with a maximum payload of 10.000 characters at once before we hit the defined constraint. So far this is acceptable for the introduced SAaaS scenario.

These tests show the time performance of single agents. Especially in the given cloud scenario scalability of the proposed system is from high interest. To prove this, scalability test with different number of VMs and agents (10, 50, 100, ...) are developed right now. These tests were not finished by the time of this manuscript. Nevertheless, to make the results available to the readers of this journal they will be available online as a tech report at the SAaaS project website [53].

Message reduction by business flow awareness

In case of a monitoring event is produced it first will be processed by the SAaaS agent, which is initiating the

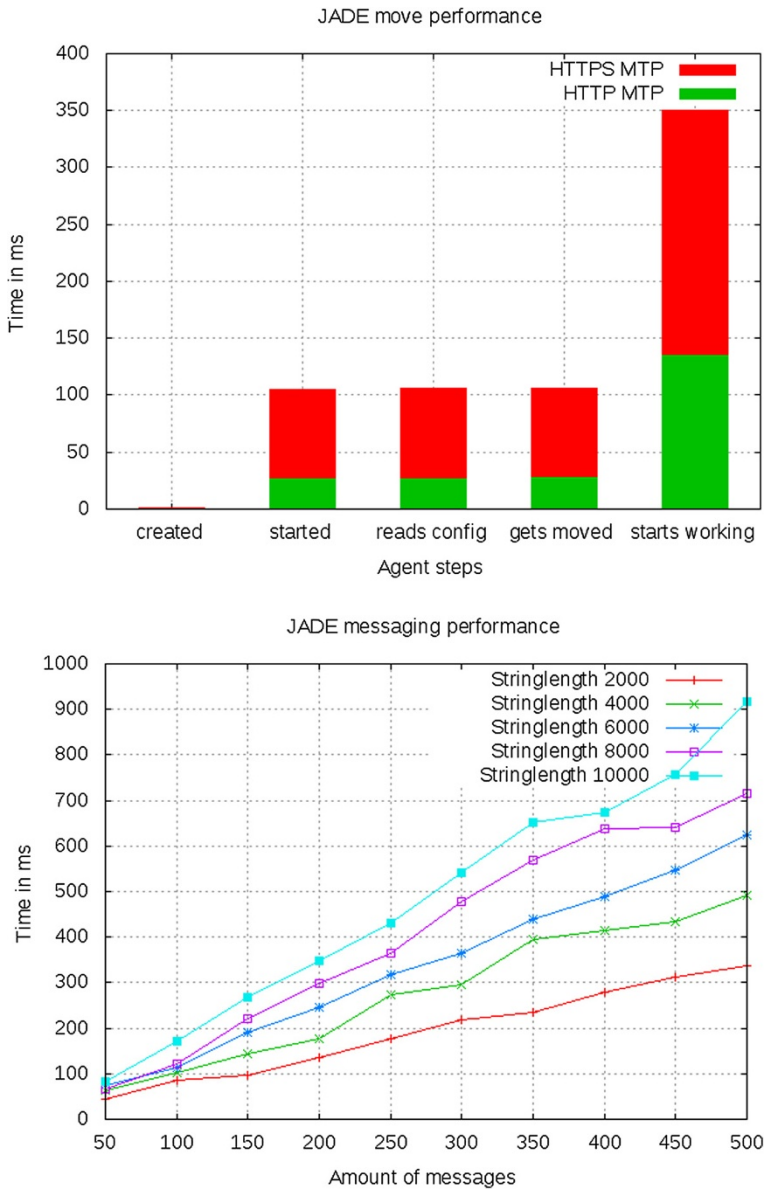

Figure 10 Performance tests of SAaaS agent performance. (a) Agent deploy time **(b)** Agent message roundtrip time.

event. Afterwards this agent informs all other agents which are also involved in the current business case (agent group). This is important to reduce the overall messages sent to the cloud event processing system especially in large cloud computing environments. Imagine an expected high load on the load balancer can result in a high number of events produced by the load balancer's agent. Since the events are expected they again result in a high load on the webserver and the database whose corresponding agents could produce again a high number of events. By informing the business flow participating agents (webserver agent, database agent) with an abstract message (e.g., 100 expected events registered) false positive event messages will be prevented.

Cloud wide incident detection

Furthermore, abstracted business flow events are distributed to a cloud infrastructure monitoring agent. This could be a started web shop request at customer *Bob's* web server from *Src-IP 1.2.3.4*. A more abstracted event gets sent to the cloud event processing system (CEP engine) to detect (possible) user overlapping security incidences. This could be the number of not completed web shop transactions originated by *SRC-IP 1.2.3.4* to predetect a

Denial of Service attack. If the CEP engine classifies a behaviour as anomalous certain actions can be executed, like warning the cloud provider's Computer Emergency Response Team, adjusting firewall settings or informing the cloud customer's admin.

How SAaaS addresses the cloud security issues

1) *Monitoring of cloud instances* User VMs running in a cloud infrastructure are equipped with SAaaS agents. The user utilizes a Security Business Flow Language describing which VM components are to be monitored, which behaviour of this VM is considered "normal" and how to alert in case of system security suspicion e.g., open network connection without a preceding legitimate request. The status gets conditioned in a user friendly format in a web portal - the SAaaS security dashboard. Continuous monitoring creates transparency about the security status of a user's cloud instances hence increasing the user's trust into the cloud environment. This addresses the cloud security problem *2) Missing security monitoring in cloud infrastructure* and mitigates problem *3) Defective isolation of shared resources* presented in (Section Cloud specific security issues in focus).
2) *Environment awareness of cloud instances* One reason of using a cloud infrastructure is to benefit from its scalability attributes. In this context it is most often used to deal with usage peeks, for example if a new version of a software gets released and huge download requests are expected. Characteristic to peeks is that they are mostly foreseeable and limited to a certain time frame. Therefore cloud user design their cloud application to start new instances if a certain threshold is reached to provide service availability. This results in two challenges for cloud security:
    2.1) *IaaS upscaling - business driven* Since a user's infrastructure can change rapidly (grow, shrink) in case of a peek scenario the incident detection system needs to be aware of the peek situation and the defined scalability thresholds. Thus, false positive incident alarms can be avoided if service requests due to newly created VM instances get detected. The monitoring system is aware of the changing infrastructure.
    2.2) *IaaS upscaling - attacker driven* Most of the time scalability thresholds, like "maximum number new VMs to be created" get defined once. Mostly during the design phase for the first peek event. If the peek was managed well by the thresholds they just stay, like defined, although they might be not needed anymore (e.g., until the next software release). This enables a new cloud specific attack: Financial damage due to nefarious abuse of cloud resources. Attacker can cause the creation of new cloud instances up to the scalability threshold by creating a huge number of allowed requests, which do not result in any successful business case e.g., distribution of malicious software. A cloud monitoring needs to be aware of business processes to detect an event of possible misuse of cloud scalability. By enhancing agents with environment awareness cloud security problem *1) Abuse of cloud resources* and *2) Missing security monitoring in cloud infrastructure* are addressed by the SAaaS incident detection architecture.
3.) *Cloud infrastructure monitoring and audit* With the presented SAaaS the security state of the entire cloud environment, especially the cloud management system will be monitored. Of interest are customer data and data path, administrative actions concerning customer's instances (e.g., patch management), incident response time, backup restore time, etc. . This way cross-customer monitoring is used by the cloud provider as well as a 3rd parties, like a security service provider to ensure the overall cloud security state. Standardized interfaces enable security audits of a cloud infrastructure, which can lead to a cloud security certification. This addresses the presented cloud security problem *2) Missing security monitoring in cloud infrastructure* and helps to bring assessable security features to cloud computing.

Conclusion and future work

In this paper, we introduced cloud specific security problems and the Security Audit as a Service incident detection system, which aims to solve them. To mitigate the shortcomings of traditional intrusion detection systems we showed the advantages of using autonomous agents as a source for sensor information. We explained how incident detection in clouds can be enhanced by adding business flow information to the presented agents. Business flows correlated with monitoring configuration can be modeled by the introduced Security Service Level Agreements. A first meta model was shown. Since SAaaS agents can be moved business flow dependent to cloud instances during runtime the system acknowledges a cloud flexibility and scalability advantages. It has been shown in several examples, that behaviour monitoring can detect cross customer incidents, which for example can help to limit Denial of Service attacks. An evaluation showed that

the selected agent framework is lightweight enough to support a cloud's changing infrastructure and how the SAaaS architecture addresses the presented cloud specific security problems.

As for future work, we identified the following tasks: a) comprehensive research in anomaly detection algorithms, b) comprehensive research in complex event processing, c) development of the SSLA policy modeler, d) development of SAaaS agents.

Additionally, to reduce network load and to avoid event storms the event data could be compressed before sending. This can be done using standard compression algorithms or to gain even higher compression ratios, by aggregating and preprocessing the data to create higher level information (e.g. information about the amount of connections of each host in the last x minutes). Research has to be done how this can be efficiently achieved for different types of data.

Endnote

[a] IEEE Computer Society standards organization for agent-based technology and its interoperability with other technologies.

Competing interests
The authors declare that they have no competing interests.

Authors' contribution
FD drafted the manuscript, carried out the background work to identify the cloud specific security problems, designed the SAaaS architecture and the first prototype of the Security Business Flow rules. He designed the agents' structure and participated in the system's evaluation. He also contributed to all of the other sections and took the lead on writing the article. CR participated the problem definition and contributed to the design of the SAaaS architecture. He also participated in the design of the agents' structure and contributed to the overall writing of the paper. MK participated in the problem definition and contributed to the overall writing of the paper. AP primarily contributed to the Security Anomaly Detection through Behaviour Analysis section and added to the Related work. He also optimized the SAaaS architecture. NC primarily contributed to the conceptualization of the paper and contributed to the overall writing of the paper. All authors have made substantial contributions and have been part of drafting the manuscript. All authors read and approved the final manuscript.

Acknowledgements
This research is supported by the German Federal Ministry of Education and Research (BMBF) through the research grant number 01BY1116. The authors would also thank Mr. Christian Fischer for his support with the performance tests of the SAaaS agents.

Author Details
[1]Furtwangen University, Cloud Research Lab, Robert-Gerwig-Platz 1, 78120 Furtwangen, Germany. [2]Centre for Security, Communications and Network Research University of Plymouth, Plymouth, UK.

Received: 27 January 2012 Accepted: 5 June 2012
Published: 12 July 2012

References

1. Brunette G, Mogull R (2009) Security Guidance for critical areas of focus in Cloud Computing V2. 1. CSA (Cloud Security Alliance) USA 1. http://www.cloudsecurityalliance.org/guidance/csaguide.v2
2. Catteddu D, Hogben G (2009) Cloud Computing - Benefits, risks and recommendations for information security. ENISA (Eur Network and Inf Security) Europe 1. http://www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-risk-assessment
3. Vaquero L, Rodero-Merino L, Moran D (2011) Locking the sky: a survey on IaaS cloud security. Computing 91: 93–118
4. Chen Y, Paxson V, Katz RH (2010) What's New About Cloud Computing Security? Technical Report UCB/EECS-2010-5, EECS Department, University of California, Berkeley
5. Mell P, Grance T (2009) Effectively and Securely Using the Cloud Computing Paradigm. Tech. rep., US National Institute of Standards and Technology, http://csrc.nist.gov/groups/SNS/cloud-computing
6. Business Insider (2011) Inside Amazon's Cloud Disaster. http://www.businessinsider.com/amazon-outage-enters-its-second-day-lots-of-sites-still-down-2011-4
7. Business Insider (2011) Amazon's Cloud Crash Disaster Permanently Destroyed Many Customers' Data. http://www.businessinsider.com/amazon-lost-data-2011-4
8. Amazon Service Health Dashboard (2011). http://status.aws.amazon.com,http://status.aws.amazon.com
9. Dölitzscher F, Reich C, Sulistio A (2010) Designing Cloud Services Adhering to Government Privacy Laws. In Proceedings of 10th IEEE International Conference on Computer and Information Technology (CIT 2010) 930–935. Furtwangen, Germany
10. Spafford EH, Zamboni D (2000) Intrusion detection using autonomous agents. Comput Networks 34(4): 547–570. Recent Advances in Intrusion Detection Systems
11. Wang C, Wang Q (2010) Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In INFOCOM , 2010 Proceedings IEEE 1–9. Lou W
12. Zhu Y, Ahn G, Hu H, Yau S (2011) Dynamic Audit Services for Outsourced Storages in Clouds. Services Comput, IEEE Trans on 99: 1
13. Massonet P, Naqvi S, Ponsard C, Latanicki J, Rochwerger B, Villari M (2011) A Monitoring and Audit Logging Architecture for Data Location Compliance in Federated Cloud Infrastructures. In Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW). 2011 IEEE International Symposium on: 1510–1517
14. Cloud Security Alliance (2011) Governance, Risk Management and Compliance (CSR). [https://cloudsecurityalliance.org/research/grc-stack/]
15. Hoss C, Johnston S, Reese G, Sapiro B (2010) CloudAudit 1.0 - Automated Audit, Assertion, Assessment, and Assurance API (A6) Tech. rep. Internet Engineering Task Force - Internet Draft - Experimental. [https://tools.ietf.org/html/draft-hoff-cloudaudit-00]
16. Cloud Security Alliance (2011) CloudTrust Protocol Information Overview Powerpoint. [https://cloudsecurityalliance.org/research/ctp/]
17. EuroCloud Deutschland_eco eV (2011) Eurocloud star audit saas certificate. [http://www.saas-audit.de]
18. Nascimento G, Correia M (2011) Anomaly-based intrusion detection in software as a service. In Dependable Systems and Networks Workshops (DSN-W) 2011 IEEE/IFIP 41st International Conference on 19–24. [http://dx.doi.org/10.1109/DSNW.2011.5958858]
19. Banerjee A, Chandola V, Srivastava J, Lazarevic A (2008) Anomaly Detection: A Tutorial. In SIAM International Conference on Data Mining. [http://www.siam.org/meetings/sdm08/TS2.ppt]
20. Lazarevic A, Ertoz L, Kumar V, Ozgur A, Srivastava J, A comparative study of anomaly detection schemes in network intrusion detection (2003) 25–36. [http://www.siam.org/proceedings/datamining/2003/dm03_03LazarevicA.pdf]
21. Garcia-Teodoro P, Daaz-Verdejo J, Macia-Fernandez G, Vazquez E (2009) Anomaly-based network intrusion detection: Techniques, systems and challenges. Comput Security 28(1–2): 18–28. [http://linkinghub.elsevier.com/retrieve/pii/S0167404808000692]
22. Patcha A, Park JM (2007) An overview of anomaly detection techniques: Existing solutions and latest technological trends. Comput Networks 51(12): 3448–3470. [http://linkinghub.elsevier.com/retrieve/pii/S138912860700062X]
23. Roschke S, Cheng F, Meinel C (2009) Intrusion Detection in the Cloud. In Dependable, Autonomic and Secure Computing, 2009 729–734. [http://dx.doi.org/10.1109/DASC.2009.94]
24. Vieira K, Schulter A, Westphall CB, Westphall CM (2010) Intrusion Detection for Grid and Cloud Computing. It Professional 12(4): 38–43. [http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5232794]

25. Raj H, Schwan K (2009) Extending virtualization services with trust guarantees via behavioral monitoring. In Proceedings of the 1st EuroSys Workshop on Virtualization Technology for Dependable Systems, VDTS'09 24–29. New York: ACM
26. Wei J, Zhang X, Ammons G, Bala V, Ning P (2009) Managing security of virtual machine images in a cloud environment. In Proceedings of the 2009 ACM workshop on Cloud computing security, CCSW '09 91–96. New York: ACM. [http://doi.acm.org/10.1145/1655008.1655021]
27. Antonopoulos A (2007) Securing Virtualized Infrastructure: From Static Security to Virtual Shields. Tech. rep., Nemertes Research. [http://hackreport.net/wp-content/uploads/2007/03/nemertes-issue-paper-securing-virtualized-infrastructure.pdf]
28. Schaaf M, Koschel A, Grivas SG, Astrova I (2010) An active DBMS style activity service for cloud environments. In Cloud Computing 2010 : The First International Conference on Cloud Computing, GRIDs, and Virtualization in Cloud Computing 2010 in ComputationWorld 2010 IARIA 80–85
29. Wooldridge M Agent Applications, Research, and Technology. [http://www.csc.liv.ac.uk/~mjw/Prof_Michael_Wooldridge_-_Home_Page]
30. Rao AS, Georgeff MP (1995) BDI Agents: From Theory to Practice. In PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS.ICMAS-95 312–319
31. Lui CL, Fu TC, Cheung TY (2005) Third International Conference on, Volume 1 131– 136. ICITA 2005
32. Chirumamilla M, Ramamurthy B (2003) Agent based intrusion detection and response system for wireless LANs. In Communications, 2003. ICC '03 492–496
33. Dasgupta D, Gonzalez F, Yallapu K, Gomez J, Yarramsettii R (2005). CIDS: An agent-based intrusion detection system, Vol. 24. [http://www.sciencedirect.com/science/article/pii/S0167404805000179]
34. Krügel C, Based MA (2001) SPARTA - a mobile agent based intrusion detection system. In Proceedings OF THE IFIP CONFERENCE ON NETWORK SECURITY (I-NETSEC). Berlin, Heidelberg: Kluwer Academic Publishers
35. Balasubramaniyan J, Garcia-Fernandez J, Isacoff D, Spafford E, Zamboni D (1998) An architecture for intrusion detection using autonomous agents. In Computer Security Applications Conference, 1998, Proceedings., 14th Annual, pp 13–24
36. Mo Y, Ma Y, Xu L (2008) Design and implementation of intrusion detection based on mobile agents. In IT in Medicine and Education, 2008. ITME 2008. IEEE International Symposium on, pp 278–281
37. Sen J, Sengupta I, Chowdhury P (2006) An architecture of a distributed intrusion detection system using cooperating agents. In Computing Informatics, 2006. ICOCI '06. International Conference on, pp 1–6
38. Cloud Security Alliance (2010) Top Threats to Cloud Computing V1.0. [https://cloudsecurityalliance.org/topthreats.html]
39. Doelitzscher F, Ardelt M, Knahl M, Reich C (2011) Sicherheitsprobleme für IT Outsourcing basierend auf Cloud Computing. HMD - Praxis der Wirtschaftsinformatik 281: 62
40. Bradshaw JM (1997) An introduction to software agents. Cambridge: MIT Press. pp 3–46
41. Grimshaw D (2011) JADE Administration Tutorial. [http://jade.tilab.com/doc/tutorials/JADEAdmin]
42. Staniford-chen S, Cheung S, Crawford R, Dilger M, Frank J, Hoagl J, Levitt K, Wee C, Yip R, Zerkle D (1996) GrIDS - A Graph Based Intrusion Detection System For Large Networks. In Proceedings of the 19th National Information Systems Security Conference, pp 361–370
43. Schneider J (2007) Systemautomatisierung. In Kircher H(ed) IT, pp 110–123. Berlin Heidelberg: Springer. [10.1007/978-3-540-46165-4_9]. [http://dx.doi.org/10.1007/978-3-540-46165-4_9]
44. Sulistio A, Reich C, Dölitzscher F (2009) Cloud Infrastructure &; Applications - CloudIA. In Proceedings of the 1st International Conference on Cloud Computing (CloudCom'09), pp 583–588. Beijing, China. Springer-Verlag, Berlin, Heidelberg. [http://dx.doi.org/10.1007/978-3-642-10665-1_56]
45. Cucurull J, Marti R, Navarro-Arribas G, Robles S, Overeinder B, Borrell J (2009) Agent mobility architecture based on IEEE-FIPA standards. Comput Commun 32(4): 712–729
46. Winter P, Lampesberger H, Zeilinger M (2011) Anomalieerkennung in Computernetzen. Datenschutz und Datensicherheit-DuD 35(4): 235–239. [http://www.springerlink.com/index/K12443031LQ84V55.pdf]
47. OWASP (2010) Top 10. The Ten Most Critical Web Application Security Risks. Tech. rep., OWASP The Open Web Application Security Project 2010. [http://owasptop10.googlecode.com/files/OWASP202010.pdf]
48. Kortchinsky K (2009) Cloudburst. In Black Hat USA. [http://www.blackhat.com/presentations/bh-usa-09/KORTCHINSKY/BHUSA09-Kortchinsky-Cloudburst-PAPER.pdf]
49. Elhage N (2011) Virtunoid: A KVM Guest → Host privilege, escalation exploit. In Black Hat USA. [http://media.blackhat.com/bh-us-11/Elhage/BH_US_11_Elhage_Virtunoid_WP.pdf]
50. Hofmeyr SA, Forrest S, Somayaji A (1998) Intrusion Detection using Sequences of System Calls. J Comput Security 6: 151–180
51. Cortese E, Quarta F, Vitaglione G, Lab TI, Direzionale C, Message J, System T (2002) Scalability and Performance of JADE Message Transport System. Journal of Analysis of Suitability for Holonic Manufacturing Systems 3(3): 52–65. [http://citeseer.ist.psu.edu/vitaglione02scalability.html]
52. BootChart (2012). [http://wiki.ubuntuusers.de/BootChart]
53. Cloud Research Lab - Furtwangen University (2012) Security Audit as a Service project website. [http://wolke.hs-furtwangen.de/currentprojects/saaas]

doi:10.1186/2192-113X-1-9
Cite this article as: Doelitzscher *et al.*: An agent based business aware incident detection system for cloud environments. *Journal of Cloud Computing: Advances, Systems and Applications* 2012 **1**:9.