BEA Workshop for WebLogic Security

Current Release Information:

- What's New in 9.2
- Upgrading from 8.1

Useful Links:

- Tutorials
- Tips and Tricks

Other Resources:

- Online Docs
- Dev2Dev
- Discussion Forums
- Development Blogs

Topics in this section describe how to secure applications developed with BEA Workshop for WebLogic Platform.

Topics Included in This Section

External Event and Callback Security

Describes how to secure events and callbacks with role-based security.

Web Service Message Level Security with WS-Security Policy

Outlines how to enable message-level security for web services.

Role-based Control Access Security

Outlines role-based security for controls.

Related Topics

WebLogic Server Documentation

Web Services Security

Web Application Security

EJB Security

Control Security

External Event and Callback Security

This topic explains how to enable role-based security for callback/event methods and their handlers.

Filtering Role Access for Callback and Event Handlers

Callback and event handlers can be protected by specifying which roles are allowed to send events. Roles are specified via the annotation @ExternalCallbackTarget.CallbackSecurity.

The annotation can be applied to all of the callback/event handlers in a control by annotating the control field declaration in the control client with the above annotation, for example:

Control Client code

```
@Control
@ExternalCallbackTarget.CallbackSecurity(rolesAllowed={"manager"})
private MyControl myControl;
```

If the annotation is placed on a specific callback/event handler method, then it overrides the roles specified on the control field.

```
@EventHandler(field="newsServiceControlOnlyManagers",
              eventSet=controls.NewsServiceControl.Callback.class,
              eventName="deliverManagerNews")
@ServiceControl.CallbackSecurity(rolesAllowed={"manager,editor,cityDesk"})
public void manager(String nf)
{
   synchronized(flashes) { flashes.add(nf); }
}
```

Controls which participate in callback/event handler security must extend the @ExternalCallbackTarget interface. This makes it possible to annotate the callback/event handler using the extending control interface itself. For instance, the is valid because @ServiceControl extends (is a subinterface of) @ExternalCallbackTarget:

```
@Control
@ServiceControl.CallbackSecurity(rolesAllowed={"manager"})
private MyControl myControl;
```

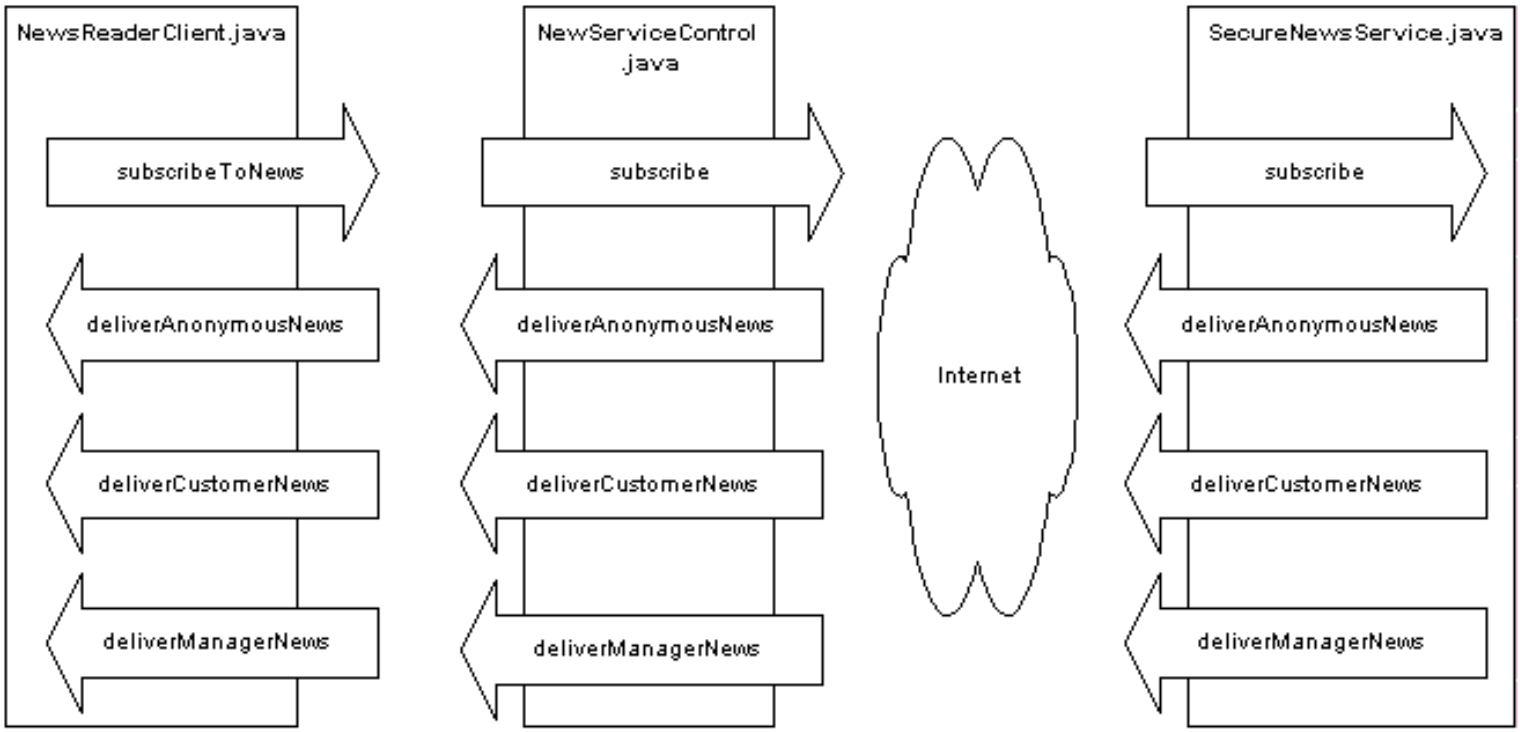
Sample Code: Scenario

Suppose you have a news reader client that invokes methods on and receives callbacks from a news delivery web service through an intermediary service control.

Also suppose there are three types of user for the new reader client:

1. testuser1: has no roles, a.k.a., the anonymous role
2. customer: has only the customer role
3. manager: has both customer and manager roles

The following diagram shows the three classes involved in this scenario: news reader client, the intermediary control, and the news provider web service. Ordinary methods are depicted as arrows pointing to the right, callbacks and event methods/event handlers are depicted as arrows pointing to the left.



The News Delivery Web Service: SecureNewService.java

The news delivery service SecureNewsService.java accepts user credentials through the method subscribe. These credentials are used to perform the appropriate callback by delivering news to managers, customers, and anonymous clients.

```
@javax.jws.WebService
public class SecureNewsService implements java.io.Serializable {

    @Callback
    private CallbackSvc callback;


    @javax.jws.WebMethod
    public void subscribe(String username, String password) {

        callback.setUsername(username);
        callback.setPassword(password);

        callback.deliverManagerNews("This is news for managers");
        callback.deliverCustomerNews("This is news for customers");
        callback.deliverAnonymousNews("This is news for anyone");
    }

    @CallbackService
    public interface CallbackSvc extends CallbackInterface {
        @WebMethod
        void deliverCustomerNews(String news) throws java.rmi.RemoteException;

        @WebMethod
        void deliverManagerNews(String news) throws java.rmi.RemoteException;
```

```
        @WebMethod
        void deliverAnonymousNews(String news) throws java.rmi.RemoteException;
    }

}
```

The Web Service Control: NewsServiceControl.java

Note that the control's event set methods must be annotated with @ExternalCallbackTarget.ExternalCallbackEvent or, in the case of web service controls, @ServiceControl.ExternalCallbackEvent. Note that the @ServiceControl.ExternalCallbackEvent annotation is automatically added when the service control is generated from a WSDL file.

```
@ControlExtension
public interface NewsServiceControl extends ServiceControl
{
    @EventSet
    public interface Callback extends ServiceControl.Callback
    {
        @ServiceControl.ExternalCallbackEvent
        public void deliverAnonymousNews(java.lang.String news);

        @ServiceControl.ExternalCallbackEvent
        public void deliverCustomerNews(java.lang.String news);

        @ServiceControl.ExternalCallbackEvent
        public void deliverManagerNews(java.lang.String news);
    }


    public void subscribe(java.lang.String username_arg,java.lang.String password_arg);
}
```

The News Reader Client: NewsReaderClient.java

The client NewsReaderClient.java contains a control declarations of the control type NewsServiceControl.java.

This control declaration, named `newsServiceControl`, is not protected at the declaration level. Instead it is protected at the method level to allow anyone to access the anonymous news delivery (even unauthenticated users), only mangers to deliver the manager news, and only customers (which includes managers) to deliver customer news. It is important to note that the calback/event security constrains who can *deliver* news (because the constraints put a filter on which roles can send callback messages to the event handlers). Any constraints on who can read news would be accomplised through the @RolesAllowed annotation.

```
@WebService
public class NewsReaderClient
{
    @Control()
    protected NewsServiceControl newsServiceControl;

    @EventHandler(field="newsServiceControl",
                  eventSet=controls.NewsServiceControl.Callback.class,
                  eventName="deliverAnonymousNews")
    @ServiceControl.CallbackSecurity(rolesAllowed={"Anonymous"})
    public void deliverAnonymousNews(String news)
    {
      System.out.print(news);
    }

   @EventHandler(field="newsServiceControl",
```

```
                eventSet=controls.NewsServiceControl.Callback.class,
                eventName="deliverManagerNews")
    @ServiceControl.CallbackSecurity(rolesAllowed={"manager"})
    public void manager(String news)
    {
       System.out.print(news);
    }

   @EventHandler(field="newsServiceControl",
                eventSet=controls.NewsServiceControl.Callback.class,
                eventName="deliverCustomerNews")
    @ServiceControl.CallbackSecurity(rolesAllowed={"customer"})
    public void customer(String news)
    {
       System.out.print(news);
    }

    @WebMethod()
    public String subscribeToNews(String username, String password)
    {
     newsServiceControl.subscribe(username,password);
     newsServiceControlOnlyManagers.subscribe(username,password);
     return "Subscribed for "+username;
    }
}
```

A related version of the sampled code above is included in the SamplesWorkspace.

Guidelines for Secure Controls

The following are requirements for a control extention that uses role-based security.

1. The control must extend the interface com.bea.control.ExternalCallbackTarget. (Note that service controls do not need to explicitly extent ExternalCallbackTarget, because they already extend ServiceControl, a subinterface of ExternalCallbackTarget.)

2. The methods in the EventSet interface of the control must be annotated with the annotation: @ExternalCallbackTarget.ExternalCallbackEvent / @ServiceControl.ExternalCallbackEvent.

3. Only the @ExternalCallbackTarget.CallbackSecurity/@WebService.CallbackSecurity annotations used by the client of the control effect security. That is, you cannot apply these annotations to the event set methods on the control.

Related Topics

Sample: Secure News Service

Web Service Message Level Security with WS-Policy

Message-level security is available for web services through the use of security policy XML files, called "WS-Policy" files. The WS-Policy file defines the signature, encryption, and tokens associated with the message.

The WS-Policy file (or files) is associated with the web service file through @Policy or @Policies annotations. Note that @Policy/@Policies annotations are not required to associate a web service and a WS-Policy file: runtime association mechanisms are also available through the Administration Console. But any policy associated with a web service through @Policy/@Policies annotations *cannot be disassociated* at runtime: the @Policy/@Policies annotations create a hardcoded association that cannot be undone at runtime.

Note that @Policy/@Policies can be applied to both web services and service controls, but runtime association through the Administration Console only applies to web services, not service controls.

WS-Policy files may be associated with entire web service or with individual methods of the web service.

In most cases you can use one of the provided WS-Policy files: Auth.xml, Sign.xml, and Encrypt.xml. For more advanced cases you can write your own WS-Policy file.

For a detailed information about message level security with WS-Policy files see Configuring Message-Level Security (Digital Signatures and Encryption) in the WebLogic Server documentation.

Upgrading WebLogic Workshop 8.1 WS-Security Files

For more information on updating WS-Security files to WS-Policy files, see Upgrading Annotations and Upgrading Security from from WS-Security to WS-Policy.

Related Topics

WebLogic Server documentation: Web Services Security

WebLogic Server documentation: Associating WS-Policy Files at Runtime Using the Administration Console

Role-based Control Access Security

For detailed information about role-based access security for Beehive controls see Control Security in the WebLogic Server documentation.

Also see the @Security annotation for reference information on role-based security for Beehive controls.

Related Topics

@Security Annotation

WebLogic Server Documentation

Control Security

Web Services Security

Associating WS-Policy Files at Runtime Using the Administration Console