Federated authorization for SaaS applications

Maarten Decat, Bert Lagaisse, Wouter Joosen

IBBT-DistriNet, KU Leuven, 3001 Leuven, Belgium

Abstract. With Software-as-a-Service (SaaS), a centrally hosted web-based application is offered to a large number of customer organizations called tenants, each using multiple applications. The tenant and provider each work in their own authoritative and administrative domain, leading to a federated architecture and raising the bar for security and access control. Access control with SaaS applications is about protecting the tenant's data at the provider's side using the tenant's policies and user information. In current practice however, all access control policies are evaluated at the provider's side, distributing and fragmenting the tenant's policies over the multiple applications it uses. Moreover, all necessary user information now has to be shared with the provider, resulting in the disclosure of confidential tenant data. Therefore, we propose the concept of *federated authorization*, a combination of externalized authorization and federated access control techniques whereby the tenant's access control policies are evaluated at the tenant's side using local data.

1 Introduction

Software-as-a-Service (SaaS) applications are a part of cloud computing in which centrally hosted web-based applications are offered to a large number of tenants (customer organizations each representing multiple end-users), each using multiple applications. From the point of view of the tenant, cloud computing and SaaS are a form of outsourcing using a pay-per-use billing scheme. From the point of view of the provider, SaaS is the next step in Application Service Provider (ASP) evolution, trying to cut operational costs by sharing resources of the application and offering it on a larger scale.
Originally, SaaS applications where mainly used by small and medium enterprises for non-core business applications. Typical examples are Google Apps (an office suite, e-mail, calendar etc.) and Salesforce (CRM). Recently however, large enterprises are increasingly starting to use SaaS for core-business applications. An example of this is a home patient monitoring system offered to health institutions like hospitals (the tenants of the application). With regards to the typical examples, these applications pose new challenges, such as the increased importance of security. While cloud computing and SaaS can provide large economical advantages, security is the major hurdle for cloud adoption [8,6,5].
SaaS requires application-level security, of which access control is an important part. From a high-level point of view, access control limits the actions a subject (e.g., a user) can take on an object in the system (e.g., a file) applying rules

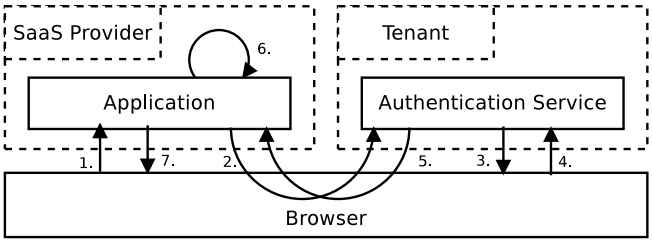

Fig. 1. Federated authentication, SAML flow [1]: the user requests a certain resource using his browser (step 1), the SaaS application redirects the user to its home domain (step 2), the user authenticates himself locally (step 3 and 4), the user is redirected back to the application together with its user information (step 5), the application uses this information for access control (step 6) and returns the resource if allowed (step 7).

defined in access control policies. The process of access control is generally divided into authentication, authorization and audit. Authentication confirms the stated identity of a subject, authorization confirms the subject is allowed to do the desired action on the object and audit offers information about the actions done in the system.
This paper focuses on authentication and authorization. Section 2 defines the problems we observe with current state of the art for authorization in SaaS applications. Section 3 sketches our proposed solution and specific objectives. Section 4 lists the challenges we predict in this research. Section 5 describes our research methodology and section 6 shows our main contributions.

2 Problem statement

SaaS applications are a form of outsourcing: the application remotely hosts and processes data actually belonging to the tenant. Therefore, access control in SaaS applications is mainly about protecting the tenant's data located at the provider's side. The tenant controls the administration and information about the users of the application[1] (e.g., employees of the tenant) and the access control rules and policies.

A first problem with access control for remote applications is *administrative scalability*. From the tenant's point of view, every application has to be provided with the necessary user information. Therefore, *federated authentication* techniques such as WS-Federation [3], OpenId [2] and SAML [1] have been developed. These techniques allow users to be authenticated in their home domain, after which the needed user information is securely exchanged with the application (see fig. 1). This centralizes user management with the tenant and offers control over the shared information and the authentication means used.

[1] There are more complex scenario's in which former unknown users access the application (e.g., ad-hoc federations). Here we purposely limit our vision and exclude this for now.

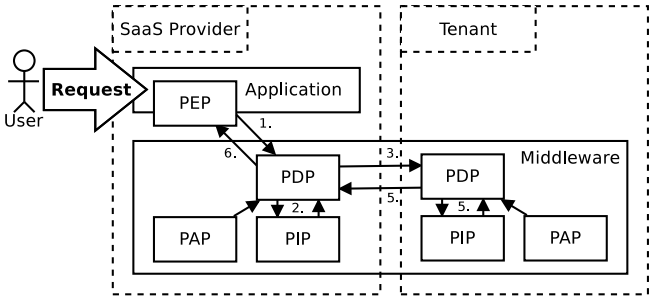

Fig. 2. Externalized authorization

With the evolution towards core-business SaaS applications, more control over the application and the data it uses is required, raising the need for authorization. While federated authentication techniques allow tenants to centralize user management, similar techniques for authorization currently do not exist and all access control policies are evaluated at the provider's side. Because of this, access control policies are distributed and fragmented over the multiple SaaS applications the tenant uses. This increases the administrative load, eventually leading to inconsistencies and security holes.
A second problem with this approach, is the necessary *disclosure of confidential tenant information.* Because the access control policy is evaluated at the provider's side, all necessary information has to be shared with the provider. While the tenant trusts the provider with the information in the application itself, it is likely that more information is needed for authorization. For example, in the patient monitoring system, the list of all patients the physician is currently treating or the fact that the patient is currently being treated by an oncologist is confidential information. Moreover, it can be the case the tenant does not want to share the policy itself, for example how the hospital handles its competitors when requesting access to its data.

3 Proposed solution

In this section we present an initial solution we envision to solve the two problems described above: their limited administrative scalability and the necessary disclosure of confidential information. The solution we envision consists of externalizing policy evaluation from the SaaS application to the tenant. With each request, the SaaS application would ask the tenant for an access control decision, after which the tenant himself evaluates the necessary policies and returns his decision. This way, tenant policies remain centralized and confidential information does not have to be shared with the application provider. Since this achieves for authorization what federated authentication did for authentication, we call this *federated authorization.*

The objective of this research is to incept and evaluate support for federated authorization in security middleware. A high-level technical overview of how we

currently envision this middleware is given in fig. 2 using XACML [11] terminology for the components. When a user sends a request to the SaaS application to perform an action, the Policy Enforcement Point (PEP) is called from the application in order to determine whether the user is allowed to do so. It therefore calls the local Policy Decision Point (PDP), which will evaluate tenant policies loaded from the Policy Administration Point (PAP) using information stored in the Policy Information Point (PIP). The provider's PDP also forwards the request to the tenant PDP, which in turn evaluates the tenant's policies using local data and returns its decision. The requested action is only allowed in case both parties allow it.
In order to achieve this objective, both the policy language and the execution environment will have to be extended. The policy has to support the necessary constructs in order to declare the needed properties of the federated authorization. Future research will have to show which constructs are minimally needed. For the execution environment, the communication between the provider and the tenant will most likely be the crucial part, since the information about the request, the subject, the object and the system are now distributed over the two parties.

4 Challenges

Federated authorization as described above builds upon and extends externalized authorization, a concept that has been researched in the past (e.g., [9]). Moreover, the XACML access control standard already defines a profile for integration with SAML for transport of both user information and authorization decisions [10]. However, no research about the practical feasibility of combining externalized authorization with federation into federated authorization is present to the best of our knowledge. An important aspect of this are the security tradeoffs implicitly made in federated authorization. For example, externalizing access control to the tenant increases administrative scalability, but also makes the system more susceptible to denial of service attacks and traffic analysis.
One major challenge we predict is the performance and performance scalability of the security middleware. This challenge is increased by taking into account concurrency and distributed data updates. Related research has led to techniques for improving access control performance (e.g., Wei [14] proposes decision recycling and inference, Brucker and Petritsch [4] focus on the retrieval of the necessary user information), some of which can also be applied for scalability. However, SaaS applications frequently use aggressive scalability techniques [12] and their applicability on access control still has to be investigated.

5 Research methodology

In this research, a case-study driven methodology is used. Two major case studies are used: (i) a home patient monitoring system and (ii) an electronic document processing system. The security analyses of both case studies offer require-

ments and policies to be applied in our prototypes. The former offers insights in the stringent legal requirements concerning medical data and the complex, fine-grained access control policies in e-health, the latter offers a tenant hierarchy, a concept which should be supported by federated authorization.
Currently, this research is in its early phase. Involvement in the case studies stated above has shown the need for solving this problem, after which a literature study of existing authorization techniques was made. As the next step in this research, our concept of federated authorization will be refined into a reference architecture. In order to build upon existing work, we will base this on widely-used standards such as XACML [11] as much as possible. This reference architecture will then be instantiated into a proof of concept using the case studies in order to evaluate the feasibility of federated authorization. The next step will be to apply and empirically evaluate performance and scalability techniques on this using metrics such as the impact of the complexity of policies (e.g., the number of subject and object attributes needed) on the response time of the whole.

6 Main contributions

Next to SaaS applications, federated access control has been researched in other domains as well, such as web applications (e.g., OpenId [2]), grid computing (e.g., PERMIS [7]) and web services (e.g., WS-Federation [3]). These domains offer relevant techniques, but their purpose differs from SaaS. For example, access control in grid computing is mainly about access to on-premise resources from other domains. Moreover, the described techniques still limit themselves to local policy enforcement based on federated authentication.
Some literature has been published about access control for multi-party systems. Zhang et al. [15] describe a framework for usage control in collaborative systems, taking into account access control data updates. Stihler et al. [13] also describe an architecture which enables the involved parties to declare access control policies on the application. However, both still employ provider-side policy enforcement, resulting into the problems described in section 2.
Existing technologies for access control also do not support federated authorization. Products such as IBM Tivoli Access Manager allow to centrally enforce access control policies on multiple applications. Moreover, by acting as an *XML gateway* these policies can be applied on external applications. This set-up is however limited to the level of messages, thereby limiting the complexity of supported policies. Moreover, this limits the mobility of the users of the external application, making it inapplicable for SaaS applications.

7 Conclusions

This paper presented a new approach to access control for SaaS applications called *federated authorization*. In this approach, policy evaluation is externalized from the SaaS application to the tenant, keeping tenant policies centralized

and its data confidential. A major challenge we predict is the performance and performance scalability of the security middleware. Using a case-study driven methodology, we will evaluate the feasibility of federated authorization for SaaS applications.

Acknowledgements This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, by the Belgian Science Policy, by the Research Fund KU Leuven, by the EU FP7 project NESSoS and by the Agency for Innovation by Science and Technology in Flanders (IWT).

References

1. Security Assertion Markup Language (SAML) v2.0 (March 2005), http://www.oasis-open.org/standards#samlv2.0
2. OpenID Authentication 2.0 - Final (December 2007), http://openid.net/specs/openid-authentication-2_0.html
3. Bajaj, S., Della-Libera, G., Dixon, B., Dusche, M., Hondo, M., Hur, M., Kaler, C., Lockhart, H., Maruyama, H., Nadalin, A., et al.: Web Services Federation Language (WS-Federation). http://specs.xmlsoap.org/ws/2006/12/federation (December 2006)
4. Brucker, A., Petritsch, H.: Idea: Efficient Evaluation of Access Control Constraints. Engineering Secure Software and Systems pp. 157–165 (2010)
5. Brunette, G., Mogull, R.: Security guidance for critical areas of focus in cloud computing v2.1. Tech. rep., Cloud Security Alliance (december 2009)
6. Catteddu, D., Hogben, G.: Cloud Computing: benefits, risks and recommendations for information security. Tech. rep., ENISA (november 2009)
7. Chadwick, D., Otenko, A.: The PERMIS X. 509 role based privilege management infrastructure. Future Generation Computer Systems 19(2), 277–289 (2003)
8. Dean, D., Saleh, T.: Capturing the Value of Cloud Computing. Tech. rep., Boston Consultancy Group (november 2009)
9. Karjoth, G.: Access control with ibm tivoli access manager. ACM Transactions on Information and System Security (TISSEC) 6(2), 232–257 (2003)
10. Lockhart, H., Parducci, B., Rissanen, E., Lockhart, H.: SAML 2.0 Profile of XACML, Version 2.0
11. Parducci, Bill and Lockhart, Hal and Rissanen, Erik: eXtensible Access Control Markup Language (XACML). http://www.oasis-open.org/committees/xacml (August 2010)
12. Shoup, R.: Scalability Best Practices: Lessons from eBay (May 2008), http://www.infoq.com/articles/ebay-scalability-best-practices
13. Stihler, M., Santin, A., Calsavara, A., Marcon, A.: Distributed usage control architecture for business coalitions. In: Communications, 2009. ICC'09. IEEE International Conference on. pp. 1–6. IEEE (2009)
14. Wei, Q.: Towards improving the availability and performance of enterprise authorization systems. Ph.D. thesis, University of British Columbia (2009)
15. Zhang, X., Nakae, M., Covington, M., Sandhu, R.: Toward a usage-based security framework for collaborative computing systems. ACM Transactions on Information and System Security (TISSEC) 11(1), 1–36 (2008)