SHIBBOLETH WEB-PROXY FOR SINGLE SIGN-ON OF CLOUD SERVICES

Thomas Rübsamen and Christoph Reich
Cloud Resarch Lab, Hochschule Furtwangen University, Furtwangen, Germany

Keywords: Cloud Computing, Shibboleth, Single Sign-on, Reverse Web Proxy.

Abstract: Single Sign-On (SSO) allows users to access services, for which they possess sufficient access rights, without re-authentication once they are authenticated successfully. Shibboleth supports SSO of web services and allows building federations.
In this paper a Shibboleth web proxy is described, which integrates a Shibboleth service provider to manage authentication and extends Cloud management systems by enabling SSO of multiple cloud services. It is shown how this approach can be used for highly dynamic Cloud environments, where services are often added and removed. The Shibboleth web proxy implementation has been contributed to the Open Source Community and is made available in the OpenNebula EcoSystem.

1 INTRODUCTION

Single sign-on solutions are more and more widely deployed in today's Internet environment. The main reason for this is the rapidly growing number of credentials (e.g. usernames and passwords) a user has to manage. Single sign-on allows the reduction of user credentials while offering a uniform access mechanism to web services.

Shibboleth(shi, 2011) is an open source single sign-on system which is commonly used in academic service infrastructures like the "Authentication and Authorization Infrastructure" provided by the German Research Network ("Deutsches Forschungsnetz, DFN-AAI")(DFN-AAI, 2011). This infrastructure forms a federation of research facilities like universities or institutes but also commercial and non-commercial entities in Germany. So far there are 68 identity providers organized in the DFN-AAI, including the identity provider of the University of Applied Science Furtwangen. In a federation, Shibboleth-enabled web services can be used by all members of a federation according to the access policies of the services, based on common agreed Shibboleth attributes.

Another important technology in today's IT infrastructures is cloud computing. One of its main characteristics is a highly dynamic service environment. The University of Applied Science Furtwangen is actively developing a private cloud infrastructure, which offers services to its employees and students (Sulistio et al., 2009) and is based on OpenNebula (OpenNebula, 2011). The integration of this private cloud infrastructure and the services hosted in this environment into the university's user management and especially into the existing Shibboleth-based single sign-on system is the focus of this paper.

To keep an overview of existing bindings between services (and their service providers) and the identity provider it is quite common to hard-code this relationship. This leads to manageability problems when Shibboleth needs to be used in very dynamic environments where services are often added and removed. Static configuration also means a lot of administrative overhead. In this scenario the identity providers need to be reconfigured every time a service (and its service provider entity) gets added or removed. The resulting downtimes are unacceptable because of the major importance of this system component for other, non-cloud-based services. In cloud-based environments the integration of services into a shibboleth-based authentication and authorization infrastructure need to be automatic and as dynamical as possible.

In this paper a system is described, which was developed to allow usage of Shibboleth-based Single Sign-On for cloud-based services in a dynamic way. With this system it is also possible to host web services in a private network, as defined by RFC 1918 (rfc, a) to overcome the limited availability of public IPv4 addresses.

This paper is structured in the following way. In

Reich C. and Rübsamen T..
SHIBBOLETH WEB-PROXY FOR SINGLE SIGN-ON OF CLOUD SERVICES.
DOI: 10.5220/0003926000890095
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CLOSER-2012), pages 89-95
ISBN: 978-989-8565-05-1

this section the necessity and the goals of the developed system were described. In section 2 follows a short introduction to the Shibboleth single sign-on system and a classification of different web proxy scenarios which are relevant for the developed system. In section 3 follows a description of the proposed solution with its associated advantages and disadvantages. Section 4 describes the OpenNebula integration, followed by section 5, which presents the results of an internal evaluation of the performance of the Shibboleth web proxy system. Related work is presented in section 6 followed by a conclusion and possible future development tasks in section 7.

2 SHIBBOLETH AND HTTP PROXY

First, in this section, an overview about the Shibboleth SSO systems (shi, 2011) is given. Second the differences between the used reverse proxy concept and the commonly used forward proxy concept is described.

2.1 Shibboleth

The two main components in a Shibboleth infrastructure are the *Identity Provider* (IdP) and the *Service Provider* (SP). The *Identity Provider* is usually connected to an organization's user management system and takes control over user authentication. Additionally the IdP provides information about users in form of attributes which are stored in the user management backend (e.g. a LDAP backend). These attributes (e.g. username, role etc.) can be used to make authorization decisions whether a user can access certain resources or not. The *Service Provider* (SP) is the second important component of Shibboleth. It is usually run as a plugin on the web server hosting the resource which needs to be protected. The SP authorizes resource access requests and also takes control of the user authentication process if no authentication token exists in the session context. These processes are usually completely independent of the resource which is to be protected. This means a resource (e.g. a web application) does not necessarily have to be customized for use in combination with Shibboleth.

If multiple resources are located on one web server it is also possible to protect those with one SP running on that host. One important thing to consider is that there always has to be a binding between a SP and its IdPs. This can be done by adding the SP's metadata to an IdP's configuration, which can either be done manually or automatically by requesting current metadata information from the SP (Shibboleth 2 Documentation, 2011a). Another way to create this binding is to register the SP in a federation which propagates the metadata information to the IdP. A Shibboleth Federation is an agreement between service providers and IdPs wishing to access those resources. All parties need to agree on a common set of acceptable authorization attributes for their users, and a schema to describe them.

2.2 HTTP Proxy

The classic web proxying differentiates two distinct communication directions: **Forward Proxy**, a classic web cache is also called a forward proxy and is usually located near the client. **Reverse Proxy**, a reverse proxy is near the requested resource and used for server performance optimization. Many of today's HTTP daemons provide reverse proxying functionality, such as the *Apache HTTP Daemon* which has been used for developing our approach described in this paper.

3 SHIBBOLETH WEB-PROXY

In this section the concept of a Shibboleth web proxy and the developed prototype will be described. A reverse proxy in combination with a Shibboleth service provider (SP) on the same host will be used to provide authentication and authorization to resources hosted in the cloud (PaaS and SaaS services). Any HTTP requests to resources or services will be forwarded through the proxy server. Figure 1 displays the architecture of the developed system. The client is a classic browser session where a protected resource has been requested. The IdP acts like in the classic scenario. One major difference to the classic Shibboleth scenario is that the web servers hosting the resource do not act as SPs anymore. The SP functionality has been moved to a single host, the reverse proxy, in the middle of figure 1. There is only one SP which is bound to the home organization's IdP. Obviously in the described architecture the reverse proxy is a single point of failure and could be a single point of attack also as all traffic must be routed over it. We are aware of this, but see solving this problem as part of future research as it is not in the scope of this paper.

The reverse proxy is responsible for forwarding requests to the web servers. One problem that arises is how to identify which resource has to be contacted. To solve this, there are a couple of different possible identification schemes which can used. These schemes will be discussed in the following. One thing to keep in mind is that all requests are sent to the re-

verse proxy. Information about the actual resource that is to be contacted must be encoded into the request URL in some way. There are two possible ways to do this. First a part of the the request URL path can be used. Second the DNS domain name part of the request URL, especially the hostname part, can be used. This way Apache's virtual host functionality can be used to identify which resource to contact.

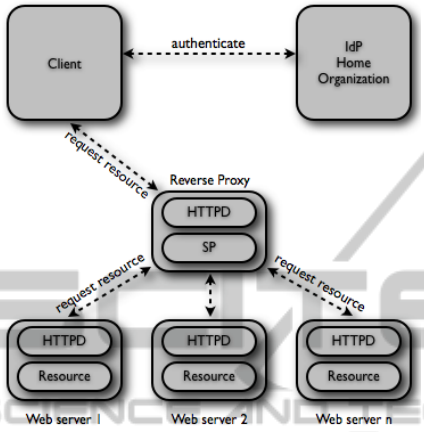

Figure 1: Shibboleth web proxy scenario.

3.1 Resource Identification using URL Path

Request are always sent directly to the proxy (e.g. "http://proxy.example.com"). To identify the actual requested resource, the path portion of the request URL is used (e.g. "http://proxy.example.org/servicea"). The proxy analyzes the path component and forwards requests to the according web server. This way different path values result in requests to different resources, and usually different web servers, behind the proxy. Figure 2 illustrates this scenario. A client requests access to the resource "servicea". The request is sent to the reverse proxy which acts as a Shibboleth SP. After authenticating the user and authorizing the access to the resource, the request is forwarded to web server 1 which hosts the requested resource. The authentication and authorization process works as previously described. Of course it is possible to use different authentication and authorization configurations for each resource (each configured path on the proxy) or no Shibboleth at all. If the authentication or authorization process fails, requests are not forwarded to the web server and access is denied.

This architecture allows relatively easy integration of a single SP into a classic reverse proxy. However there are some major downsides to this approach. Request URLs can quickly become overly complex. Another downside is that SSL cannot be used effectively. In fact there can be no end-to-end encryption using SSL. It is possible to use SSL between the client and the reverse proxy and it is also possible to use SSL between the reverse proxy and the web server hosting the resources, but the proxy server will always (temporarily) terminate the SSL connection and therefore see unencrypted data.

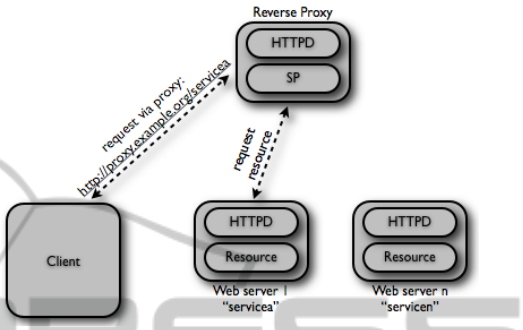

Figure 2: Proxy scenario with service identification using paths.

3.1.1 Extension: HTTP-redirect to HTTPS

As previously stated relying on paths for resource identification can become very complex very quickly and lead to unhandy URLs. A simple extension to this approach can reduce complexity by using HTTP redirects and virtual hosting capabilities on the reverse proxy. A name-based virtual host can be created on the proxy for every resource hosted behind it. Clients can now access URLs of a name-based virtual host. Those virtual hosts are combined with DNS alias records pointing to the proxy server. The proxy can now process the virtual hostname and create a request to the more complex path-based identification scheme and redirect the client to this URL. This method can also be used to force all requests to use HTTPS. The resulting less complex URLs makes it more easy for the user to handle them. Nevertheless this approach does not solve the SSL end-to-end encryption problem.

3.2 Service Identification using Virtual Hosts

Another way to identify resources behind the reverse proxy is the use of virtual hosting capabilities of the HTTP daemon (in our case Apache HTTPD). Virtual hosting was initially developed to allow more web services to be hosted on a single web server using multiple DNS aliases. There are two different types of virtual hosting:

- **IP-based Virtual Hosting.** When IP-based virtual hosting is used the web server listens for incoming connections on multiple IP addresses. These addresses can be aliases on the same network interface or addresses on different interfaces. For each address the web server is now able to host different virtual hosts (e.g. different web sites). To the client this seems like there are different web servers hosting each of those sites when there is in fact only one web server hosting them.
- **Name-based Virtual Hosting.** The second type is name-based virtual hosting. The web server listens on only one IP address. Using the fully-qualified DNS name it is also possible to host multiple sites on one web server. For this mechanism to work all DNS names point to one web server. This can be done using DNS aliases (CNAME resource records). When a resource is requested the web server now analyzes the HTTP request header. According to the hostname which is found the request is forwarded to a different virtual host.

Name-based virtual hosting is the more commonly used variant, when it is required to separate resources logically and hosting them on the same web server because it is usually easier to manage DNS aliases and there is no need for additional IP addresses.

A system which uses a Shibboleth proxy and virtual hosting for resource identification has to do access authentication and authorization on the basis of virtual hosts. Figure 3 displays this scenario. In the first step a client has to resolve a DNS alias (CNAME) like ”http://servicea.example.com”. The result will be the IP address of the proxy server. Now the client will send the resource request to the proxy server which extracts the hostname of the target resource from the request header. According to this hostname a virtual host is chosen and the request is forwarded to the right web server. During this process Shibboleth authentication and authorization is performed.

At first this seems like the more elegant solution to the problem of resource identification. But there are also some major downsides to this approach. For instance the usage of SSL or TLS between the client and the proxy is very complicated. The main reason for this is in the nature of the SSL/TLS protocol. The identification of the virtual host to which the HTTP request has to be sent is included in the HTTP request header. However, this information is not available on the proxy right away. In fact the request has to be decrypted first. The target of the request is a virtual host (e.g. ”servicea.example.com”) but the SSL connection was established to the proxy (e.g. ”proxy.example.com”). This results in a hostname mismatch error. The most simple solution to this would be using a wildcard certificate (e.g. for *.example.com) on the proxy, but this is a generally discouraged approach. A better solution would be using a TLS protocol extension which tries to solve this problem.

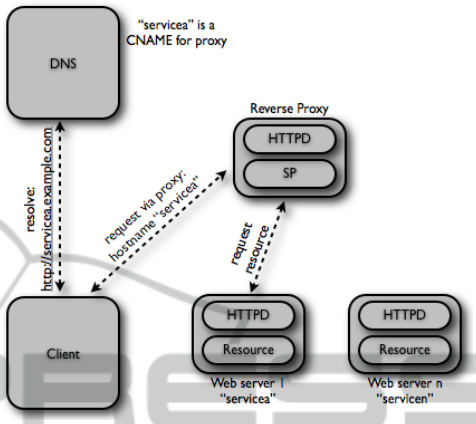

Figure 3: Proxy scenario with service identification using virtual hosts.

3.2.1 Extension: Server Name Indication

Server Name Indication (SNI) is an extension for the Transport Layer Security (TLS) (rfc, b) protocol. It allows the client to send the server name of the request URL during establishment of the the encrypted connection. The server name is usually the fully qualified DNS hostname of the requested resource. This way it is possible for the web server to choose the right certificate (e.g. the certificate matching the hostname of the virtual host).

A major problem in current browser implementations of TLS is the support for SNI. Despite the wide support of SNI on all major platforms (Microsoft Windows, Linux, Mac OS) and Browsers in their current releases it gets problematic if not the most current browser versions are used. For best compatibility it is still best to not rely on SNI support (TLS SNI Test Site: *.sni.velox.ch, 2011). The most problematic combination would be Windows XP, which is still widely used, and Internet Explorer.

4 IMPLEMENTATION OF SHIBBOLETH WEB-PROXY IN THE CLOUDIA PROJECT

The University of Applied Science Furtwangen is actively researching Cloud Computing technology. The Cloud Research Lab’s project ”Cloud Infrastructure

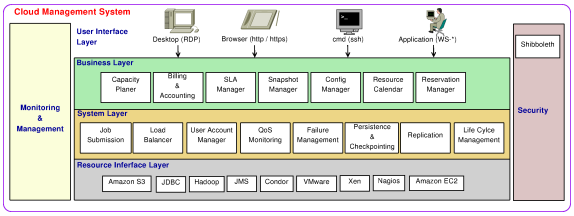

Figure 4: Cloud management system of CloudIA.

and Applications" (CloudIA) (Cloud Research Lab - Department of Computer Science, 2011) has developed a private cloud infrastructure based on OpenNebula (OpenNebula, 2011) and cloud based services to enhance this platform.

CloudIA leverages various virtualization technologies, such as Xen and KVM, and supports Service-Level Agreements (SLAs) in IaaS and SaaS models, as shown in Figure 4. In this Figure, the Cloud Management System (CMS) of CloudIA is divided into several layers (User Interface, Business, System, Resource Interface) as well as two components (Monitoring & Management, Security) for extensibility and maintainability.

Services in this cloud system are hosted on dynamically created virtual machines. That means the fluctuation of VMs (creation and deletion) is very high. As previously stated it is one of the major goals to integrate the cloud infrastructure with the user management system of the university. Therefore it was necessary to find a solution which integrates services hosted in the cloud with our Shibboleth infrastructure while maintaining the dynamic character of cloud environments. As a result of the research conducted by the Cloud Research Team, a Shibboleth web proxy was developed as shown in Figure 1. The proxy itself is running on a virtual machine hosted in the cloud.

Because of the large number of VMs which can be hosted in the cloud it became a requirement to be able to use private IP addresses. By passing all traffic through the publicly available reverse proxy it is very simple to run services in a private subnet while still being able to access them publicly through the proxy.

Resource identification in our private cloud is done using the previously described path-based identification mechanism. The main reason for this was the better support for SSL without relying on SNI. Although, it would also be possible to use the virtual host mechanism in our architecture. In fact, both the path- and virtual host based approaches are implemented in our private cloud system.

Users who run services in the cloud can activate Shibboleth for protecting those services via a web-based management console. It is also possible to configure authorization rules using this console. This concept of a Shibboleth web proxy has been implemented and provided as Open Source under the name *StudiCloud* (Cloud Research Lab HFU,) as an extention of Opennebula (OpenNebula, 2011).

5 EVALUATION

To demonstrate whether the reverse proxy based approach to shibboleth authentication is feasible in practice or not, an evaluation was performed on the HFU Cloud Research infrastructure.

The evaluation scenario is composed of one reverse proxy and one target web server which hosts a simple HTML page approximately 10KB in size. Both servers are using the Apache 2.2 Web Server on Debian 5.0 and are hosted as virtual machines on the HFU cloud. To eliminate any interference between virtual machines, these two have been deployed on separate cloud hosts, interconnected with GBit-Ethernet.

Three different evaluation scenarios for accessing the target web server via the proxy have been developed:

- *Shibboleth Authentication.* In this scenario a shibboleth authentication workflow using the reverse proxy is performed before access to the target web server is granted.
- *Basic Authentication.* Apache supports a simple way of authenticating users. This mechanism is used to compare the complexity against the shibboleth scenario.
- *No Authentication.* The direct access scenario without any authentication needed is used as a reference to compare to.

JMeter is a well-known and well established tool for stress testing and performance testing of web sites and web applications. It provides enough flexibility and customizability options to perform benchmarks on the above mentioned scenarios.

During the evaluation, test plans for shibboleth, basic and no authentication workflows have been developed using JMeter. The primary focus of the evaluation is to show whether shibboleth in a reverse proxy scenario performs well enough to be used in high load, high traffic setups, as using it as a primary gateway in PaaS and SaaS scenarios would be. To eliminate any interferences, like load fluctuations on the cloud hosts, KVM process scheduling, these separate test plans have been merged into a single test plan and run in parallel. For each of the scenarios a setup of 10 threads requesting the same page sequentially 500

Table 1: JMeter evaluation.

	Average request time (ms)
No authentication	51
Basic authentication	52
Shibboleth authentication	54

times, which results in 5000 requests per scenario and 15000 requests overall, has been implemented.

Table 1 shows average request/response times of the tested scenarios. Obviously and expectedly using a reverse proxy which is forwarding requests to other web servers without any authentication performs best. Also using basic authentication which does no more than sending additional login credentials in the HTTP request header is only insignificantly slower. In the shibboleth scenario a slightly worse result could have been expected because of the complexity of shibboleth authentication. For instance there is additional communication between the apache *mod_shib* module and the service provider daemon *shibd*, which holds session information. But obviously this overhead does not pose much of a negative impact on the overall performance of this setup.

These results make the aforementioned integration of Shibboleth authentication into cloud computing infrastructures using a reverse proxy, an attractive way to maintain manageability and security.

Things to consider when integrating a Shibboleth reverse proxy are proxy resources, bandwidth and failover. Scaling up is the most common way to enhance the performance of a reverse proxy, when demands are rising. Scaling out (e.g. distributing load on multiple reverse proxies) can be a bit problematic because Shibboleth SPs store session information, which in this case need to be distributed among multiple reverse proxies. Shibboleth supports some mechanisms to cluster SPs. One way is to choose a master SP which stores session information and let other SPs communicate via TCP to it in case they need additional information. Another way would be to store session data in a central database and connect SPs to it. Either way those approaches add a lot of complexity and it needs to be seen, whether or not they are feasible performancewise. Another approach is to only protect the web application's login script and let the application itself handle authentication and authorization in its session management. This way the SP is only needed for initial authentication and attribute exchange at the very beginning of a session with the web application. Adding session stickiness to this timeframe solves a lot of problems, but requires the application to have session management and a login script has to be developed, which handles session creation. (see (Shibboleth 2 Documentation, 2011b) for further information on SP clustering)

The approach described in this paper suffers from one major problem. Requiring all requests to be handled by the reverse proxy makes it a single point of failure. This can be solved by applying common failover strategies. In fact deploying the proxy as a virtual machine has the advantage of hot migratability of the VM in case the hosting cloud node fails. Further research has to be conducted in this area.

6 RELATED WORK

Guanxi (Jie et al., 2008) is an open source implementation of the Shibboleth protocol and architecture for e-social science. Guanxi Shibboleth is integrated into the Sakai collaborative and learning environment and PERMIS technology is enabling a policy-driven, role-based, fine-grained access control. Our approach especially enables the OpenNebula Cloud Management System with Shibboleth but enhances the access to Cloud services by policies.

The national grid service (NGS) provides access to compute and data resources for UK academics. The work from Xiao Dong Wang et.al. (Wang et al., 2009) describes an architecture by which users are authenticated by the UK access management federation to acquire low assurance credentials to access Grid resources on the NGS. Technically it is integrated into a portal, whereas our approach is a proxy which can be integrated in other portals.

The work from Takaaki, K. et.al. (Takaaki et al., 2011) developed a web forward proxy server with authentication method using Shibboleth. This proxy solves problems in basic access authentication and digest access authentication supported by existing web forward proxy servers. Here the user already has to use a proxy and it is not supporting federations.

The approach discussed in this paper is exclusively based on open source technologies which makes it particularly interesting in an academic environment. For example TU Munich also uses Shibboleth as a single sign-on system (Hommel, 2010). This proxy solution could also be interesting for such infrastructures outside of cloud environments if use cases exist where additional flexibility is needed.

7 CONCLUSIONS AND FUTURE WORK

In this paper a solution to reduce the number of ser-

service provider instances in a dynamic cloud environment using a Shibboleth reverse proxy is presented. This reduction results in a more flexible, easily automatable and simpler administration of services while using Shibboleth-based single sign-on, authentication and authorization.

Two mechanisms for identifying services using URLs have been presented. The path-based mechanism has proven to be more flexible while providing best compatibility even to non-current web browsers.

Additionally, an evaluation has been conducted which has shown, that using a shibboleth service provider on a reverse proxy is insignificantly slower than directly accessing web sites and services. Compared to the gains in manageability, flexibility and security these performance drops are very acceptable.

The presented solution obviously suffers from a single point of failure, the proxy. Future work will analyze a clustered architecture to overcome performance and single point of failure problems.

REFERENCES

RFC1918 - Address Allocation for Private Internets.

Transport Layer Security (TLS) Extensions.

(2011). Shibboleth. http://shibboleth.internet2.edu/.

Cloud Research Lab - Department of Computer Science (2011). http://www.wolke.hs-furtwangen.de/.

Cloud Research Lab HFU. StudiCloud. http://opennebula.org/software:ecosystem:studicloud.

DFN-AAI (2011). https://www.aai.dfn.de/.

Hommel, W. (2010). Campus single sign-on und hochschulübergreifendes identity management. In Bode, A. and Borgeest, R., editors, *Informationsmanagement in Hochschulen*, pages 221–232. Springer Berlin Heidelberg. 10.1007/978-3-642-04720-6_19.

Jie, W., Young, A., Arshad, J., Finch, J., Procter, R., and Turner, A. (2008). A guanxi shibboleth based security infrastructure. In *Enterprise Distributed Object Computing Conference Workshops, 2008 12th*, pages 151 –158.

OpenNebula (2011). http://www.opennebula.org/.

Shibboleth 2 Documentation (2011a). Communicating with a Service Provider. https://spaces.internet2.edu/display/SHIB2/IdPSPCommunicate.

Shibboleth 2 Documentation (2011b). Shibboleth SP clustering. https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPClustering.

Sulistio, A., Reich, C., and Doelitzscher, F. (2009). Cloud infrastructure & applications – cloudia. In Jaatun, M., Zhao, G., and Rong, C., editors, *Cloud Computing*, volume 5931 of *Lecture Notes in Computer Science*, pages 583–588. Springer Berlin / Heidelberg. 10.1007/978-3-642-10665-1_56.

Takaaki, K., Hiroaki, S., Noritoshi, D., and Ken, M. (2011). Design and implementation of web forward proxy with shibboleth authentication. In *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on*, pages 321 –326.

TLS SNI Test Site: *.sni.velox.ch (2011). https://sni.velox.ch/.

Wang, X. D., Jones, M., Jensen, J., Richards, A., Wallom, D., Ma, T., Frank, R., Spence, D., Young, S., Devereux, C., and Geddes, N. (2009). Shibboleth access for resources on the national grid service (sarongs). In *Information Assurance and Security, 2009. IAS '09. Fifth International Conference on*, volume 2, pages 338 –341.