Position Paper

Toward a More Secure Web 15/16 March 2006
W3C Workshop on Transparency and Usability of Web Authentication

Approaches to Simplify Server Authentication

Frederick Hirsch, Nokia
Hubert A. Le Van Gong, Sun

Abstract

Much work has been dedicated to authenticating clients to service providers, ranging from the use of client TLS in the Web to the development of Single Sign-On mechanisms to enable usable authentication in a circle of trust, as outlined in the Liberty Alliance ID-FF [ID-FF] and SAML Single Sign-On profiles [SAML2]. What remains hard is the effective authentication of Service Providers to clients, due in part to client device limitations, provisioning issues and user understanding and education. For these reasons PKI based solutions including the use of TLS server authentication, have not been particularly effective, even though such processing has been developed for years.

We suggest two approaches. One is creating and re-using a shared client secret unique to each server to enable the client to authenticate the server (an approach we did not invent). The second is what we call Simplified Server Authentication (SSA), an analog of SSO for server authentication that can reuse many of those mechanisms and infrastructure and which can be integrated with shared client secrets. This approach is typically appropriate within a community of trust, such as a group of enterprise servers that have established relationships with an identity provider.

Introduction

The need for server authentication is clear with the many risks associated with a client accessing a false site purporting to be a real site and sharing private information at that false site, be it passwords, financial information, or other private information leading to identity theft and other losses of privacy.

The traditional approach has been to use SSL/TLS [TLS] that includes a PKI based mechanism for establishing the identity of the server. This relies upon the server supporting SSL/TLS and being configured with a key pair and certificate suitable for TLS. If all goes well the client TLS software executes the TLS protocol, validates a signature as part of that protocol and establishes that the server is as expected.

There are a few potential flaws with this approach:

1. In addition to validating a signature, additional steps are required and not always performed – steps to establish trust in the entire mechanism. These include certificate path validation, leading to a trusted root authority and timely certificate status validation.

2. The browser needs to be configured with the correct root certificates and not with inappropriate certificates. Management of this configuration is beyond most users. A related concern is the use of self-signed certificates or sites that do not use certificates issued by a commonly accepted certificate issuer.

3. SSL/TLS checks that the domain name accessed matches information in the certificate, but this does not prevent attacks against DNS in conjunction with the use of inappropriate certificates.

4. Export regulations and configuration can cause unexpectedly weak cryptography to be used.

5. Finally the user has no idea how to confirm that the process is operating properly or what to do if a pop-up message occurs, other than to simply go on, regardless.

6. The user must trust the entire infrastructure, including the browser security implementation.

These concerns also exist if an alternative but similar approach is deployed, such as signing metadata with an XML Signature and then providing that signature with the client. Similar validation, trust and end-user issues remain, although use of services like XKMS [XKMS] can move some of the problems from the client.

There are two potential solutions that can be used in different circumstances.

In the case that a client establishes a long-standing relationship with relatively few sites, the user can manage this with minimal cooperation of each site. By sharing a secret with the site, and expecting the site to provide this information upon each access, a user can recognize a false site by the fact that it is unable to provide this information. This information may be protected and provided in a variety of means discussed below.

For a more scalable solution within a community, SSO mechanisms may be re-used to provide server authentication on behalf of the client. For example, when a client accesses a service provider the initial request can be redirected to an identity provider that can perform authentication of the service provider before redirecting the client back to the it. In this case it is reasonable to expect the identity provider to support a variety of mechanisms for server authentication and to be able to correctly perform PKI or other validation checks in conjunction with signature verification. This approach may be used to establish a shared client secret to simplify subsequent server authentication and access.

Shared Secret Approach

The use of shared secret requires a registration process at the server, where the user establishes an “account” and provides a private piece of information (essentially a password for the server to use to authenticate to the client). Many common web sites establish accounts for a variety of reasons – buying (e.g. amazon.com), travel (e.g. aa.com), online communities (to have a handle and maintain state) and so on, so this is an incremental change to the data maintained at the server.

This does require a standard means for the server to present the password, both in format and protocol. The password also needs to be protected on the wire, in storage, and care must be taken to prevent replay attacks. Thus one could expect an additional HTTP header on the request containing a nonce, with an additional response header containing a Hash of the nonce, password hash, and server domain name for example.

The browser must also be modified to generate the nonce and HTTP header on the request, and to recognize and process the response value. It is also necessary for the browser user to maintain a list of sites where such server authentication is required. Modern browsers offer the possibility to develop relatively sophisticated plugins that could provide such functionality.
Although the simplified server authentication approach we described below offers more flexibility and a smaller *impact* on the browser we believe the shared secret approach is relevant when 3rd party authentication is undesirable (or not possible).

Simplified Server Authentication (SSA)

Much work has been done on simplifying and making client authentication to servers usable, especially when a client needs to access multiple service providers that share a business relationship. Standards from the Liberty Alliance on the Identity Federation Framework (ID-FF) have addressed this problem, taking special care to address privacy requirements based on laws, regulations and best practices. This work has also been standardized and reflected in the latest OASIS SAML 2.0 single sign-on profile.

A single sign-on approach like this makes use of the ability of Web servers to redirect browser requests, enabling a service provider to redirect a client request to an infrastructure component, an Identity Provider, obtain authentication information for the user based on a previous authentication act, or to require a new authentication interaction with the client. Once all the necessary steps have been taken by the identity provider (IDP), the client is redirected back to the service provider and able to take advantage of the service.

This approach allows authentication mechanisms to be implemented by an IDP, simplifying service provider implementation, and most importantly avoids the need for the client to establish passwords at multiple sites and to authenticate repeatedly to multiple sites. This increases usability and reduces the risks associated with password re-use and repeated authentication.

This concept, and in fact the infrastructure, may be also applied to server authentication to the client. In this case when a client accesses a service provider it may require that the service provider authenticate to a trusted identity provider (not to the client) and that the response for the request come from the identity provider, containing an authentication assertion for that provider. This may be an XML Signature signed XML quantity for example. In this case key management is simplified since there are many fewer trusted identity providers. Once the authentication artifact is verified, the client can reconnect to the IDP to be redirected to the correct service provider. Alternatively, the authentication process could have provided the SP with a shared secret that it can now present to the client as mentioned in the shared secret approach, allowing a subsequent direct

connection to the SP. In this case the client may rely upon the IDP and use a trusted SSL/TLS channel to obtain a server secret from the IDP to be used in subsequent authentication. (In this case more care may be used in establishing and securing the use of SSL/TLS).
If the Identity Provider and the service provider had already been federated (around the customer's identity) then the same operation could lead to mutual (and seamless) authentication between the service provider and the customer: the end-result of the aforementioned sequence is that the user is directly logged in to his account at the service provider.

This SSA approach will require the definition of some HTTP headers and other mechanisms similar to SSO approaches, and can re-use assertions such as SAML 2.0 assertions, but much of the work may already have been done. We have not implemented this, but are speculating that this is the case.

The Liberty Alliance and OASIS SSTC have also defined the concept of an "Enhanced Client or Proxy", a component that is either integrated into a client (e.g. the browser) or that exists as a proxy. This component knows how to reach an IDP on behalf of a client. A similar component could be defined that knows how to reach an IDP on behalf of a Service Provider, and integrated as a server component or as part of a proxy. More thought on this is required, but we suggest that many of the mechanisms used for SSO can be recast to approach the server problem.

We find the combination of a mechanism to establish and share a server password and then subsequently use it particularly promising.

References

[ID-FF] Liberty Alliance Federation Framework ID-FF 1.1
https://www.projectliberty.org/resources/specifications.php#box1

[SAML2] Security Assertions Markup Language, OASIS
http://www.oasis-open.org/apps/org/workgroup/security/#samlv20

[SimplifiedClientAuth] see http://weblog.infoworld.com/udell/2005/05/31.html#a1241

[TLS], RFC 2246, The TLS Protocol,
http://www.ietf.org/rfc/rfc2246.txt

[XKMS] XML Key Management, W3C
http://www.w3.org/2001/XKMS/