SPReverseProxy

Deploy the Service Provider behind a Reverse Web Proxy

A reverse proxy (called "proxy" below) is installed in front of a web server (called "resource" below), only the latter is hosting the resource and is running the Shibboleth Service Provider software. All traffic to that web server goes through the reverse proxy – there should be no way to access the web server directly (i.e., you must use packet filters, firewalls, web server configuration, etc. to prevent access from anywhere but the proxy). The proxy could also be used for SSL offloading, handling all (HTTP and) HTTPS traffic, speaking only plain HTTP to the web server – if you wanted to rely solely on a trusted network. (In case your proxy itself is Apache httpd you can also enable HTTPS while proxying to the webserver, see the mod_ssl documentation).

Here is a description of the SSO flow (leaving out IdP Discovery for brevity):

1. The client attempts to access https://proxy.example.org/secure
2. The reverse proxy at proxy.example.org internally forwards the request to http://resource.example.org/secure
3. The location /secure on the resource is protected by a Shibboleth SP
4. The Shibboleth SP intercepts the request and generates a SAML2 AuthnRequest with an AssertionConsumerServiceURL of https://proxy.example.org/Shibboleth.sso/SAML2/POST (assuming default locations and a properly configured web environment on proxy.example.org. If proxy.example.org's web server configuration is not correct, a variety of wrong URL’s may be generated here.)
5. Also the relayState for the requested URL is set (e.g in a HTTP cookie).
   - Note that the path (/secure) to the requested resource is set by the Shibboleth SP and hence is specific to the protected resource on the web server. This mandates that the proxy either proxies the resource with the exact same path (/secure to /secure), or that the proxy is able to rewrite HTTP response headers (e.g. the ones containing the relayState) before returning results to the client.
6. The client authenticates at an IdP and bounces back to https://proxy.example.org/Shibboleth.sso/SAML/POST with an authentication (and probably also an attribute) assertion.
7. The resource gets the request forwarded from proxy.example.org
8. If there's no attribute assertion the Shibboleth SP at the resource may also query the IdP for attributes (Note that queries from the Shibboleth SP will not go through the proxy described in this document). The SP redirects to the resource specified in the relayState, applies any authorization logic and returns the page (to the proxy, and the proxy to the client).

Reverse Proxy

Apache httpd with mod_proxy

Building a basic reverse proxy with the Apache httpd web server:

| ProxyPass | /Shibboleth.sso/ http://resource.example.org/Shibboleth.sso/ |
| ProxyPassReverse | /Shibboleth.sso/ http://resource.example.org/Shibboleth.sso/ |
| ProxyPass | /secure/ http://resource.example.org/secure/ |
| ProxyPassReverse | /secure/ http://resource.example.org/secure/ |

Lighttpd with lighttpd_mod_proxy

Building a basic reverse proxy with the lighttpd web server:

```
server.modules  += ( "mod_proxy" )
$HTTP["url"] =~ "^/secure/" {  
    proxy.server = ( "" => (( "host" => "resource.example.org", "port" => 80 )))  
}
```

Note: Proxying the Shibboleth handlerURL is not part of this example, but will still need to be done when following the general direction of this document.

Resource

Apache httpd 2.2

On the web server with the Shibboleth SP set the ServerName directive to the scheme, host name and port of the proxy, cf. httpd documentation:

```
ServerName https://proxy.example.org:443
UseCanonicalName On
```

(In case of SSL offloading to the proxy, the resource's web server will only have a plain HTTP vhost configured – since any HTTPS traffic will be terminated at the proxy – but the ServerName directive will still need to be set as specified above.)
**shibboleth2.xml**

With SSL offloaded to the proxy, also set `handlerSSL="false"` in `shibboleth2.xml`, so the Shibboleth handler will accept protocol messages on plain HTTP.

### Metadata

Any protocol endpoints in the Metadata describing the SP must point to `proxy.example.org`. Unless the proxy itself does not handle HTTPS at all (i.e., access to the resource is not protected by TLS/SSL), all endpoints in the metadata should be set to HTTPS URLs. If the process by which you generate metadata does not do this for you, you'll need to perform this change yourself.

```xml
<AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    Location="https://proxy.example.org/Shibboleth.sso/SAML2/POST" index="1"/>
<AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign"
    Location="https://proxy.example.org/Shibboleth.sso/SAML2/POST-SimpleSign" index="2"/>
<AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
    Location="https://proxy.example.org/Shibboleth.sso/SAML2/Artifact" index="3"/>
<AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:PAOS"
    Location="https://proxy.example.org/Shibboleth.sso/SAML2/ECP" index="4"/>
<AssertionConsumerService Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post"
    Location="https://proxy.example.org/Shibboleth.sso/SAML/POST" index="5"/>
<AssertionConsumerService Binding="urn:oasis:names:tc:SAML:1.0:profiles:artifact-01"
    Location="https://proxy.example.org/Shibboleth.sso/SAML/Artifact" index="6"/>
```