Windows Builtin SSO + Duo 2FA for Java HTTP Applications using the Jespa DuoHttpSecurityService

The DuoHttpSecurityService, and its Filter version DuoHttpSecurityFilter, combines the password-free experience of Windows builtin SSO with the two-factor authentication provided by Duo from duo.com / duosecurity.com.

This combination is considerably more secure than conventional cloud SSO for several reasons:

- Resistant to session hijacking because each TCP connection is first authenticated using Windows builtin SSO (and for non-SSO clients if the browser password dialog is used).
- Passwords do not exist in browser-space or on the authorization server. NTLMSSP tokens are produced and consumed by the Windows SSPI regardless of browser.
- Users are not typing passwords that might be logged or observed.
- Using one Active Directory identity reduces attack surface, prevents leaking data through uncontrolled accounts and makes credential management and policy enforcement easier.

Additional benefits specific to this solution are:

- 100% Java provides a self-contained solution that runs equally well on Linux, Windows or any other Java platform.
- Easier to install. NTLM requires only creating a Computer account in Active Directory and setting its password. NTLM is peer-to-peer unlike Kerberos[1] which is relatively complicated to setup and has numerous dependencies that limit its utility with an HTTP service (clients must have access to a domain controller).
- Jespa is a generic security library that allows creating highly customized solutions (extend the DuoHttpSecurityService to create a Filter that handles the specific needs of your application).
- Sophisticated Windows specific functionality for locating domain controllers and transparent domain controller and name server failover.
- Performance is excellent. The core HttpSecurityService can handle many authentications concurrently. Efficient implementation minimizes network communication and memory usage. Fast Windows group based access control using `groups.denied` and `groups.allowed` properties and at code-level using `request.isMemberOf(“domain\\group“)`.
- With an OEM license Jespa can be transparently distributed with your product.

[1] While Kerberos is cryptographically superior to NTLM, using HTTPS almost completely negates this weakness. Even if NetNTLMv2 tokens were exposed, in real-world practice, they are difficult to reverse.

Windows Builtin SSO vs Conventional Cloud SSO

Windows clients that are joined to a Windows domain can access HTTP resources using the credentials the end-user entered when they logged into their workstation or device. Unlike many cloud SSO schemes, Windows builtin SSO occurs transparently for all resources being accessed. There is no single entry point like a login page that triggers authentication. Windows builtin SSO occurs for every TCP connection between the user-agent and the HTTP server. Even if the user-agent is not challenged for authentication, it can still proactively initiate Windows builtin SSO and the server must process it.

With conventional cloud SSO solutions, users typically have to type credentials into a login form at an authorization server. Windows builtin SSO eliminates this password entry step to provide the security and convenience of a truly password-free experience.

External User-Agents

For clients that are not joined to a Windows domain, end-users can still enter their credentials using the browser password dialog or a conventional login form. If the browser password dialog is used, the NTLMSSP handshake will occur just as it would if Windows builtin SSO were used such that passwords do not exist in browser-space. With a conventional login form, passwords are exposed to browser-space and exist, albeit momentarily, on the authorization server in plaintext.

The DuoHttpSecurtyService can also be configured to allow end-users to selectively *not* use Windows builtin SSO and instead login with alternative Windows credentials using a conventional login form. At any time they can then logout and go back to doing Windows builtin SSO.

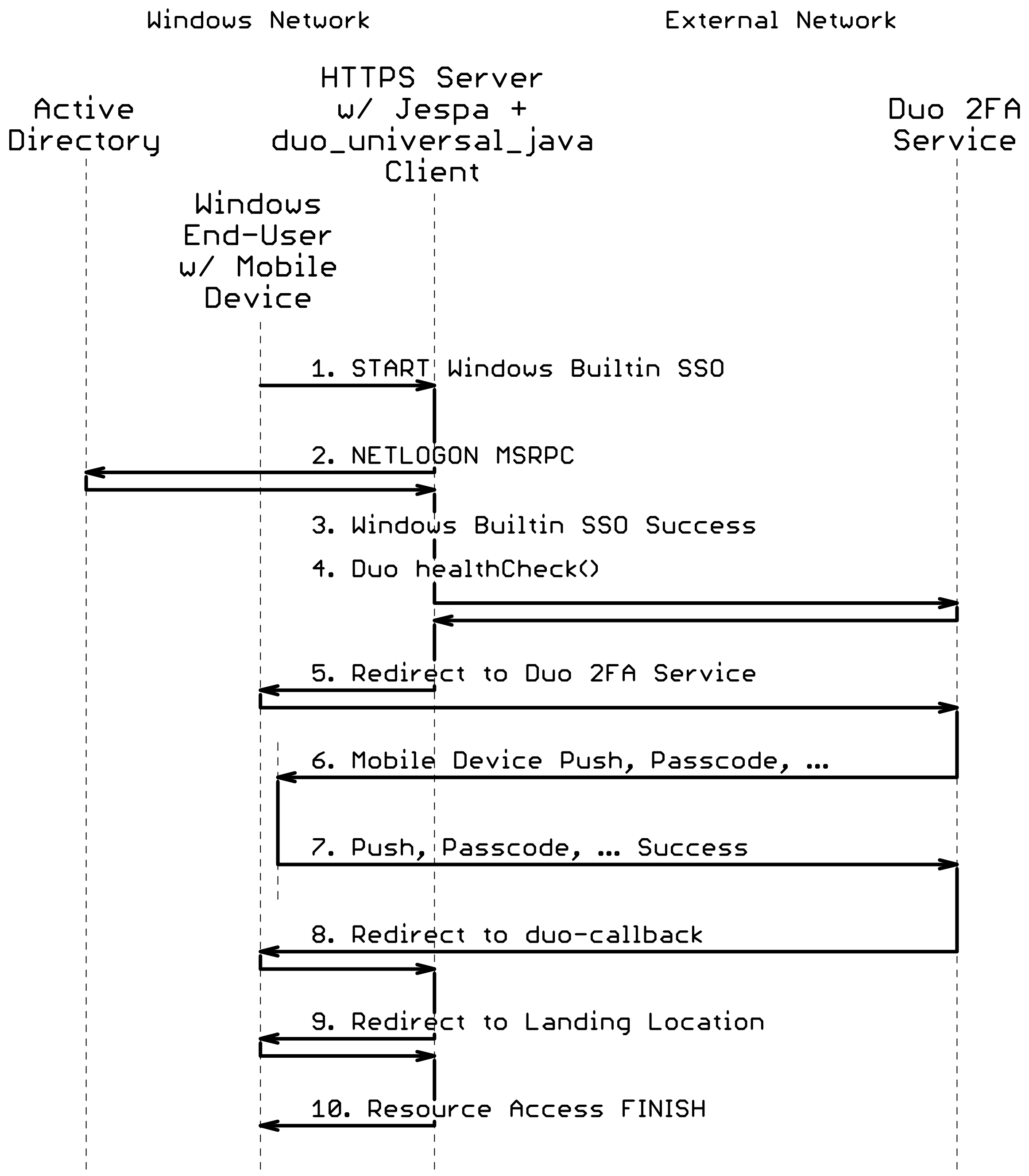
Duo 2FA

After the end-user has successfully performed any permutation of the above described first factor authentication, they will be redirected to the Duo API host to perform 2FA such as through the Duo app on their mobile device, entering a hardware code, by email or other method configured through the Duo admin dashboard.

If 2FA is successful, the resulting token is stored in the user's session to suppress 2FA until the session expires or the application removes the user's token from the session or the user invokes an explicit logout to remove their token from the session.

See also the `jespa.http.DuoHttpSecurityService` Java API documentation under `docs/api` within the Jespa package and on the Support page at ioplex.com.

Sequence Diagram of Jespa + Duo 2FA using the DuoHttpSecurityService


Windows Network
External Network
Active Directory
HTTPS Server w/ Jespa + duo_universal_java Client
Duo 2FA Service
Windows End-User w/ Mobile Device
1. START Windows Builtin SSO
2. NETLOGON MSRPC
3. Windows Builtin SSO Success
4. Duo healthCheck()
5. Redirect to Duo 2FA Service
6. Mobile Device Push, Passcode, ...
7. Push, Passcode, ... Success
8. Redirect to duo-callback
9. Redirect to Landing Location
10. Resource Access FINISH

Requirements

Java 1.5 or later (Jespa is 100% Java that runs equally well on Linux, Windows or any OS that supports Java).

A Computer account created specifically for Jespa in the Active Directory domain of users accessing protected resources (cross domain authentication works transparently but can be slower).

Note: A regular User account will not work. Only a Computer account can bind the NETLOGON service of Windows domain controllers.

HTTPS is required. Cloud SSO protocols exchange sensitive data as GET parameters. The Duo 2FA server will not redirect clients to non-HTTPS servers. For InterAnet installations, an Active Directory Certificate Authority certificate is ideal. Browsers of Windows clients joined to the domain will automatically have access to and use the domain CA certificate. But a conventional Internet CA certificate is required if clients will access the server from the Internet.

Keep-Alive behavior is required. The NTLMSSP over HTTPS SSO protocol is a three-request "handshake" that must occur over the same TCP connection. If this connection is broken for any reason, Windows builtin SSO will not be successful. General purpose web servers like Tomcat support and use Keep-Alive by default. But a simple load balancer will not. See the *Requirements* section in the Jespa Operator's Manual for details.

Ports for MSRPC TCP transport to the NETLOGON service (and lsarpc service if group-based access control features are used) must be open between the Jespa host and the domain controllers being used. See the *Jespa 1.2 Port Requirements* section in the Jespa Operator's Manual for details.

Maven is not technically required but it is the recommended tool for retrieving the duo_universal_java client package and its numerous dependencies. The jespaduo-example provides a Maven `pom.xml` file to easily build a WAR file.

A Duo administrator account on duosecurity.com must create a "Web SDK" application entry to obtain a Client ID, Client Secret and API host URL for use with the Duo 2FA service.

Installation

The following instructions are mostly specific to Tomcat on Windows. Console commands are used for clarity but files can be copied and edited by conventional UI methods. See Appendix A for a concise list of commands for Jetty on Linux. Interpolate as necessary for other environments.

If necessary, install Java on the HTTP server host.

If necessary, install a suitable Java HTTP server such as Jetty or Tomcat.

HTTPS is required although the rest of the installation can be completed before it will be used.

Install Maven

Download and install Apache Maven from https://maven.apache.org/. Very generally, this means setting user environment variables for `JAVA_HOME` and adding the Maven `bin` directory to `PATH`. Search for "env" or navigate to Control Panel > User Accounts and select "Edit environment variables for your account" or “Change my environment variables”. Add a new variable for `JAVA_HOME` with the path to your Java installation like:

Variable name: `JAVA_HOME`
Variable value: `%ProgramFiles%\Java\jre1.8.0_311`

Edit the `PATH` viable and add the Maven bin directory:

Variable name: *`PATH`*
Variable value: *`...ata;C:\path\to\maven\bin`*

Install the jespaduo-example Webapp

Download and unpack the latest version of Jespa. This example references Jespa version 1.2.8. Adjust these references to match the version of Jespa that you use.

Start a `cmd.exe` console and change to the directory jespaduo-example:

```
cmd> cd c:\path\to\jespa-1.2.8\examples\jespaduo-example\
```

All of the commands that follow are relative to the jespaduo-example directory.

Copy the Jespa jar file into the source tree.

```
cmd> copy ..\..\jespa-1.2.8.jar src\main\webapp\WEB-INF\lib
             1 file(s) copied.
```

Copy the example `jespaduo.prp` file from the source tree to a location suitable for configuration files.

Note: Do not edit the file in the source tree. In this example, we make the jespaduo.prp file external to the WAR file so that it can be modified at runtime.

The commands used here put the `jespaduo.prp` file in `$CATALINA_BASE\conf` where `$CATALINA_BASE = C:\path\to\tomcat-base` but it can be anywhere on the local drive[2].

```
cmd> copy src\main\webapp\WEB-INF\jespaduo.prp c:\path\to\tomcat-base\conf\
             1 file(s) copied.
```

Create the Computer Account for Jespa in Active Directory

As an Administrator, create the Computer account in Active Directory as described in the Installation section of the Jespa Operator's Manual. The `bindstr`, `service.acctname` and various other properties will be updated in the `jespaduo.prp` file in the next step (if you use the `SetupWizard.vbs` script, all of these properties should be saved in the file `SetupWizard.txt`).

[2] The Jespa properties file should not be located on a network drive because the HttpSecurityService checks the file every 5 seconds to see if it has changed. A slow network drive or a partial network failure may impact performance or cause avoidable errors.

Replace Jespa Properties in the jespaduo.prp File

Open the `jespaduo.prp` file in a text editor:

```
cmd> notepad c:\path\to\tomcat-base\conf\jespaduo.prp
```

Update the block of Jespa properties generated when creating the Computer account in the previous step. The result should look something like the default example (uncomment `jespa.dns.site` if you are using Active Directory Sites and Services):

```
jespa.bindstr = busi.corp
jespa.dns.servers = 192.168.15.110,192.168.15.115
#jespa.dns.site = Paris
jespa.service.acctname = jespa1$@busi.corp
jespa.service.password = ALongRandomPassword
```

See *The NtlmSecurityProvider Properties* section in the Jespa Operator's Manual for details.

Replace Duo Properties in the jespaduo.prp File

Log into your duosecurity.com Admin account, creating one if necessary, select *Applications* > *Protect an Application* (upper right button), scroll to the bottom for the *Web SDK 2FA* row and select *Protect* > *Activate Now*.

Copy the Client ID, Client secret and API hostname into the `jespaduo.prp` file. Adjust the redirect URL hostname and port to match your Tomcat installation. HTTPS is required so this must be an `https://` URL. The result should look something like the default example:

```
duo.clientId = DIRSJWINVALIDK72JSU2
duo.clientSecret = kSKf93kInValidPDfKNdI2lsLmVUiT90011sMCn
duo.api.host = api-1234abcd.duosecurity.com
duo.redirect.uri = https://apps1.busi.corp:8443/jespaduo-example/duo-callback
```

Save the `jespaduo.prp`.

See *The HttpSecurityService Properties* section in the Jespa Operator's Manual for details about the other properties in the jespaduo.prp file.

Build the jespaduo-example.war File with Maven

Now build the WAR file using the below Maven command. There are two Maven properties supplied as arguments that specify paths that will be external to the WAR file (just used for simple substitution in the `src\main\webapp\WEB-INF\web.xml` file).

The jespa.properties.path Maven Property

The `jespa.properties.path` property specifies the path to the jespaduo.prp file. The default value is `/WEB-INF/jespaduo.prp`. For this example, an external properties file is used so that it can be modified at runtime without reloading the webapp (if an external `jespaduo.prp` file is modified, the DuoHttpSecurityService will automatically reload it within 5 seconds).

IMPORTANT: If the properties file will be external from the WAR file, it must be prefixed with “`file:/`” (one slash only) followed by the platform specific path like “`file:/C:\path\…`”. This is not a URL. The prefix is simply an escape sequence to indicate to the underlying HttpSecurityService that the path is an absolute path.

The jespa.log.path Maven Property

The `jespa.log.path` property specifies the path to the Jespa log file. The default value is `/tmp/jespaduo.log` (which on Windows equates to `C:\tmp\jespaduo.log`). The `jespa.log.path` is *not* prefixed with “`file:/`” as it is always an absolute path. This property should be changed to a more appropriate location such as within the application server logs directory. Viewing the Jespa log file will be important for verifying that everything is working correctly and for debugging if necessary.

The following is an example of the Maven command for building the WAR file.

Note: The following command must be typed as one continuous line

```
cmd> mvn package "-Djespa.properties.path=file:/C:\path\to\tomcat-
     base\conf\jespaduo.prp" "-Djespa.log.path=C:\path\to\tomcat-
     base\logs\jespaduo.log"
     [INFO] Scanning for projects...
     ...
     Downloaded from central: https://repo.maven.apac...
     Downloaded from central: https://repo.maven.apac...
     [INFO] Packaging webapp
     ...
     [INFO] Building war: c:\path\to\examples\jespaduo-example\target\jespad
     uo-example.war
     [INFO] ------------------------------------------------------------------
     [INFO] BUILD SUCCESS
```

The above Maven command will download numerous packages (especially the first time it executes) and may take at least 30 seconds even with a fast Internet connection. The command should complete with "`BUILD SUCCESS`" as shown above.

Note: If you get an error “'mvn' is not recognized as an internal or external command”, you may need to launch a fresh cmd.exe console to inherit any new or edited environment variables. Run cmd> SET, look at your PATH and JAVA_HOME environment variables and verify that they are correct as per the section above about installing Maven.

Deploy the jespaduo-example.war File

The Maven command should produce a `target\jespaduo-example.war` file. To deploy the WAR file in Tomcat, simply copying it into the webapps directory with a command like:

```
cmd> copy target\jespaduo-example.war c:\path\to\tomcat-base\webapps\
             1 file(s) copied.
```

and (re)start Tomcat.

Note: When re-deploying a WAR file using the simply copy method, it may be necessary to first clean the exploded files and restart Tomcat. An alternative method is to use the Tomcat Manager webapp. There are numerous methods for deploying WAR files in Tomcat but the simple copy method should be sufficient for this example.

Test the jespaduo-example Webapp

If the WAR file is successfully deployed, test the jespaduo-example webapp by visiting the context root with a URL like https://app.server.name:8443/jespaduo-example/.

Note: If the page displays your identity as "null", this means that Java classes did not load successfully such as because a jar file is missing or file permissions are incorrect or similar. View the application server stderr logs for errors.

Note: If you get the browser password dialog, this indicates that the browser is not configured to perform SSO (although entering Active Directory credentials should work). Add the target site to Internet Options > Local Intranet Zone. See the Requirements and Browser Settings for SSO in the Jespa Operator's Manual for details.

If the installation is successful, the webapp should redirect the user-agent to the Duo 2FA service specified with the duo.api.host property where the end-user will be asked to enroll if they have not already. If 2FA is ultimately successful, the end-user will be redirected back to the jespaduo-example where their Active Directory identity will be displayed along with raw data about their Duo token.

Note: If the return redirect results in a 404 Not found error, this indicates that the duo.redirect.uri property value is incorrect.

Check the `logs/jespaduo.log` file for errors.

Delete the `SetupWizard.txt` file or at least remove the Computer account password from the file.

Protect the `jespaduo.prp` file and anything else that contains your Duo Client secret.

The installation of the jespaduo-example is complete!

Integration with your Application

Copy the filter/filter-mapping sections from the `src\main\webapp\WEB-INF\web.xml` into your application's `web.xml` adjusting parameter values as necessary. Copy all jar files from `WEB-INF\lib` into your application `WEB-INF\lib` directory (or just the Jespa jar and then merge the Maven `pom.xml` into your project to get the duo_universal_java client and its dependencies). Copy the `jespaduo.prp` file adjusting values as necessary.

After seeing everything working in production, gradually reduce the `jespa.log.level`. Eventually this value might be 1 to indicate that only exceptions should be logged.

Appendix A: Jetty on Linux Command Summary

The following is a concise list of commands for installing the jespaduo-example in a Linux environment.

```
$ cd jespa-1.2.8/examples/jespaduo-example/
$ cp ../../jespa-1.2.8.jar src/main/webapp/WEB-INF/lib/
$ cp src/main/webapp/WEB-INF/jespaduo.prp /path/to/jetty-base/etc/
$ gedit /path/to/jetty-base/etc/jespaduo.prp
$ mvn package "-Djespa.properties.path=file://path/to/jetty-
base/etc/jespaduo.prp" "-Djespa.log.path=/path/to/jetty-
base/logs/jespaduo.log"
[INFO] Scanning for projects...
...
$ cp target/jespaduo-example.war /path/to/jetty-base/webapps/
```

Appendix B: Example jespaduo.prp

The following is the default jespaduo.prp file with fictitious example values.

```
# This is an example properties file for the DuoHttpSecurityFilter.
# The NtlmSecurityProvider is used for Windows built-in SSO and the
# Duo Universal Prompt Java Client is used for 2FA with duosecurty.com.

#
# DuoHttpSecurityFilter properties
#
provider.classname = jespa.ntlm.NtlmSecurityProvider
http.parameter.username.name = username
http.parameter.password.name = password
http.parameter.logout.name = logout
fallback.location = /jespaduo-example/Login.jsp
excludes = /Login.jsp
#groups.allowed = BUSICORP\\Domain Admins

#
# NtlmSecurityProvider properties
#
#jespa.log.path = /tmp/jespaduo.log
jespa.log.level = 4
jespa.account.canonicalForm = 3

#
# Replace the following with properties determined in Step 1 of
# Installation section in the Jespa Operator's Manual.
#
jespa.bindstr = busi.corp
jespa.dns.servers = 192.168.15.110,192.168.15.115
#jespa.dns.site = Paris
jespa.service.acctname = jespa1$@busi.corp
jespa.service.password = ALongRandomPassword

#
# DuoHttpSecurityService properties
#
# Replace the clientId, clientSecret and api.host properties with values
# from the the Duo admin dashboard under Applications > Web SDK.
# Adjust the redirect.uri to match your webapp.
#
duo.clientId = DIRSJWINVALIDK72JSU2
duo.clientSecret = kSKf93kInValidPDfKNdI2lsLmVUiT90011sMCn
duo.api.host = api-1234abcd.duosecurity.com
duo.redirect.uri = https://apps1.busi.corp:8443/jespaduo-example/duo-callback
```