Shibboleth & Portals

David L. Wasley, et. al.

May 12, 2002

Shibboleth[1] provides a powerful and flexible mechanism for trusted parties to exchange information about a potential user of a web-based on-line resource. A "portal" is a web-based service but also, in some cases, must act as a proxy for the user in gaining access to information resources. Shibboleth clearly provides a solution for identifying the user upon initial access to the portal. This note suggests a way in which Shibboleth also can support the required proxy functionality.

In the basic Shibboleth scenario (Figure 1), the user's initial request to the target resource is acted upon by a SHIRE. If the SHIRE has no previous knowledge of the user, it asks the WAYF to redirect the user to that user's local domain HS and specifies where the HS should return the user after authentication is complete. If the WAYF doesn't know about the user, it will have to ask "Where are you from?" in order to look up the HS. If the HS doesn't yet know who the user is, it will need to interact with the user before generating a handle to return to the SHIRE.

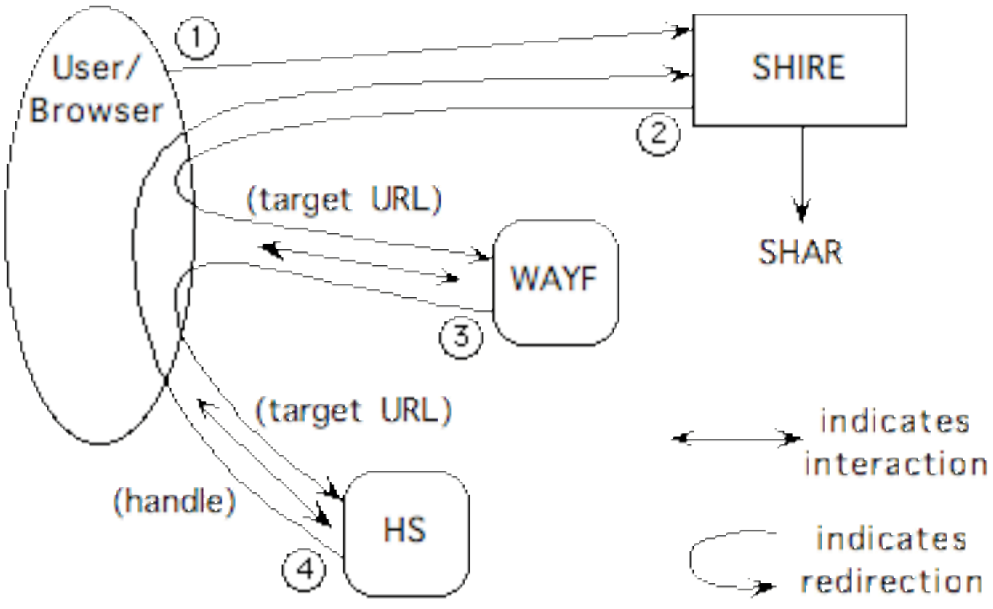

Figure 1: Basic Shibboleth Phase 1 Flows

[1] See `http://middleware.internet2.edu/shibboleth/`

Portal-First Scenarios

It is anticipated that a common scenario might be for the user's origin site to run a web server "starting point" that lists resources available to the user. Where the target resource does not need access management or is not Shibbolized, the page merely takes the user to the resource as is done now. If, however, the target resource is known to be in a Shibboleth Club with the origin site, the starting point page can take the user directly to the user's HS and specify where the HS should take the user after authentication is complete. This bypasses the WAYF step as a convenience for the user. This scenario is illustrated in Figure 2.

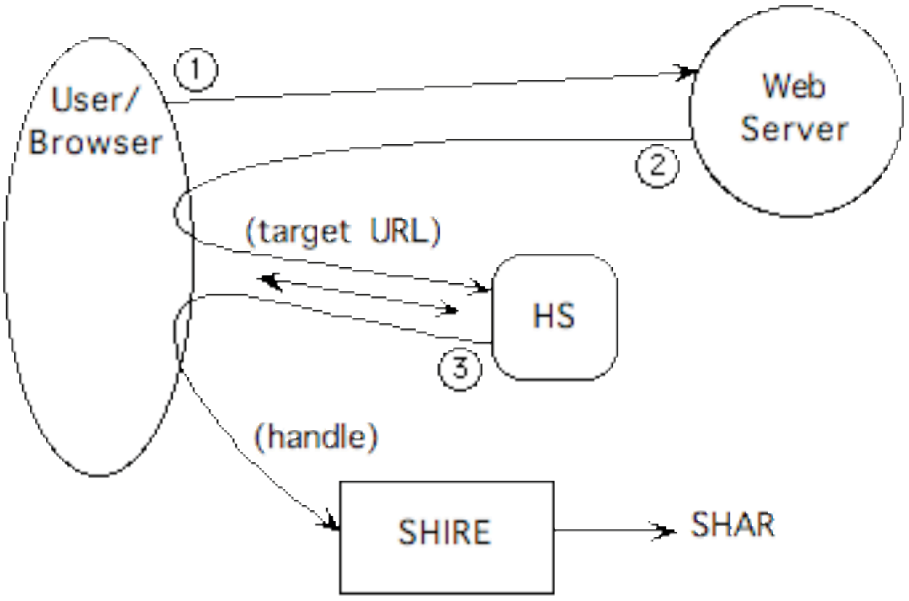

Figure 2: "Web Server First" Shibboleth Phase 1 Flows

The scenario in which a portal merely directs a user to other resources is identical to the above. However, if the portal wants to intercept the information stream coming back from the resource, e.g. in order to render it in a sub-window ("channel") or for a specific type of user platform, then the portal must convince the resource that it is acting on behalf of the real user. This is often referred to as acting as a proxy for the user.

If the target resource is not Shibbolized, then some form of credential or shared secret caching might be employed but this presents a security vulnerability and must be implemented very carefully. If the target is PKI aware, then it might be modified to accept a credential identifying the Portal as a trusted source, and then accept an assertion of the user's identity to enable the portal to gain access to the user's information. This might work readily if the target shares the same security domain as the portal but becomes more problematic otherwise. Both of these scenarios involve

giving the resource specific identity of the user which might be neither required nor appropriate, a problem which Shibboleth solves.

If the portal is in the user's origin security domain and Shibboleth aware, and the target is in the same Shibboleth Club as the origin site, then the portal could (in theory) contact the origin site HS directly, establish itself as a trusted agent, and provide an assertion of the user's identity to the HS along with the URL for the target resource. The HS would act exactly as in the basic scenario except that it would accept the assertion of the user's identity from the trusted portal instead of interacting with the user. This is illustrated in Figure 3. The result would be that the resource returns information to the portal instead of directly to the user's browser since the handle message is received by the SHIRE from the portal.

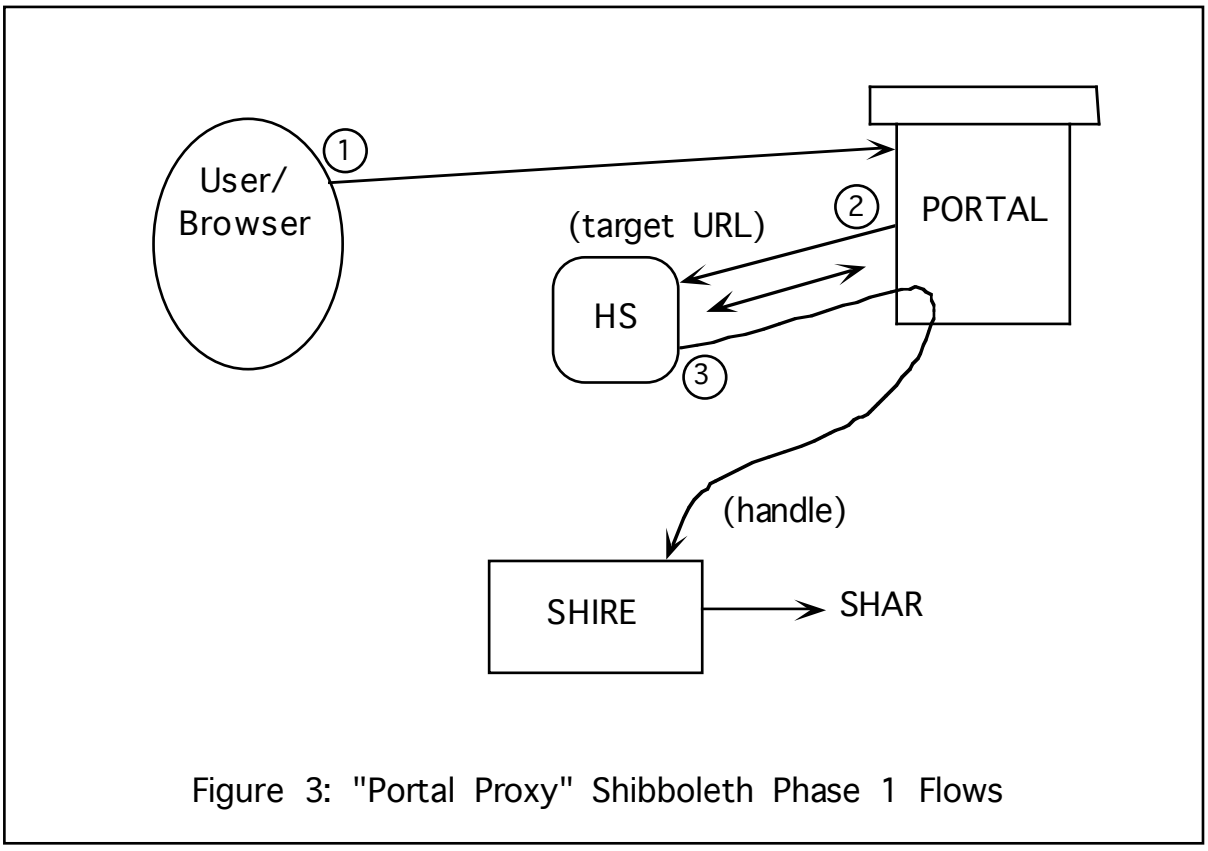

Figure 3: "Portal Proxy" Shibboleth Phase 1 Flows

It would make sense for the HS to establish an SSL/TLS connection to the portal so that it can receive the portal's credentials and assertions safely. If the origin site is PKI enabled, the SSL/TLS certificate from the portal also could establish the portal's identity.

The new Shibboleth subcomponent needed to make this work would be the portal's assertion to the HS of the user's identity, either by asserting a specific identity or by offering the handle that the portal received when the user connected to it (see below for further discussion of this strategy). There are a number of different scenarios and trust models to consider and it might be necessary to develop a configurable HS in order to handle all of them appropriately.

If the portal is not in the user's security domain, it may not know the URL of the user's HS nor be trusted by that HS. Furthermore, the user might be a member of several security domains and the portal would not know *a priori* which one(s) to contact. These are primarily information management issues and might be solved in a number of ways. However, for simplicity at this stage of Shibboleth support for portals we will not address the "remote portal rendering third party information" problem.

		Rendering Portal with respect to User's Security Domain	
		Same Domain	Different Domain
Information source with respect to Portal	Same Security Domain	Trust can be managed easily and common credentials, e.g. SSO, can be understood by resources.	User may have to manage trust with the portal, e.g. under a Shibboleth scenario. Resources can trust the portal and might share a common SSO.
	Different Security Domain	Trust can be managed easily but application specific credentials must be available to the portal. Shibboleth can mitigate risk and make access transparent to user.	User may have to manage trust with the portal, e.g. under a Shibboleth scenario. Trust relationships between the User's HS and the portal, and between the portal and resources are complex. Shibboleth might help.

One way of parsing the Rendering Portal Problem

Portals and Trust

When a portal is asked to render personal information on behalf of a user, there is at least an implicit trust relationship between the user and the portal: the user must assume that the portal will not inappropriately cache that information nor pass it on to other parties. By the same token, the user is implicitly delegating authority to the portal to gain access to that information from the server or resource where it resides. The embodiment of this trust in the portal software involves thoughtful design and clear notification to users of its behavior. In addition, the portal must be designed to mitigate risks of a security compromise to the greatest extent possible within the constraints of each access scenario.

A basic trust issue is how much the portal needs to "know" about the user, either for personalization or for third party information access. Clearly the user must have identified him/herself to the portal sufficiently to allow the portal to "personalize" correctly the page it presented, but the portal should not ask for more than that. If personalization was based on group identity, e.g. "student," then that identity alone might not be sufficient to enable access to resources that require specific identity. Therefore, portals providing specific user information either will have determined specific user identity upon initial contact or must have a way to provide it indirectly to the information source (or both).

The following table summarizes some of the information access scenarios.

Information Source	Access Method	Notes
Thin Client or terminal based	UserID & password	The portal must have access to the user's application specific UserID and password. This can be a significant security risk. Trust is needed only between the user and the portal.
	Initial Sign-on based	The portal must be trusted to assert the ISO handle to the resource server. There is no constraint on which server(s) except through the trust mechanism.
Client/Server	UserID & password	See above. May also require protocol support, e.g. CHAP, APOP, etc.
	Initial Sign-on based	See above.
	Kerberos	??
	PKI	The portal could offer the user's PKI cert to the resource but there would be no way for the resource to verify the owner. This would require even more trust than the ISO model since the cert has a longer validity period.
	Shibboleth	As noted in Figure 3, the portal could assert an identity to the HS which would result in a new handle being created and sent to the resource. The portal would never see the user's attributes given to the resource. The HS must trust the portal. (See also ISO above.)

How the portal might convince the holder of personal information to release it will depend on the options available for each specific source. Only the most modern information sources will have the capability of minimizing risks to the user. An important design goal is to define the circumstances under which the portal can make use of a user's delegation of authority, and to enable the user to constrain that delegation to an appropriate set of actions.

In practice, there will be many 'legacy' resources for which the user simply will have to place a great deal of trust in the behavior of the portal. For example, to gain access to a thin client-based resource that only understands a "userID and password in clear ASCII text" will require that the user give the portal his/her userID and password. This allows the portal to "pretend to be the user" and creates the risk that it could do so at any time or that a third party might intercept the user's credentials.

Newer applications might have a more complex API and use secure transmission technology but the trust issue remains: how does the portal convince the information

source to release personal information belonging to the user? Any scenario will require a certain degree of trust between the user, the portal, and the resource. A design goal should be to minimize the required degree of such trust so as to minimize risks.

A Proposed Shibboleth Trust Model for 'Rendering Portal' Applications

The lowest risk scenario might be for the user to authenticate separately to each resource and then direct them to send encrypted information streams to the rendering portal. With current technology, this may be impractical. Next best might be a scenario wherein the portal is trusted by an authorization server so the portal can ask it to arrange for access to a resource on behalf of a user whom the portal only knows referentially. If the authorization server can do so without ever giving authorization data to the portal, then there is no increased risk of misuse of that information.

Referential identity is an essential aspect of how Shibboleth works. In a fully Shibbolized scenario, the user would use Shibboleth to become authorized to the portal. Then, when seeking access to a third party resource, the portal would simply send back to the HS the handle created for the portal when the user first connected to it. The HS, like the AA, can map between the handle and the user's directory information.

The HS would verify the portal's identity, using it's PKI certificate, and then create another handle for the same original user but identifying the new target as the handle consumer. The third party resource (the new target) then retrieves authorization data directly from the AA, bypassing the portal.

For even greater security, the HS might insist that the validity period of the original handle not have expired in order to prevent the portal (possibly via an intruder) from reusing an old handle. However, portal sessions might be very long lived so provision would need to be made for creating special long lived handles for the portal, or for obtaining a new handle when the original expires.

If the portal is not Shibbolized but the third party resource is, the HS would have to accept some other assertion of the user's identity. This is less secure than the fully Shibbolized scenario since the identity token would not likely be opaque to the portal and would not likely have a validity period.

Conclusion

If this type of support for 'rendering portals' makes sense, then it should be added to the next phase of prototype code development. And of course we'll need a name for the new component. I suggest:

SHIP – Shibboleth Identity Proxy