Enabling the Autonomic Management of Federated Identity Providers

Christopher Bailey, David W. Chadwick, Rogério De Lemos, Kristy S. Siu

Enabling the Autonomic Management of Federated Identity Providers

Christopher Bailey, David W. Chadwick, Rogério de Lemos, Kristy W. S. Siu

School of Computing, University of Kent, UK
{c.bailey, d.w.chadwick, r.delemos, k.w.s.siu}@kent.ac.uk

Abstract. The autonomic management of federated authorization infrastructures (federations) is seen as a means for improving the monitoring and use of a service provider's resources. However, federations are comprised of independent management domains with varying scopes of control and data ownership. The focus of this paper is on the autonomic management of federated identity providers by service providers located in other domains, when the identity providers have been diagnosed as the source of abuse. In particular, we describe how an autonomic controller, external to the domain of the identity provider, exercises control over the issuing of privilege attributes. The paper presents a conceptual design and implementation of an effector for an identity provider that is capable of enabling cross-domain autonomic management. The implementation of an effector for a SimpleSAMLphp identity provider is evaluated by demonstrating how an autonomic controller, together with the effector, is capable of responding to malicious abuse.

Keywords. identity management, self-adaptive authorization, federated authorization, computing security, autonomic computing

1 Introduction

Autonomic computing is fast becoming a means of improving traditional methods for repairing, managing and evolving systems in a plethora of application domains [1]. One particular interest within autonomic computing is solutions that enable autonomic management of entities within complex systems, such as the autonomic management of federated authorization infrastructures (federations). Federations can be represented as a network of identity providers (IdPs) that identify and authenticate subjects (users) in order to facilitate their access to remote service providers' (SPs) resources.

One aspect of managing federated authorization infrastructures is how to respond to subjects whose interactions and usage of resources becomes abusive, or malicious, whilst being within the bounds of their access privileges. For example, in the case of Wikileaks, an army intelligence officer allegedly accessed (then subsequently leaked) hundreds of thousands of classified U.S. Department of Defence cables [2]. Since each individual access was granted by the SP's access control system, it did not detect any abuse. Had it done so, and the system had been federated, then the SP would have

faced a dilemma, since the user's privilege attributes would have been assigned by a trusted IdP, and not by itself. The SP consequently loses some control over exactly who the subjects are and how they are authorised. It is a challenging task for human administrators to monitor, and respond to these potential malicious events today. They may only resolve these by either 1) removing the trust they have placed in the IdP, 2) by personally requesting the IdP to limit the offending subjects' privilege attributes, or 3) by stopping all accesses by anyone with these privilege attributes (unless they can uniquely identify the particular user, which is not always the case in federated systems). This is clearly time consuming and unsatisfactory.

Previous work [3] identified the need for autonomic management of (federated) authorization infrastructures, and described the Self-Adaptive Authorization Framework (SAAF). SAAF analyses subject behaviour via subject usage of authorization services (i.e., from authorization decisions). It considers various adaptation strategies against the IdPs' and SP's components within federations. There are several challenges when considering the autonomic management of IdPs. Whilst SPs own the resources where the malicious behaviour is identified, they do not own the subject's privilege attributes that confer access. These belong to the IdPs. Yet SPs are required to limit these privileges in order to prevent further malicious events within their own domain. Assuming a SP deploys an autonomic controller, the controller is normally restricted in its operation to the SP's domain whilst the IdP is outside this domain. Therefore, adaptation strategies can only be executed on the IdP with its permission. Without this, the autonomic controller will need to resort to high consequence adaptations within its own domain (such as removing all trust in the IdP). Increasing the likelihood that an IdP will permit the requested adaptations requires a secure and configurable solution, in which the IdP maintains ownership of its data, and can act on adaptation requests through varying means, which it ultimately controls.

The contribution of this paper is to define and implement the enabling concepts of automated and semi-automated management of subjects' privilege attributes within IdP domains, by SP domains. We describe the enabling solution as an effector, to be deployed within an IdP's domain. An implementation of the effector is deployed as part of an extended SimpleSAMLphp [4] IdP. An instance of SAAF, the autonomic controller, is deployed as part of a SimpleSAMLphp SP. We show that the performance of this system is good.

The rest of this paper is structured as follows. In Section 2, we review background and related work. In Section 3 we describe a conceptual design to the problem area. In Section 4 we detail an implementation of the conceptual design. Section 5 describes the experimental results. Finally, Section 6 concludes by summarising the work done so far, and indicating future directions of research.

2 Background and Related Work

This section details a brief review of background and current work, which motivates this research, within the areas of authorization infrastructures, identity management, and autonomic computing.

2.1 Federated Authorization Infrastructures

Federated authorization infrastructures (federations) refer to a collection of distributed services and assets (such as privilege attributes and authorization policies) that enable the sharing and protection of organisational resources, across organisational domains [5]. Organisations, known as SPs, share their resources with users authenticated by trusted third party organisations, known as IdPs. Authorization is given in conformance to an authorization model, such as the Attribute Based Access Control (ABAC) model [6]. ABAC authorization policies state the permissions (actions executable against a resource) assigned to various attribute types and values, which the IdPs are required to store and provide on behalf of their subjects.

There are various technologies that exist to enable federations. X.509 [7] defines a distributed privilege management infrastructure built with attribute certificates, upon which SAML attribute assertions [8] were modelled. Shibboleth [9] uses the SAML standard to protect web services over a network, requiring users accessing Shibboleth protected resources to authenticate against their IdP in order for the latter to provide attribute assertions to the former. SimpleSAMLphp [4] is an alternative implementation of the same SAML standard. PERMIS [10] was originally an implementation of the X.509 privilege management infrastructure, but was subsequently enhanced to support SAML attribute assertions as well. OpenID Connect [11] and IETF Abfab [12] are two of the latest federation protocols, which are in the final stages of being standardised.

2.2 Self-Adaptation and Authorization

The Self-Adaptive Authorization Framework (SAAF) [3] is a solution for improving the monitoring and regulation of resource usage within federations, through autonomic management. SAAF adapts authorization assets (i.e., privilege attributes and authorization policies) in response to identifying malicious/abusive behaviour. Malicious behaviour is identified by the monitoring of subject usage in conformance to behaviour rules (defined at deployment) that classify malicious patterns of usage (e.g., high rate of access requests). The deployment of SAAF (Figure 1) comprises an autonomic controller, owned by a SP, monitoring the use of its authorization services in relation to its protected resources. This is achieved through a feedback control loop [13], adapting authorization assets to further prevent or mitigate malicious behaviour.

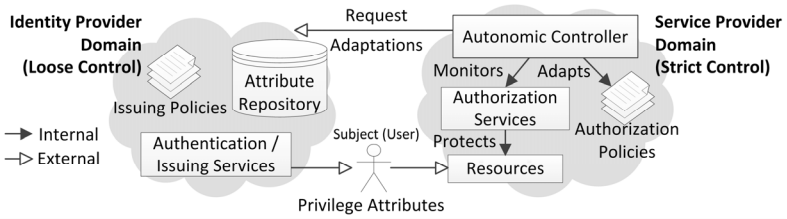

Fig. 1. Autonomic management in federated authorization infrastructures

In the case of adapting SP assets (authorization policies), the SAAF autonomic controller is trusted by the SP to carry out the necessary adaptations, implying strict control. However, a critical adaptation within SAAF is the adaptation of authorization assets belonging to IdPs where control is restricted (loose control).

2.3 Related Work

To the best of our knowledge no other works explore the role of autonomic controllers across different management domains, in particular, within the area of federated identity management. However, similar works exist which explore the autonomic management of complex systems. For example, an autonomic management framework [14] for web services describes autonomic controllers deployed at the point of service, enabling services to identify and resolve their own management problems. Our work differs in that autonomic controllers are not applicable for all types of services within a federated authorization infrastructure, as malicious behaviour identified by SPs cannot be identified by the source (IdPs). This requires external autonomic controllers to operate across management domains. Other papers explore the role of autonomic management and cooperation between differing services within a network [15], whereby trust and reputation is relied upon to increase the favourability of cooperation (in our case, adaptation). In comparison, our work provides a platform for autonomic management in which trust already exists for issuing of privilege attributes, as a fundamental component of federations.

3 Managing Identity Providers

This section details the conceptual design for enabling the autonomic management of identity providers.

3.1 Conceptual Design

The ability to manage IdPs relies specifically on the trust that an IdP has in the (autonomic controller of the) requesting SP. For example, a SP identifies malicious/abusive activity associated with a subject belonging to an IdP. The SP might request the IdP to remove the subject's identity attribute(s) which grant the subject access rights at the SP. However, these identity attributes may give the subject access rights at many SPs, and not only at the abused SP. In the latter case the IdP might easily decide to grant the removal request. In the former case the decision is more difficult and hinges partially on whether the IdP is more concerned about upsetting its subject or the many SPs that it has trust relationships with (and which the subject might similarly be abusing). If the request is refused the SP is left with several options:

- allow the malicious activity to continue (for example, when the alternative options have a greater cost when compared to the malicious activity), or
- ask the IdP to alter its attribute release / issuing policy so that it does not issue attribute assertions for this subject, or

- remove access rights from this specific subject (challenging, as it depends on how subjects are identified, i.e., through persistent or transient IDs) or
- remove access rights from all subjects who share the same set of identity attributes with the abusive subject, or
- remove all trust from this particular IdP (for example, the IdP has refused numerous adaptation requests and the abusive behaviour continues).

To avoid the last option being taken, it is in an IdP's interest to comply with requests for management changes in relation to either its attribute release policy or one of its subject's identity attributes, otherwise SPs may associate too much risk in using the IdP. It is for these complex reasons that we have defined the autonomic management to be about the IdP's output i.e., its assertions about a subject's privilege attributes, so that it is independent of the actual internal mechanisms employed by the IdP to achieve this. Autonomic controllers only depend on the final outcome, which is to control the privilege attributes that the IdP will assert for a particular subject in the future. The IdP therefore remains in control of the corrective action that is to be taken, and deciding how to achieve the desired objective. We therefore propose the following two definitions:

Definition 1. We define the automated management of a subject's privilege attribute assertions within a federated identity management infrastructure as: *the ability for an autonomic controller, situated in a SP's domain, to issue adaptations to an IdP's domain in order to immediately control the privilege attribute assertions that the IdP will issue for that subject when it subsequently requests access to the SP's resources.*

Definition 2. We define the semi-automated management of a subject's privilege attribute assertions within a federated identity management infrastructure as: *a variant of definition 1, whereby the IdP's domain queues adaptations for a human controller to review, before execution.*

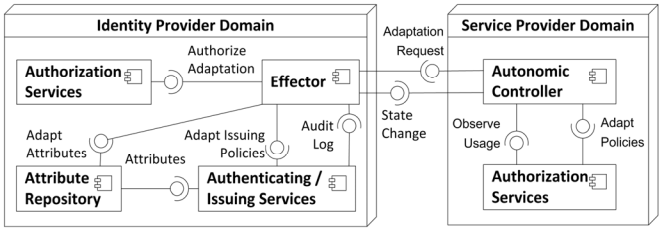

Fig. 2. Conceptual design

Figure 2 details the conceptual components of a managed IdP, which are required both to provide information to the *autonomic controller* (within the domain of a SP), and to control and enable it to request changes to a subject's asserted privilege attributes. The *effector* is the enabler for adaptations concerning an IdP. The *authorization service* at the IdP authorizes the autonomic controller, via the effector, to change either the *issuing policy* (which controls the subject's attribute assertions) or the *attribute repository* (which holds the subject's attributes). The *audit log* provides the effector with mappings between the local IDs of subjects and the IDs presented to the SP in the security assertions. The *authorization services* at the SP utilise the subject's

security assertions provided by the IdP's *authenticating and issuing services*. The autonomic controller requests adaptations against the IdP's effector, and receives state changes (i.e., subject no longer has privilege attribute 'x'), to confirm adaptations.

3.2 Identity Provider

We assume an IdP is capable of authenticating a user as being one of its subjects, and of providing attribute assertions about an authenticated subject to SPs. The IdP is capable of utilising supporting technologies that facilitate the storage and access of subject credentials/privilege attributes, for example, the Lightweight Directory Access Protocol (LDAP). These privilege attributes are assumed to be cryptographically secured and provided to trusted SPs as security assertions, following a standard protocol, such as SAML [8]. We also assume IdPs are able to log and audit security assertion assignments, as well as the authentications made through the IdP authentication services and any random, transient or session identifiers that are assigned to the subjects in the security assertions. Without these auditing capabilities, IdPs are unable to map session usage to actual subjects, in case they need to identify subjects when responding to notifications of malicious activity.

3.3 Autonomic Controller and Service Provider

The autonomic controller is capable of observing activity within the SP's resources to produce a state, specifically in relation to the accessing subjects and the use of subject privileges. The autonomic controller is able to classify malicious/abusive behaviour as behaviour rules. Behaviour rules are defined at deployment by sources of authority within the SP domain, and relevant to the SP's environment (i.e., academic / governmental). The autonomic controller is able to assess conformance to behaviour rules by observing subject usage, and respond when abusive behaviour has been identified. We make the assumption that the responses made by the autonomic controller are necessary, although the method in which abusive behaviour is identified, and the response chosen, is not covered by this paper. The autonomic controller is placed in the SP domain, as it is intrinsic to identification of malicious activity attributed through the subject's direct actions against the SP.

In the case of managing IdPs, an autonomic controller's adaptations refer to the modification of privilege attribute assertions. Each request made by an autonomic controller specifies an abstract adaptation operation along with enabling information, such as the persistent ID to which malicious behaviour is attributed, and the privilege attributes used. Requests are made over a reliable communications protocol and are idempotent, meaning that the autonomic controller will expect to always get a response. The autonomic controller may continue to make the same request (until a timeout is reached) if a response is not received, without adapting the final state of the IdP's system. Upon timeout or a failure response the adaptation is classed as failed. Request-responses may be synchronous or asynchronous. Synchronous communications are used to implement the automated management of a subject's attribute asser-

tions, whereas asynchronous communications are used to implement semi-automated management.

3.4 Identity Provider Effector

The IdP's effector is under the full control of the IdP administrator. He/she configures it to process adaptations requested by a SP's autonomic controller, either synchronously, or asynchronously. Communication flows between the IdP's effector and IdP software are made internally and rely on a host's operating system to ensure security. Communication between an autonomic controller and an IdP's effector are executed via secure communication, such as TLS/SSL, and require mutual authentication.

The effector requires access to issuing policies, attribute repositories and audit logs, within the IdP. Access to issuing policies is required in order to adapt the policy controlling the subjects' privilege attributes asserted by the IdP (if allowed by the administrator). Access to logs is required to map between an identifier (persistent or transient) that the SP has received, and the internal identifier of the subject. Access to attribute repositories is needed to modify a subject's privilege attributes (if allowed).

The effector supports a set of abstract adaptations that are necessary when managing an IdP. It is expected to translate these abstract adaptations into concrete adaptions that are supported by the underlying technology. For example, 'remove subject's privilege attribute assertion' may be translated into the relevant LDAP modify command in order to be executed against the LDAP directory, or into the appropriate Shibboleth attribute release policy to stop the SAML attribute assertion being created. The list of executable adaptations is as follows, and these are referred to as the effector operations: 1) Remove privilege attribute assertion from all subjects, 2) Remove privilege attribute assertion from identified subject, 3) Add privilege attribute assertion for all subjects, and 4) Add privilege attribute assertion for identified subject.

A consequence of defining such a set of abstract operations is that it allows the IdP to utilise an authorization service to determine which operations to allow and which to deny, and then to determine how to implement the allowed ones. The addition of privilege attribute assertions is provided in order to specify a subject with reduced privileges (as a new attribute), where attributes exist within a hierarchy. For example, a Supervisor attribute inherits from an Employee attribute.

4 Implementation

This section describes the implementation of the effector for a SimpleSAMLphp [4] IdP, and shows how it can be integrated with an autonomic controller.

4.1 Federated Authorization Infrastructure

The effector together with a single SP and a single IdP are implemented as a SAML compliant federation. SimpleSAMLphp is used as the unifying technology to enable communication between the two providers. This is a basic federated authorization

infrastructure to demonstrate the effector, however the effector could potentially be used in setups with multiple services and IdPs.

The IdP is implemented on a single host machine, whereby an instance of SimpleSAMLphp is installed and configured to provide IdP services. An open LDAP server is installed to store subject privilege attributes and authentication information. Finally, an implementation of a SimpleSAMLphp IdP effector is installed, compliant with our conceptual design, to enable cross-domain management. The effector makes use of open LDAP's access control lists in order to manage the extent of adaptations a client is permitted to request.

The SP is implemented across two host machines, one to host the SP's resources (resource host), and one to host an autonomic controller and authorization services (authorization host). The authorization host deploys an implementation of SAAF [3] and an instance of PERMIS [10], which is used to protect the SP's resources deployed on the resource host. PERMIS is capable of utilising ABAC authorization policies to provide the validation of SAML attribute assertions issued by IdPs, and access control decisions to the resource host.

4.2 Extending SimpleSAMLphp

To facilitate operations by the IdP's effector, we extended the logging capabilities of SimpleSAMLphp in order to always ensure the correct retrieval of a subject's LDAP distinguished (unique) name. SimpleSAMLphp stores its log information in a relational database (SQLite). In its original configuration, SimpleSAMLphp was only capable of mapping persistent IDs to subject attribute values. Additional information, such as attribute type, LDAP host, and LDAP search base, is needed in order to locate the actual subjects' LDAP entries for both transient and persistent IDs. Whilst some of this information e.g. LDAP host names, is available in the SimpleSAMLphp configuration file, it is not persistent to configuration changes. For this reason we decided to record all this additional information in the log DB, so that the effector is always able to identify the abusive subject's distinguished name.

4.3 SimpleSAMLphp Effector

The SimpleSAMLphp effector, shown in Figure 3, implements a subset of the effector component shown in Figure 2. It is a PHP web service hosted alongside the SimpleSAMLphp IdP service. It has access to the log database stored within the SimpleSAMLphp directory, which enables it to map between persistent and transient IDs and a subject's distinguished name. Web service clients, such as the SAAF controller, can access the effector providing they have been issued with a trusted client X.509 certificate. Mutual SSL/TLS authentication is required and the client's certificate distinguished name is used to identify the requesting client.

Although the effector component conforms to the conceptual design described in Section 3, it is somewhat restricted due to the limited capabilities of SimpleSAMLphp. SimpleSAMLphp relies upon an attribute repository, such as LDAP, along with an attribute release / issuing policy which is represented by a PHP configu-

ration file. However, the attribute release policy is constrained to stating only which attributes can be released to which SPs, regardless of the individual subject. As a result the effector adapts subject attributes held in the LDAP repository in order to achieve the *per subject* granularity. Modifying the privilege attribute assertions for *all subjects* is implemented by changing the SP's PERMIS credential validation policy rather than the SimpleSAMLphp attribute release policy. However, if the SP's authorization services do not provide credential validation policies, then adaptation of attribute release policies will be needed.

When operating synchronously, the effector utilises the LDAP access control lists in order to authorize the subject level adaptation requests, notifying requesting clients of failure in case the client is unauthorized. When operating asynchronously, meaning manual review is required, the effector queues requests and notifies administrators via email when new requests are received. Human administrators then review the queued requests before allowing the effector to execute an adaptation and inform the client of success or failure. The effector is initialised once it receives a SOAP message request from a client. From here SOAP requests are processed in the following manner: 1) mutually authenticate the requesting client over TLS and obtain the requestor's distinguished name (DN) from its certificate, 2) verify the requested operation is valid, 3) retrieve the target subject's unique attribute mapping from the persistent/transient ID stored in the SimpleSAMLphp audit log database, 4) retrieve the subjects' DN(s) using the relevant LDAP host name and search base, 5) translate the requested operation into LDAP executable operations, 6) bind the requestor's DN to the relevant LDAP server, 7) execute the update operation against LDAP, providing the access control list allows it, 8) respond to the client with confirmation of the state changes.

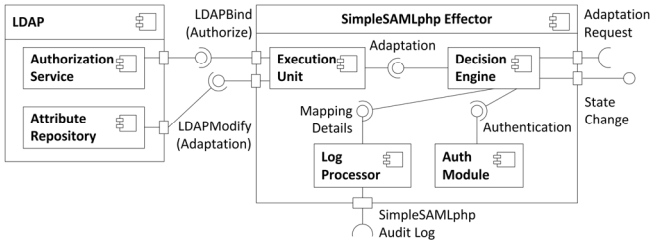

Fig. 3. Effector for SimpleSAMLphp IdP

5 Experiments

In this section, we discuss the deployment of the SimpleSAMLphp IdP and its effector in relation to a case of abuse identified with a SAAF controller.

5.1 Adaptation Scenario

The SimpleSAMLphp IdP is configured to issue persistent IDs with the release of privilege attributes for authenticated subjects. An LDAP directory is populated with

subject authentication and privilege attributes. The effector is deployed, configured to run synchronously, and rely on an LDAP access control list to restrict the actions of a SAAF autonomic controller.

The SP is configured to host a payroll web application that utilises a policy enforcement point (PEP). The PEP requires subjects to 1) authenticate against the subject's IdP, 2) obtain the subject's releasable privilege attributes in the form of a SAML assertion (via SimpleSAMLphp), and 3) utilise the SP's authorization services to provide an authorization decision. PERMIS is deployed with an authorization policy that states the IdP is trusted to assign the privilege attribute 'permisRole=employee' to its subjects. This privilege attribute can be used to execute the permission of 'get employee payslip' on the payroll web application. The SAAF autonomic controller is deployed with a simple behaviour policy stating that no single subject belonging to the IdP may request access to any of the SP's resources, greater than 10 requests per minute. This is to stop automated attacks. SAAF profiles usage based on subjects' persistent IDs associated with the federated access requests. Should this rule be broken SAAF identifies the subject as committing abuse and can respond through various adaptation strategies. The best adaptation strategy is chosen based on a weighted decision problem solving algorithm, for example, considering the cost of realising the adaptation strategy against the cost of allowing abuse to continue.

In this scenario, a subject registered with the IdP, requests access to 'get employee payslip' more than 10 times within a minute interval. Each time the subject requests access, PERMIS logs the request, detailing the subject's attributes from the subject's SAML assertion, the subject's persistent ID, and access decision given. The SAAF autonomic controller builds up the subject's pattern of access based on these logged events, checking conformance access against its behaviour policy. SAAF identifies that the stated behaviour rule has been broken, and reacts by requesting the SimpleSAMLphp effector to prevent the subject from using the privilege attribute of 'permisRole=employee'. The SAAF autonomic controller encapsulates this request in a SOAP message, which is sent over a mutually authenticated HTTPS connection to the effector. It contains an operation (remove privilege attribute), the subject's persistent ID observed from the subject's SAML assertions, the SP's ID to identify where the persistent ID was used, attribute type (permisRole) and attribute value (employee).

Providing the effector's response to the client indicates a successful adaptation (i.e., subject will no longer be issued permisRole=employee), the SAAF controller assumes the adaptation has been successful. However, if the response indicates an unsuccessful state, the offending subject is free to continue committing malicious behaviour. If the subject's behaviour continues, SAAF may take steps to remove the trustworthiness of the IdP in question, but this is not addressed here.

5.2 Performance and Load Tests

We have executed four types of load and performance tests. These tests are categorised as T1 – successful adaptation, T2 – invalid operation, T3 – invalid subject mapping, and T4 – LDAP error (either not authorized or unable to execute action). Tests were performed on two virtual machines (Debian 6.0.5 512MB memory hosted on a

2.4Ghz, 3GB memory MS Windows machine), as server and client, where threads on the client machine were used to depict multiple virtual clients.

We measured the average response times within an interval of one second (reflecting the minimum SAAF autonomic controller adaptation cycle), issuing requests within a single initial burst until the interval was complete. On average, with the minimum load of one client (one SAAF) issuing one request per second, we found performance of T1 requests could be executed in 65ms, T2 in 49ms, T3 in 50ms, and finally T4 in 62ms. We identified that the maximum load (Figure 4) was reached with 18 clients executing one request within the one-second interval.

In practice we do not expect SAAF to create a high load on this effector, due to the nature in which it executes adaptation strategies. As more adaptation requests are made, it is likely to coincide with increased levels of malicious activity, causing the autonomic controller to resort to high consequence adaptations that are out of scope of the effector, such as changing its local PERMIS policy.

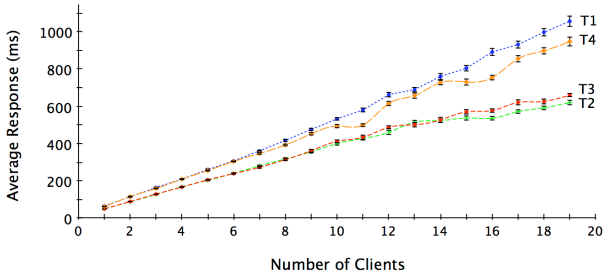

Fig. 4. Average (mean) response time, with standard error, against number of clients

6 Conclusion

This paper has presented an approach for enabling the autonomic management of federated identity providers (IdPs) across independent management domains. The motivation for this work is the fact that service provider (SP) domains can diagnose IdP domains as the source of malicious abuse. At the conceptual level, the basis of the proposed approach is the integration of an autonomic controller, positioned in the domain of a SP, with an effector, positioned in the domain of an IdP. We present the conceptual design of the effector, whilst satisfying key safeguards such as, ensuring the IdP remains in complete control of its assets. This effector has been implemented and evaluated through the deployment of a federated authorization infrastructure, which incorporates a SimpleSAMLphp IdP. We have shown that an autonomic controller is able to manage, via the effector, an IdP's ability to assign privilege attributes to its subjects. Through performance and load testing, we have shown that the IdP's effector is capable of operating with multiple autonomic controllers when handling adaptation requests within an autonomic controller's minimum adaptation cycle.

In the work described in this paper, it is recognised that the autonomic controller does not have strict control over the IdP, and relies on the IdP's goodwill. In order for

control to be more effectively applied, it would be necessary to have a legal service agreement or similar between the SP and IdP, whereby the IdP agrees to enact the SP's adaptation requests. In this way, the sphere of control exercised by the SP's autonomic controller would extend beyond the domain of the SP with which it is associated, to that of the IdPs to which the SP is contractually bound. Our future work aims to explore the requirements of service agreements between SPs and IdPs in order to ensure control when managing subjects' access rights between different domains.

References

1. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. In Computer 36, pp 41--50 (2003).
2. G. Adams.: Private Memo Exposes US Fears over Wikileaks. In The Independent. Available at http://www.independent.co.uk/news/world/americas/private-memo-exposes-us-fears-over-wikileaks-2177041.html (2011).
3. Bailey, C., Chadwick, D.W., de Lemos, R.: Self-Adaptive Authorization Framework for Policy Based RBAC/ABAC Models. In Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, pp. 37--44 (2011).
4. SimpleSAMLphp Version 1.9.2. Available at http://simplesamlphp.org/.
5. Chadwick, D.W.: Federated Identity Management. In A. Aldini, G. Barthe, and R. Gorrieri (Eds.): FOSAD 2008/2009, LNCS 5705, pp. 96--120, (2009).
6. ITU-T Rec X.812 (1995) | ISO/IEC 10181-3:1996 "Security Frameworks for open systems: Access control framework".
7. ISO 9594-8/ITU-T Rec. X.509 (2001) The Directory: Public-key and attribute certificate frameworks.
8. OASIS "Security Assertion Markup Language (SAML) Version 2.0".
9. Morgan, R. L., Cantor, S., Carmody, S., Hoen, W., Klingenstein, K.: Federated Security: The Shibboleth Approach. EDUCAUSE Quarterly, (2004).
10. Chadwick, D.W., Zhao, G., Otenko, S., Laborde, R., Su, L., Nguyen, T.A.: PERMIS: A modular Authorization Infrastructure. In Concurrency and Computation: Practice and Experience, pp 1341--1357 (2008).
11. Sakimura, N. et al. "OpenID Connect Standard 1.0 - draft 18". 26 March 2013. Available at http://openid.net/specs/openid-connect-standard-1_0.html.
12. Howlett, J. et al. "Application Bridging for Federated Access Beyond Web (ABFAB) Architecture". draft-ietf-abfab-arch-05.txt, 25 Feb, 2013.
13. Brun, Y., Serugendo, G.M., Gacek, C., Giese, H., Keinie, H., Litoiu, M., Muller, H., Peeze, M., Shaw, M.: Engineering Self-Adaptive Systems through Feedback Loops. In Software Engineering for Self-Adaptive Systems, pp 48--70 (2009).
14. Cheng, Y., Leon-Garcia, A., Foster, I.: Toward an Autonomic Service Management Framework: A Holistic Vision of SOA, AON, and Autonomic Computing. In IEEE Communications Magazine 46, Issue 5, pp 138--146 (2008).
15. Psaier, H., Juszczyk, L., Skopik, F., Schall, D., Dustdar, S.: Runtime Behaviour Monitoring and Self-Adaptation in Service-Oriented Systems. In Proceedings of the 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems, pp 164--173 (2010).