Policy Based Management for Security in Cloud Computing

Adrian Waller[1], Ian Sandy[1], Eamonn Power[2], Efthimia Aivaloglou[3], Charalampos Skianis[3], Antonio Muñoz[4], and Antonio Maña[4]

[1] Thales UK, Research and Technology, Reading, UK
[2] TSSG, Waterford Institute of Technology, Waterford, Ireland
[3] Department of Information and Communication Systems Engineering, University of the Aegean, Samos, Greece
[4] University of Málaga
{adrian.waller,ian.sandy}@thalesgroup.com, epower@tssg.org, {eaiv,cskianis}@aegean.gr, {amunoz,amg}@lcc.uma.es

Abstract. Cloud computing is one of the biggest trends in information technology, with individuals, companies and even governments moving towards their use to save costs and increase flexibility. Cloud infrastructures are typically based on virtualised environments, to allow physical infrastructure to be shared by multiple end users. These infrastructures can be very large and complex, with many end users, making their configuration difficult, error-prone and time-consuming. At the same time, the fact that diverse end users share the same physical infrastructure raises security concerns, and can lead to a significant impact from misconfiguration or being slow to react to attacks. In this paper, we focus on the use of Policy Based Management techniques to manage cloud infrastructure, identifying the requirements, surveying the state-of-the-art, identifying the challenges and proposing potential solutions.

Keywords: Policy Based Management; Virtualisation; Cloud Computing.

1 Introduction

Cloud computing is one of the biggest trends in Information Technology (IT) today. By enabling data and services to reside on outsourced and shared computing platforms, significant cost savings and more flexibility can be achieved compared to deploying and maintaining one's own infrastructure. For this reason, companies and even governments are moving towards their use, but the potential sensitivity of their data means that cloud providers must manage their large and complex infrastructures in a robust way. Current trends in IT suggest that software systems will become very different from their counterparts today, due to a greater adoption of Service-Oriented Architectures (SOAs), the wider deployment of Software as a Service (SaaS), and the increased use of wireless and mobile technologies [1][2]. In line with these trends, cloud computing platforms are built on top of large-scale, heterogeneous infrastructures that are made available to a large number of end users with very disparate needs. In this setting, the management of non-functional properties such as security and

privacy will be of an increased and critical importance. In this paper we look at the use of Policy Based Management (PBM) techniques to securely manage cloud infrastructure. In section 2, we describe the background to cloud management, the use of PBM in this context, and the requirements for a solution based on PBM. In section 3, we survey the state-of-the-art and identify the key challenges for such a solution. Finally, in section 4 we outline some potential solution approaches and future work that we are pursuing in the PASSIVE project [3].

2 Background and Requirements

The NIST definition of cloud computing [4] refers to a model of resource management that enables convenient access to a shared pool of configurable computing resources that can be easily provisioned and released with minimal effort from the service provider. It goes on to categorise the service models as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). IaaS allows the provisioning of servers (using virtual machines (VM)), storage and network resources rapidly using either a console interface or an API. The goal of this paper is to outline a component that resides beneath the console/API and spans the underlying resources to enable fine-grained resource control and provide assurance regarding the integrity of the resources being managed. We propose an approach using PBM of the virtualisation resources for cloud providers. In essence, PBM is a technique for specifying the behaviour of a system under different circumstances. The use of policies allows the response of the system to a given situation to be changed quite simply, by changing the policy, without the need to modify the underlying software. In a dynamic system such as presented by cloud computing, the system must handle changing policies as the system runs, which gives rise to a number of issues that have to be solved in order to create an effective system:

- The PBM system has to take in policies covering a variety of topics in addition to security (e.g. resource allocation), and from a variety of sources. These policies may be expressed in multiple languages at different levels of abstraction, and must be translated into a common language for use at the point decisions are made.
- The decision making process using the defined policies must be correct, and the implementation of the policy actually has to happen (i.e. be enforced, and be consistent throughout the cloud). This implies the need for assurance in both the PBM decision making, as well as the selection, reconfiguration and composition of the components that are used to implement the decision.
- Having multiple policies from multiple sources will almost certainly result in a conflict at some stage, which will need to be resolved.
- Last but not least, a PBM system's activities will, of course, need to be performed in such a way so as not to impact on the performance and cost of the cloud.

In the following section, we consider the relevant state-of-the-art and major challenges in developing a PBM solution to meet these requirements.

3 State-of-the-Art and Challenges

3.1 Policy Based Management

A common theme in the state-of-the-art is the use of formal or logic-based methods. Systems with a rigorous formal foundation both for the specifications and for the semantics of authorisation allow rigorous guarantees of the security policies [5]. A problem is that the policy to be applied may actually be a composite of different requirements from different sources. One proposed solution in access control is an algebra with formal semantics which allows a number of simple policies to be combined into the required complex policy [6]. A significant challenge remains to develop a formal policy language which is suitable for expressing policies for a range of areas such as security, access control, monitoring and resource management. Part of the challenge would be to make the language as easy to use as possible without compromising its formal properties, which may require the development of a natural-language front-end with associated translation, such as proposed for the PERMIS editor [7]. Another potential benefit of a formal language would be in making the detection and resolution of conflicts between policies easier, which is in itself a challenge that needs addressing [8]. A recent survey of conflict resolution techniques found them to be mostly unsuitable for live management systems [9]. Algorithms and techniques for conflict detection and resolution are needed both when the policies are being created or edited and when they are being evaluated. The best approach to conflicts may be to avoid them altogether by paying close attention to writing policies to ensure that they cannot conflict (e.g. [9]) but this is unlikely to be a successful strategy in an environment as complex and dynamic as a cloud. Within a cloud, Virtualised Environments (VEs) are typically used, and PBM has been proposed for managing them. Performance for such an approach is a key challenge within VEs, and an example of work in this area is to transfer the security enforcement and program analysis roles to a policy-directed FPGA [10].

3.2 Cloud PBM Architectures

Policies can be enforced at various layers of the systems architecture of cloud computing environments. Policies controlling resource access or inter-VM communication can be enforced at the hypervisor layer, while more fine-grained policies can be enforced at the VM operating system layer. Policies controlling the formation of coalitions of VMs or setting restrictions on their collocation may be defined on a central management VM instead of on each host of the infrastructure. The sHype security architecture [11] enables the enforcement of policy based access control for the shared virtual resources and the information flows between operating systems hosted on common hardware platforms. Following the FLASK access control architecture [12], sHype keeps the access control policy separate from access control enforcement. The policy management function offers the means to create and maintain policy instantiations that are efficient to use at the hypervisor level. The OpenTC architecture [13] enables the definition and enforcement of a wide range of security policies. It includes a trusted virtualisation layer, a Trusted Platform Module (TPM) with strong isolation properties between virtual machines, and a security services layer. Similar to the

sHype architecture, the definition and management of the security policies is performed at the application layer, in a dedicated management virtual machine. A layered architecture for access control in virtualised systems running sHype for mandatory access control (MAC) was proposed in [14]. The operating system kernel (SELinux) layer implements MAC to confine data received from the other VMs. The Shamon shared reference monitor [15] that has been proposed for enforcing MAC policies across a distributed set of VMs also implements a layered approach. It enforces MAC reference monitoring from the hypervisor (Xen) and the operating system (SELinux) and IPsec network controls. Shamon offers support for coalitions of VMs on multiple physical hypervisors. In more recent proposed solutions enabling trusted multi-tenant virtual datacentres [16], the notion of coalitions of VMs has evolved to the concept of Trusted Virtual Domains (TVDs) [17] that allow the grouping of VMs that collaborate. The Trusted Virtual Datacentre (TVDc) security solution [16] groups VMs into TVDs and relies on the enforcement of MAC policies by sHype for isolating them. While the architectures that enable the formation of coalitions [15] or TVDs [16] allow the enforcement of fine-grained policies for controlling cooperation among the coalitions, one challenge that remains is the flexible organisation and management of the coalition members which could be useful for scenarios with frequent VM membership changes, such as for cloud infrastructures hosting virtual desktops. An additional challenge for controlling VM placement and collocation is to enable the definition of placement rules based on both static and dynamic attributes of the hosts and the VMs, and the security characteristics supported by the platform.

3.3 Assurance in Decision Making

To achieve high assurance, policies need to be precisely and unambiguously specified, and accurately implemented. Policies may also conflict, and therefore these conflicts need to be detected and resolved if correct behaviour is to be observed. The ideal approach to achieve this would be the use of a logic-based formal language, allowing the correctness of the policies to be mathematically proven. Unfortunately, there appear to be no readily available formal policy languages suitable for an environment such as cloud computing, where policies cover a range of activities from access control to resource management. DHARMA [18] is a formal language, but since it is principally a reference monitor it is not really suitable or easily adaptable to meet these needs. More general purpose policy languages such as APPEL [19] and PONDER [20] do exist, as do more specialised ones such as XACML [21] for access control or UCON [22] for usage control. However, none of these are formal. Another difficulty with formal languages, or indeed any language that can be implemented in an automated policy system, is that they require a lot of skill to be used effectively, which is unlikely to be found in a user who is not a programming or technical expert (or, indeed, a formal methods expert). A potential solution would be a natural language front-end as the interface to the user. This would, by necessity, have a restricted vocabulary and grammar and would need to be translated into the underlying machine-readable language. Ideally, there would only be one translation step, built on formal methods, that would generate machine instructions from user input and be provably correct. This is unlikely to be realised in the short-term and intermediate stages will be needed, with a consequent greater difficulty in showing that the policy

has been correctly interpreted and enforced. Verification and validation of the low-level policies is also needed, and should include detecting and resolving conflicts between policies. In the absence of a formal language with its inherent property of proof of correctness, testing will have to be more rigorous and more extensive to provide this. Even so, it is not possible to provide the same level of confidence with any realistic testing regime, although this approach can be less expensive as it does not require specialist staff to be available.

3.4 Software Security Certification

In addition to assurance in policy decision making, assurance in the security and privacy properties of the modified system resulting from a policy decision is also needed. In principle, certification appears a plausible, practical and well-established solution for increasing users' trust and confidence, where a certificate attests security properties of entities (software and hardware products, systems and services). However, looking more closely at the specific characteristics of cloud computing scenarios, we see that current software system certification schemes are not appropriate. Software certification is currently based on evaluation processes carried out by experts following pre–defined and publicly accepted criteria that analyse the software using different techniques, ranging from testing to formal modelling. These processes are mostly manual and require considerable amounts of effort, and thus time and investment. The relying party of a certificate needs not only to trust the authenticity of the certificate, but also the experts, and the certification scheme. This trust is established by the scheme being run by accredited authorities, the accreditation of the experts themselves, and the certificate being officially approved. In current schemes certificates are awarded to traditional, monolithic software systems and become invalid when a system performs run time selection and composition of components [23]. However, in a cloud computing scenario, several independently produced applications may coexist on a virtualised environment, which in turn is supported by a distributed computing architecture. Clearly, this approach of providing certificate–based assurance of security does not scale well to scenarios that are characterised by dynamism, high degrees of distribution, and ever–changing environments. The main reasons for this are that existing schemes produce certificates and explanations intended for human users and aim to help them decide whether or not to use/buy the system. Also, certificates refer to a particular version of the product or system. In general, changes in the system structure require a process of re–certification. Certification schemes like the Common Criteria (CC)[24] contain an assurance class on flaw remediation, but it is rarely used and does not provide methodological support for analysing the security impact of system changes. An additional challenge is the need to cover both individual software services and the environment in which they operate at execution time. Some support exists in CC to deal with composite systems (i.e. derive a system certification from certificates of its components), but a perfect match between assumptions and component guarantees is required, which is still too restrictive to be practical in our scenarios. An important aspect of cloud scenarios is dynamism. Unfortunately, current software certification schemes do not support dynamic replacement of components or runtime binding of systems. Even in CC v3.1 [24], changing components requires new evaluator/expert interaction and repetition of (or parts of) the evaluation

and certification. Moreover, current certificates lack a machine-readable, semantics-aware format for expressing security properties. Thus, they cannot be used to support and automate run time security assessment, although this issue of providing machine-readable versions of security certifications is being addressed in the ASSERT4SOA project [25]. As a result, today's certification schemes simply do not provide, from an end user perspective, a reliable way to assess the trustworthiness of a composite application in the context where (and at the moment when) it will be actually executed.

4 Future Work and Acknowledgements

The work in this paper arises from the PASSIVE project [3]. PASSIVE is developing a policy-based security architecture for cloud computing which will address many of the challenges raised in this paper. PASSIVE is a Specific Targeted Research Project (STREP) supported by the European 7th Framework Programme, Contract number ICT-2.1.4-257644, Project starting date 1st June 2010 (duration 24 months).

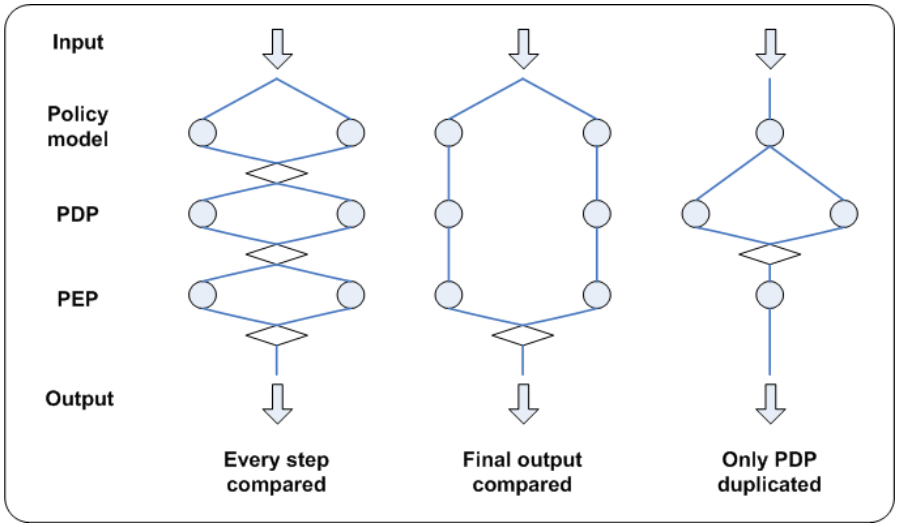

Fig. 1. Options for multiple redundant implementations in PBM

One approach we are pursing to providing high-assurance is the provision of multiple independent implementations of important components, whose outputs are compared and must agree. Unanimity will ensure that only the correct output is obtained or an error condition will be raised. Majority voting can allow continued operation, albeit with a perhaps less than ideal output with the discrepancy flagged for urgent investigation. The implementations need to be as independent as possible (e.g. carried out by different teams possibly using different programming languages), giving much greater confidence that the outcome is correct. Potentially, the whole section between the user natural language-based interface and the resulting machine instructions (i.e. the whole policy system) could be done this way. This would suggest a need for multiple policy languages in addition to the code that makes decisions based on the policies, and that which enforces those decisions. There are different ways of exploiting the duplication, the two extremes being that the different implementations run sepa-

rately and only the final outcomes are compared or that the outcomes of each step are compared as the process runs. The diagram illustrates this as well as the situation with only one part duplicated (the Policy Decision Point, (PDP)).

Another approach we will take is the so-called 'policy continuum' [26]. This approach provides a means to represent the various constituency languages needed to support security policy definition at various levels. It also supports the mapping of high-level goals to low-level tasks and actions. This mapping is supported by the use of a common information model, which seeks to represent, in an abstract way, the behaviour and characteristics of a system without regard to details such as platform, language etc. Such information models have been used and demonstrated in projects such as AutoI [27] where virtual infrastructure and associated management policies were modelled and used to manage, monitor and orchestrate Internet services. The information model allows data to be harmonised between constituencies. This permits access to information gathered from outside of the constituency to be associated with current constituency entities. This, in turn, allows more useful information to be inferred and used. An example here would be the use of intrusion detection system data on a given node to decide how resources are allocated in surrounding nodes by provisioning systems. This ability for common information sharing between diverse components such as those described above in the duplicated decision point approach would support such a solution. Both components could have their output represented in common terms and thus compared or prioritised. PASSIVE is currently designing a solution and a demonstrator will be available at the end of the project (Summer 2012).

References

1. Software as a Service Market Will Expand Rather than Contract Despite the Economic Crisis, IDC Finds (January 2009), http://www.idc.com/getdoc.jsp?containerId=prUS21641409 (accessed March 2010)
2. Robinson, J.J.: Demand for software-as-a-service still growing (May 2009), http://www.information-age.com/channels/commsand-networking/perspectives-and-trends/1046687/demand-forsoftwareasaservice-still-growing.thtml (accessed March 2010)
3. PASSIVE project, http://ict-passive.eu/
4. http://csrc.nist.gov/groups/SNS/cloud-computing/ (July 10, 2009)
5. Chapin, P.C., Shalka, C., Wang, X.S.: Authorization in Trust Management: Features and Foundations. ACM Comput. Surv. 40(3), Article 9 (August 2008)
6. Bonatti, P., De Capitani Di Vimercati, S., Samarati, P.: An Algebra for Composing Access Control Policies. ACM Trans. Inf. Syst. Secur. 5(1), 1–35 (2002)
7. Inglesant, P., Sasse, M.A., Chadwick, D., Shi, L.L.: Expressions of Expertness: The Virtuous Circle of Natural Language for Access Control Policy Specification. In: Symposium On Usable Privacy and Security (SOUPS) 2008, Pittsburgh, PA, USA, July 23-25 (2008)
8. Dunlop, N., Indulska, J., Raymond, K.: Methods for Conflict Resolution in Policy-Based Management Systems. In: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing (EDOC 2003), pp. 1–12 (2003)

9. Chadha, R.: A Cautionary Note about Policy Conflict Resolution. In: Proc. IEEE Military Comms Conference 2006, MILCOM 2006, Washington, DC, October 23-25 (2006)
10. Bratus, S., Locasto, M.E., Ramaswamy, A., Smith, S.W.: Traps, Events, Emulation, and Enforcement: Managing the Yin and Yang of Virtualization-based Security. In: VMSEC 2008, Fairfax, Virginia, USA, pp. 49–58 (October 31, 2008)
11. Sailer, R., Valdez, E., Jaeger, T., Perez, R., van Doorn, L., Griffin, J.L., Berger, S.: sHype: Secure Hypervisor Approach to Trusted Virtualized Systems. IBM Research Report RC23511, 2005 (2005)
12. Spencer, R., Smalley, S., Loscocco, P., Hibler, M., Andersen, D., Lepreau, J.: The flask security architecture: system support for diverse security policies. In: Proceedings of the 8th Conference on USENIX Security Symposium, vol. 8 (1999)
13. Kuhlmann, D., Landfermann, R., Ramasamy, H. V., Schunter, M., Ramunno, G., Vernizzi, D.: An Open Trusted Computing Architecture - Secure Virtual Machines Enabling User-Defined Policy Enforcement. OpenTC report, 2006 (2006)
14. Payne, A.D., Sailer, R., Cáceres, R., Perez, R., Lee, W.: A layered approach to simplified access control in virtualized systems. ACM SIGOPS Operating Systems Review 41(7), 12–19 (2007)
15. McCune, J.M., Jaeger, T., Berger, S., Caceres, R., Sailer, R.: Shamon: A System for Distributed Mandatory Access Control. In: Computer Security Applications Conference, pp. 23–32 (2006)
16. Berger, S., Cáceres, R., Pendarakis, D., Sailer, R., Valdez, E., Perez, R., Schildhauer, W., Srinivasan, D.: TVDc: Managing Security in the Trusted Virtual Datacenter. ACM SIGOPS Operating Systems Review 42(1), 40–47 (2008)
17. Bussani, A., Griffin, J.L., Jansen, B., Julisch, K., Karjoth, G., Maruyama, H., Nakamura, M., Perez, R., Schunter, M., Tanner, A., van Doorn, L., Herreweghen, E.V., Waidner, M., Yoshihama, S.: Trusted Virtual Domains: Secure foundation for business and IT services, Research Report RC 23792, IBM Research (November 2005)
18. Chander, A., Dean, D., Mitchell, J.C.: A distributed high assurance reference monitor. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 231–244. Springer, Heidelberg (2004)
19. Montangero, C., Reiff-Marganiec, S., Semini, L.: Logic–Based Detection of Conflicts in APPEL Policies. In: Arbab, F., Sirjani, M. (eds.) FSEN 2007. LNCS, vol. 4767, pp. 257–271. Springer, Heidelberg (2007)
20. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: Ponder: A Language for Specifying Security and Management Policies for Distributed Systems. The Language Specification Version 2.3. Imperial College Research Report DoC 2000/1 (October 20, 2000)
21. OASIS website (February 2011), `http://www.oasis-open.org`
22. Zhang, X., Parisi-Presicce, F., Sandhu, R., Park, J.: Formal Model and Policy Specification of Usage Control. ACM Trans. Inf. Syst. Secur. 8(4), 351–387 (2005)
23. Alvaro, A., de Almeida, E.S., de Lemos Meira, S.R.: Software component certification: A survey. In: Proc. of 31st EUROMICRO Conference on Software Engineering and Advanced Applications, Porto, Portugal (August–September 2005)
24. Common Criteria for Information Technology Security Evaluation, ISO/IEC Standard 15408, version 3.1(2008)
25. ASSERT4SOA Project (March 2011), `http://www.assert4soa.eu/`
26. Davy, S., Jennings, B., Strassner, J.: The Policy Continuum - A Formal Model. In: Proc. Modelling Autonomic Communications Environments, Multlicon, Multicon, Berlin. Lecture Notes, vol. 6, pp. 65–78 (2007)
27. AUTOI ICT-216404, Deliverable D4.1 - Initial Management Plane (December 2008)