Using webLock's Security Realm and Authorization Filters for Application-Level Security

September 2006
www.2ab.com

Introduction

Access management is a simple concept. Every business has information that needs to be protected from unauthorized disclosure. To protect information, companies define policies that govern who can access specific classes of business and/or personal information. For example, if a customer seeks proprietary contract information (such as their negotiated pricing discounts), they should probably have authorization to obtain this information, however, they should not be authorized to access the same information about another company's contract. Traditionally, these requests have been made through people who were responsible for the enforcing access policy related to each request. We expect someone in authority to be able to classify the request for contract information and make a decision regarding release to the requestor.

Access Management software has a simple goal. It allows the human who previously acted as a guardian of sensitive information to be removed from the process without loss of access control. This sounds simple, but most businesses are struggling with the implementation of access management as they integrate and extend their Java applications to provide access from the Web. This is because machines cannot classify information or make access decisions unless they are explicitly programmed with algorithms to accomplish this. When you take the responsibility for access decisions away from human beings (who are skilled at generalizations), it becomes necessary to insert software guards into your applications.

Web access management solutions, such as webLock, which provides a commercial implementation of the Java Authentication and Authorization Service (JAAS), provide scalable alternatives to the reference implementations delivered with Web/application servers and/or the costly prospect of embedding access control mechanisms and access policy into the Web application. They allow application software guards to leverage services that enable access policy to be modified, tested and deployed dynamically without application code changes. This enables your developers to concentrate on providing business services on the Web. Access management solutions efficiently enable high performance access controls in distributed environments while allowing centralized management of access policy.

webLock may be configured as part of the Web infrastructure or as the access management component of a Web-based business application. The focus of this paper is to provide you with an understanding of how to use the webLock security realm and authorization filters as part of the container-managed security infrastructure[1]. The ability to control this at the Web infrastructure level is appealing if you wish to provide a consistent framework for access control across multiple Web applications. The container-managed authentication model is often assumed to be the default by Web developers. For this reason, many Web developers will be comfortable with using a pre-configured realm for authentication. To protect an application using the webLock security realm, the realm is configured in your server.xml file at the appropriate level (e.g. HOST, CONTEXT, …). Tools for configuration and management of users, groups, roles and access policy for URLs are provided using webLock's graphical management tool. In this paper, we describe the steps necessary to use webLock to manage the complexity associated with software access management. We hope this paper will help you determine if webLock will meet the requirements you have for container-managed security for your Web application.

[1] If you wish to use webLock in your application with no changes to your server.xml file, other webLock whitepapers describe how to configure and deploy webLock as part of your Web application.

Application Security Often Requires Both Authentication and Authorization

Application security must address any security-related requirements not provided by the runtime security infrastructure. In the area of access management, any requirement to restrict a) the usage of application features or b) access to business and personal information is part of "application security." In this paper, we will focus on using authentication and authorization to provide the fine-grain access control requirements of "application security" in a Web environment.

First let's make sure we are clear on the difference between Authentication and Authorization.

- Authentication answers the question: "How do I know that you are who you say you are?" The goal of authentication is to securely determine who is making the request for access to information or some functionality of the Web application.
- Authorization answers the question: "Now that I know who you are, how do I know if you are allowed to access the information or application feature that you are requesting?" The goal of authorization is to protect business and personal information and sensitive application features from being used by people who legitimately have access to the application – but should only be allowed some subset of its information and/or functionality.

This paper will focus on how a Web infrastructure can use webLock to provide the authentication and authorization necessary for Web application-level security.

A Document Center as an Example Application

For the example application, we will focus on securing a Web site for a software company that wishes to provide an on-line area they call the "Document Center." In this section, we want to introduce you to the Document Center and the access control policy that webLock will enforce.

The Document Center includes a variety of information that the company makes available electronically to customers, potential customers and/or business partners. The company wishes to protect all of the information in the Document Center, however, different types of information are governed by different access policy. Authentication of the user (login) is all that is necessary for access to some of the information. That is, some of the information will be made available to anyone that has been provided a User ID. Information in this classification includes product technical specifications, frequently asked questions and whitepapers. The Document Center also has a "Download Center" whose access is restricted to customers and companies who are currently engaging in a product evaluation. The Download Center provides product binaries, evaluation licenses, demonstration programs and product manuals. Finally, the Document Center includes a "Customer Center" that contains information that should only be available to customers. Within the Customer Center, there are different classes of information as well. There are general documents that all customers may access, such as support briefings, product specific technical briefings and maintenance patches, and there are also documents that must be restricted to individuals who work for a specific company. For example, if Acme Corporation is a customer, there are documents that only Acme employees (as a group or by named individuals) can access. These documents typically pertain to a support issue that is being worked for a particular customer and/or information that is relevant to the business relationship (e.g. the support agreement).

A user is required to authenticate to enter the Document Center. From the home page of the center, the user can access the general documents or choose to navigate to the Download Center or the Customer Center. The authorization checks performed by webLock restrict access to the Download and Customer Centers appropriately. In the following section, we will look at how webLock is configured to protect the Document Center.

Authentication Using the WebAgentRealm

First let's look at how to configure webLock to require that a user log in before they are allowed to access the Document Center. To support authentication, webLock provides a pre-built Java Authentication and Authorization Service (JAAS) LoginModule which is named "WebAgentLoginModule" and is used by the WebAgentRealm. The realm ensures that the user is authenticated when protected URLs are requested. It also manages the login, credentials acquisition and logout phases of authentication.

The WebAgentRealm is configured in your tomcat **server.xml** file. The **server.xml** filter definition for the example application is shown below. It specifies a realm that will be used for authentication of the user when they try to access protected URLs (or request a login). The realm can be configured at any level; here we choose to place it at the Context level so that it will be the default for the "2ab" Web application. If placed at the "HOST" level, it would be the default for all Web applications within this container.

```
<Context path="/2ab" docBase="2ab">
   <Realm className="com.twoab.ilock.wl.tomcat.WebAgentRealm"
      propertiesFile="C:\jakarta-tomcat-5.0.28\webapps\2ab\WEB- INF\
                                                servletContainerManaged.properties"/>
</Context>
```

WebAgentRealm Definition (server.xml)

The servletContainerManaged.properties file referenced in the web.xml file provides configuration information such as name of the instance of the iLock Security Center that manages identity information for this application. It also defines the level of trace and/or auditing information that will be reported in the webLock log files.

Now that the security realm is defined, we must also configure the type of authentication for the protected Web pages (those URLs that will trigger the realm) and the name of the realm. The login configuration determines whether or not the application will use the default (BASIC) login form or an application provided form (FORM). For this example, we will use the BASIC authentication dialog provided by the container.

```
<login-config>
      <auth-method>BASIC</auth-method>
       <realm-name>WebAgentRealm</realm-name>
</login-config>
```

Login Configuration Options (web.xml)

The last step is to ensure that the User ID and Password are not passed in clear text when they are entered on the login form and to specify which URLs require authentication. To do this, the following is added to the web.xml file. This entry will ensure that all requests to a directory under the "/secure" URL path will be automatically redirected to a secure channel (i.e. https). The auth-constraint forces authentication for access to this area and allows all authenticated users to access the Document Center home page.

```
<security-constraint>
 <web-resource-collection>
   <web-resource-name>secure</web-resource-name>
   <url-pattern>/secure/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
      <role-name>*</role-name>
  </auth-constraint>
  <user-data-constraint>
   <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
 </security-constraint>
```

Ensuring Authentication is Required and https Will Be Used for the Secure Area of the Site

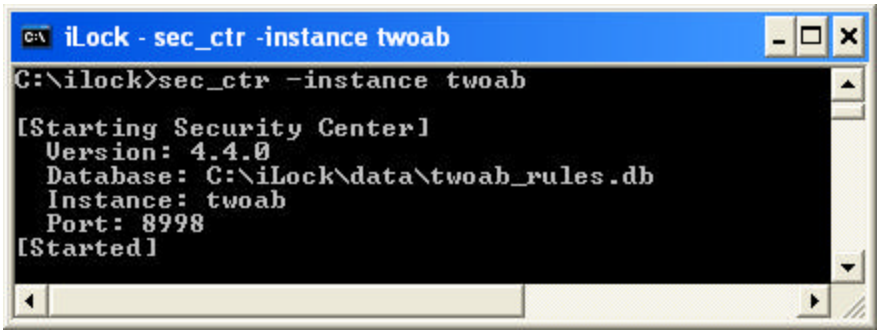
In webLock, identity information and access policies are managed by the iLock Security Center. webLock properties that define which Security Center to use are defined in the *servletContainerManaged.properties* file referenced by the *web.xml* filter definitions. The credentialFromPrincipal will ensure that webLock defined groups and roles are available for use by the authorization filter we define in the next session.

```
credentialFromPrincipal=true

# Connection to the Security Center
securityCenterHost=localhost
securityCenterPort=8998
securityCenterInstance=twoab
```

The Properties That Define the iLock Security Center to Use (weblock.properties)

We will see later how identity is administered using the iLock graphical tools. The Security Center is typically run as a service (although it may be deployed as an embedded database) so that it is accessible by multiple applications. Prior to accessing the Web application, we ensure that the appropriate Security Center instance is available for use by the Web application by starting the twoab instance of the Security Center.

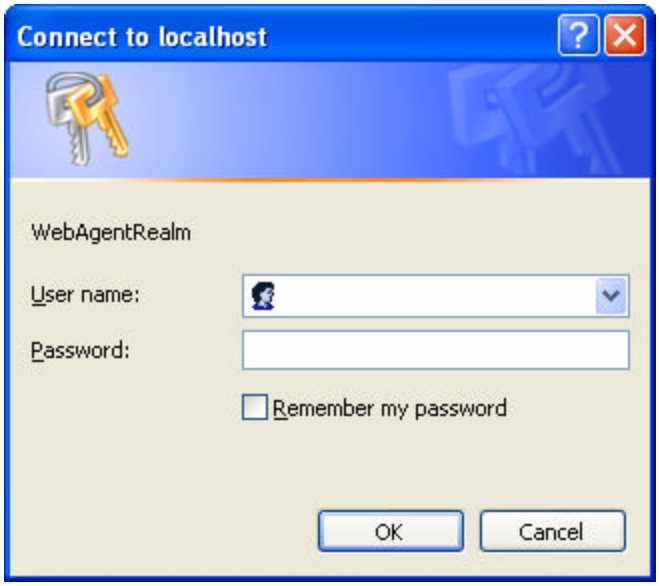


The iLock Security Center (instance = twoab) Running as a Service

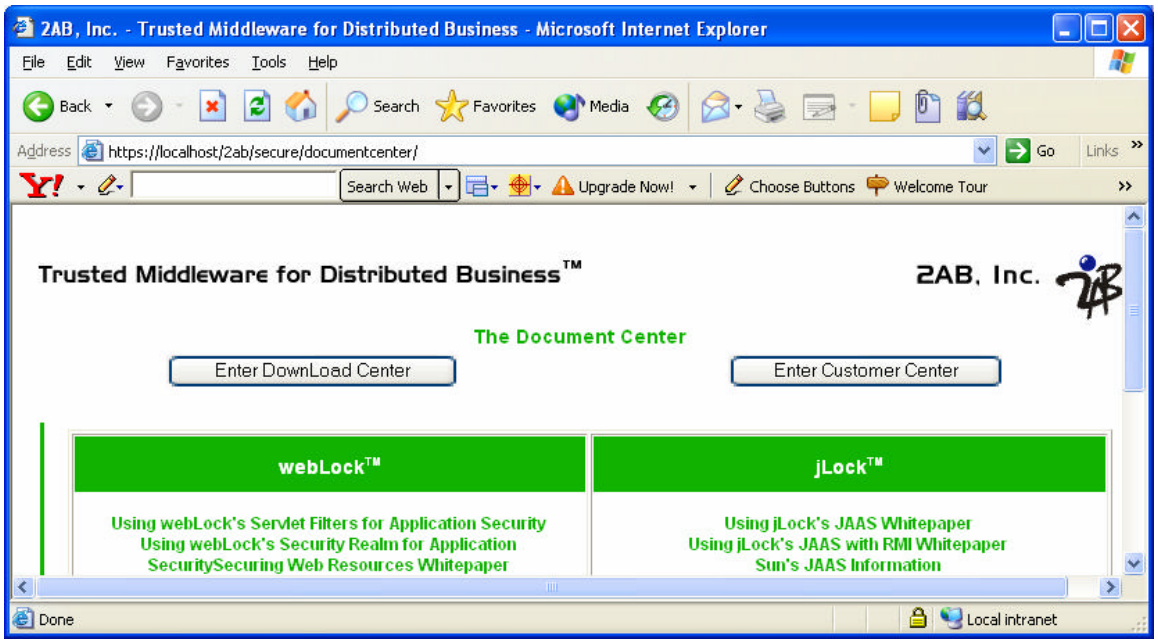
Now, when we make a request to the Document Center URL, the login form appears (note that although the requestor may have specified http, we are now using https). This is the result of the security-constraint we configured.

The form you see here is the default Login form provided by webLock – that is, the application did not have to create this form. It is possible to provide a custom form as part of the configuration if you wish.

If a valid User ID and Password are supplied, the request is re-directed to the originally requested site as shown below. Of course, invalid User IDs and/or passwords are forwarded to an error form.



The Default Tomcat Login Form

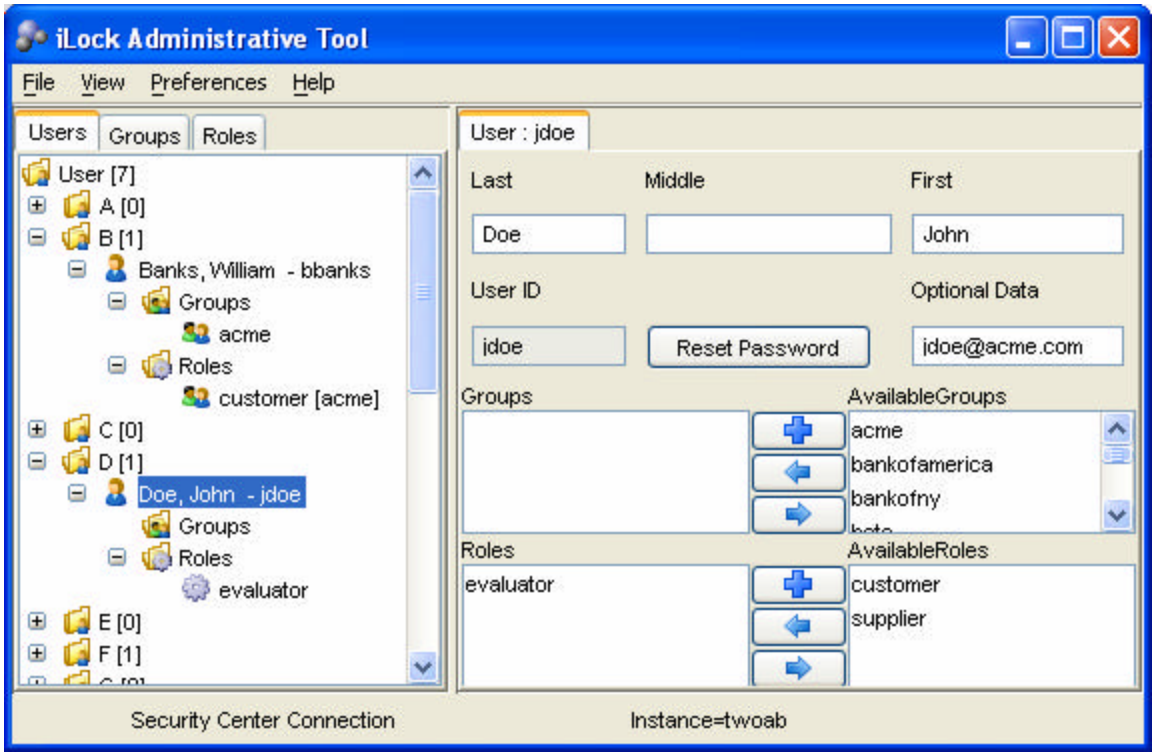


The Document Center Home Page

.

Managing Identity Information

Now let's look at how identity information is administered. For authentication, we obviously need to be able to define users and their passwords. We do this with the Identity Manager which is a view in the iLock Administrative tool shipped with webLock.

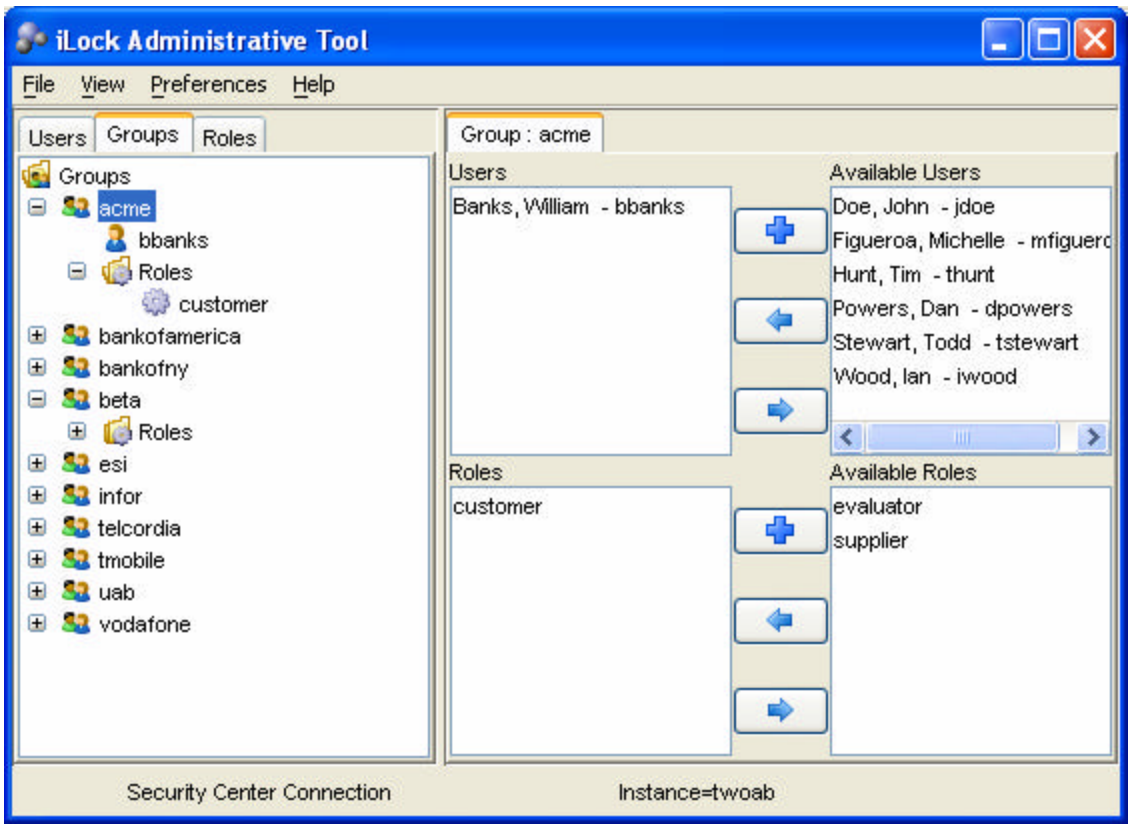


The iLock Identity Manager User Panel

This tool provides a graphical interface for defining users, groups and roles. Notice that in the lower right hand corner, the tool reports the Security Center instance that it is connected to. You may manage many different Security Centers; you may share Security Centers between Web applications, or you may use a different Security Center instance for each Web application.

The Identity Manager User Panel allows you to create users and their passwords. An administrator using the **Reset Password** button on the Identity Manager can also reset the passwords. Password format requirements can be set using the **Preferences** menu.

The Identity Manager also supports the configuration of the groups and roles that are associated with a user. These are the fundamental building blocks of scalable access policy, and so we will look at them before discussing authorization. Users can be assigned groups and/or roles by selecting an available group/role and moving it to the assigned groups and roles using the arrows.



The Identity Manager Group Panel

webLock also allows roles to be assigned to groups. In the sample application, we have decided to use "groups" to represent the company the user works for. For that reason, it is useful to assign the role "customer" to any group created to represent a company that is a customer. This means that when we associate a user with a group they will inherit the role of customer if it is on the group (vs. having to configure it for each user). The screen shot above shows the Group view of "acme." If you look back at the previous Identity Manager User view screen shot, you can see the user William Banks. Notice that this user has the group "acme" and it shows that he has inherited the role of "customer" from that group.

Authorization Using the WebAgentFilter

If we wanted to allow everyone with a valid User ID and Password access to all the documents in the Document Center, then the Document Center would be adequately secured with authentication (login) only. However, we know that some of the documents must be restricted based upon the credentials (userid, group and/or role) of the authenticated user. To do that, we need more than authentication – we need an authorization service. webLock provides this capability through the configuration of the WebAgentFilter.

To enable authorization checks, we configure the webLock provided filter named WebAgentFilter to perform authorization checks. The configuration for this authorization filter is done in the web.xml file [2].

```
<filter>
  <filter-name>AccessAuthorizationFilter</filter-name>
  <filter-class>com.twoab.ilock.wl.servletapi.WebAgentFilter</filter-class>
  <init-param>
    <param-name>propertiesFile</param-name>
    <param-value>
       [servletContainerInstallDir]/webapps/2ab/WEBINF/servletContainerManaged.properties
    </param-value>   </init-param>
</filter>                              ……
```

WebAgentFilter Defined for Authorization (web.xml)

Now that the filter is defined, we must also define the filter mappings for the protected Web pages (those URLs that will trigger the authorization filter).

```
<filter-mapping>
   <filter-name>AccessAuthorizationFilter</filter-name>
   <url-pattern>/secure/documentcenter/*</url-pattern>
</filter-mapping>
```

WebAgentFilter Filter Mapping for Authorization (web.xml)

After configuring the authorization filter, if you access the Document Center with the log level set to DEBUG or TRACE, you will see the trace details of the authentication and authorization checks done by the security realm and authorization filter. Below are some key excerpts from that log.

```
2006-10-05 11:00:08 INFO - WebAgentLoginModule - 13842387 - authenticate? - jdoe - true - null - null - null - null

2006-10-05 11:00:08 INFO - WebAgentRealm - /2ab - authenticate? - jdoe - true - null - null - null - null

2006-10-05 11:00:08 DEBUG - WebAgentRealm - /2ab - authenticate() end jdoe GenericPrincipal[jdoe()]

…

2006-10-05 11:00:08 StandardContext[/2ab]INFO - WebAgentFilter - AccessAuthorizationFilter - authorize? - jdoe - true - https://localhost:443/2ab/secure/documentcenter/index.htm? - GET - 127.0.0.1 - 127.0.0.1
```

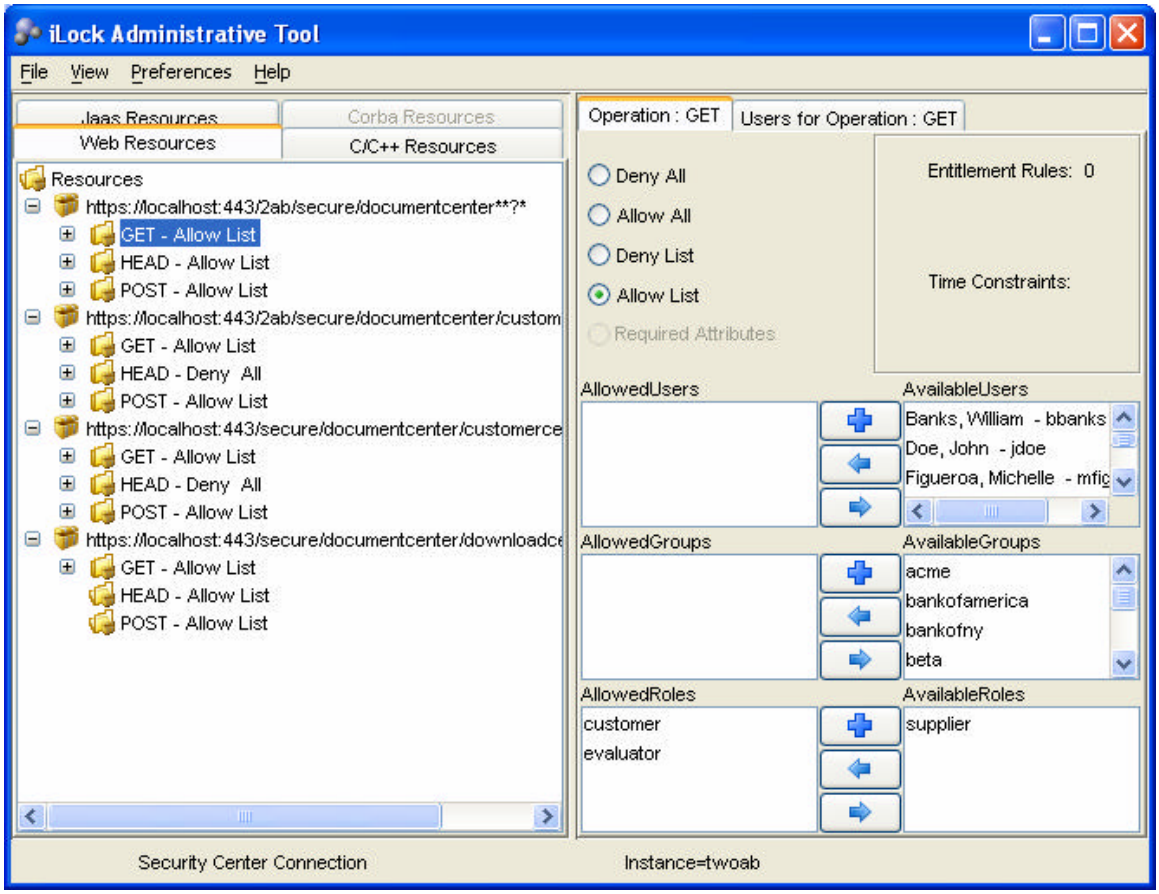
Entries from the log showing authentication and authorization checks

Now let's look at how access policy is administered in webLock.

[2] For tomcat containers, you may use the WebAgentValve and configure the value in the server.xml file. Using a filter (i.e. WebAgentFilter) is a more portable solution and is therefore chosen for this example.

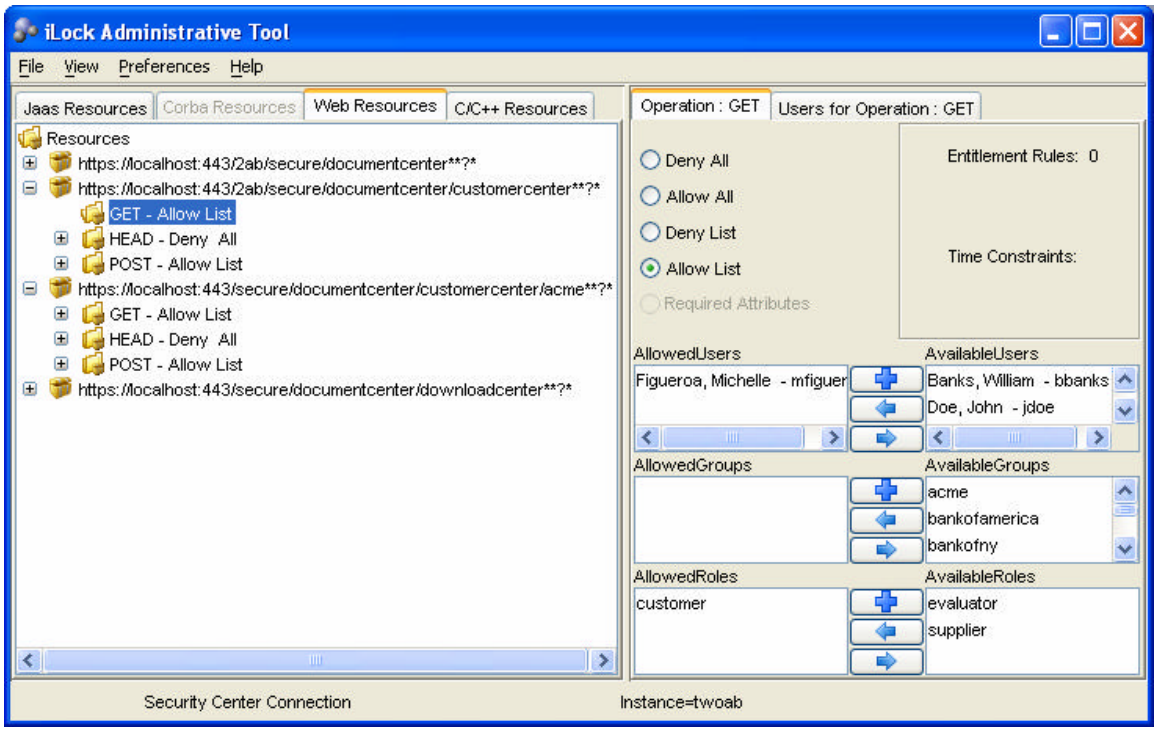
Managing Access Policy

At this point, webLock will perform access control checks for any URL (document or sub-directory) that is in the Document Center section of the Web site. To control who can access specific documents or directories, you must define access policy for the Document Center URL and any URLs that have unique access policy within the Document Center. We do this with the Resource Manager, which is a view in the same webLock Administrative tool we used to configure identity information. For this demonstration, we wish to restrict the Document Center to users who have the role "customer" or the role "evaluator." We define the http operations (GET, HEAD and POST) and define our access policy for each operation. Below you see the screen shot for the GET operation.



Defining the Policy for the GET Operation on the Document Center URL

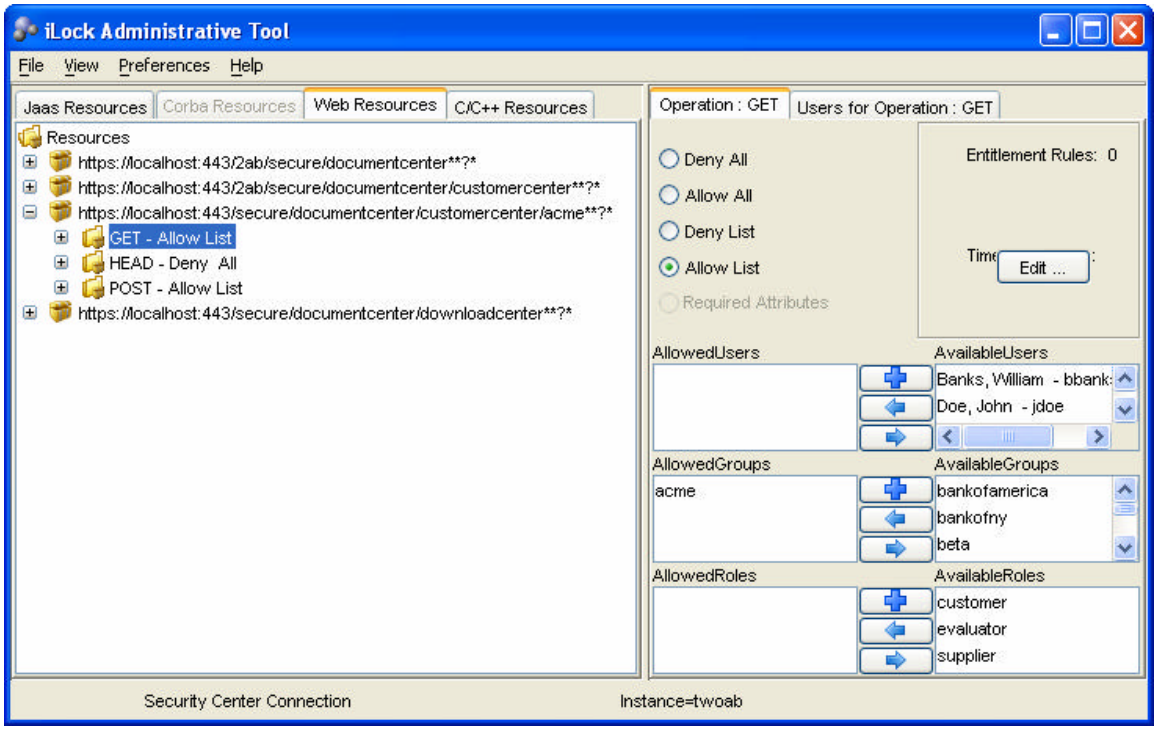
We also have a section in our Document Center that we wish to restrict to customers only (and perhaps a few individuals we wish to grant access to). We call this the "Customer Center." Below we show the access policy for the Customer Center section of the Document Center. Notice that to access the URL; you must have the role "customer" or the User ID "mfigueroa."



’

Access Policy for the Customer Center.

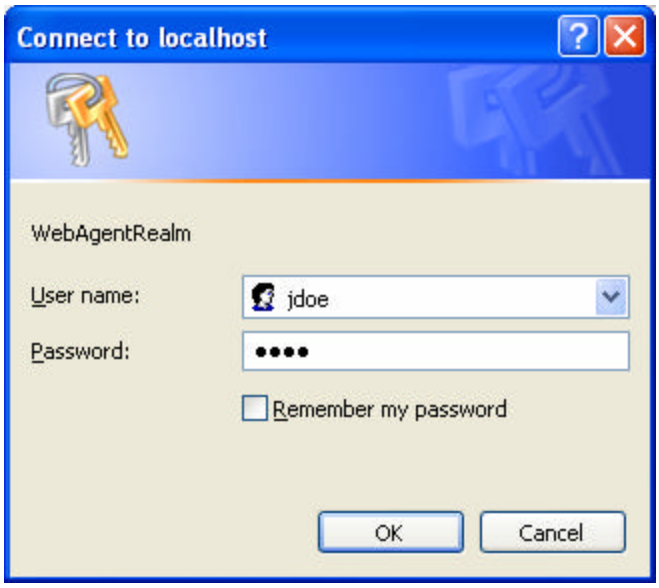
To create customer specific areas, we create subdirectories within the Customer Center for each customer and restrict those areas to users who are in the “group” for that company. For example, to provide an area for the Acme Corporation, we create the subdirectory named “acme” and define an access policy for that URL that allows only members of the group “acme.”



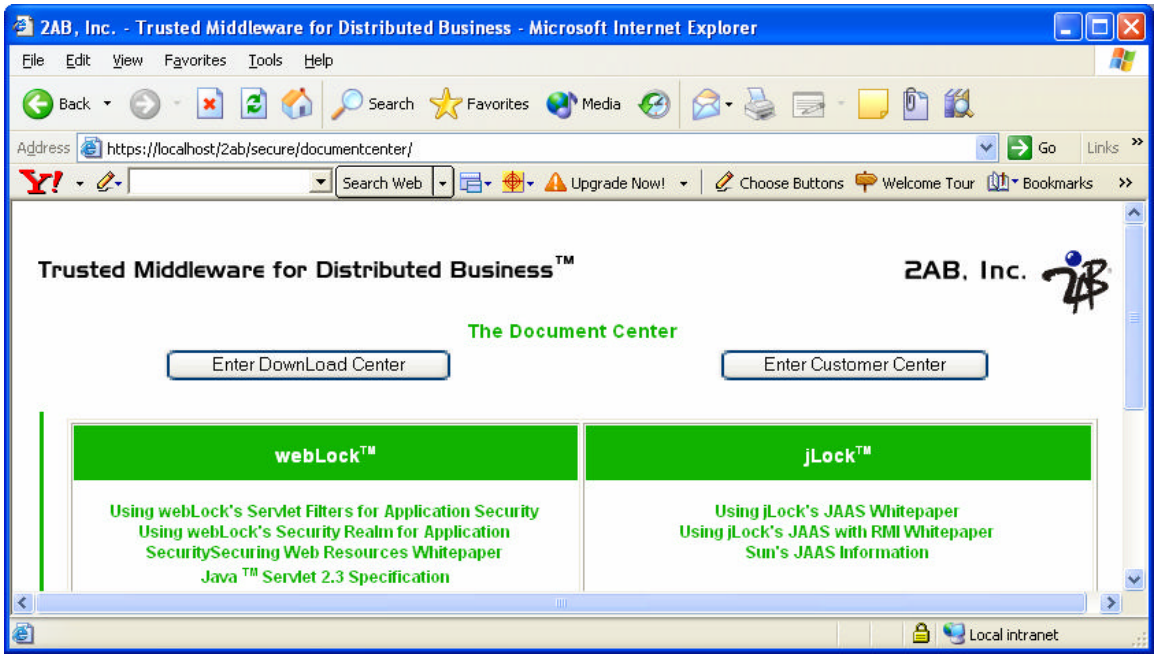
Access Policy for Acme Corporation’s Restricted Area of the Customer Center.

Exploring the webLock Protected Document Center

Now that we have defined our access policy, let's look at what happens when we use different User IDs to log in. First, we try to access the Document Center and get the Login panel shown to the right (access to the Document Center requires authentication).

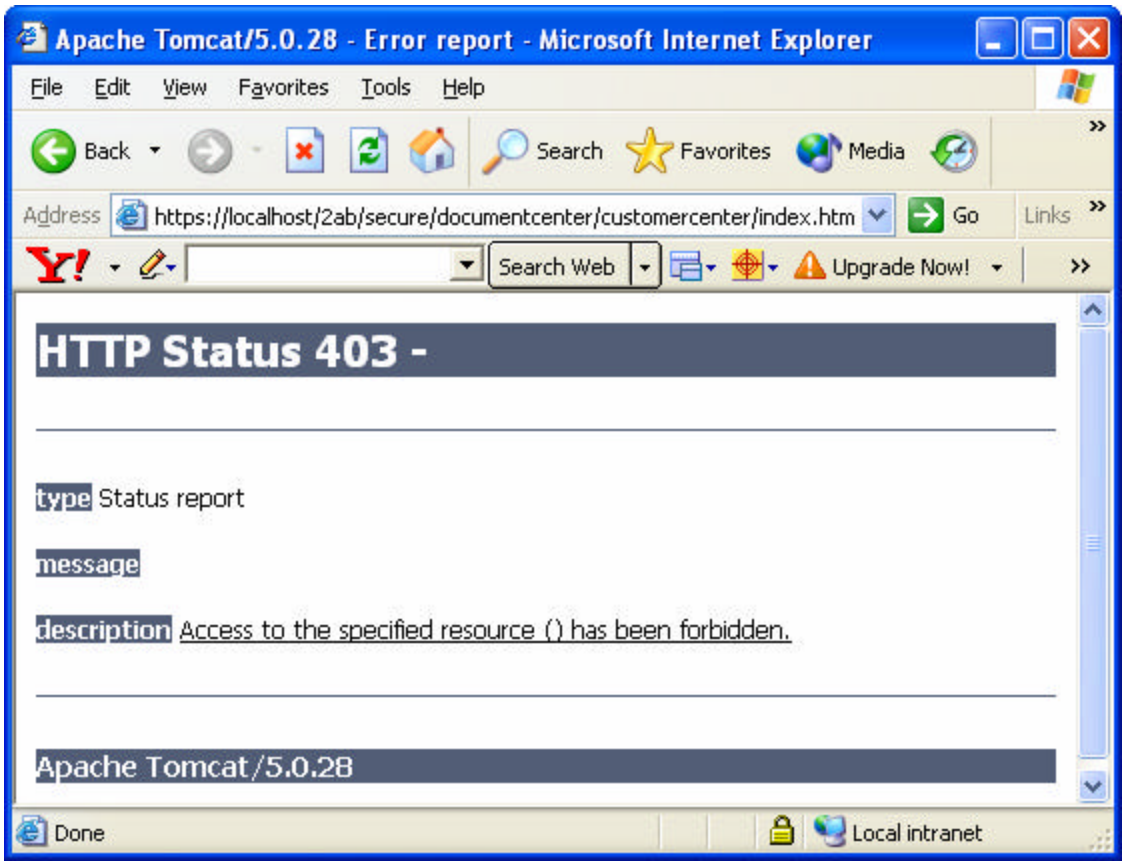


First, let's login as jdoe. As shown in the Identity Manager screen shot, John Doe (user ID: jdoe) has the role "evaluator," but does not have the role "customer." Based on the access policy we outlined for our Document Center, John Doe should be allowed to enter the Document Center and the Download Center; however, access should be denied if he tries to enter the Customer Center. That is exactly what happens. After authentication, the Document Center home page is displayed.



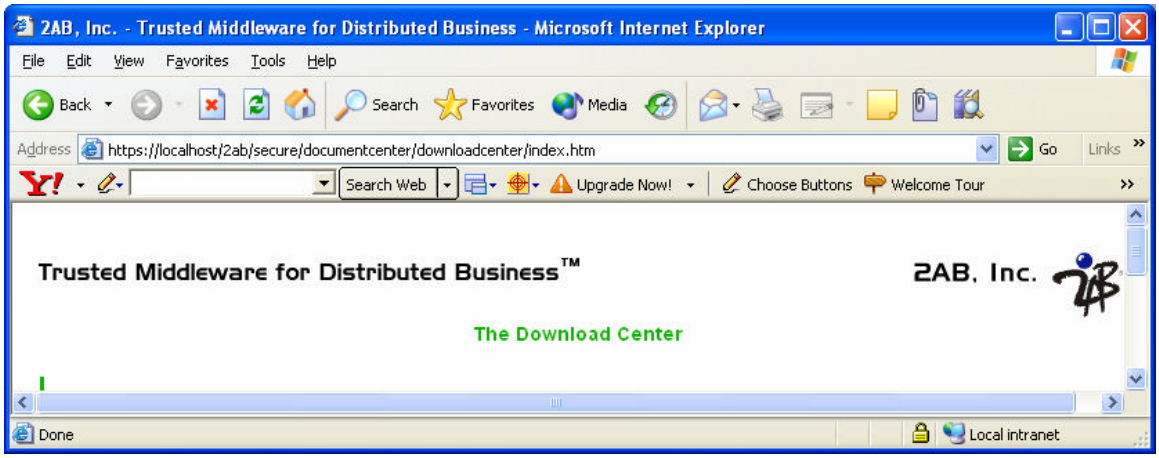
The Document Center Home Page

Now that John Doe is logged in, he tries to access the "Customer Center" by selecting the button on the right of the home page. Access is denied as shown below.



John Doe is Denied Access to the Customer Center

John Doe clicks the "Back" arrow and now tries to enter the Download Center using this button on the Document Center home page. This request succeeds as shown below:



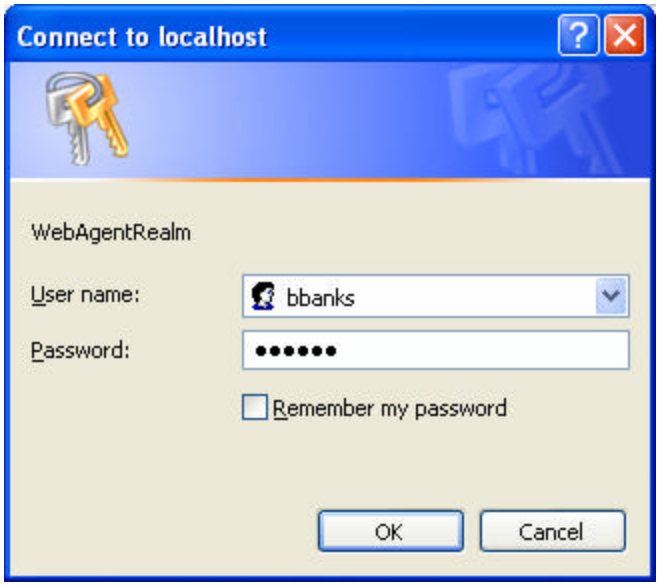
The Download Center Home Page

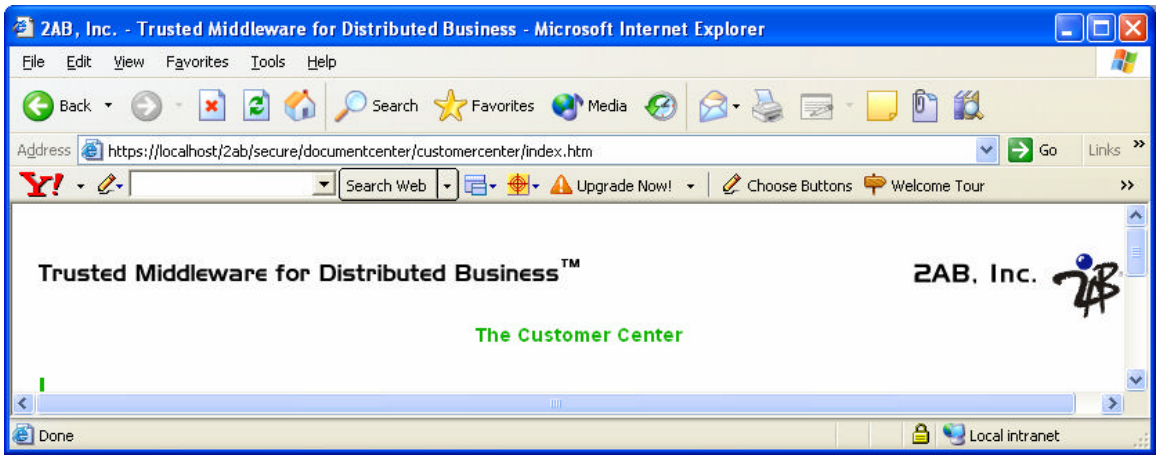
The request for access to the Download Center was authorized because the access policy for the URL allows anyone with the role of "evaluator" or "customer" to have access. Although John Doe is not a customer, he does have the role evaluator and is therefore granted access to the Download Center.

Now let's login as William Banks (userid bbanks).

We looked at the identity configuration for William Banks earlier. He is in the group "acme" which has the role "customer" assigned to it. For that reason, the he has the role "customer" which allows him access to the Customer Center as shown below.

If you log in as Michelle Figueroa you will also be granted access to the Customer Center as mfigueroa is one of the User IDs which has been directly authorized for access to the Customer Center. That is, access has been configured for anyone who has the role "customer" or the userid "mfigueroa."





The Customer Center is only available for users with the role "customer" or the userid "mfigueroa"

Summary

webLock supports a service-oriented architectural approach to providing application-level security for Web applications. webLock enable you to provide authentication and authorization access control checks within your Web application without writing any software. webLock provides a security realm and authorization filters that make it simple to deploy and manage container managed access controls. Your Web developers simply configure the pre-built security realm and servlet authorization filter (which act as Guards) and the Web access policy that you wish to implement.

We hope that this paper has helped you understand how webLock can be leveraged for application-level security in a Web environment. If you would like to evaluate webLock in your environment, please contact sales@2ab.com