Using Apache mod_plsql for SSO via NTLM in Oracle Application Express (Proof-of-Concept)

Table of contents

1 Introduction

mod_plsql is an Apache module that is used by Oracle Application Express to interact with the Oracle Database. This module also can be used to read the authorization token of the client (browser).

1.1 Links for additional information

- Blog Entry 'NTLM HTTP Authentication and Application Express' of Jason Straub:
  http://jastraub.blogspot.com/2008/03/ntlm-http-authentication-and.html

- Oracle White Paper:
  http://ebookbrowse.com/apex-ntlm-authentication-wp-pdf-d108853936

- Several Information found on Oracle Discussion Forum 'Application Express':
  https://forums.oracle.com/forums/forum.jspa?forumID=137

1.2 Further information

greenIT Est.
www.greenit.li | info@greenit.li

ORACLE PARTNER

2 Installation

2.1 Stop the Oracle HTTP Server (Apache Web server)

2.2 Extend the Directive in dads.conf for APEX

File: <OHS home>\ohs\conf\mod_plsql\dads.conf
Extend the Location /pls/apex:

```
PlsqlCGIEnvironmentList AUTHORIZATION
```

2.3 Restart the Oracle HTTP Server

Look at the log files:

<OHS home>\instances\instance1\diagnostics\logs\OHS\ohs1\

access_log.*

console~OHS~1.log

ohs.log

3 Test SSO with APEX

We will now need to test to see that our user details have been set correctly.

3.1 Notes on Browser

3.1.1 Configure Firefox Browser

Firefox will work with NTLM, but each browser has to be configured to trust each server where you want to employ NTLM.
To configure Firefox to negotiate NTLM with a specific server:

1. Type **about:config** in the address bar
2. Type **ntlm** in the filter text box
3. Double click the preference **network.automatic-ntlm-auth.trusted-uri's** and enter a comma separated list of trusted servers on your network

3.2 NTLM Authentication via PL/SQL in an APEX Application

3.2.1 Create an Authentication Function in the Database

```
CREATE OR REPLACE FUNCTION pageSentryNtlmPlsql
    RETURN BOOLEAN
IS
  vAuthorization     VARCHAR2(512);
  vCharset           VARCHAR2(128);
  vCurrentSessionId  NUMBER;
  vDecode            VARCHAR2(2000);
  vDomain            VARCHAR2(128);
  vLengthDomain      PLS_INTEGER;
  vLengthUsername    PLS_INTEGER;
  vHtpBuffer         htp.htbuf_arr;
  vHtpRows           INTEGER;
  vOffsetDomain      PLS_INTEGER;
  vOffsetUsername    PLS_INTEGER;
  vRaw               RAW(1000);
  vRemoteHost        VARCHAR2(512);
  vRemoteUsername    VARCHAR2(512);
  vUrl               VARCHAR2(500);
  vUsername          VARCHAR2(128);
BEGIN
  -- Check to ensure that we are running as the correct database user.
  IF USER ^= 'APEX_PUBLIC_USER' THEN
    RETURN FALSE;
  END IF;
  -- Get SessionId of current user session based on cookie.
  vCurrentSessionId := wwv_flow_custom_auth_std.get_session_id_from_cookie;
  -- Check if session exists and is valid.
  IF wwv_flow_custom_auth_std.is_session_valid THEN
    apex_application.g_instance := vCurrentSessionId;
    vUsername := wwv_flow_custom_auth_std.get_username;
    wwv_flow_custom_auth.define_user_session(p_user => vUsername,
        p_session_id => vCurrentSessionId);
    RETURN TRUE;
  ELSE
    -- Get username using NTLM.
    vAuthorization := owa_util.get_cgi_env('AUTHORIZATION');
    IF vAuthorization IS NULL THEN
      -- The browser receives the 1st 401.
      owa_util.status_line(nstatus => 401, creason => 'Unauthorized',
        bclose_header => FALSE);
      -- The browser receives the hint to 'WWW-Authenticate: NTLM'.
      htp.p('WWW-Authenticate: NTLM');
      owa_util.mime_header('text/html', FALSE, 'utf-8');
      owa_util.http_header_close;
      wwv_flow.g_unrecoverable_error := TRUE;
      RETURN FALSE;
    END IF;
    IF SUBSTR(vAuthorization, 1, 5) = 'NTLM ' THEN
      vDecode := utl_encode.text_decode(buf => SUBSTR(vAuthorization, 6),
```

```
      encoding => UTL_ENCODE.BASE64);
    vRaw := utl_raw.cast_to_raw(vDecode);
    IF utl_raw.cast_to_binary_integer(utl_raw.substr(vRaw, 14, 1)) IN (0, 130) THEN
      IF utl_raw.cast_to_binary_integer(utl_raw.substr(vRaw, 9, 1)) = 1 THEN
        owa_util.mime_header('text/html', FALSE, 'utf-8');
        -- The browser receives the 2nd 401.
        --   The browser receives the hint 'WWW-Authenticate: NTLM' with a more concrete
        --   hint including a token.
        owa_util.status_line(nstatus => 401, creason => 'Unauthorized',
          bclose_header => FALSE);
        htp.p('WWW-Authenticate: NTLM TlRMTVNTUAACAAAAAAAAACgAAAABggAAU3J2Tm9uY2UAAAAAAAAAAA==');
        owa_util.http_header_close;
        wwv_flow.g_unrecoverable_error := TRUE;
        RETURN FALSE;
      END IF;
      -- Determine DB charset, convert raw to WE8MSWIN1252 and parse
      --   domain and username from authorization token.
      SELECT value INTO vCharset
        FROM nls_database_parameters
        WHERE parameter = 'NLS_CHARACTERSET';
      vLengthDomain := utl_raw.cast_to_binary_integer(utl_raw.substr(vRaw, 32, 1)) * 256+
        utl_raw.cast_to_binary_integer(utl_raw.substr(vRaw, 31, 1));
      vOffsetDomain := utl_raw.cast_to_binary_integer(utl_raw.substr(vRaw, 34, 1)) * 256+
        utl_raw.cast_to_binary_integer(utl_raw.substr(vRaw, 33, 1));
      vDomain := REPLACE(REPLACE(SUBSTR(CONVERT(utl_raw.cast_to_varchar2(vRaw),
        vCharset, 'WE8MSWIN1252'), vOffsetDomain + 1, vLengthDomain), CHR(0), NULL), CHR(15), NULL);
      vLengthUsername := utl_raw.cast_to_binary_integer(utl_raw.substr(vRaw, 40, 1)) * 256+
        utl_raw.cast_to_binary_integer(utl_raw.substr(vRaw, 39, 1));
      vOffsetDomain := utl_raw.cast_to_binary_integer(utl_raw.substr(vRaw, 42, 1)) * 256+
        utl_raw.cast_to_binary_integer(utl_raw.substr(vRaw, 41, 1));
      vUsername := UPPER(REPLACE(SUBSTR(CONVERT(utl_raw.cast_to_varchar2(vRaw),
        vCharset, 'WE8MSWIN1252'), vOffsetDomain, vLengthUsername), CHR(0), NULL));
    ELSE
      vUsername := 'NOBODY';
    END IF;
  END IF;
  -- Define a new APEX user session.
  wwv_flow_custom_auth.define_user_session(p_user => vUsername,
    p_session_id => wwv_flow_custom_auth.get_next_session_id);
  -- Tell APEX engine to quit.
  apex_application.g_unrecoverable_error := TRUE;
  IF owa_util.get_cgi_env('REQUEST_METHOD') = 'GET' THEN
    vUrl := 'f?' || wwv_flow_utilities.url_decode2(owa_util.get_cgi_env('QUERY_STRING'));
    wwv_flow_custom_auth.remember_deep_link(p_url => vUrl);
  ELSE
    vUrl := 'f?p=' || TO_CHAR(apex_application.g_flow_id) || ':' ||
      TO_CHAR(NVL(apex_application.g_flow_step_id, 0)) || ':' ||
      TO_CHAR(apex_application.g_instance);
    wwv_flow_custom_auth.remember_deep_link(p_url => vUrl);
  END IF;
  -- Register the Session in Apex Sessions Table, set Cookie and redirect back.
  wwv_flow_custom_auth_std.post_login(p_uname => vUsername,
    p_session_id => nv('APP_SESSION'), p_flow_page => apex_application.g_flow_id
    || ':' || NVL(apex_application.g_flow_step_id, 0), p_preserve_case => TRUE);
  -- Get HTP output (written by wwv_flow_custom_auth_std.post_login).
  --   It contains the session cookie we need.
  vHtpRows := 15;
  htp.get_page(thepage => vHtpBuffer, irows => vHtpRows);
  -- Reset the HTP buffer so that we can write our own header.
  htp.init;
  sys.owa_util.mime_header('text/html', FALSE);
  -- We have to trick IE that he thinks the authentication fails, otherwise
  --   he doesn't send any data when issueing a POST because he wants to
  --   do the NTLM stuff again.
  IF wwv_flow.get_browser_version ^= 'NSCP' THEN
    -- The browser receives the 3rd 401.
    owa_util.status_line(nstatus => 401, creason => 'Unauthorized',
      bclose_header => FALSE);
  END IF;
  -- Write the session cookie into our output.
  FOR ii IN 1 .. vHtpRows LOOP
    IF vHtpBuffer(ii) LIKE 'Set-Cookie:%' THEN
      htp.p(RTRIM(vHtpBuffer(ii), CHR(10)));
    END IF;
  END LOOP;
  vUrl := 'f?p=' || apex_application.g_flow_id || ':' ||
    NVL(apex_application.g_flow_step_id, 0) || ':' || apex_application.g_instance;
  IF wwv_flow.get_browser_version = 'NSCP' THEN
    -- Firefox: Redirect can be set with a HTTP header attribute.
    htp.p('Location: ' || vUrl);
    owa_util.http_header_close;
  ELSE
    -- IE: The javascript is required so that we are redirected to the page as
```

```
      --   the wwv_flow_custom_auth_std.post_login would normally do with the
      --   HTTP 302 redirect.
      owa_util.http_header_close;
      htp.p('<html><head>');
      htp.p('<script type="text/javascript">');
      htp.p('  location.href="' || vUrl || '";');
      htp.p('</script>');
      htp.p('<noscript>');
      htp.p('<meta http-equiv="refresh" content="0; URL=' || vUrl || '" />');
      htp.p('</noscript>');
      htp.p('</head>');
      htp.p('<body>');
      htp.p('You were logged in successfully. Click <a href="' || vUrl ||
        '">here</a> to continue.');
      htp.p('</body>');
      htp.p('</html>');
    END IF;
    RETURN FALSE;
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    RAISE;
END pageSentryNtlmPlsql;
/
```

3.2.2 Make an APEX Application using NTLM Authentication via PL/SQL

- Go to Shared Components → Authentication Schemes
- Create a new Authentication Scheme

  Scheme Name:

  `NTLM`

  Page Sentry Function:

  `RETURN pageSentryNtlmPlsql`

  Login Processing, Authentication Function:

  `-BUILTIN-`

  Logout URL, Logout URL:

  `wwv_flow_custom_auth_std.logout?p_this_flow=&APP_ID.&p_next_flow_page_sess=&APP_ID.:101`

  Make this scheme current

3.2.3 Run the Application

The application should start without needing to login.