xCP 2.0 SSO Integrations

RAJAKUMAR THIRUVASAGAM

Contents

Overview 4

General Information 5

Kerberos Integration 6

- Snapshots 6
- Demo Environment 7
- Setup Instructions 7
  - Kerberos setup 7
  - SPN 7
  - Deploy xCP App 8
  - Artifacts 8
  - Changes to applicationContext-security.xml 8
  - Content Server Setup 9
  - Browser Setup 9
- Known Issues 9

WebSEAL Integration 10

- Snapshots 10
- Demo Environment 11
- Setup Instructions 11
  - WebSEAL setup 11
  - Deploy xCP App 11
  - Artifacts 11
  - Changes to applicationContext-security.xml 11
  - Setup Privilege DFC Principal 12
  - Setup WebSEAL junction 12
- Known Issues 13

CA SiteMinder Integration 14

- Snapshots 14
- Demo Environment 14
- Setup Instructions 15

CA SiteMinder setup 15
Deploy xCP App 15
Artifacts 15
Changes to applicationContext-security.xml 15
Content Server Setup 16
Known Issues 16
J2EE Container Integration 17
Snapshots 17
Demo Environment 17
Setup Instructions 18
Tomcat setup 18
Deploy xCP App 18
Artifacts 18
Changes to web.xml 18
Changes to applicationContext-security.xml 18
Known Issues 20
Single user login 21
Snapshots 21
Demo Environment 21
Setup Instructions 21
Deploy xCP App 21
Artifacts 21
Changes to applicationContext-security.xml 21
Known Issues 22
Encrypting password using TrustedAuthenticatorTool 23
Prerequisite 23
Steps 23
RSA ClearTrust Integration 24

Overview

xCP 2.0 supports only form based authentication. The absence of SSO integration in xCP 2.0 has already been noticed. This accelerator tries to bridge the gap by supporting the following SSO authentication mechanisms in xCP 2.0 applications:

1. Kerberos
2. WebSEAL
3. CA SiteMinder
4. J2EE Container
5. Single User (Guest/Anonymous) Login

Thus this accelerator will help to achieve parity in SSO authentication mechanism with xCP 1.x.

General Information

All the setup instructions are based on tomcat server. For other app servers, refer the corresponding app server guide.

The xCP persistence layer has a bug which needs to be patched for all the SSO integrations to work. The fixed persistence jar has been provided with this accelerator.

Kerberos Integration

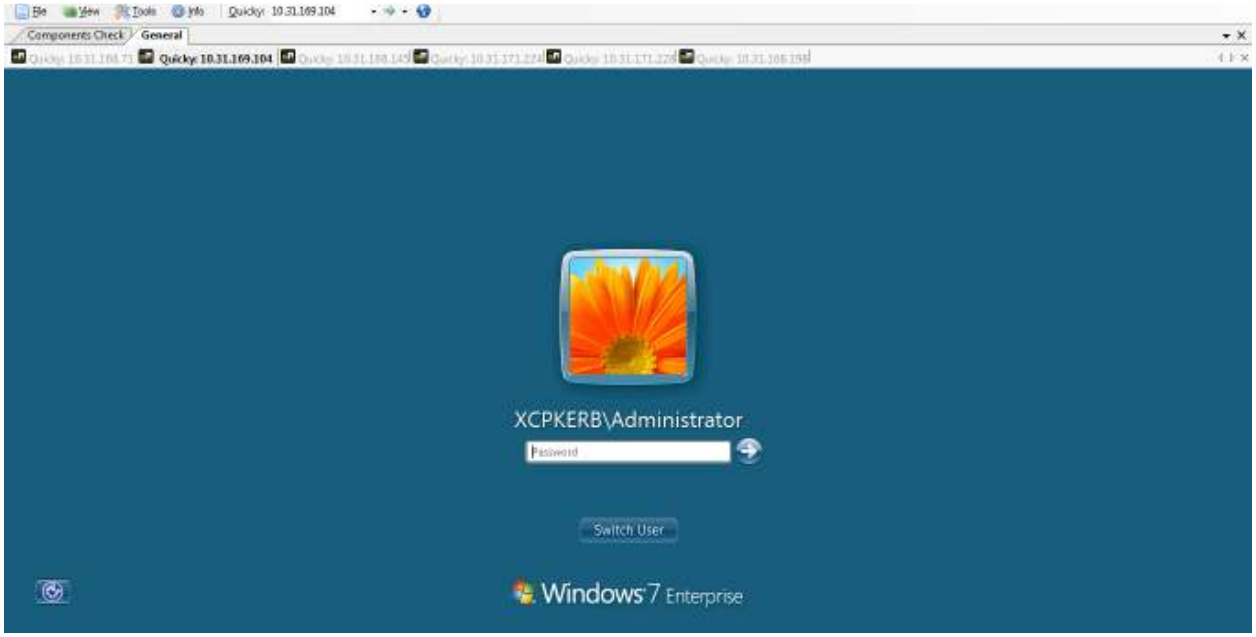
Kerberos is a computer network authentication protocol, which allows individuals communicating over a non-secure network to prove their identity to one another in a secure manner. The most visible benefit of Kerberos for end users is Single Sign On. The end user need not sign on to each application but instead can sign on to their computer once. Kerberos accomplishes single sign on by storing credentials in a secure manner.

In xCP Kerberos integration, the end users are automatically logged in to the docbase using credentials stored in windows private credential area. Unlike the other SSO solutions, Kerberos SSO does not show any authentication challenge to the user. The only authentication challenge that the user encounters is the login to his or her computer by providing Windows domain credentials.

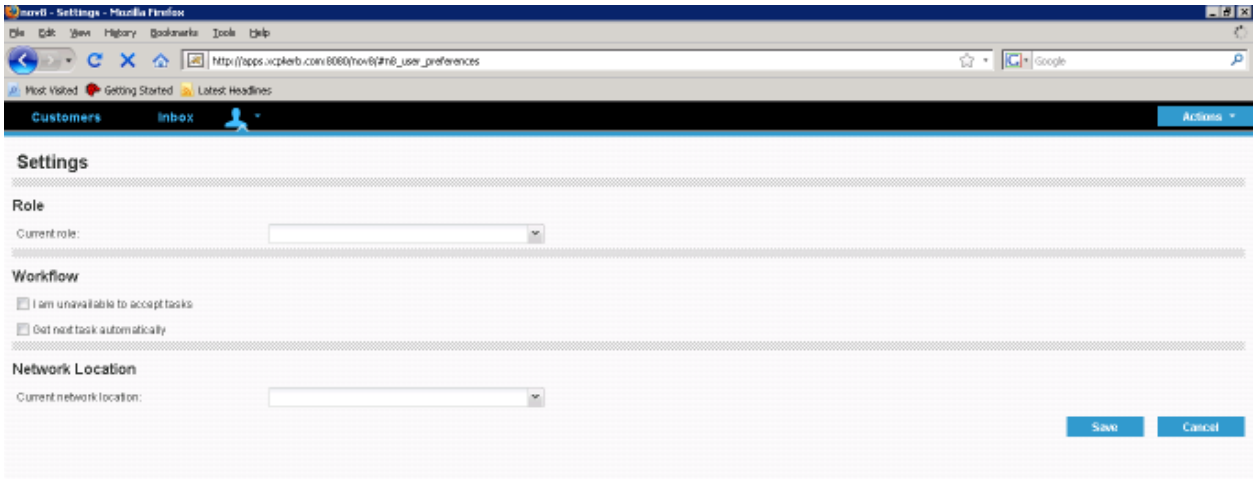
This implementation can support both single domain and multi domain Kerberos configuration. This implementation relies on spring security extension project that support Kerberos authentication.

Snapshots

Login to machine 10.31.169.104 as Administrator under XCPKERB domain.



On accessing the xCP app URL



Demo Environment

Login to machine 10.31.169.104

Username: Administrator
Password: emc101!
Domain: XCPKERB

Use Firefox or IE to access the xCP app
http://apps.xcpkerb.com:8080/nov8

Note: There is no authentication challenge and no login page.

The demo environment for Kerberos integration uses single domain setup. The domain controller, appserver and content server machines are Win 2008 R2. The client machine is Win 7. RC4 HMAC encryption standard is used in this Kerberos setup.

Setup Instructions

Kerberos setup

Install and configure Windows domain controller, add participating machines and create required user accounts. Refer to EMC Documentum Kerberos SSO Authentication white paper for instructions on installation and configuration.

SPN

Service Principal Name (SPN) is a unique name that identifies an instance of a service and is associated with the login account under which the service instance runs. Windows user account names are not multipart as Kerberos principal names. Because of this, it is not possible to directly create an account with the principal name like HTTP/hostname.dns.com. Hence such a principal instance is created

through service principal name mappings. In this case, a user account is created with a meaningful name and a service principal name mapping is added for HTTP/hostname.dns.com

xCP uses browser SPNEGO support to implement Kerberos SSO. In this case, browser requests the service token from the KDC for the requested app using SPN. Browser prepares the SPN based on the app URL. The SPN prepared by the browser is of the following format:

```
HTTP/fully qualified URL@REALM
```

For instance, if the app server URL is 'apps.xcpkerb.com' and realm is 'XCPKERB.COM', then the SPN framed by browser is HTTP/apps.xcpkerb.com@XCPKERB.COM.

Deploy xCP App

Deploy the xCP app in tomcat. Refer to tomcat webapp deployment guide.

Artifacts

1. Copy com.emc.xcp.services.security.ssoauth.jar to <TOMCAT HOME>\webapps\<app root>\WEB-INF\lib.
2. Copy spring-security-kerberos-core-1.0.0.M2.jar to <TOMCAT HOME>\webapps\<app root>\WEB-INF\lib.
3. Copy the updated com.emc.xcp.services.persistence.jar to <TOMCAT HOME>\webapps\<app root>\WEB-INF\lib. Take a backup of this jar before copying the updated jar.

Changes to applicationContext-security.xml

Add spring's SpnegoEntryPoint bean and refer the bean as entry point reference in HTTP security configuration.

```
<http use-expressions="true" realm="xCP" entry-point-ref="spnegoEntryPoint" disable-
url-rewriting="true">
       …
       …
</http>

<beans:bean id="spnegoEntryPoint"
class="org.springframework.security.extensions.kerberos.web.SpnegoEntryPoint" />
```

Add SpnegoDocumentumAuthenticationProcessingFilter bean and refer the bean in HTTP security configuration as pre authentication filter.

```
 <http use-expressions="true" realm="xCP" entry-point-ref="spnegoEntryPoint" disable-
url-rewriting="true">
       …
       …
       <custom-filter ref="spnegoAuthenticationProcessingFilter"
       before="PRE_AUTH_FILTER" />
 </http>


<beans:bean id="spnegoAuthenticationProcessingFilter"
class="com.emc.xcp.security.SpnegoDocumentumAuthenticationProcessingFilter">
        <beans:property name="authenticationManager" ref="authenticationManager" />
</beans:bean>
```

Add KerberosDocumentumAuthenticationProvider beans and refer it authentication provider in the authentication manager.

```
 <authentication-manager alias="authenticationManager">
       <authentication-provider ref="kerberosServiceAuthenticationProvider" />
       <authentication-provider ref="documentumAuthenticationProvider"/>
</authentication-manager>


<beans:bean id="kerberosServiceAuthenticationProvider"
class="com.emc.xcp.security.KerberosDocumentumAuthenticationProvider">
        <beans:property name="repository"
value="#{deployment['dfc.globalregistry.repository']}"/>
        <beans:property name="servicePrincipal" value=""/>
</beans:bean>
```

Add the configured SPN under the servicePrincipal property.

Refer to the provided sample applicationContext-security.xml to make these changes.

Restart your app server for these changes to take effect. Accessing the xCP application through the client machine in the Kerberos domain should login without showing any user authentication challenge.

Content Server Setup

Configure the content server dm_krb authentication plug-in. Refer the content server administration guide and EMC Documentum Kerberos SSO Authentication white paper for details.

Browser Setup

The client machine should be added to the Windows Kerberos domain. Follow the instructions in the EMC Documentum Kerberos SSO Authentication white paper to setup the browser. There are specific instructions for both IE and Firefox.

Known Issues

None

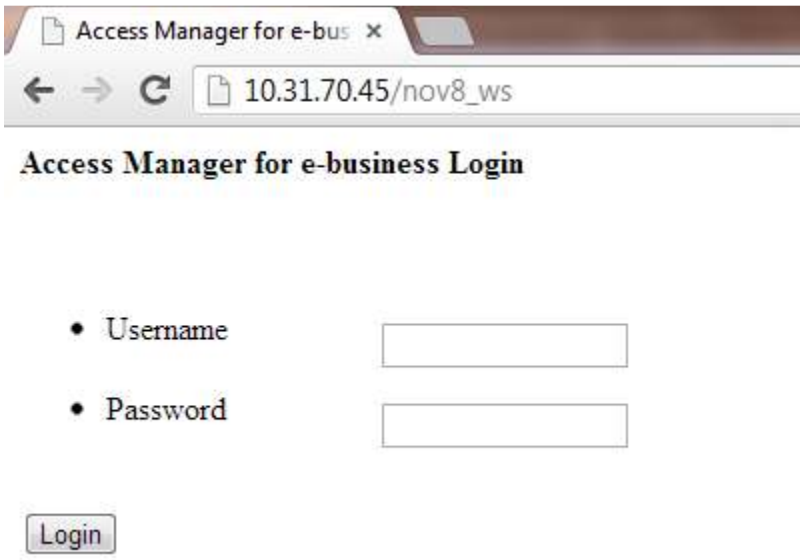
WebSEAL Integration

WebSEAL is a HTTP proxy based SSO solution integrating with its own SSO implementation or other SSO implementations like Kerberos. WebSEAL abstracts the responsibility of authenticating and authorizing users accessing protected resources. WebSEAL recommends trusting all requests landing in webapp domain as authenticated and authorized for access. Hence, there is no second pass authentication at the xCP application layer or DFC. xCP will leverage the privilege DFC principal facility to obtain a valid docbase session.

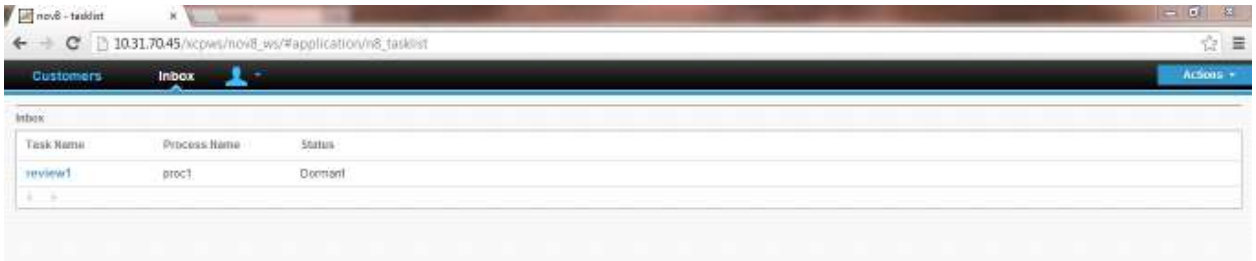
This integration is capable of supporting both one-way and two-way SSL connection between WebSEAL proxy and app server.

Snapshots

On accessing the xCP app URL



Enter ‘sec_master’ as user name
Enter ‘Password@123’ for password
Click Login



Demo Environment

http://10.31.70.45/xcpws/nov8_ws

Username: sec_master
Password: Password@123

Note: These are the credentials for webseal authentication. The webseal proxy will inject its login form when accessing the xCP app.

The demo environment for WebSEAL integration uses non-SSL junction between the WebSEAL proxy and tomcat appserver.

Setup Instructions

WebSEAL setup

Install and configure WebSEAL. Refer to the WebSEAL installation and administration guide for details.

Deploy xCP App

Deploy the xCP app in tomcat. Refer to tomcat webapp deployment guide.

Artifacts

1. Copy com.emc.xcp.services.security.ssoauth.jar to <TOMCAT HOME>\webapps\<app root>\WEB-INF\lib.
2. Copy the updated com.emc.xcp.services.persistence.jar to <TOMCAT HOME>\webapps\<app root>\WEB-INF\lib. Take a backup of this jar before copying the updated jar.

Changes to applicationContext-security.xml

Add the spring's RequestHeaderAuthenticationFilter bean and refer the bean in HTTP security configuration as pre authentication filter.

```
 <http use-expressions="true" realm="xCP" entry-point-ref="xcpAuthenticationEntryPoint"
disable-url-rewriting="true">
       …
       …
       <custom-filter ref="websealFilter" position="PRE_AUTH_FILTER" />
 </http>


<beans:bean id="websealFilter"
class="org.springframework.security.web.authentication.preauth.RequestHeaderAuthentica
tionFilter">
        <beans:property name="principalRequestHeader" value="iv-user"/>
        <beans:property name="authenticationManager" ref="authenticationManager"/>
</beans:bean>
```

Add WebsealDocumentumAuthenticationProvider beans and refer it authentication provider in the authentication manager.

```
 <authentication-manager alias="authenticationManager">
       <authentication-provider ref="websealDocumentumAuthenticationProvider" />
```

```
        <authentication-provider ref="documentumAuthenticationProvider"/>
</authentication-manager>


<beans:bean id="websealDocumentumAuthenticationProvider"
class="com.emc.xcp.security.WebsealDocumentumAuthenticationProvider">
         <beans:property name="repository"
value="#{deployment['dfc.globalregistry.repository']}"/>
</beans:bean>
```

Refer to the provided sample applicationContext-security.xml to make these changes

Setup Privilege DFC Principal

The DFC instance that is used by the xCP app integrated with WebSEAL should be made as privilege client. The DFC instance should also be enabled to have Trusted Login.

Refer to DFC guide on how to setup privilege client and trusted login for DFC instances.

Setup WebSEAL junction

In the WebSEAL server, a junction has to be created pointing to the app server. WebSEAL acts as a proxy and injects authentication when the request passes through the proxy. Refer to WebSEAL administration guide on how to setup junctions.

Restart your app server for these changes to take effect. The xCP application URL will be pointing to WebSEAL proxy. Accessing the xCP application should show WebSEAL login page. Note the user will not be shown any xCP sign-in page.

Known Issues

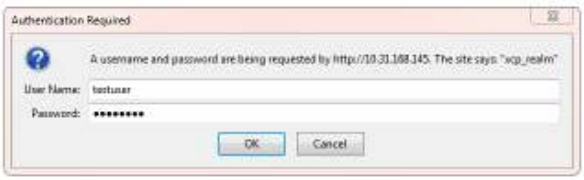
1. This authentication scheme does not seem to work with Firefox. Webseal seems to inject its session cookie as JavaScript code into the application service response which fails loading in ExtJS.
2. On login, the redirection to the application page does not seem to happen. WebSEAL tries to load fav.ico which fails. Refer WEBTOP-22334 resolution.

CA SiteMinder Integration

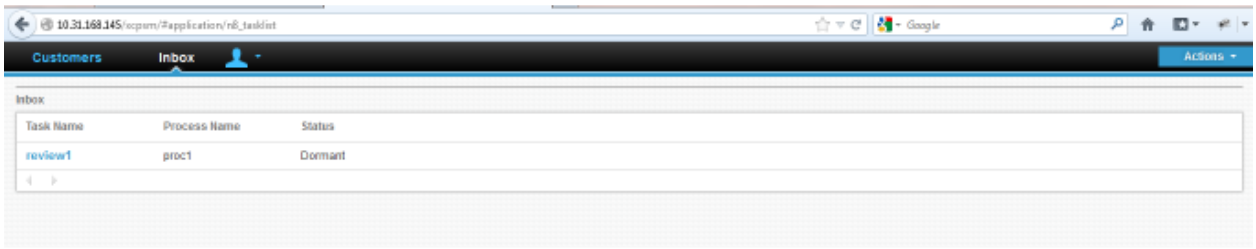
CA SiteMinder is a highly scalable solution that provides best practice identity and access management components for web single sign-on, authentication, authorization, auditing and administration. This integration with xCP uses two-pass authentication. In the first pass, the CA SiteMinder webagent interrupts the request to the xCP app and verify the user credentials before redirecting the request to the app server. In the second pass the authenticated session credential is used to authenticate and get a DFC/Content server session.

Snapshots

On accessing the xCP app URL



Enter ‘testuser’ in User Name
Enter ‘password’ in password
Click OK



Demo Environment

http://10.31.168.145/xcpsm

Username: testuser
Password: password

Note: There will be a basic authentication box displayed to enter the credentials. There will not be any xCP sign-in page.

Setup Instructions

CA SiteMinder setup

Install and configure CA SiteMinder policy server. Refer to the CA SiteMinder policy server installation and administration guide for details.

Install and configure CA SiteMinder webagent on a web proxy. Refer to the CA SiteMinder webagent installation guide for details.

Refer to Netegrity Configuration for EMC Documentum Webtop solutions guide for configuration details.

Deploy xCP App

Deploy the xCP app in tomcat. Refer to tomcat webapp deployment guide.

Artifacts

1. Copy com.emc.xcp.services.security.ssoauth.jar to <TOMCAT HOME>\webapps\<app root>\WEB-INF\lib.
2. Copy the updated com.emc.xcp.services.persistence.jar to <TOMCAT HOME>\webapps\<app root>\WEB-INF\lib. Take a backup of this jar before copying the updated jar.

Changes to applicationContext-security.xml

Add SMRequestHeaderAuthenticationFilter bean and refer the bean in HTTP security configuration as pre authentication filter.

```
 <http use-expressions="true" realm="xCP" entry-point-
ref="xcpAuthenticationEntryPoint" disable-url-rewriting="true">
       …
       …
       <custom-filter ref="siteminderFilter" position="PRE_AUTH_FILTER" />
 </http>


<beans:bean id="siteminderFilter"
class="com.emc.xcp.security.SMRequestHeaderAuthenticationFilter">
        <beans:property name="principalRequestHeader" value="SM_USER"/>
        <beans:property name="authenticationManager" ref="authenticationManager" />
</beans:bean>
```

Add SMDocumentumAuthenticationProviderbeans and refer it authentication provider in the authentication manager.

```
 <authentication-manager alias="authenticationManager">
       <authentication-provider ref="smDocumentumAuthenticationProvider" />
       <authentication-provider ref="documentumAuthenticationProvider"/>
</authentication-manager>


<beans:bean id="smDocumentumAuthenticationProvider"
class="com.emc.xcp.security.SMDocumentumAuthenticationProvider">
```

```
        <beans:property name="repository"
value="#{deployment['dfc.globalregistry.repository']}"/>
</beans:bean>
```

Refer to the provided sample applicationContext-security.xml to make these changes

Content Server Setup

Configure the content server dm_netegrity authentication plug-in. Refer the content server administration guide for details.

Restart your app server for these changes to take effect. Accessing the xCP application should now show CA SiteMinder login page. Note the user will not be shown any xCP sign-in page.

Known Issues

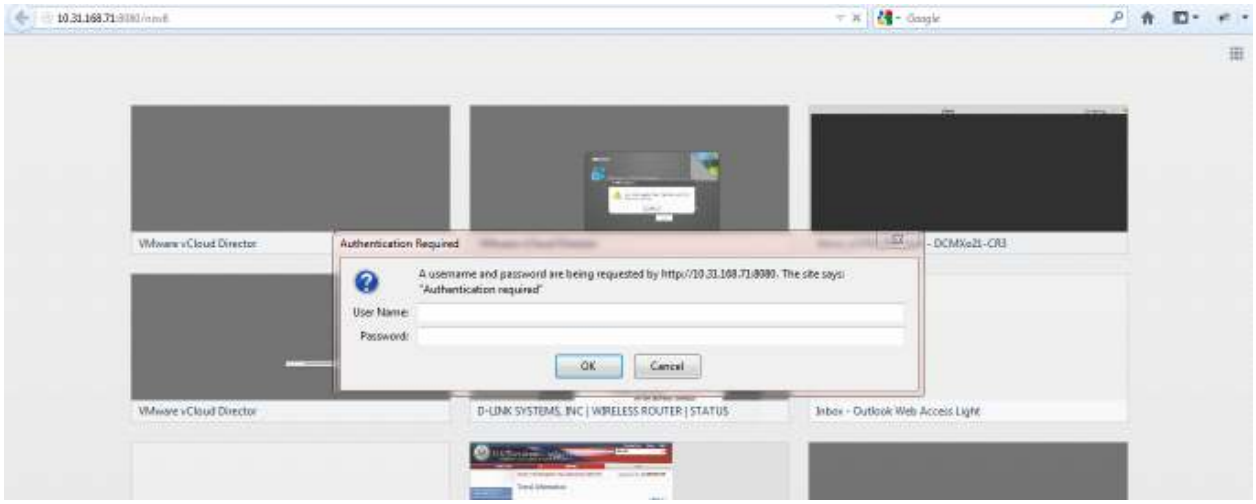
None

J2EE Container Integration

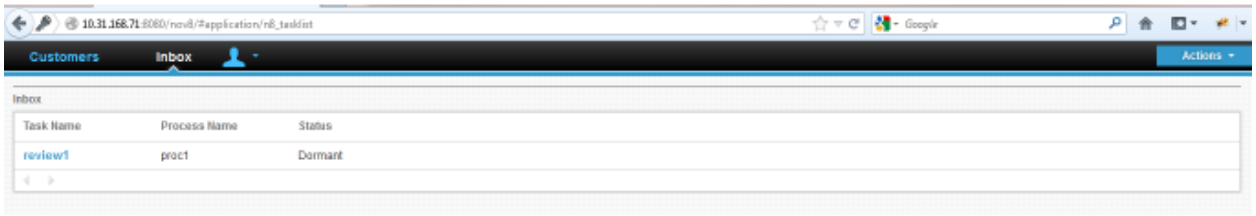
Many J2EE application containers inherently supports authentication. The webapp deployed in such containers can leverage the container authentication. With this integration, the application layer relies on the container authentication and the DFC/CS authentication relies on the principal authentication mode.

Snapshots

On accessing the xCP app URL



Enter ‘dmadmin’ as user name.
Enter ‘123’ as password.
Click OK.



Note: On accessing the app URL, you will be prompted by browser login dialog and there is no xCP sign-in page.

Demo Environment

http://10.31.168.71:8080/nov8

Username: dmadmin
Password: 123

Setup Instructions

Tomcat setup

Modify tomcat-users.xml located in <TOMCAT HOME>\conf\tomcat-users.xml to add a user. For instance, add the following line to add “dmadmin” user.

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="everyone"/>
  <user username="admin" password="admin" roles="manager"/>
  <user username="dmadmin" password="123" roles="everyone" />
</tomcat-users>
```

Deploy xCP App

Deploy the xCP app in tomcat. Refer to tomcat webapp deployment guide.

Artifacts

1. Copy com.emc.xcp.services.security.ssoauth.jar to <TOMCAT HOME>\webapps\<app root>\WEB-INF\lib.
2. Copy the updated com.emc.xcp.services.persistence.jar to <TOMCAT HOME>\webapps\<app root>\WEB-INF\lib. Take a backup of this jar before copying the updated jar.

Changes to web.xml

The web.xml needs to be modified to trigger the container authentication. Modify <TOMCAT HOME>\webapps\<app root>\WEB-INF\web.xml, and add the following lines to bottom of the file within ‘web-app’ tag.

```
<web-app>
…
…
 <security-constraint>
       <web-resource-collection>
         <web-resource-name>xCP</web-resource-name>
         <url-pattern>/*</url-pattern>
         <http-method>POST</http-method>
         <http-method>GET</http-method>
         <http-method>PUT</http-method>
       </web-resource-collection>
       <auth-constraint>
         <role-name>everyone</role-name>
       </auth-constraint>
     </security-constraint>

     <login-config>
       <auth-method>BASIC</auth-method>
     </login-config>
</web-app>
```

Changes to applicationContext-security.xml

Add J2eePreAuthenticatedProcessingFilter bean and refer the bean in HTTP security configuration as pre authentication filter.

```
 <http use-expressions="true" realm="xCP" entry-point-
ref="xcpAuthenticationEntryPoint" disable-url-rewriting="true">
       …
       …
       <custom-filter ref="jeePreAuthenticatedFilter" position="PRE_AUTH_FILTER" />
 </http>


 <beans:bean id="jeePreAuthenticatedFilter"
class="org.springframework.security.web.authentication.preauth.j2ee.J2eePreAuthenticat
edProcessingFilter">
        <beans:property name="authenticationManager" ref="authenticationManager" />
 </beans:bean>
```

Add JeePrincipalAuthenticationProvider bean and refer it authentication provider in the authentication manager.

```
 <authentication-manager alias="authenticationManager">
       <authentication-provider ref="jeePrincipalAuthenticationProvider" />
       <authentication-provider ref="documentumAuthenticationProvider"/>
</authentication-manager>


<beans:bean id="jeePrincipalAuthenticationProvider" class
="com.emc.xcp.security.JeePrincipalAuthenticationProvider">
       <beans:property name="repository"
       value="#{deployment['dfc.globalregistry.repository']}"/>
              <beans:property name="trustProperties">
              <beans:props>
                     <beans:prop key="xcp2012.user">dmadmin</beans:prop>
                     <beans:prop
                     key="xcp2012.password">1/7yU1p+aYdcpff423JXog==</beans:prop>
                     <beans:prop key="xcp2012.domain"></beans:prop>
              </beans:props>
       </beans:property>
</beans:bean>
```

Provide super user credentials in trustProperties for DFC principal authentication mode. Refer to DFC guide on principal authentication mode.

The trust properties should have user, password and domain in the following format. The key should be named as <docbase name>.user, <docbase name>.password and <docbase name>.domain. In the example above, ‘xcp2012’ is the docbase name. The corresponding values for all these keys should be provided as attribute value.

The encrypted password for the super user account should be specified rather than plain text password. Refer to section “Encrypting password using TrustedAuthenticatorTool” for steps to encrypt the password.

Refer to the provided sample applicationContext-security.xml to make these changes.

Restart the app server for these changes to take effect. Accessing the xCP application should now show appserver login dialog. In this case, since the auth-method has been set to BASIC, browser login dialog is shown to the user. Note the user is not shown any xCP sign-in page.

Known Issues

None

Single user login

This scheme will allow automatic login using pre-configured single user credentials. All the users hitting the xCP app will automatically sign in using the anonymous/guest account.

Snapshots

On accessing the xCP app URL

Note: On accessing the app URL, you will not be prompted for user credentials and there is no xCP sign-in page.

Demo Environment

http://10.31.168.71:8080/nov8_su

Setup Instructions

Deploy xCP App

Deploy the xCP app in tomcat. Refer to tomcat webapp deployment guide.

Artifacts

1. Copy com.emc.xcp.services.security.ssoauth.jar to <TOMCAT HOME>\webapps\<app root>\WEB-INF\lib.
2. Copy the updated com.emc.xcp.services.persistence.jar to <TOMCAT HOME>\webapps\<app root>\WEB-INF\lib. Take a backup of this jar before copying the updated jar.

Changes to applicationContext-security.xml

Add SingleUserAuthenticationProcessingFilter bean and refer the bean in HTTP security configuration as pre authentication filter.

```
 <http use-expressions="true" realm="xCP" entry-point-
ref="xcpAuthenticationEntryPoint" disable-url-rewriting="true">
       …
       …
       <custom-filter ref="singleUserAuthenticationProcessingFilter"
       position="PRE_AUTH_FILTER" />
 </http>
```

```
 <beans:bean id="singleUserAuthenticationProcessingFilter"
class="com.emc.xcp.security.SingleUserAuthenticationProcessingFilter">
        <beans:property name="authenticationManager" ref="authenticationManager" />
        <beans:property name="userCredentialProperties">
            <beans:props>
                <beans:prop key="username">dmadmin</beans:prop>
                <beans:prop key="password">S09jH2dAas76MuUVeBlP+Q==</beans:prop>
            </beans:props>
        </beans:property>
    </beans:bean>
```

Provide the single user credentials for the automatic login.

The encrypted password for the single user account should be specified rather than plain text password. Refer to section “Encrypting password using TrustedAuthenticatorTool” for steps to encrypt the password.

Add SingleUserAuthenticationProvider bean and refer it authentication provider in the authentication manager.

```
 <authentication-manager alias="authenticationManager">
      <authentication-provider ref="singleUserAuthenticationProvider" />
      <authentication-provider ref="documentumAuthenticationProvider"/>
</authentication-manager>


<beans:bean id="singleUserAuthenticationProvider"
class="com.emc.xcp.security.SingleUserAuthenticationProvider">
        <beans:property name="repository"
value="#{deployment['dfc.globalregistry.repository']}"/>
    </beans:bean>
```

Refer to the provided sample applicationContext-security.xml to make these changes.

Restart the app server for these changes to take effect. Accessing the xCP application should now automatically login using the pre-configured user credentials. Note the user is not shown any xCP sign-in page or other authentication challenges.

Known Issues

None

Encrypting password using TrustedAuthenticatorTool

The user account passwords specified in the config files should be encrypted using standard encryption techniques. This enhances the security and makes the xCP application security compliant.

TrustedAuthenticatorTool is a utility used to encrypt the plain text passwords using Triple DES 168 bit encryption standard. This tool relies on symmetric encryption key which is stored in the keystore.

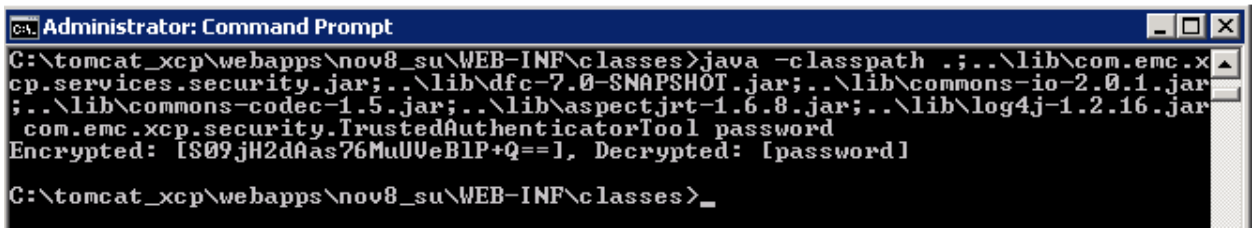
Prerequisite

JRE 1.6 or above

Steps

1. Open a command prompt
2. Navigate to <TOMCAT HOME>\webapps\<app root>\WEB-INF\classes directory
3. Run the following command: java -classpath .;..\lib\com.emc.xcp.services.security.ssoauth.jar;..\lib\dfc-7.0-SNAPSHOT.jar;..\lib\commons-io-2.0.1.jar;..\lib\commons-codec-1.5.jar;..\lib\aspectjrt-1.6.8.jar;..\lib\log4j-1.2.16.jar com.emc.xcp.security.TrustedAuthenticatorTool <password1> <password2>
4. This will print the encrypted password string on the command prompt
5. Copy and paste this encrypted password string into the config file.

The JEE Container Integration and Single user login will be able to decrypt these passwords and use.



Note: Executing the TrustedAuthenticatorTool for the first time will create 'xcp.keystore' file in the <TOMCAT HOME>\webapps\<app root>\WEB-INF\classes directory. This file is required for decryption.

RSA ClearTrust Integration

RSA ClearTrust can be integrated with xCP on the similar lines of CA SiteMinder integration. This accelerator version does not support this integration. However, this can be achieved with minimal effort.