An Oracle White Paper
May 2011

Oracle ADF 11g Desktop Integration Security Whitepaper

ORACLE

Introduction

Oracle ADF 11g Desktop Integration (ADFdi) is a framework for integrating ADF-based web applications with Microsoft Excel. In delivering integrated workbooks, developers sometimes have questions about how the security features work internally. Likewise, systems administrators often have detailed questions about product security before deploying a solution.

This document attempts to answer questions about how ADFdi security works. This document does not address development or configuration topics. Nor does it address web application security. The reader is encouraged to refer to the appropriate developer's guide for such information.

ADFdi Architecture Summary

The ADFdi framework is composed of a client side portion running on top of Microsoft .NET and two server components: the ADFdi remote servlet and the ADFdi download filter. The server components run within the context of an ADF Fusion web application. The client component acts as the View/Controller and communicates with the servlet to synchronize data and execute business logic in the web application's Model. Communication between the client and server takes the form of HTTP requests and responses.

Session Management

An ADFdi integrated workbook extends and always runs within the context of an ADF Fusion web application. Offline/disconnected use is fully supported, but certain data synchronization and user interface interactions require that a valid user session be established with the web application. ADFdi relies on HTTP cookie-based session management for all of its interactions with the ADF Fusion web application, regardless of whether the web application is configured to enforce authentication or not. This document is focused on how ADFdi interacts with web applications that enforce authentication, but much of the content also applies to non-authenticated session management.

Authentication

ADFdi leverages whatever authentication mechanism has been configured for the web application. That is, if end users are challenged with Basic authentication when accessing the application from a browser, workbook users will see the same challenge when a session is established due to some triggering event in the workbook. The same holds true for Digest, Form-Based, and certificate-based authentication methods.

Overview

ADFdi uses the same mechanism employed by a web browser to identify the current user when accessing the web server: session cookies. The following outlines the approach used by ADFdi to leverage the web application's (browser-based) authentication and session management.

- The ADFdi client framework makes an initial request to the server to test whether the web application is enforcing authentication or not. For web applications that are enforcing authentication, the response status will indicate as much. Conversely, if the initial request receives a 200 OK response, ADFdi takes that as an indication that the web application is not enforcing authentication.
- A response indicating an authenticated web application will trigger the Login Sequence:
  - Just as with a standalone web browser, the user is prompted for credentials
  - ADFdi captures the cookies issued during the authentication sequence
  - (For a non-authenticated web application the cookies are also captured).
- Subsequent requests sent by ADFdi from client to server include those captured cookies

The following sections describe in more detail how ADFdi establishes a user session.

Open an Integrated Workbook

When the workbook first attempts to communicate with the web application the following steps occur:

1. Once per workbook session, the end-user is prompted to provide confirmation that ADFdi may connect to the web application's URL.
2. Assuming that the confirmation is given:
   a. ADFdi uses a WinInet GET request to attempt to access the ADFdi remote servlet URL, for example:
   `http://myHost.myCompany.com:123/myApp/adfdiRemoteServlet`

3. The response from this initial request will dictate whether the ADFdi client treats the web application as authenticated or not. Responses that indicate an authenticated web application include:
    a. Authorization (401, 407)
    b. Redirection (301, 302, 303, 305, 307)
    c. Specific error codes, for example: `ERROR_INTERNET_CLIENT_AUTH_NEEDED` (WinInet error 12044)
    d. If the initial request receives a 200 OK response, ADFdi treats the web application as non-authenticated.
4. To establish a valid, authenticated user session, the Login Sequence is triggered (see below). For non-authenticated web applications, ADFdi makes one or more Session Status check requests to establish a user session.
5. At this point, the original communication that triggered the creation of a user session proceeds (step 6).

On subsequent attempts to communicate with the web application the following steps occur:

6. If not already performed, a Tamper Check request is issued to verify that the workbook's metadata is intact.
    a. On success, the flow continues with the original request in step 7
    b. On failure, communication with the web application is halted and the ADFdi logic in the workbook stops functioning.
7. A POST request is sent to the ADFdi servlet. This request may be an attempt to synchronize Binding Container data, execute an action binding, retrieve a resource bundle, etc. This request uses the cookies captured during the Session Status call.
8. If the cookies sent are acceptable to the web application container, the request is allowed through to the ADFdi remote servlet.
9. When the ADFdi remote servlet receives a request, it:
    a. Verifies the request type from a list of known types
    b. Verifies the expected schema version of the request's XML payload

Login Sequence

The Login Sequence takes place at various points during the workbook lifecycle. The sequence is triggered when the workbook is first opened and/or first needs to contact the authenticated web application for metadata, data, or to invoke business logic. The sequence may also be triggered

as the result of the user having explicitly logged out, or due to a web application session timeout. The next contact to the web application in these cases will trigger the Login Sequence.

Tech Note: The.NET WebBrowser Control

ADFdi uses Windows Forms dialogs that contain an instance of the .Net Web Browser Control (WBC). The WBC uses Microsoft's WinInet library. Microsoft Internet Explorer (MSIE) also uses WinInet, so some MSIE settings affect the WBC.

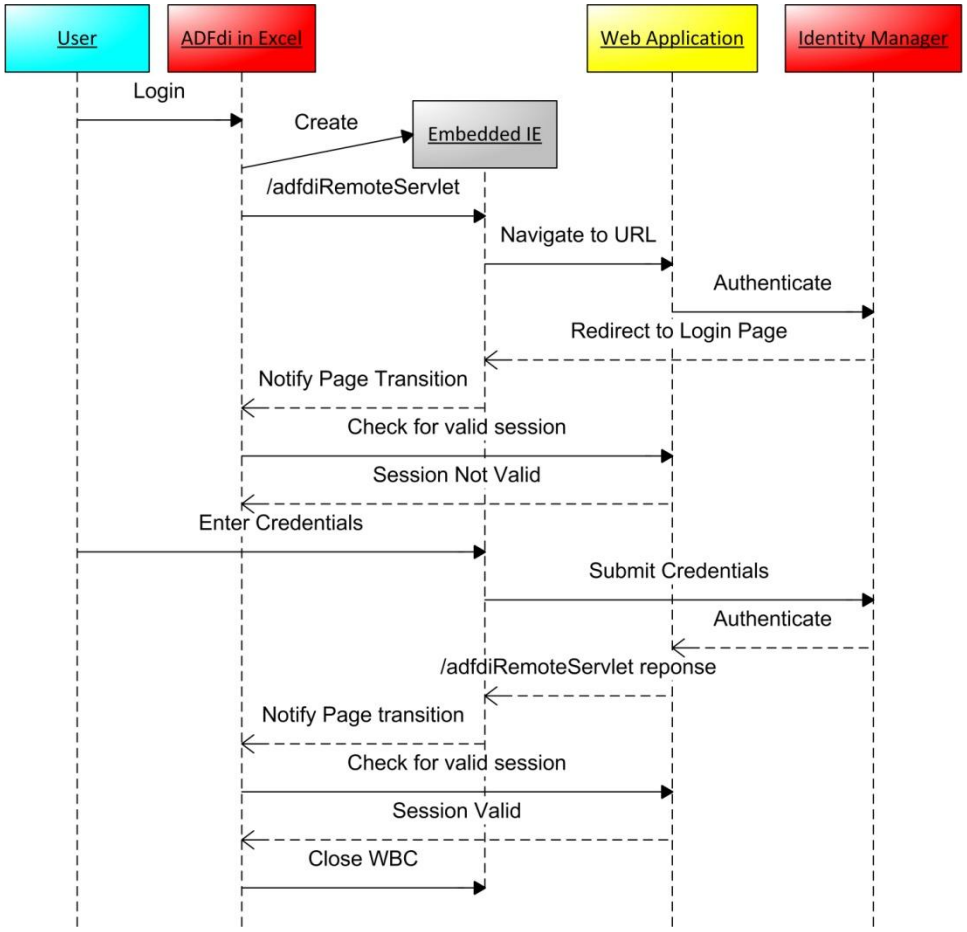

Figure 1. Login Sequence Diagram

1. ADFdi opens a Windows Forms dialog from within the Excel process.
2. ADFdi hooks into the browser control's page transition event.
3. ADFdi instructs the browser control to navigate to the ADFdi remote servlet URL.

Note: The ADFdi remote servlet entry point must be protected by the same authentication and security constraints used for other protected resources in the application. The content from the ADFdi remote servlet entry point should never be displayed. The attempt to navigate to the servlet entry point provokes the web application's authentication response. This response typically takes the form of displaying a login page of some kind.

4. For each page transition, ADFdi performs a "session status check" (see below)
5. When the session status check succeeds, the dialog is closed automatically

At this point, the WinInet library (in the current Excel process) is holding cookies corresponding to a valid, authenticated web application session. The Login sequence is complete and control is returned to the primary ADFdi flow. The ADFdi client side framework also has an in-memory list of cookies that match WinInet's.

Session Status Check

At various times during the workbook session, the ADFdi client framework checks the validity of the user session using the following logic:

1. Construct an HTTP POST request using WinInet to leverage all WinInet cookies registered for the web application URL, including those marked HttpOnly.
2. If the cookies supplied on the request are invalid or represent a timed-out session, a 401 status is returned from the web application container and the ADFdi remote servlet never processes the request.
3. Otherwise a reply is returned to the client and the response payload contains the set of cookies echoed back from the request. The ADFdi client framework captures the list of cookies for use on subsequent managed POST requests to the ADFdi remote servlet.

Managed/Unmanaged HTTP Requests

The ADFdi client framework communicates with the ADFdi remote servlet (running in the context of the web application) using two distinct channels. The vast majority of the communication takes place using .NET's HttpWebRequest APIs (**managed**). These APIs support streamed requests and responses, but unfortunately are not directly integrated with WinInet. All the cookies issued by the web application during the Login Sequence are effectively invisible to these managed APIs. ADFdi's SessionStatus calls use WinInet's **unmanaged** APIs and so are able to gain access to the cookies issued during the Login Sequence.

HttpOnly Cookies

When the web application/container is configured to issue HttpOnly cookies, Microsoft has done an exceptional job at blocking all client side application logic from gaining access to those cookies. In particular, the ADFdi client framework cannot inspect the WinInet cookie list and

see HttpOnly cookies. However, requests made to the web application via the WBC (built on WinInet) or directly via the WinInet library, will send all the cookies appropriate for that web application, including those marked as HttpOnly. Since ADFdi uses the .NET managed classes for its other requests to the ADFdi servlet, the Session Status entry point echoes back all request cookies in its response payload. This approach allows the ADFdi client to capture the cookies (including HttpOnly) for use on subsequent managed requests.

Establish Session In Non-authenticated Application

The ADFdi client framework follows a slightly different code path when the initial request it makes determines that the web application is not enforcing authentication. ADFdi still needs to establish and use a valid user session for all subsequent requests. In this code path, rather than displaying a login dialog, a series of WinInet requests (including Session Status Check) are made in order to capture cookies that identify the user to the web application so that those cookies can be sent on subsequent requests to the ADFdi remote servlet.

Multiple Workbooks in one Excel process

If a user opens two or more integrated workbooks at the same time, the workbooks will generally run in the same Excel process (under Windows). The Excel process has a single state with respect to the WinInet library. So, in effect, both workbooks will share the same web application session. This situation is analogous to opening two "tabs" in Microsoft Internet Explorer. Both tabs share a set of cookies from the MSIE process.

The following sequence illustrates the flow when a second workbook is opened (assuming that the first workbook established a valid user session with the web application).

When the second workbook first attempts to communicate with the same web application the following steps occur:

1. Once per workbook session, the end-user is prompted to provide confirmation that ADFdi can attempt to connect to the web application's URL.
2. Assuming the confirmation is given, ADFdi makes a GET request to the servlet entry point using WinInet.
3. Since the user has already established a valid session with the web application (via the first workbook), the request made in step 2 will succeed, and a Session Status check will capture the relevant cookies.
4. If -- for some reason, say, an explicit Log Out from the first workbook, or a session timeout – the Session Status check fails, the Login Sequence is triggered (see above).
5. At this point, the original communication that triggered the login sequence proceeds.

Web Dialogs

ADFdi provides a feature, known as the "Dialog action" that application developers can use to configure worksheet controls to display web dialogs. Web dialogs employ an embedded .Net Web Browser Control to render a web page. For external web pages (those outside the context of the web application) authentication and user sessions are not managed at all by ADFdi. For web pages within the web application context, the user session is shared with the workbook, since the dialog's WBC shares the session cookie state with the WBC used by the Login Sequence (above). In particular, no additional login is required to render web pages from the web application in a web dialog.

Logout

When ADFdi's client action Logout is invoked:

- ADFdi sends an "invalidate" request to the Session Status entry point (if a connection was previously attempted).
- The ADFdi servlet code gets the `HttpSession` instance from the request and invokes the `invalidate()` method on it.
- The ADFdi client framework clears the cookie list from WinInet and from its in-memory cookie list (in the Excel process). All non-permanent cookies are cleared.

When the ADFdi-enabled workbook is closed:

- If there are no other ADFdi-enabled workbooks open, then a Session Status "invalidate" request is sent to the ADFdi remote servlet (if a connection was ever attempted).
- If at least one other ADFdi-enabled workbook is open, then the session is not invalidated, since it is shared across workbooks.

Authorization

ADFdi does not explicitly manage or enforce Authorization control. Instead ADFdi relies on the web application's Model tier to enforce authorization.

Workbook Tamper Checking

The ADFdi metadata that describes the workbook-to-web-application integration is stored in hidden worksheets in plain text. This metadata is vulnerable to tampering by a malicious person or program. In order for the user of a given workbook to have confidence in the integrity of a workbook, ADFdi includes a feature to check for metadata tampering.

At the end of the workbook integration design process, the developer publishes the workbook for use in the web application. The publish process computes a reliable hash code of the workbook metadata. This hash code is registered with the web application.

At runtime, the workbook re-computes the metadata hash code. The new value is provided to the server with the first request. If the server is unable to get a perfect match on this hash code, an error is returned to the client.

If the client receives an error from the tamper check process, it reports this failure to the user directly. Then, the integration framework shuts down for that workbook.

So, while the ADFdi framework does not prevent tampering of workbook metadata, it does detect such tampering and prevent the user from using a tampered workbook.

Additional Security Considerations

.NET WebBrowser Control

ADFdi displays WinForms dialogs that contain an instance of the .Net Web Browser Control (WBC). These dialogs are used during the Login Sequence and also to render web pages (Web Dialogs). The WBC uses Microsoft's WinInet library which is also shared with Microsoft Internet Explorer (MSIE). Much of the HTTP connection implementation and many configuration settings used by WinInet are common between the WBC and MSIE. In particular, for a given URL, once a set-cookie response header is received, that cookie will be sent on subsequent requests by both MSIE and the WBC. The ADFdi client relies on WinInet for various Session Status operations and utilizes any cookies set during the Login Sequence orchestrated by the WBC.

SSL/HTTPS

ADFdi-integrated workbooks work seamlessly with secure HTTP communications. As with Internet Explorer accessing a web application over HTTPS, this support may require that the necessary certificates are installed using Internet Explorer's Internet Options.

Single Sign-On

ADFdi-integrated workbook works seamlessly with Oracle Access Manager and other single-sign-on solutions with as long as the identity management product supports standard J2EE cookie-based session identifiers. For Oracle Access Manager environments, it is critical that the ADFdi remote servlet entry point (eg. /myApp/adfdiRemoteServlet) is added as a protected resource in the OAM configuration.

Note: In order for ADFdi to operate properly in an OAM/SSO environment that is using WebGate 11g, the user-defined configuration parameter **"filterOAMAuthnCookie"** must be set to false to allow the ADFdi servlet to receive, capture, and echo the OAM authentication cookie(s) during Session Status requests.

Self-Healing Sessions

When the web application is configured to use certain authentication types, the end-user's session will be automatically re-established after the session timeout period specified for the web application. This behavior applies to Basic and Digest, but not Form-Based authentication methods.

Document Data Security

Before creating integrated solutions that download sensitive data to Excel workbooks, due consideration must be given to the handling and protection of these workbooks.

- Are the desktop machines of the authorized users secured?
- Will these users employ Excel password protection in the integrated workbooks?

References

- Oracle® Fusion Middleware Desktop Integration Developer's Guide for Oracle Application Development Framework
- Oracle® Fusion Middleware Programming Security for Oracle WebLogic Server
- Oracle® Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework
    - Enabling ADF Security in a Fusion Web Application

Conclusion

Oracle ADF 11g Desktop Integration includes a variety of mechanisms intended to enforce high standards of security between the user, the integrated spreadsheet, and the host web application. ADFdi leverages web application security and session management to provide a consistent experience and a single source of truth for authentication.

Security in Oracle ADF 11g Desktop Integration
May 2011
Author: Shaun Logan
Contributing Authors: E. Alex Davis

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
oracle.com

Oracle is committed to developing practices and products that help protect the environment

0109