Incident Detection for Cloud Environments

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

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

***Abstract*—Security and privacy concerns hinder a broad adoption of cloud computing in industry. In this paper we identify cloud specific security risks and introduce the cloud incident detection system Security Audit as a Service (SAaaS). SAaaS is built on autonomous distributed agents feeding a complex event processing engine, informing about a cloud's security state. In addition to technical monitoring factors like number of open network connections business process flows can be modelled to detect customer overlapping security incidents. In case of identified attacks actions can be defined to protect the cloud service assets. As contribution of this paper we provide a high-level design of the SAaaS architecture and a first prototype of a virtual machine agent. We show how an incident detection system for a cloud environment should be designed to address cloud specific security problems.**

***Keywords*-cloud computing; security; autonomous agents.**

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 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 [5][1] is showing they are not sufficient for cloud environments. 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 considered as normal. Knowledge of underlying business processes is needed, for example that a new Virtual Machine (VM) gets created after a user's scalability threshold for its Webshop has been exceeded.

Therefore, we introduce an incident detection system for cloud environments based on autonomous agents, which collect data directly at the source, analyse and aggregate information and distribute it considering the underlying business process. To achieve this data interpretation gets supported by a Security Service Level Agreements (SSLA) policy modelling engine that allows to define monitoring events which 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:

- high number of distributed systems
- Frequently changing infrastructure due to the scalability advantages
- Multitenancy of users who are "owning" participating systems with administrator rights.

In the remainder of this paper, we first describe related work (Section II). Section III introduces the Security Audit as a Service (SAaaS) architecture which targets to solve the mentioned problems above. Why the paradigm of autonomous agents is valuable for incident detection in cloud environments is discussed in Section IV and a first SAaaS agent prototype gets presented. Subsequently (Section V), we discuss cloud specific security issues, which are addressed by the presented SAaaS architecture. Section VI concludes the paper and informs about future work.

II. Related Work

This section covers related research work. First, we show current literature identifying cloud security issues. Following, we are discussing 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 categorises the most widely accepted cloud security issues into three different domains of the Infrastructure as a Service (IaaS) model: machine virtualization, network

 ISBN: 978-1-61208-174-8

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.

Raj et al. [6] introduce a virtualization service implemented as Xen VM extensions, which provides Role Based Access Control (RBAC) based on a trust value of a VM. This trust is based upon a VMs attributes like number of open network connections. Access to different cloud services like file access is given on a VMs' 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 [7] 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 an open (cloud) environment where incidents like abuse of resources needs to be detected. Mo et al. introduce in [8] an IDS based on distributed agents using the mobile technology. They show how mobile agents can support anomaly detection thereby overcoming the flaws of traditional intrusion detection in accuracy and performance. The paradigm of cooperating distributed autonomous agents and its corresponding advantages for IDS' is shown by Sengupta et al. in [9]. The presented advantages apply for our SAaaS agents as well.

III. Security Audit as a Service architecture

While distributed monitoring sensors are a well known procedure in intrusion detection systems (IDS) for traditional IT systems they do not cover the security needs of cloud environments. They are not flexible enough to monitor such a complex environment in a user manageable fashion. Mostly because existing architectures are built around a single monolithic entity which is not scalable enough to do data collection and processing in an efficient and meaningful way [10]. To mitigate this, we propose an autonomous agent-based intrusion detection system for cloud computing: Security Audit as a Service (SAaaS). The SAaaS architecture aims to support the following scenarios.

A. SAaaS Target Scenarios

A) Monitoring and audit of cloud instances User VMs run in a cloud infrastructure are equipped with an SAaaS agent. The user defines Security Service Level Agreements defining which behaviour of this VM in considered "normal", which VM components are to be monitored and how to alert in case of system deviation from the defined manner. The status gets conditioned in a user friendly format in a webportal - the SAaaS security dashboard. This continuous monitoring creates transparency about the security status of a user's cloud VMs hence increasing the user's trust into the cloud environment.

B.) Cloud infrastructure monitoring and audit The security status of the entire cloud environment, especially the cloud management system, access to customer data and data paths are monitored. This way customer-spanning monitoring is used by the cloud provider as well as a 3rd party, like a security service provider to ensure the overall cloud security status. Standardised interfaces enable security audits of a cloud infrastructure which can lead to a cloud security certification.

B. Typical SAaaS Use Case

To fullfill the presented scenarios we are proposing to use an autonomous agent system to monitor cloud environments. Before explaining the advantages of autonomous agents in detail we briefly want to explain the whole SAaaS event processing sequence. To support this consider the following example. Given a typical web application system consisting of a webserver, a load balancer and a database backend deployed at three VMs in a cloud. All VMs are equipped with SAaaS agents. The user's administrator installs the three VMs with the necessary software, e.g., Apache webserver, Tomcat load balancer, MySQL database. After the functional configuration the monitoring configuration gets designed in form of **S**ecurity **S**ervice **L**evel **A**greements (SSLAs). 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 which does not occur in a valid business process flow. Therefore, a monitoring event should be generated. If an event gets generated it first will be preprocessed by the SAaaS agent which is responsible for the monitoring target. This is important to reduce the overall messages sent to the cloud event processing system especially in large cloud computing environments. The SAaaS agent filters out possible VM dependent events like a started web application session from *IP 1.2.3.4*. A

 ISBN: 978-1-61208-174-8

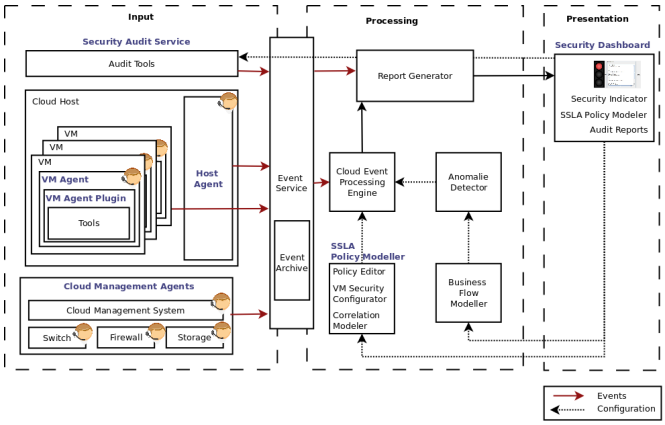

Figure 1. SAaaS event processing sequence

more abstracted event gets send to the cloud event processing system (a complex event processing (CEP) engine) to detect (possible) user overlapping security incidences. This could be a message containing the number of not completed web shop transactions by IP *IP 1.2.3.4* to pre-detect a Denial of Service attack. If the CEP engine detects an abnormal behaviour actions can be executed like warning the cloud provider's Computer Emergency Response Team, adjusting firewall settings or informing the cloud customer's admin.

Figure 1 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.

Input: The SAaaS architecture gets its monitoring information from distributed agents which are positioned at key points of the cloud's infrastructure to detect abnormal activities in a cloud environment. Possible key points are: running VMs of cloud users, the VM hosting systems, 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: 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 Security Service Level Agreements 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: 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`.

IV. Anomaly Detection Using Autonomous Agents

In this section, we are showing the advantages of using distributed autonomous agents for incident detection in a cloud environment. Therefore, we first give a definition what can be considered as an autonomous agent.

A. Agent Definition

An agent can be defined as [11]:
"... 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

 ISBN: 978-1-61208-174-8

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 its characteristics as discussed in Section I. Therefore, it is important to have a high number of sensors capturing simple events. Preprocessed and combined complex events can be generated reducing the possibility of "event storms". Combined with knowledge about business process flows it will be possible to detect security incidents like discussed in Section V, while keeping the network load low.

The usage of autonomous agents delivers this possibility because agents are *independent units* that 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` 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 gets executed as part of business process `P1`. An agent `A` monitoring database access can get moved at the beginning of `R` to the request-executing VM `V1`, monitoring the data access during process time and gets deleted from `V1` after `P` is 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.

While single agents can monitor simple events (e.g., user login on VM) and share them with other agents *complex events can be detected*. Given the scenario of a successful 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 quarantine environment, alerting the user and simultaneously starting a fresh instance of `VM2` based on its VM image.

By ordering agents in a hierarchical structure and preprocessing of detected events reduces network load originated from the incidents detection system. Furthermore this makes the system more scalable by reducing data sent to upper system layers. This is introduced and used in [12]. Combining events from system deployed agents (e.g., VM agent, host agent) and infrastructure monitoring agents (network agent, firewall agent) incident detection is not limited to either host or network based sensors which is especially important for the characteristics of cloud environments.

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 the damage is restricted to this single agent or a small subset of connected agents which are requiring information from this agent.

C. 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 [13] 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 (IEEE Computer Society standards organisation for agent-based technology and its interoperability with other technologies) specifications. Furthermore it already provides a user interface which alleviates agents creation, deployment and testing.

Figure 2 illustrates a basic agent architecture we already assumed in the SAaaS Use Case presentation in Section III-B. It shows three SAaaS VM agents. Agents life 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 [14]. 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

 ISBN: 978-1-61208-174-8

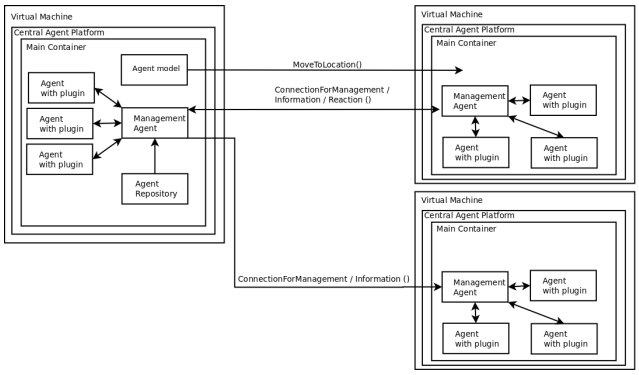

Figure 2. Basic SAaaS agent design

2. 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. [15] 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, showing a signal light indicator informed about occurring events. When started it shows a green light. After launching an attack, the Security Dashboard indicator light changes its colour to yellow or red as defined in a severity matrix given the type of detected attack.

D. Agent Performance Test

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 [16]. 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. [17] 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.

V. Discussion - Cloud Specific Security Issues addressed by SAaaS

The German Federal Office for Information Security publishes the IT baseline protection catalogues enabling enterprises to achieve an appropriate security level for all types of information. The catalogues were extended by a special module covering virtualization in 2010. In a comprehensive study on all IT baseline protection catalogues as well as current scientific literature available [1][18][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:

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

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

which informs user 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. The presented SAaaS scenario B) Cloud infrastructure monitoring and audit addresses this problem.

Defective isolation of shared resources In cloud computing isolation in depth is not easily achievable due to usage of rather complex virtualization technology like VMware, Xen or KVM. Persistent storage is shared between customers as well. Cloud 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 contain its damage by the presented actions.

VI. Conclusion and Future Work

In this paper, we introduced the Security Audit as a Service architecture to mitigate the shortcoming traditional audit systems suffer to audit cloud computing environments. We showed the advantages of using autonomous agents as a source for sensor information. We explained how incident detection in clouds can be done by adding business process information to technical monitored events to perform anomaly detection in clouds.

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 modeller, 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.

[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. Doelitzscher, C. Reich, and A. Sulistio, "Designing cloud services adhering to government privacy laws," in *Proceedings of 10th IEEE International Conference on Computer and Information Technology (CIT 2010)*, 2010, pp. 930–935.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

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

[14] D. Grimshaw, "JADE Administration Tutorial," http://jade.tilab.com/doc/tutorials/JADEAdmin, 06.09.2011.

[15] 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.

[16] 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.

[17] K. Jurasovic, G. Jezic, and M. Kusek, "A performance analysis of multi-agent systems." *ITSSA*, vol. 1, no. 4, pp. 335–342, 2006, http://dblp.uni-trier.de/db/ journals/itssa/itssa1.html#JurasovicJK06, 06.09.2011.

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

 ISBN: 978-1-61208-174-8