Governing Bot-as-a-Service in Sustainability Platforms–Issues and Approaches

Downloaded from: https://research.chalmers.se, 2024-04-27 04:10 UTC

Citation for the original published paper (version of record):
Truong, H., Phung, P., Dustdar, S. (2012). Governing Bot-as-a-Service in Sustainability Platforms–Issues and Approaches. Procedia Computer Science, 10: 561-568.
http://dx.doi.org/10.1016/j.procs.2012.06.072

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology.
It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004.
research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 10 (2012) 561 – 568

Procedia Computer Science

The 9th International Conference on Mobile Web Information Systems (MobiWIS)

Governing Bot-as-a-Service in Sustainability Platforms – Issues and Approaches☆

Hong-Linh Truong[a], Phu H. Phung[b], Schahram Dustdar[a]

[a]*Distributed Systems Group, Vienna University of Technology, Austria*
[b]*ProSec Security Group, Chalmers University of Technology, Sweden*

Abstract

The emerging cloud computing models for Internet-of-Things have fostered the development of lightweight applications using cloud services for monitoring and optimizing devices and equipment hosted in distributed facilities. Such applications – called bots in our work – can be composed and deployed with multiple types of governance policies from cloud platforms to distributed hosting environments and they can access not only local data and devices but also cloud data and features. Therefore, it is a great challenge to govern them. In this paper, we discuss governance issues and state-of-the-art on supporting the emerging Bot-as-a-Service in sustainability governance platforms. Based on that we outline our approaches to policy development and enforcement for the Bot-as-a-Service model.

Keywords: bot-as-a-service, governance, policy enforcement, runtime monitoring, cloud computing

1. Introduction

Advances in networking, sensor, data management and cloud computing techniques have fostered the integration of Internet-of-Things into cloud platforms for facility management. Cloud-based sustainability governance platforms [1] have been developed and provided under the cloud computing models for facility monitoring and management. Various sensors and applications for analyzing diverse types of data and for managing devices and equipment for different stakeholders have been introduced in such platforms.

1.1. Motivation

While many techniques have concentrated on sensor integration, data integration and data analytics on top of these platforms [2], we are interested in the development, deployment and execution of (intelligent and context-aware) lightweight applications that can be developed, composed and deployed on-the-fly based on context-specific situations captured from the monitoring and analysis of near-realtime sensor data. Together with techniques for connecting and monitoring devices and equipment, which are concentrated on getting information from these devices and equipment, such lightweight applications can be used to maintain and

☆The work mentioned in this paper is partially funded by the Pacific Control Cloud Computing Lab and by the European WebSand project. We thank our colleagues in the Pacific Control Cloud Computing Lab for fruitful discussion about bots and their applications.

Email addresses: truong@infosys.tuwien.ac.at (Hong-Linh Truong), phung@chalmers.se (Phu H. Phung), dustdar@infosys.tuwien.ac.at (Schahram Dustdar)

1877-0509 © 2012 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.procs.2012.06.072

optimize these devices and equipment. They are crucial as nowadays several types of equipment and devices in large-scale facilities, such as freezers, chillers, and backup power systems, are expensive to maintain and optimize. With a huge amount of monitoring data managed in sustainability governance platforms and multiple stakeholders (e.g., manufactures, maintainers, and operators) using such platforms to monitor and maintain their devices and equipment, it is possible for these applications to automatically learn the operational history of, detect errors of, and to automate repair and support of devices and equipment.

However, these lightweight applications can be developed and deployed to cloud platforms by third-parties, can be created by composing existing applications and functions offered by cloud services and then configured and deployed to facility sites on-the-fly, or may be pre-installed in hosting environments deployed at facility sites while being reconfigured for a specific context. Thus the development, deployment and execution of these applications must be monitored and controlled at runtime to ensure they comply with different governance policies. In our work, these lightweight applications are called bots and they can be developed and executed in the cloud via a Bot Platform-as-a-Service (BoP), offering the Bot-as-a-Service (BaaS) business model for sustainability governance.

1.2. Related work

While many research efforts have concentrated on security for monitoring sensors and data protection in the facility side, such as smart Grids [3], and in the cloud side [4, 5], very little effort has been spent on understanding issues in governing such bots in the cloud. In fact, the concept of combining cloud computing models and bots development has just been emerging: such concepts are not similar to, e.g., mobile agents and their applications for building management [6], due to the complexity and diversity of cloud data, devices, equipment, and stakeholders, although the concept of BaaS will naturally be built upon existing work. Related work on understanding governance issues either focus only on part of the BaaS model (e.g., hosting environments for mobile agents [7] and application stores [8]) or one aspect of governance issues (e.g., security [3, 9, 10], data access [11], or performance [12]). We do a comprehensive review of several governance issues in different phases of bot development, deployment and execution in clouds. Most importantly, we consider relevant governance issues in an emerging model - BaaS in which governance issues span across layers and places as well as associated with business models in the cloud.

1.3. Contributions and paper structure

In this paper, we systematically motivate the need for policy enforcement for bots in BoP. Our contributions are (i) a deep analysis of the scenarios of the bot-as-a-service architecture and its governance issues, (ii) a review of the state-of-the-art enforcement techniques in literature and identified the emerging issues that have not been solved, and (iii) approaches to the policy definition, management, and enforcement framework for the identified requirements and potential future developments.

The rest of this paper is organized as follows: Section 2 overviews BoP. Section 3 characterizes components and interactions in BoP. Section 4 discusses governance issues and state-of-the-art. Section 5 presents our approaches to the governance of bots. We conclude the paper and outline our future work in Section 6.

2. Overview of Bot Platform-as-a-Service

We are focusing on facility monitoring in smart cities, such as using the Galaxy platform (`http://pacificcontrols.net/products/galaxy.html`). In our work, several sensors are deployed in different buildings to monitor building Mechanical, Electrical and Plumbing (MEP) systems and surrounding environments, such as temperature and air quality. In a cloud facility management model, at each facility site, sensor data are aggregated and significant information is propagated to cloud services where online monitoring is performed. At the moment, most of monitoring tasks for devices and equipment are conducted by operators who are going to fix problems detected in facilities. In a recent emerging concept, calling intelligent and context-aware bots, bots can be deployed at the facility sites to detect problems and fix them automatically.

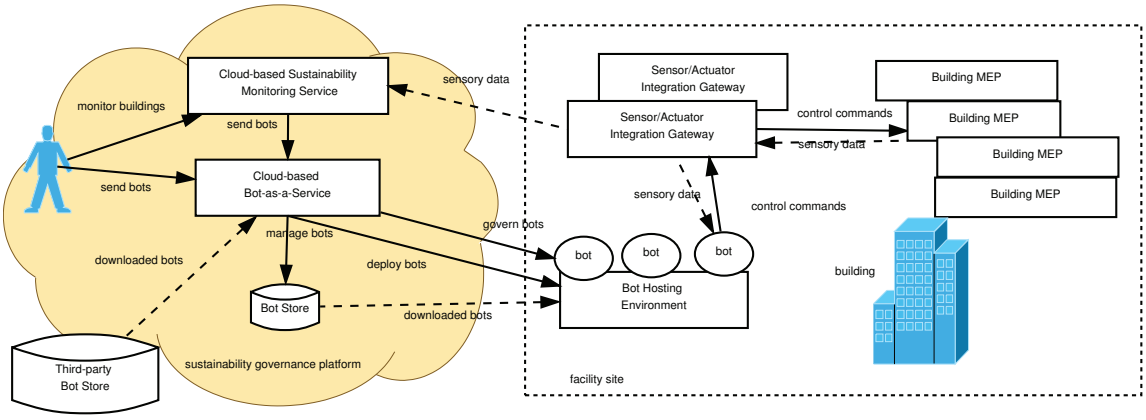

Fig. 1. High level view of Bot Platform-as-a-Service (BoP) consisting of bots, Bot Store, BaaS, and Bot Hosting Environment

A bot is a lightweight application that is executed by a hosting environment. In real systems, bot can be built based on mobile agent platforms, OSGi (`http://http://www.osgi.org`), or plain Java objects. To support the development and execution of bots in the cloud, a Bot Platform-as-a-Service (BoP) can be provided. Figure 1 describes a high level view of BoP, including the following conceptual components:

- Bot-as-a-Service (BaaS): supports the development, composition of bots, management, and deployment of bots, and the definition and management of governance policies for bots.
- Bot Store: stores bots and templates for building bots. Bot Stores can be hosted by the sustainability governance platform or third parties (e.g., a specific company which manages its chillers)
- Bot Hosting Environment: deploys, verifies and executes bots, and monitors bot execution.

Bots and BaaS interact with several other services, sensor/actuator integration gateways, devices and equipment at the cloud and facility sides. For example, when the Monitoring Service detects a possible problem, it can control BaaS to find suitable rules and algorithms to build a bot, which is deployed to the Bot Hosting Environment to analyze device's data and control the device. An important point is that, at runtime, bots are instantiated based on existing problems and bots can be launched from bot stores in cloud service to bot hosting environments at facility sites.

The BoP we present here conceptually describes how bots, BaaS, and other components in the cloud platform and sensor integration gateways interact. From the implementation perspective, bots and their hosting environments can be implemented using different frameworks, such as mobile agents, OSGi or plain Java objects. For the sake of understanding general governance issues of BoP, we will not examine languages and frameworks for bots as well the context-aware and intelligent capabilities of bots, but focus on governance issues associated with BoP.

3. Characterizing Components and Interactions in Bot Platform-as-a-Service

In order to analyze possible issues in governing bots during their lifecycle – from the development to the execution – it is important to understand requirements and scenarios for which bots are required. In BoP described in Figure 1, the lifecycle of a bot has three main phases (i) *Development* – bots are compiled from source code or bots are composed from existing objects/bots, (ii) *Deployment* – bots are transferred from clouds to hosting environments for execution, and (iii) *Execution* – bots are running in hosting environments. Due to the diversity of devices and equipment, on the one hand, multiple stakeholders will develop different types of bots, specific for particular devices and equipment. On the other hand, multiple stakeholders will have different service contracts for using bots. A single BoP must be able to support such diverse stakeholders, bots, and hosting environments.

Phase	Scenarios	Governance support
Development	Bots are written by third parties for specific types of devices/equipment and are stored either in cloud-based Bot Store or third-party Bot Store	Bot integrity must be ensured. Many bots service contract terms can only be defined for specific cases, e.g., which types of devices and data analytics algorithms supported
	Bots are composed, potentially from other bots, by cloud consumers using BaaS	Governance policies for composite bots must be compatible with bots to be composed.
Deployment	Bots are deployed from Bot Stores to Bot Hosting Environments.	Policies for bots are defined on the fly, e.g. for specific hosting environments, devices, local sensor data access and remote cloud features.
	Bots are deployed to Bot Hosting Environments on-the-fly by BaaS to work with specific types of devices	We need to verify if bots are trusted and if bots are really sent by trusted BaaS.
Execution	Bots can only deal with specific devices and can utilize data and services from the cloud platform.	Access to specific types of data is monitored and controlled.
	Multiple bots are concurrent executed	Bad performance of bots should not bring down the hosting environment. Bot's CPU and memory consumption must be controlled.

Table 1. Scenarios in the development, deployment, execution and dissolution of bots

Table 1 describes some representative scenarios in the development, deployment, and execution of bots. In these scenarios, several governance supports are required. We categorize types of governance issues that should be consider in BoP in the following:

1. *System/network security and access control*: protect systems and networks in order to prevent unauthorized access that can compromise BoP.
2. *Application integrity and service verification*: ensure that the bot content is sent by the trusted party and is unchanged during bot transfer processes, e.g. from Bot Store to Bot Hosting Environment
3. *Service contract management*: due the pay-per-use model of cloud computing, bot capabilities are depending on a service contract. This service contract can cover different terms related to, e.g., application performance, and data acquisition and devices to be controlled.
4. *System and application performance*: ensure that the execution of bots will not prevent the correct operation and the availability of hosting environments.
5. *Data acquisition and control*: Bots will access data from local hosting environments and sensor integration gateways as well as data from the cloud platform. Furthermore, bots will be able to control building MEP (via actuators) and initiate certain features in the cloud platform (e.g., launch new bots or escalate an emergency response processes).

Several governance supports require integrated solutions due to the intersection between cloud computing model, mobile code, and Internet-of-thing. For example, except service contract management, most governance issues look similar to that for mobile code. However, we see that the issues of application performance and cloud access are different, not to mention that the business service contract has a strong influence on all other governance issues.

4. Governance Issues and State-of-the Art

4.1. Current solutions for governance policy enforcement

In order to examine how existing techniques could support governance of bots in BoP, we need to examine identified governance issues in different places in BoP. Table 2 summarizes main techniques that

Governance issues	Places	Existing techniques
Application integrity and service authenticity	BaaS; Bot Store; Bot Hosting Environment	code signing; certification authority
Application composition based on service contract	BaaS	dynamic software product lines [13]
Service contract enforcement	BaaS; Bot Hosting Environment	static analysis; sandboxing; mediation; security by contract [9, 10]; Model-Carrying Code (MCC) [14]
System/network security and access control	BaaS;Bot Store; Bot Hosting Environment	secured connection; message signature; role-based access control
Application performance	Bot Hosting Environment	sandboxing [12]; virtual machine [15]; specific runtime systems [16]; reference monitoring [17]
Data acquisition and control	Bot Hosting Environment	inlined reference monitoring [17]; safe interpreter; Model-Carrying Code (MCC) [14]; security by contract [9, 10]

Table 2. Summary of main existing techniques for governing bots in BoP

could be used for governing bots in BoP. Basically, a bot is built from a number of existing pieces of binary code which might come from untrusted and trusted sources and the function and runtime of bots are dependent on specific context and service contract between the consumer and the cloud provider. The main issue is that the execution of bots must be monitored and controlled at runtime in order to prevent unintended behaviors. Several techniques can be ready to support BoP, such as secured network connection, message signature, and role based access controls. Therefore, we do not discuss them here. Instead, we will focus on main critical points that are specific to BoP.

4.1.1. Governing application integrity, service identity, and application security

There are several techniques that can be applied to bots, however, these techniques are suitable only certain phases of bots.

- *Development phase*: Static analysis can check bots before executing so that only those not violating pre-defined policies are allowed to be executed. However, it cannot check runtime violations.
- *Deployment phase:* Code signing can certify the integrity of the code but cannot prevent bad behavior of the code execution. Using certification authority, signatures of bots and BaaS can be checked again with hosting environment policies, e.g., based on concepts in [18]. However, it can only allow to bots to be installed/deployed/executed or not but no other steps.
- *Execution phase*: Execution monitoring techniques, e.g., based on inlined reference monitoring [17], can enforce bot-specific security policies to prevent bad behavior at runtime. However, they are designed specific for governing security only and are not targeted to our BaaS model. Model-Carrying Code (MCC) [14] can be adaptable to Bot Hosting Environments, however, it requires the base system to be modified. Sandboxing techniques can confine a bot instance within a Bot Hosting Environment that can only access to a limited functionality. The confinement mechanism is based on ”all-or-nothing” approach which is not suitable for fine-grained monitoring. Safe interpreter can also be used to control bots, but it requires bots and Bot Hosting Environments to support interpretation languages.

4.1.2. Governing application performance

Governing application performance occurs mainly in execution phase. However, policies for application performance can be defined in the development or deployment phases. There exist several techniques to measure the performance of hosting environments and their processes. Many techniques, such as based on sandboxing model [12], to control CPU/memory usage by applications require external tools to interact with OS and applications but such tools are heavy. It is also possible to use virtual machine for hosting environments and control the performance of virtual machines [15]. However, we will not be able to prioritize

bots. Specific runtime systems [16] also enable bot-specific performance monitoring but it means that Bot Hosting Environments must be based on these systems.

4.1.3. Governing service contract

Some contractual terms can be used to limit functions when a bot is developed and the other terms are used to check runtime properties, such as number of data points, types of devices and specific facilities. In particular, for on-the-fly composition and deployment, contract terms can strongly influence bot's functions. Potentially, dynamic variability techniques for supporting building limited functions, e.g., based on context [13], can be utilized. However, most dynamic composition and variability models are designed for service-oriented software systems and large-scale software product lines, not lightweight applications. Furthermore, there is a lack of techniques to generate bot software features from specific service contracts in specific situations. At runtime, while existing techniques discussed in other sections can be used to enforce certain contract terms, such as security and performance, we need to apply different techniques for enforcing a service contract for bot instances of a consumer, as the contract covers multiple types of governance policies.

4.1.4. Governing data acquisition and control

Governing data acquisition and control is mainly related to the execution phase, although it might need some support, such as policy generation and code rewriting, from the development phase. The key point is that, dependent on service contract, data acquisition and control can be governed at the system APIs for accessing data and sending control commands or at specific types of data/commands. For example, a bot might be allowed to use `read()` API for reading any sensor data type or to use `read` with only chiller data of chiller manufactured by a specific company (e.g., `read(datatype=chiller, chillerType=company`$_A$`)`). While APIs can be protected and checked by using existing techniques, e.g., static analysis, sandboxing and inlined reference monitoring, they do not support well, e.g., how much data or which control commands a specific bot instance should be allowed. Thus, advanced techniques, e.g., application-level data access monitoring [11], could be used. However, this may require extra components to intercept bot level data acquisition and control messages.

4.2. Current solutions for governance policy definition and management

In order to govern bot executions, the unintended behaviors must be specified in policies for each bot so that its execution can be controlled at runtime. As we have multiple governance issues, we also need to consider multiple types of policies, including data acquisition and control, safety, and service contract policies. These policies are bot-specific and the policies are established in case-to-case basis depending on each particular service contract between the consumer and the cloud platform provider. However, BoP must be able to handle multiple types of policies for multiple bots from different consumers while these bots are possible executed in the same hosting environment for the same facility (e.g., a bot for chiller and a bot for electricity backup system can come from different stakeholders but are executed in the same hosting environment at the same time). Another important point is that policies are used at different places, such as BaaS and Bot Hosting Environments, at different phases, as inputs for different enforcement techniques. Although there have been a number policy languages including research prototypes such as [19, 20, 21], and industry standards such as WS-Policy (`http://www.w3.org/Submission/WS-Policy/`), XACML (`http://labs.oracle.com/projects/xacml/`), none of these considers all of the above multiple types of governance policies. In mobile application development, such as Android (`http://developer.android.com`), applications are associated with use permission which specifies resources and functions can be accessed from the hosting environment. However, such policies are static while BoP need dynamic policies.

4.3. Discussion

From the analysis in the previous section, on the one hand, we observed that while certain techniques can in principle be used, it is not clear if they can be engineered in BoP, in particular, BoP needs to support multiple types of governance and diverse types of hosting environments whose capabilities are limited. For example, model-carrying code techniques require hosting environments to inspect bots; if using safe

interpreters, hosting environments must support interpretation languages; or using encryption to transfer bots would require a heavy cryptosystem in hosting environments; or limiting CPU and memory of bots might not be implemented in all hosting environments. On the other hand, we see that we need multiple types of governance support carried out by cloud services and Bot Hosting Environments at different phases of bot's lifecycle. As a result, BoP needs to have governance policy specifications that allows different types of governance. Furthermore, to enforce such policies, various techniques must be integrated, as typically a type of techniques is suitable only for a specific governance support at a specific phase of bot's lifecycle.

5. Towards Multi-phased Policy Management and Enforcement for Bots

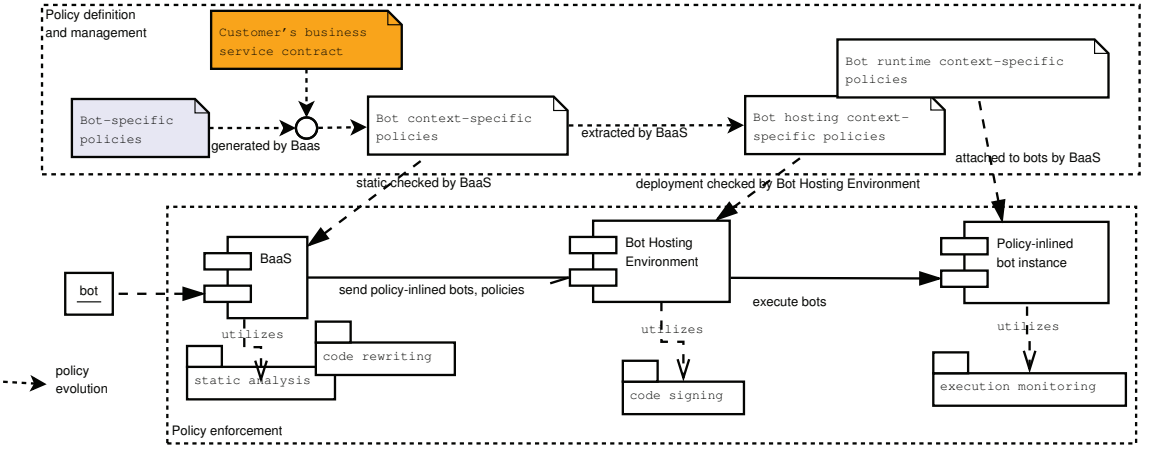

Fig. 2. Conceptual framework for multi-phased policy management and enforcement for bots

As discussed in the previous section, different types of policies need to be enforced for bots. Our approaches are to address (i) policy definition and management and (ii) policy enforcement for multiple types of governance through different phases of bot's lifecycle in an integrated framework. Figure 2 depicts our framework which is divided into two main blocks for *policy definition and management* and for *policy enforcement*. In these two blocks, policy definitions are evolved through different phases and in each phase, at different places, different components will use different techniques to enforce policies.

In summary, the framework works as follows. After a bot is built and stored, we already have static, common *bot-specific policies* – bp_s. Such policies are obtained from the development and defined for a bot. When a bot named b is selected for deployment for a particular consumer under a particular situation, BaaS can take bot-specific policies and combine with consumer's business service contract (e.g., cost and service level agreements) to generate *bot context-specific policies* – bp_{cxt}. In cases, bots are composed and deployed on the fly, then the above-mentioned way is still valid, except that BaaS can just consider bp_s as a part of bp_{cxt}. Many policies $p_i \in bp_{cxt}$ can be checked by BaaS before BaaS starts to deploy b. Before deploying b, BaaS produces two subset of policies, $bp_{cxt}(h)$ and $bp_{cxt}(r)$. The first set of policies $bp_{cxt}(h)$ can be checked by Bot Hosting Environment before running b, while $bp_{cxt}(r)$ can be used to check b at its runtime. In our framework, $bp_{cxt}(r)$ is attached to b and bots will self-regulate their operations based on $bp_{cxt}(r)$. For checking policies, suitable techniques will be employed at different places. Thus, our framework will provide extensible mechanisms to enable plug-ins of different techniques.

With this approach, we will provide templates to define such policies. The policy templates for bot-specific policies and context-specific policies are based on API calls provided by the hosting environment and by cloud services. Templates for bot instance policies need further investigation to combine between event sequences and their parameters, together with business rules. In our framework, an enforcement service is integrated into BaaS to enforce desired policies. A bot is first checked by a static analysis technique to ensure that it does not violate the defined static policies, and then context-specific policies. A bot passing

the check will be rewritten to embed inlined reference monitor into the code so that the execution will be controlled and monitored by the monitor at runtime to ensure that its execution will not violate the runtime policies. When deploying into the hosting environment, the authenticity and integrity of the bot code must be ensured. Our framework will support this feature by employing a code signing technique. Our proposed framework provides an end-to-end enforcement solution for bots construction and execution. While several feasible techniques are chosen, the challenging issue is how to integrate these techniques into the framework so that they can work together to enforce desired policies for the bot architecture.

6. Conclusions and Future Work

The emerging Bot-as-a-Service model for monitoring and managing devices and equipment by utilizing cloud computing offerings calls for a careful investigation on governance issues. In this paper, we analyze possible governance issues and existing techniques that can be reused and should be improved in order to support governance of bots in sustainability platforms. Our future work is to focus on the development our policy definition, management and enforcement framework that support cross governance issues for bots.

References

[1] H.-L. Truong, S. Dustdar, A Survey on Cloud-based Sustainability Governance Systems, International Journal of Web Information Systems, 2012. To appear.
[2] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Computer Networks 54 (15) (2010) 2787 – 2805.
[3] Y. Simmhan, A. G. Kumbhare, B. Cao, V. K. Prasanna, An analysis of security and privacy issues in smart grid software architectures on clouds, in: L. Liu, M. Parashar (Eds.), IEEE CLOUD, IEEE, 2011, pp. 582–589.
[4] D. Song, E. Shi, I. Fischer, U. Shankar, Cloud data protection for the masses, Computer 45 (1) (2012) 39 –45.
[5] M. Christodorescu, R. Sailer, D. L. Schales, D. Sgandurra, D. Zamboni, Cloud security is not (just) virtualization security: a short paper, in: Proceedings of the 2009 ACM workshop on Cloud computing security, CCSW '09, ACM, New York, NY, USA, 2009, pp. 97–102.
[6] D. Booy, K. Liu, B. Qiao, C. Guy, A semiotic multi-agent system for intelligent building control, in: Proceedings of the 1st international conference on Ambient media and systems, Ambi-Sys '08, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2008, pp. 24:1–24:7.
[7] A. Bürkle, A. Hertel, W. Müller, M. Wieser, Evaluating the security of mobile agent platforms, Autonomous Agents and Multi-Agent Systems 18 (2) (2009) 295–311.
[8] K. P. N. Puttaswamy, C. Kruegel, B. Y. Zhao, Silverline: toward data confidentiality in storage-intensive cloud applications, in: Proceedings of the 2nd ACM Symposium on Cloud Computing, SOCC '11, ACM, New York, NY, USA, 2011, pp. 10:1–10:13.
[9] N. Dragoni, F. Martinelli, F. Massacci, P. Mori, C. Schaefer, T. Walter, E. Vetillard, Security-by-Contract (SxC) for software and services of mobile systems, At your service - Service-Oriented Computing from an EU Perspective. MIT Press, 2008.
[10] L. Desmet, W. Joosen, F. Massacci, F. Piessens, I. Siahaan, D. Vanoverberghe, Security by contract on the.net platform, in: Information Security Technical Report, Elsevier, 2008.
[11] D. Y. Zhu, J. Jung, D. Song, T. Kohno, D. Wetherall, Tainteraser: protecting sensitive data leaks using application-level taint tracking, SIGOPS Oper. Syst. Rev. 45 (2011) 142–154.
[12] F. Chang, A. Itzkovitz, V. Karamcheti, User-level resource-constrained sandboxing, in: Proceedings of the 4th conference on USENIX Windows Systems Symposium - Volume 4, USENIX Association, Berkeley, CA, USA, 2000, pp. 3–3.
[13] C. Parra, X. Blanc, L. Duchien, Context awareness for dynamic service-oriented product lines, in: Proceedings of the 13th International Software Product Line Conference, SPLC '09, Carnegie Mellon University, 2009, pp. 131–140.
[14] R. Sekar, V. Venkatakrishnan, S. Basu, S. Bhatkar, D. C. DuVarney, Model-carrying code: a practical approach for safe execution of untrusted applications, in: Proceedings of the nineteenth ACM symposium on Operating systems principles, SOSP '03, ACM, New York, NY, USA, 2003, pp. 15–28. doi:http://doi.acm.org/10.1145/945445.945448.
[15] K. Onoue, Y. Oyama, A. Yonezawa, Control of system calls from outside of virtual machines, in: Proceedings of the 2008 ACM symposium on Applied computing, SAC '08, ACM, New York, NY, USA, 2008, pp. 2116–1221.
[16] G. Back, W. C. Hsieh, The kaffeos java runtime system, ACM Trans. Program. Lang. Syst. 27 (2005) 583–630.
[17] Úlfar Erlingsson, The Inlined Reference Monitors Approach to Security Policy Enforcement, Ph.D. thesis, Cornell University, Ithaca, New York (2004).
[18] W. Enck, M. Ongtang, P. McDaniel, On lightweight mobile phone application certification, in: Proceedings of the 16th ACM conference on Computer and communications security, CCS '09, ACM, New York, NY, USA, 2009, pp. 235–245.
[19] L. Bauer, J. Ligatti, D. Walker, Composing security policies with Polymer, in: PLDI '05: Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation, ACM, New York, NY, USA, 2005, pp. 305–314.
[20] I. Aktug, K. Naliuka, Conspec – a formal language for policy specification, Electron. Notes Theor. Comput. Sci. 197 (1) (2008) 45–58. doi:http://dx.doi.org/10.1016/j.entcs.2007.10.013.
[21] K. W. Hamlen, M. Jones, Aspect-oriented in-lined reference monitors, in: Proceedings of the third ACM SIGPLAN workshop on Programming languages and analysis for security, PLAS '08, ACM, New York, NY, USA, 2008, pp. 11–20.