Authentication for web services

Drew Dean
Yahoo!, Inc.
ddean@yahoo-inc.com

January 26, 2006

Introduction

As the World Wide Web evolves into the 21st century, usage models of the web are changing. These changes will, of necessity, carry over into the way web authentication works. In this note, we briefly examine some of the issues that developers of "mashups" will face.

Third-party Delegation

A rapidly growing number of web services[1] are being built by a wide variety of developers. These services typically offer an extremely low cost-of-entry for third party developers to bring exciting new applications to WWW users. As web services become more sophisticated, we expect them to integrate data from multiple sources, some of which will require authentication.

The obvious way to build such a web service is, unfortunately, indistinguishable from a phishing site: ask the user for their credentials for another website. For some legacy websites, where the information has to be obtained via screen-scraping, this will be the simplest solution, and may be the only viable solution, as the browser security model (the "same-origin" policy [1]) may prevent client-side solutions.

As we desire to do better for new applications, we need an authentication and authorization architecture that supports delegation. In fact, we desire support for pseudonymous delegation: third party applications should not require the user to disclose their primary credentials, but rather provide a pseudonym that can authenticate with the larger service and authorize access to a subset of a given user's data. By using a pseudonym, the user can protect their privacy by not allowing linkage across services. Additionally, the third party cannot be liable for anything that goes wrong with the primary account outside the scope of the delegation, as they simply don't have access to it.

One important detail that delegation always raises is revocation [2, 3]. How can a user revoke an existing delegation if either they no longer desire the third-party service, or the original delegation was erroneous, perhaps because the user suspects their account was compromised and the original delegation was made by an unknown party? To make this manageable for the user, delegations should be revalidated or revoked upon password changes to the primary account.

[1] We use web service to include any service that offers an API accessible via HTTP or a similar interface, reserving Web service for those that use WSDL, etc.

One of the great benefits of web services is that they support decoupled evolution of the underlying software. In order to get the full benefits of decoupled evolution, the authentication mechanism needs to use opaque authenticators. This prevents arbitrary third parties from relying on anything in the format of an authenticator. The origin of the authenticator may wish to provide a service to map the authenticator to various meaningful representations.

Conclusion

Authentication in a world of web services presents many new challenges. We have sketched a few requirements for web service authentication to allow web services to achieve their full potential.

Acknowledgement

The author wishes to thank Scott Renfro for fruitful discussions on this topic.

References

[1] Vinod Anupam and Alain Mayer. Security of web browser scripting languages: Vulnerabilities, attacks, and remedies. In *Proceedings of the 7th USENIX Security Symposium*, pages 187–200, San Antonio, TX, January 1998. USENIX. `http://www.usenix.org/publications/library/proceedings/sec98/full_papers/anupam/anupam.pdf`.

[2] Ajay Chander, Drew Dean, and John C. Mitchell. A state-transition model of trust management and access control. In *Proceedings of the 14th IEEE Foundations of Computer Security Workshop*, Cape Breton, Nova Scotia, 2001. `http://theory.stanford.edu/~jcm/papers/csfw01a.ps`.

[3] Ajay Chander, Drew Dean, and John C. Mitchell. Reconstructing trust management. *Journal of Computer Security*, 12(1):131–164, 2004. `http://theory.stanford.edu/~jcm/papers/chander-dean-jcs-04.pdf`.