SLOWebappAdaptation

Web application Adaptation for SLO

This page provides instructions on how web applications can be adapted in order to support a clean (Single) Log Out with Shibboleth. Please note that there are many issues involved with logout as summarized here but in some use cases logout nevertheless could make sense to be implemented.

General Logout Approaches

There are two cases of web application when it comes to support logout:

Web application without session management

Applications that rely on Shibboleth's session management and access control, won't need any adaptation to support logout. This is because the Shibboleth Service Provider by itself already supports logout.
These applications often are completely protected by Shibboleth either by a required or lazy session. In both cases Shibboleth will handle session management and thus also logout.

Web application with own session management

In this category fall a lot if not most of all web application with some sort of internal user management including most e-learning applications. These applications use Shibboleth mostly only to authenticate users and from then on use their own session management. One advantage of this is that the application is independent from Shibboleth and can control the session itself. However, this also can be problematic in cases where the web application's session timeout is longer than the one that the Service Provider has set. And it also requires some adaptations when it comes to support logout.

What such applications need to be able to log out users that were authenticated via Shibboleth is to have a mapping that maps a Shibboleth session ID to an internally used session. Only then the Service Provider can notify this application to delete the session of a Shibboleth user via an HTTP/HTTPS or SOAP request.

The Service Provider can notify an application in two different ways. Either by a front-channel (via the user's web browser) HTTP/HTTPS request or a back-channel (direct SP - web application connection) SOAP request. While the front-channel logout request usually is easier to implement. This is because the user's web browser will send cookies to the logout script of the application that then can easily identify and destroy the user's session. However, front-channel logout will - if at all - only be supported in in later versions of Shibboleth. The better but also more complex approach to a clean SLO is to use the back-channel notification. In this case the logout script receives a SOAP request bearing a Shibboleth session ID of the user whose session shall be destroyed. Thus, the web application needs to be able to identify a user's session based on the Shibboleth session ID. Therefore, the above mapping is required.

Since the web application cannot delete the user's session by deleting its cookie (remember that in back-channel notification the user's browser is not involved), the application only can delete the server-side session data from the filesystem or in the database.

Therefore, what is needed to implement the recommended back-channel notification in your web application is:

1. The code that implements user login via Shibboleth needs also to store a user's **Shib-Session-ID** , which should be available in the web server environment of a protected page.
2. The code that implements logout via Shibboleth must then be able to handle a SOAP request like the following:

   ```
   <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
     <s:Body>
       <LogoutNotification xmlns="urn:mace:shibboleth:2.0:sp:notify" type="global">
         <SessionID>
           _d5628602323819f716fcee04103ad5ef
         </SessionID>
       </LogoutNotification>
     </s:Body>
   </s:Envelope>
   ```

   If the user's session could be successfully deleted, the response should be:

   ```
   <soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:notify="urn:mace:
   shibboleth:2.0:sp:notify">
     <soap-env:Body>
       <soap-env:LogoutNotificationResponse>
         <notify:OK/>
       </soap-env:LogoutNotificationResponse>
     </soap-env:Body>
   </soap-env:Envelope>
   ```

3. If the above requirements can be met, the one last thing that remains to do is to configure the logout script in a Service Provider's Notify element as explained on the Notify page.

A code example for PHP that fulfills the above requirements is given below.

Sample PHP 5 logout code

The following code can be used as a sample to adapt an application for front and back-channel logout. This requires, that the Shibboleth name identifier that was used during login is stored in the user's session within the application. Only then is it possible that this code can be called using the name identifier as input so that the code can then delete the application session. If back channel is used, the script should not live under a location that is protected by the SP, as it will not be executed in the web browser context but called by `shibd` directly (SOAP call using `libcurl`).

```
<?
// Sample PHP 5 Shibboleth logout code by lukas.haemmerle@switch.change_user
// History:
// - December 8 2008:    Uploaded initial version that also was used for Moodle

// Just for debugging the WSDL part
ini_set("soap.wsdl_cache_enabled", "0"); // disabling WSDL cache

/************************************************/
/* Sample code to log out user from application */
/************************************************/

// Requirements:
// PHP 5 with SOAP support (should be available in default deployment)


//////////////////////////
// Front channel logout //
//////////////////////////

// Note: Generally the back-channel logout should be used once the Shibboleth
//       Identity Provider supports Single Log Out!
//       Front-channel logout is not of much use.

if (
                isset($_GET['return'])
                && isset($_GET['action'])
                && $_GET['action'] == 'logout'
   ){

        // Logout out user from application
        // E.g. destroy application session/cookie etc

        /* Example Code: Start */

        // Unset all of the session variables.
        $_SESSION = array();

        // Destroy cookie
        if (isset($_COOKIE[session_name()])) {
                setcookie(session_name(), '', time()-42000, '/');
        }

        session_destroy();

        /* Example Code: End */

        // Finally, send user to the return URL
        header('Location: '.$_GET['return']);
        exit;
}

/////////////////////////
// Back channel logout //
/////////////////////////

// Note: This is the preferred logout channel because it also allows
//       administrative logout. However, it requires your application to be
//       adapted in the sense that the user's Shibboleth session ID must be
//       stored in the application's session data.
//       See function LogoutNotification below

elseif (!empty($HTTP_RAW_POST_DATA)) {
        // Set SOAP header
        $server = new SoapServer('https://'.$_SERVER['HTTP_HOST'].$_SERVER['PHP_SELF'].'/LogoutNotification.
wsdl');
        $server->addFunction("LogoutNotification");
        $server->handle();
}

/////////////////
// Return WSDL //
/////////////////

// Note: This is needed for the PHP SoapServer class.
//       Since I'm not a web service guru it might be that the code below is not
//       absolutely correct but at least it seems to to its job properly when it
//       comes to Shibboleth logout

else {

        header('Content-Type: text/xml');

        echo <<<WSDL
<?xml version ="1.0" encoding ="UTF-8" ?>
<definitions name="LogoutNotification"
  targetNamespace="urn:mace:shibboleth:2.0:sp:notify"
  xmlns:notify="urn:mace:shibboleth:2.0:sp:notify"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

        <types>
           <schema targetNamespace="urn:mace:shibboleth:2.0:sp:notify"
                   xmlns="http://www.w3.org/2000/10/XMLSchema"
                   xmlns:notify="urn:mace:shibboleth:2.0:sp:notify">

                        <simpleType name="string">
                                <restriction base="string">
                                        <minLength value="1"/>
                                </restriction>
                        </simpleType>

                        <element name="OK" type="notify:OKType"/>
                        <complexType name="OKType">
                                <sequence/>
                        </complexType>

                </schema>
        </types>

        <message name="getLogoutNotificationRequest">
                <part name="SessionID" type="notify:string" />
        </message>

        <message name="getLogoutNotificationResponse" >
                <part name="OK"/>
        </message>

        <portType name="LogoutNotificationPortType">
                <operation name="LogoutNotification">
                        <input message="getLogoutNotificationRequest"/>
                        <output message="getLogoutNotificationResponse"/>
                </operation>
        </portType>

        <binding name="LogoutNotificationBinding" type="notify:LogoutNotificationPortType">
                <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
                <operation name="LogoutNotification">
                        <soap:operation soapAction="urn:xmethods-logout-notification#LogoutNotification"/>
                </operation>
        </binding>
        <service name="LogoutNotificationService">
                  <port name="LogoutNotificationPort" binding="notify:LogoutNotificationBinding">
                        <soap:address location="https://{$_SERVER['HTTP_HOST']}{$_SERVER['PHP_SELF']}"/>
                  </port>
        </service>
</definitions>
WSDL;
        exit;

}

/******************************************************************************/
/// This function does the actual logout
function LogoutNotification($SessionID){

        // Delete session of user using $SessionID to locate the user's session file
        // on the file system or in the database
        // Then delete this entry or record to clear the session
        // However, for that to work it is essential that the user's Shibboleth
        // SessionID is stored in the user session data!

        $dbsession = true;
        if ($dbsession){
                // INSERT AND ADAPT CODE HERE
                /*
                if ($user_session_data = #YOUR-DB-QUERY-FUNCTION#('SELECT sesskey, sessdata FROM '.#YOUR-
SESSION-TABLE#.' WHERE expiry > NOW()')) {
                        foreach ($user_session_data as $session_data) {

                                $user_session = unserialize($session_data->sessdata);

                                // We assume that the user's Shibboleth session ID is stored in $user_session
['SESSION']->shibboleth_session_id
                                if (isset($user_session['SESSION']) && isset($user_session['SESSION']-
>shibboleth_session_id)){
                                        // If there is a match, delete entry
                                        if ($user_session['SESSION']->shibboleth_session_id == $SessionID){
                                                // Delete this session entry
                                                if (#YOUR-SESSION-DESTRUCTION-FUNCTION#($session_data->sesskey)
!== true){
                                                        return new SoapFault('LogoutError', 'Could not delete
session entry in database.');
                                                }
                                        }
                                }
                        }
                }
                */
        } else {
                $dir = session_save_path();
                if (is_dir($dir)) {
                        if ($dh = opendir($dir)) {
                                while (($file = readdir($dh)) !== false) {
                                        if (is_file($dir.'/'.$file)){
                                                $session_key = ereg_replace('sess_', '', $file);

                                                $data = file($dir.'/'.$file);

                                                if (isset($data[0])){
                                                        $session = unserializesession ($data[0]);
                                                        // If there is a match, delete file
                                                        if (isset($session['user']['sessionID']) && $session
['user']['sessionID'] == $SessionID){
                                                                // Delete this file
                                                                if (!unlink($dir.'/'.$file)){
                                                                        return new SoapFault('LogoutError',
'Could not delete session file.');
                                                                }
                                                        }
                                                }
                                        }
                                }
                                closedir($dh);
                        } else {
                                return new SoapFault('LogoutError', 'Couldn\'t read directory content of
'.$dir);
                        }
                } else {
                        return new SoapFault('LogoutError', 'There is no directory '.$dir);
                }
        }
}

/*******************************************************************************/
// Deserializes session data and returns it in a hash array of arrays
function unserializesession( $serialized_string ){
        $variables = array( );
        $a = preg_split( "/(\w+)\|/", $serialized_string, -1, PREG_SPLIT_NO_EMPTY | PREG_SPLIT_DELIM_CAPTURE );
        for( $i = 0; $i < count( $a ); $i = $i+2 ) {
                $variables[$a[$i]] = unserialize( $a[$i+1] );
        }
        return( $variables );
}

?>
```

With code like the above one then has to configure the Service Provider in order to notify the adapted web application via a Notify option like the following:

```
<Notify
        Channel="back"
        Location="https://#YOUR_HOSTNAME#/#PATH_TO#/logout.php" />

<!--
If possible, you should use only the back channel logout once it is working.
-->
<!--
<Notify
        Channel="front"
        Location="https://#YOUR_HOSTNAME#/#PATH_TO#/logout.php" />
-->
```

As pointed out above, one should only use the back channel notification because the Service Provider will support (at least at first) only the back channel communication which would make it impossible to use front-channel notification in cases where logout is initiated on another Service Provider than the one where the Notify element is configured.