Oracle Application Express Authentication Scheme Using NTLM Token Decoding

An Oracle White Paper
June 2008

Oracle Application Express and NTLM Authentication

TERMINOLOGY

NTLM	NT LAN Manager is a Microsoft authentication protocol
Hash	A fixed-size result obtained by applying a mathematical function called the hashing algorithm to an arbitrary amount of data.
Message Authentication Code or MAC	A cryptographic message authentication code. A short piece of information used to authenticate a message. A MAC algorithm accepts as input a secret key and an arbitrary-length message to be authenticated, and outputs a MAC (sometimes known as a tag). The MAC value protects both a message's integrity as well as its authenticity, by allowing verifiers (who also possess the secret key) to detect any changes to the message content.
Nonce	A randomly generated value used to defeat "replay" attacks.
Group Policy	There are certain settings that administrators can apply to groups of computers or users using the infrastructure provided by the Microsoft Active Directory. The administrator can manage these settings from a single central location. These include: Registry-based policy, security, software installation, scripts, remote installation services, internet explorer maintenance and folder redirection.
DCE/RPC	DCE/RPC stands for “Distributed Computing Environment/ Remote Procedural Calls”. This is a Remote Procedure Call system that allows software to work across multiple

computers, as if it were all working on the same computer. This system allows programmers to write distributed software without having to worry about the underlying network code.

DCE/RPC should not be confused with the Distributed Computing Environment (DCE) as a whole; DCE includes, in addition to DCE/RPC, a suite of DCE/RPC services that provide, amongst other things, Cell Directory Services (CDS) and the DCE Distributed File System (DFS). It should also not be confused with RPC as a whole, as RPC refers to a wide range of different and often incompatible technologies such as ONC RPC and XML-RPC.

INTRODUCTION

NTLM is a suite of authentication and session security protocols used in various Microsoft network protocol implementations and supported by the NTLM

Security Support Provider ("NTLMSSP"). Originally used for authentication and negotiation of secure DCE/RPC, NTLM is also used throughout Microsoft's systems as an integrated single sign-on mechanism. It is probably best recognized as part of the "Integrated Windows Authentication" stack for HTTP authentication; however, it is also used in Microsoft implementations of SMTP, POP3, IMAP (all part of Exchange), CIFS/SMB, Telnet, SIP, and possibly others.

The NTLM Security Support Provider provides authentication, integrity, and confidentiality services within the Window Security Support Provider Interface (SSPI) framework. SSPI specifies a core set of security functionality that is implemented by supporting providers; the NTLMSSP is such a provider. The SSPI specifies, and the NTLMSSP implements, the following core operations:

1. Authentication -- NTLM provides a challenge-response authentication mechanism, in which clients are able to prove their identities without sending a password to the server. Challenge-response authentication mechanism (CRAM) is the two-level scheme for authenticating network users that is used as part of the Web's Hypertext Transfer Protocol (HTTP). Using the CRAM, the server (or, alternatively, a proxy server or gateway) issues a challenge to a user in the form of a "401 unauthorized" request for a password which is a string of characters known only to the client and the server. When the server receives the client response, it checks to be sure the password is correct. If so, the user is authenticated. If not, or if for any other reason the network does not want to accept the password, a "403 forbidden" message is issued, and access to the site is denied. The CRAM can be used in addition to other security features, such as strong encryption.
2. Signing -- The NTLMSSP provides a means of applying a digital "signature" to a message. This ensures that the signed message has not been modified (either accidentally or intentionally) and that the signing party has knowledge of a shared secret. NTLM implements a symmetric signature scheme (Message Authentication Code, or MAC); That is, a valid signature can only be generated and verified by parties that possess the common shared key.
3. Sealing -- The NTLMSSP implements a symmetric-key encryption mechanism, which provides message confidentiality. In the case of NTLM, sealing also implies signing (a signed message is not necessarily sealed, but all sealed messages are signed).

NTLM has been largely supplanted by Kerberos as the authentication protocol of choice for domain-based scenarios. However, Kerberos is a trusted-third-party scheme, and cannot be used in situations where no trusted third party exists; for example, member servers (servers that are not part of a domain), local accounts, and authentication to resources in an untrusted domain. In such scenarios, NTLM continues to be the primary authentication mechanism.

OVERVIEW

NTLM credentials are based on data obtained during the interactive logon process and consist of a domain name, a user name, and a one-way hash of the user's password. NTLM uses an encrypted challenge/response protocol to authenticate a user without sending the user's password over the wire. Instead, the system requesting authentication must perform a calculation that proves it has access to the secured NTLM credentials.

NTLM Authentication Process

Interactive NTLM authentication over a network typically involves two systems: a client system, where the user is requesting authentication, and a domain controller, where information related to the user's password is kept. Non-interactive authentication, which may be required to permit an already logged-on user to access a resource such as a server application, typically involves three systems: a client, a server, and a domain controller that does the authentication calculations on behalf of the server.

NTLM non-interactive authentication has the following steps with the first step providing the user's NTLM credentials and occurring only as part of the interactive authentication (logon) process:

1. In Interactive authentication a user accesses a client computer and provides a domain name, user name, and password. The client computes a cryptographic hash of the password and discards the actual password.
2. The client then sends the user name to the server in plaintext.
3. The server generates a 16-byte random number, called a challenge or nonce, and sends it to the client.
4. The client encrypts this challenge with the hash of the user's password and returns the result to the server. This is called the response.
5. The server sends the following three items to the domain controller: User name, Challenge sent to the client, Response received from the client.
6. The domain controller uses the user name to retrieve the hash of the user's password from the Security Account Manager database. It uses this password hash to encrypt the challenge.
7. The domain controller compares the encrypted challenge it computed (in step 6) to the response computed by the client (in step 4). If they are identical, authentication is successful.

NTLM Authentication in Web Applications

Many customers use NTLM authentication for .NET intranet applications so that they do not have to supply their domain credentials again, and the application recognizes who they are. Many customers have deployed Apache and mod_ntlm, and used a custom authentication scheme. A common security model allows seamless integration and application interoperability. If you have multiple applications then you don’t need to provide login credentials for each application separately. For example single sign-on (SSO) is a method of access control that enables a user to authenticate once and gain access to the resources of multiple software systems. In an organization, which stores its user database in a Lightweight Directory Access Protocol (LDAP) database, all users would have a single set of authentication credentials. All information processing systems can use an LDAP Directory for user authentication and authorization so single sign-on can be achieved organization-wide.

Benefits of this include reducing the amount of internal fraud by malicious employee contact, convenience of password access, security on all levels of entry/exit/access to systems, and centralized reporting for compliance adherence.

So if you have a suite of Microsoft applications in .NET and are also using Oracle Application Express applications then you can make use of NTLM authentication for common authentication.

With Apache modules like mod_ntlm you should specify the allowed users exactly as the authentication module delivers them. If the module delivers MYDOMAIN\GBell then this is the syntax you need to enter after the Require directive. If your user name contains spaces, you need to enclose it in quotes: Require "MYDOMAIN\Graham Bell". Also do not suffix the NTLMServer and NTLMBackup directive values otherwise it will throw an internal server error.

This paper presents a pure PL/SQL code solution for decoding an NTLM token and using that decoded value as the authenticated user in APEX applications. The function will set the username to "nobody" if it detects that the browser prompted the user for their credentials instead of just silently negotiating them. You can then write authorization schemes that deny access to the "nobody" user. Note that unlike the mod_ntlm Apache module, this solution does not pass along credentials to a domain controller for authentication. This solution requests that the browser present an NTLM authentication token and decodes the username and domain from that token.

NTLM AUTHENTICATION IN APEX APPLICATIONS

Configuring the dads.conf

First you need to configure your DAD used for Application Express so that mod_plsql can be aware of a CGI environment variable called "Authorization." Make the following changes:

1. Search for the file that contains the DAD description used for Oracle Application Express i.e., $ORACLE_HTTPSERVER_HOME/Apache/modplsql/conf/dads.conf
2. Edit the DAD entry for Application Express adding: PlsqlCGIEnvironmentList AUTHORIZATION
3. Save the file
4. Stop and Start the Apache/ Oracle HTTP Server

Now you can access the CGI environment variable "Authorization."

Creating a Page Sentry Function

Next compile the following function that will be used as a page sentry function for a custom authentication scheme, in the same schema as your application.

```
create or replace function ntlm_page_sentry
return boolean
is
   l_username      varchar2(512);
   l_session_id    number;
   l_raw           raw(1000);
   l_domain        varchar2(128);
   l_user          varchar2(128);
   l_auth          varchar2(512);
   l_decode        varchar2(2000);
   l_off           pls_integer := 0;
   l_length        pls_integer;
   l_offset        pls_integer;
   l_htp_buffer    htp.htbuf_arr;
   l_htp_rows      INTEGER;
   l_url           VARCHAR2(500);
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.
   l_session_id := wwv_flow_custom_auth_std.get_session_id_from_cookie;
   -- check application session cookie.
   if wwv_flow_custom_auth_std.is_session_valid then
      apex_application.g_instance := l_session_id;
```

```
        l_username := wwv_flow_custom_auth_std.get_username;
        wwv_flow_custom_auth.define_user_session(p_user => l_username,
                p_session_id => l_session_id);
        return true;
    else
        -- get username using NTLM
        l_auth := owa_util.get_cgi_env('AUTHORIZATION');
        if l_auth is null then
            owa_util.status_line(nstatus => 401,
                    creason => 'Unauthorized',
                    bclose_header => false);
            htp.p('WWW-Authenticate: NTLM');
            owa_util.http_header_close;
            wwv_flow.g_unrecoverable_error := TRUE;
            return false;
        end if;

        if substr(l_auth,1,5) = 'NTLM ' and l_auth != 'NTLM
TlRMTVNTUAABAAAAB4IIAAAAAAAAAAAAAAAAAAAAAAAAA=' then
            l_decode := utl_encode.text_decode(buf => substr(l_auth,6), encoding
=> UTL_ENCODE.BASE64);
            l_raw := utl_raw.cast_to_raw(l_decode);
            if utl_raw.cast_to_binary_integer(utl_raw.substr(l_raw,9,1)) = 1 then
                owa_util.status_line(nstatus => 401,
                        creason => 'Unauthorized',
                        bclose_header => false);
                htp.p('WWW-Authenticate: NTLM
TlRMTVNTUAACAAAAAAAAACgAAAABggAAAICAgAAAAAAAAAA
AAAAAA==');
                owa_util.http_header_close;
                wwv_flow.g_unrecoverable_error := TRUE;
                return false;
            end if;
            l_length :=
utl_raw.cast_to_binary_integer(utl_raw.substr(l_raw,32,1))*256 +
utl_raw.cast_to_binary_integer(utl_raw.substr(l_raw,31,1));
            l_offset :=
utl_raw.cast_to_binary_integer(utl_raw.substr(l_raw,34,1))*256 +
utl_raw.cast_to_binary_integer(utl_raw.substr(l_raw,33,1));
            l_domain :=
replace(replace(substr(utl_raw.cast_to_varchar2(l_raw),l_offset +
1,l_length),chr(0),null),chr(15),null);
            l_length :=
utl_raw.cast_to_binary_integer(utl_raw.substr(l_raw,40,1))*256 +
utl_raw.cast_to_binary_integer(utl_raw.substr(l_raw,39,1));
            l_offset :=
utl_raw.cast_to_binary_integer(utl_raw.substr(l_raw,42,1))*256 +
utl_raw.cast_to_binary_integer(utl_raw.substr(l_raw,41,1));
            l_user :=
replace(substr(utl_raw.cast_to_varchar2(l_raw),l_offset,l_length),chr(0),null);
            l_username := l_domain||'\'||l_user;
        else
            l_username := 'nobody';
        end if;
        -- application session cookie not valid --> define a new apex session.
        wwv_flow_custom_auth.define_user_session(p_user => l_username,
            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
            wwv_flow_custom_auth.remember_deep_link(p_url => 'f?'||
wwv_flow_utilities.url_decode2(owa_util.get_cgi_env('QUERY_STRING')));
        else
        wwv_flow_custom_auth.remember_deep_link(p_url => '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));
      end if;
      -- register the session in apex sessions table, set cookie, redirect back.
      wwv_flow_custom_auth_std.post_login(p_uname => l_username,
        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 wwv_flow_custom_auth_std.post_login has written,
        -- it contains the session cookie we need.
        l_htp_rows := 15;
        htp.get_page
          ( thepage => l_htp_buffer
          , irows   => l_htp_rows
          );
        -- reset the HTP buffer so that we can write our own header, ...
        htp.init;
        -- See http://www.nabble.com/Empty-POST-requests-on-IE-td15332680.html
        -- We have to trick IE that he thinks the authentication fails, otherwise
        -- he doesn't send any data when issuing a POST because he wants to
        -- do the NTLM stuff again
        owa_util.status_line
          ( nstatus => 401,
            creason => 'Unauthorized',
            bclose_header => FALSE
          );
        -- write the session cookie into our output
        FOR ii IN 1 .. l_htp_rows
        LOOP
            IF l_htp_buffer(ii) LIKE 'Set-Cookie:%'
            THEN
                htp.p(l_htp_buffer(ii));
            END IF;
        END LOOP;
        --
        l_url := '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: '||l_url);
      owa_util.http_header_close;
    ELSE
      -- For 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="'||l_url||'";');
      htp.p('</script>');
      htp.p('<noscript>');
      htp.p('<meta http-equiv="Refresh" content="0; URL="'||l_url||'">');
      htp.p('</noscript>');
      htp.p('</head>');
      htp.p('<body>');
      htp.p('You were logged in successfully. Click <a
href="'||l_url||'">here</a> to continue.');
      htp.p('</body>');
      htp.p('</html>');
    END IF;
   return false;
  end if;
end ntlm_page_sentry;
```

Creating Custom Authentication Scheme

Next create a custom authentication scheme that uses the above function as the page sentry function. To create a custom authentication scheme:

1. Click Shared Components from the Application Builder home page
2. Click Authentication Schemes under Security
3. Click Create >
4. Choose From scratch and click Next >
5. Enter NTLM in the Name field and click Next >
6. Enter return ntlm_page_sentry in the Page Sentry Function text area and click Next >
7. Click Next > until the Confirm step
8. Click Create Scheme
9. Click Change Current

10. Choose NTLM and Click Next >
11. Click Make Current

Now, run the application and you should see your username in the format of DOMAIN\username provided you are using a browser that is configured to support NTLM negotiation.

In an application that uses NTLM Authentication it is possible for someone to sniff traffic on the network. Someone can see the NTLM authorization token for a specific user, and then use that token to spoof the identity of someone and use the application. Use NTLM over HTTPS (SSL) to avoid this vulnerability, but make sure that the SSL is terminated on the web server, not some SSL accelerator (which may in itself facilitate the attack, e.g. if it shares a TCP connection to the server among several clients). Another alternative is to configure the web server not to use persistent HTTP connections for resources that are protected by NTLM authentication.

Browser Support and NTLM

For Internet Explorer to automatically negotiate NTLM, the security settings of the browser must be set to Medium-low or Low. By default, IE is set to Medium-low for local intranet sites, and this authentication really only makes sense for local intranet sites.

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

Windows Vista has local security policies that, by default, do not allow browser negotiation of NTLM authentication. You don’t need to change the setting on each Vista client making use of the Group Policy. NTLM authentication is really only relevant for clients that are part of an Active Directory domain, and therefore, group policy would apply.

NTLM and the Embedded PL/SQL Gateway

Currently, this solution will not work with the XDB HTTP Server with the Embedded PL/SQL Gateway(EPG).

The database rejects access if the browser user attempts to connect explicitly with the HTTP Authorization header. For more information, see the Oracle XML DB Developers Guide, "Configuring Static Authentication with DBMS_EPG."

Oracle Application Express Authentication
Scheme Using NTLM Token Decoding
May 2008
Author: Priyanka Sharma
Contributing Authors: Jason Straub

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com