2011 Third IEEE International Conference on Coud Computing Technology and Science

An autonomous agent based incident detection system for cloud environments

Frank Doelitzscher*, Christoph Reich*, Martin Knahl* and Nathan Clarke‡
*Furtwangen University - Cloud Research Lab
Robert-Gerwig-Platz 1, 78120 Furtwangen, Germany
Email: {Frank.Doelitzscher, Christoph.Reich, Martin.Knahl}@hs-furtwangen.de
†Centre for Security, Communications and Network Research - University of Plymouth
Plymouth PL4 8AA, United Kingdom
Email: N.Clarke@plymouth.ac.uk

***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 build upon intelligent autonomous agents, which are aware of underlying business flows of deployed cloud instances. Business flows are modelled in form of Security Service Level Agreements, which enable the SAaaS architecture to be flexible and to supported cross customer event monitoring of a cloud infrastructure. As contribution of this paper we provide a high-level design of the SAaaS architecture, an introduction into the concept of Security Service Level Agreements, a first prototype of an autonomous agent and an evaluation about, which cloud specific security problems are addressed by the presented architecture.**

I. Introduction

Enterprise analysts and research identified cloud specific security problems as the major research area in cloud computing [1], [2], [3], [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, multitenancy, access from everywhere, on-demand availability and 3rd party hosting. Although existing recommendations (ITIL), standards (ISO 27001:2005 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], [5]. In classic IT infrastructures security audits and penetration tests are used to document a datacenter'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 it 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 an up scaled cloud service is caused by a higher demand of business requests or by a hacker misuse. As a first approach to a continuous auditing system for cloud environments we are presenting a monitoring environment in this paper.

While monitoring the security of large IT infrastructures with distributed sensors as input feeds of an 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 [6]. 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, analyse and aggregate information and distribute it considering the underlying business process. This data interpretation is achieved using a Security Service Level Agreements (SSLA) policy modelling engine that allows to define monitoring events and consider business process flows. The usage of autonomous agents enables a behaviour anomaly detection of cloud components while maintaining the cloud specific flexibility. Our system respects the following cloud specific attributes:

- A high number and complexity of distributed systems
- A often changing infrastructure (e.g., service scalability or user driven)
- An interpretation of the cloud activation in respect to business processes

In the remainder of this paper, we first describe related work (Section II). Continuing with selected cloud specific security issues (Section III), which are targeted by our approach. Section IV 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 V evaluates why the paradigm of autonomous agents is valuable for incident detection in cloud environments and how the presented SAaaS architecture addresses the presented cloud security issues. Section VI concludes the paper and informs about future work.

978-0-7695-4622-3/11 $26.00 © 2011 IEEE
DOI 10.1109/CloudCom.2011.35

II. Related Work

First, current literature identifying cloud security issues is shown. Followed by a discussion of other cloud security research projects in contrast to SAaaS and the usage of autonomous agents for systems security.

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 traditionally computing solved by presented technology frameworks it also demands an architecture enabling "mutual trust" for cloud user and cloud provider. Both papers confirm and complete the cloud specific security issues identified by our research. Furthermore, they demand a two-way trust enabling architecture for cloud infrastructures and the ability of "choosable security primitives with well considered defaults" [4]. The SAaaS architecture will provide this mutual trust. SAaaS' Security Service Level Agreements enable the user to define its own security levels and to choose from a spectrum of security subsystems as demanded by [4].

Raj et al. [7] introduce a virtualization service implemented as Xen Virtual Machine (VM) extensions, which provides Role Based Access Control (RBAC) 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.

Zamboni et al. present in [8] 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. Mo et al. introduce in [9] 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 [10]. The presented advantages apply for our SAaaS agents as well.

In [11] 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 [12] and [3] 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.

III. Cloud Specific Security Issues In Focus

This section describes cloud specific problems which we target by our approach. 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], [13], [2], [3], [4]) we made a comparison between classic IT-Housing, IT-Outsourcing and cloud computing. The following cloud specific security issues were identified as solvable by the SAaaS system.

1) 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 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: authorisation, integrity, non-repudiation and privacy. Strong monitoring of user activities on all cloud infrastructure components is necessary to increase transparency.

2) 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 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 customer blindly depend on a working patch management of the cloud provider. Cloud provider usually do not disclose information on this processes

to customers. By this issue the information security core principles integrity, availability and non-repudiation.

3) 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 provider advertise implemented reliability measures to pretend data loss, like replicating data up to six times. In contrast customer 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.

IV. 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 Service Level Agreements (SSLA). Following, a first SAaaS agent prototype is presented.

A. Agent Definition

An agent can be defined as [14]:
"... 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 data from other instances e.g., the policy module, and send information to other instances, like other 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.

B. How agents can improve incident detection

Incident detection in cloud environments is a non trivial task due to a cloud's characteristics. Especially the frequently changing infrastructure poses a big challenge to define something, like normal behaviour and detect abnormal behaviour. Therefore it is important to have a high number of sensors capturing simple events. Simple events need to be preprocessed and abstracted to complex events, reducing the possibility

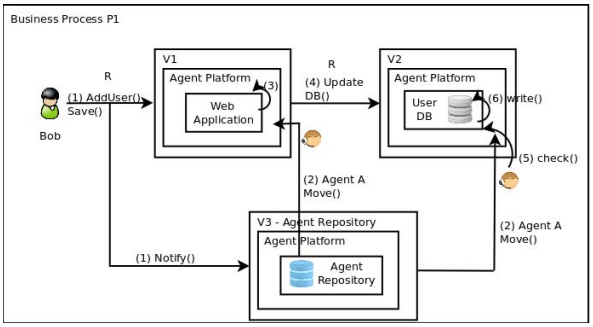


Fig. 1. Business flow dependend agent deployment

"of event storms". 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 and also can 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.

Imagine a business process of a web application `P1` (depicted in Figure 1) where user Bob adds a new user to a user database by filling out a web form. By pressing the "Save" button a legal request R (1) gets executed as part of business process `P1`. Agents `A` that monitor the database access get moved (2) at the process start `R` to the request-executing VM(s) `V1,V2`, monitoring the data access (3),(5) during the process time. After the data process has been comleted (4), (6) they delete themselves from `V1, V2` after `P1` has been finished.

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 IV-D SSLAs). 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 *complex events* can be created. Given the scenario of a successful unauthorised login of an attacker at a virtual machine `VM2`, misusing a webserver'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

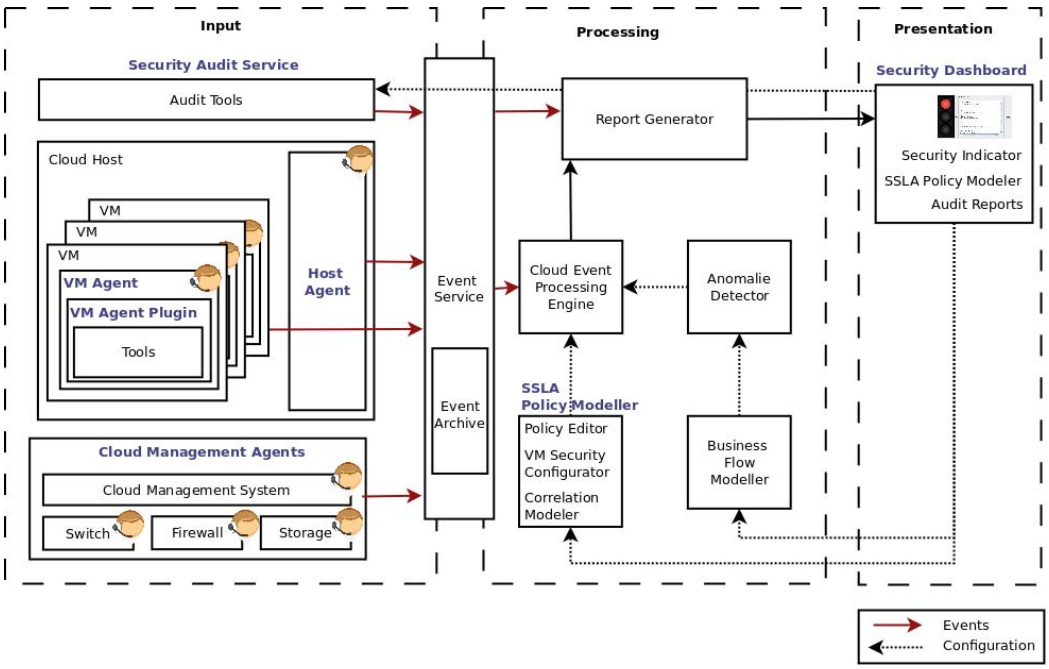


Fig. 2. SAaaS event processing sequence

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 (multiple simple agents can exits on the same platform e.g., inside a VM), preprocessing detected simple events and sharing of information between logical associated agents the network load can be reduced. Furthermore, this makes the system more scalable by reducing data sent to upper system layers. This is introduced and used in [15]. Combining events from system agents (VM agent, host agent - see Figure 2) 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.

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.

C. SAaaS Architecture Components

Figure 2 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.

`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 abnormal 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 monitor and policy enforcing tools. Therefore it loads 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 SSLA 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 freezed config file). Thus, every occurrence gets first preprocessed by an agent, which reduces communication between VM agent and Cloud Management Agent. Self learning algorithms will be evaluated to improve an agents' intelligence. The *Security Service Level Agreements policy modeller* consists of a policy editor, a VM security configurator and a semantic correlation modeller to enable cloud user to design SSLA and security policies. An example for a SSLA 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 send from the Security Audit Service to the corresponding agents. Using the monitoring information of the distributes agents in combination with the SSLAs a *cloud behaviour model* is build up for every cloud user. SSLAs 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 modelled business flows from the *Business Flow 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 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 are organised 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 standardised message format which is not defined yet. Our first prototype implements the Intrusion Detection Message Exchange Format (IDMEF). Events are also stored in an *Event Archive*.

D. Security Service Level Agreements

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 Security Service Level Agreements (SSLA) will be developed. Consider the following example:

Given a typical web application system consisting of a load balancer, a webserver and a database backend deployed at three VMs in a cloud as depicted in Figure 3. All VMs are equipped with SAaaS agents. The user's administrator installs each VM with the necessary software, e.g., Apache webserver, Tomcat load balancer, MySQL database.

After the functional configuration of the installed software is finished the monitoring configuration gets designed in form of Security Service Level Agreements. 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, SSLAs allow 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 abnormal behaviour

```
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>
```

Listing 1. SSLA modelling example

which does not occur in a valid business process flow. Therefore a monitoring event should be generated. SSLAs need to be modelled 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 a modelled SSLA is shown in Listing 1. Line 4 - 19 describe possible technical rules, while line 21 - 26 modells 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.

This is a first example - definition of the SSLA modelling language is due to future work. To reduce complexity a graphical policy modeller needs to be developed. For typical cloud usage components e.g., a webserver, profiles will be prepared, which modells necessary dependencies, like config directory, associated processes or used network protocols.

E. 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 [16] is build 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[1] specifications. Furthermore, it already provides a user interface, which alleviates agents creation, deployment and testing.

Figure 3 illustrates a basic agent architecture we already assumed in the SAaaS Use Case presented in Section IV-D. 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 [17]. To provide monitoring functionality a VM agent interacts through agent plugins with stand-alone tools, like process monitor, intrusion detection system or anti virus scanner, as depicted in Figure 3. To harness the potential of cloud computing an agent can be deployed to a VM on-demand according to the SSLA 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. [18] was integrated. This supports the presented advantage of deploying agents on-demand if a designed business process flow was started (as described in Section IV-B). Though this implementation is up to future work.

As a first prototype, a two layered agent platform was developed, consisting of a *VM agent* running inside a VM, a *Cloud Management Agent* running as a service at a dedicated VM feeding information to 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 agent 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 4 informs about occurring events. Figure 4(a) shows the VM's state before an attack. After launching an attack, the Security Dashboard indicator light changes its colour as set defined in a simple severity matrix and gives short information about the monitored event (Figure 4(b)).

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

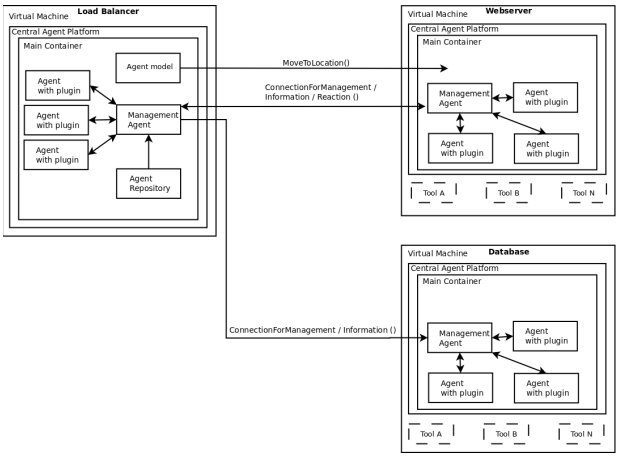


Fig. 3. Basic SAaaS agent design

V. Evaluation

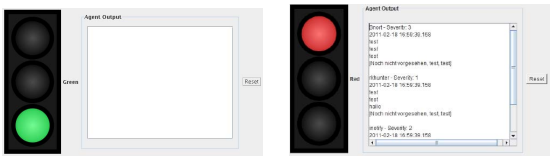
In this section we first evaluate the usage of autonomous agents for the SAaaS architecture. We then discuss scenarios to show how the SAaaS architecture addresses the introduced cloud security issues presented in Section III.

A. Evaluation of the SAaaS Architecture

1) Agent Performance: It is essential for the SAaaS architecture that the agents are very efficient not causing a high offset of resource consumption. JADE agent performance is very low as demonstrated by E.Cortese et al. in [19]. They show that CPU overhead is very low. Average round trip time of a message between to agents (request message, answer message) with a message content of seven characters takes only 13,4 ms. Jurasovic et al. [20] show that even with increasing message size the round trip time does not increase significantly. Also the message overhead by the agent communication does not increase significantly with increasing message size. Details about the used test lab are given in the mentioned literature.

In our first prototype, we wanted to see how fast an agent can be deployed to a new platform. 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. Over all test runs we confirmed that the average time for an agent move is below 1,5 seconds. This proves the applicability of the JADE agent platform to support the presented SAaaS use case.

2) Message Reduction by Business Process Awareness: In case of a monitoring event is produced it first will be processed by the SAaaS agent, which is initiating the 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.



(a) No events registered. (b) Events detected.

Fig. 4. Cloud security dashboard prototype

Imagine a 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.

3) 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 abnormal 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.

B. 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 defines Security Service Level Agreements 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 III.

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 cloud security problem *2) Missing security monitoring in cloud infrastructure* and helps to bring assessable security features to cloud computing.

VI. 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. Furthermore, complex

cross customer events can be produced to identify abnormal behaviour in a frequently changing cloud infrastructure. This 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.

ACKNOWLEDGMENT

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

REFERENCES

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

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

[3] L. Vaquero, L. Rodero-Merino, and D. Morn, "Locking the sky: a survey on iaas cloud security," *Computing*, vol. 91, pp. 93–118, 2011.

[4] 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.

[5] F. Dölitzscher, 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.

[6] E. H. Spafford and D. Zamboni, "Intrusion detection using autonomous agents," *Computer Networks*, vol. 34, no. 4, pp. 547 – 570, 2000, recent Advances in Intrusion Detection Systems.

[7] H. Raj and K. Schwan, "Extending virtualization services with trust guarantees via behavioral monitoring," in *Proceedings of the 1st EuroSys Workshop on Virtualization Technology for Dependable Systems*, ser. VDTS '09. New York, NY, USA: ACM, 2009, pp. 24–29.

[8] 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.

[9] 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.

[10] 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.

[11] 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. [Online]. Available: http://doi.acm.org/10.1145/1655008.1655021

[12] Andreas Antonopoulos, "Securing Virtualized Infrastructure: From Static Security to Virtual Shields," Nemertes Research, Tech. Rep., 2007.

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

[14] J. M. Bradshaw, *An introduction to software agents*. Cambridge, MA, USA: MIT Press, 1997, pp. 3–46.

[15] 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.

[16] Anthony Sulistio and Christoph Reich and Frank Dlitzscher, "Cloud Infrastructure & Applications - CloudIA," in *Proceedings of the 1st International Conference on Cloud Computing (CloudCom'09)*, Beijing, China, 12 2009.

[17] David Grimshaw, "JADE Administration Tutorial," http://jade.tilab.com/doc/tutorials/JADEAdmin, 07 2011.

[18] J. Cucurull, R. Mart, 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.

[19] E. Cortese, F. Quarta, G. Vitaglione, T. I. Lab, C. Direzionale, J. Message, and T. System, "Scalability and performance of jade message transport system," 2002.

[20] K. Jurasovic, G. Jezic, and M. Kusek, "A performance analysis of multi-agent systems." *ITSSA*, vol. 1, no. 4, pp. 335–342, 2006. [Online]. Available: http://dblp.uni-trier.de/db/journals/itssa/itssa1.html\#JurasovicJK06