database solutions	CIRCABC – Conceptual Problem
total pages: 6 date: 28.03.2009 revision: 1.2	**Cannot open or edit Documents with Security Ranking higher than PUBLIC**

Table of Content

1 Non-technical Problem Description

If you try to open a document which has a security ranking higher than PUBLIC (i.e. INTERNAL), you get a error message from the browser that the connection failed.

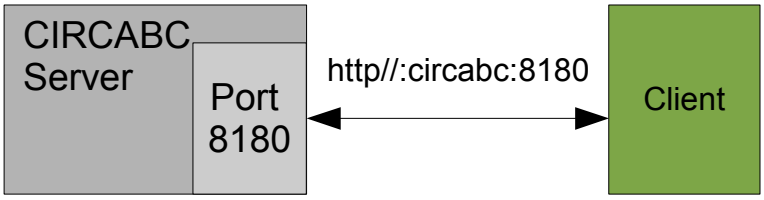
2 Technical Problem Description

If the security ranking of the document is higher than PUBLIC, the connection will be redirected to a SSL secured connection via “https”. In the URL where the error message from the browser appears you will see the “https” at the beginning of the URL in the browser. The port in this URL will be the same as the port for the standard “http” connection.

The problem will occur if you are running tomcat not with the standard web configuration (http: port 80, https: port 443). This will be the case if you have used the standard installation instructions.

To illustrate the conceptional flaw in this we will consider some pictures of a possible system-landscape.

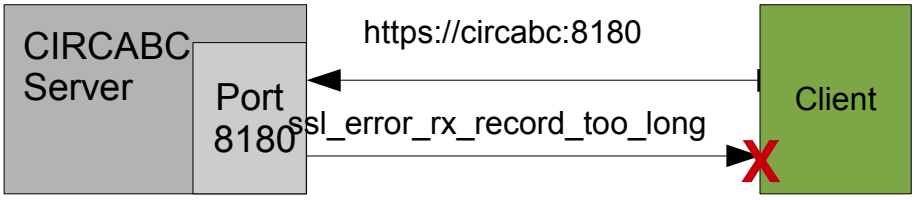
In picture 1 we have a simple client which just talks http to the default tomcat-port.



Picture 1. good-case http communcation

As long as non“security-relevant” documents are downloaded everything works out of the box.

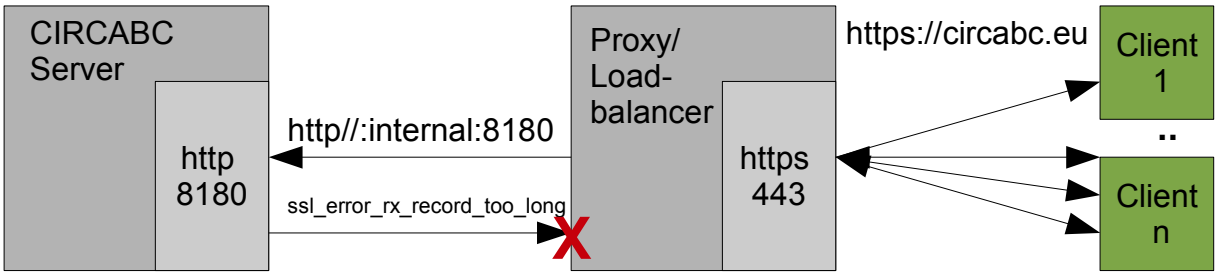
If one tries to download a “security-relevant” document the scenario shown in picture 2 will occur.



Picture 2.: Trying to get a secure document with the default CIRCABC installation throws a ssl_error_rx_record_too_long error

The client tries to establish a https-connection over http which will result in a ssl_error_rx_record_too_long error (technical details: http://www.mozilla.org/projects/security/pki/nss/ref/ssl/sslerr.html).

Let's go a step further and assume that we are in a sophisticated environment and we have a real need for proxy and/or loadbalancing technics.

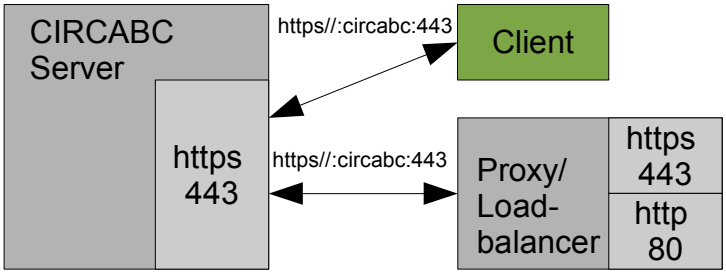


Picture 3: Unable to build up a scalable installation with the help of server caches

As we can see in picture 3 the simple administrative approach “just-use-it-out-of-the-box” and handle all the security and comfort-stuff with network-technology would fail.

Even if every communication to the internet is secured with https the circabc-application would think that it is not safe and would reject the proxy's request (if they share the same host).

A possible solution to this feature is to only allow https-connections to the circabc-server. In the case of the Proxy the security-feature can then be disabled.



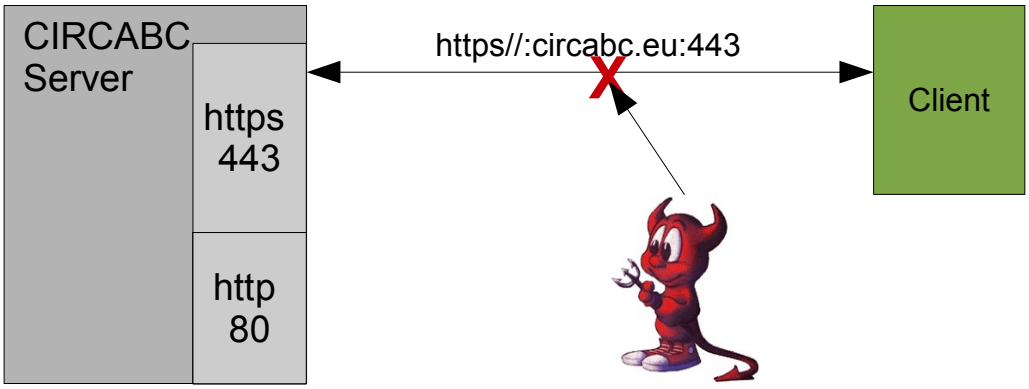
Picture 4: Allow only https connection in general.

As we can see in picture 4 the application now thinks everything is secure but the proxy can deliver the information via http. Furthermore there is a slight drawback in terms of performance because we are forced to setup a internal system via https.

All communication via https is the good scenario, but to emphasize the conceptional flaw of this idea in general, we look at the problem from a different point of view:

What was the intention of this feature?

Of course, the answer is to secure critical data. But against what? Sure, the evil and well-known man-in-the-middle-wiretapper. Let's consider picture 5:



Picture 5: CIRCABC is secured against all evil ....

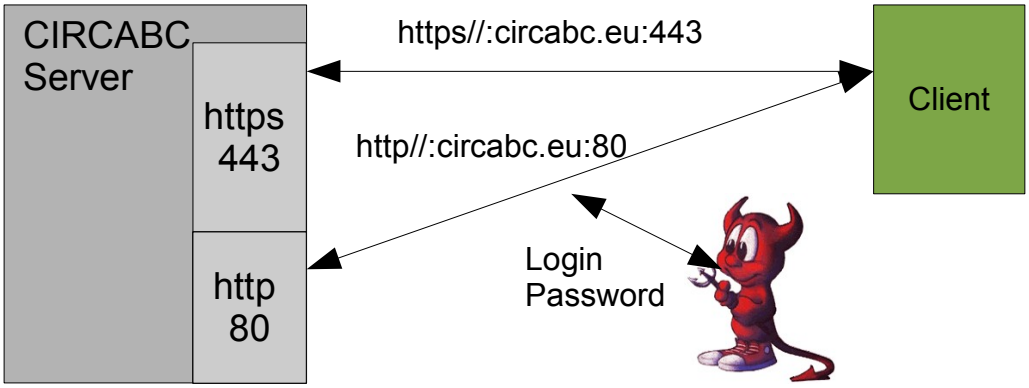
Yes, here everything is safe.. The software checks if this connection is secure and we can send critical data via the internet.

Nope.

A security chain is only as secure as the weakest link in the chain.

Therefore the check is far too late and as shown in picture 4 we can even trick the software into thinking that the connection is secure. To put it in a nutshell: Network security checks are not in the scope of the application.

But back to the idea that the checks are far too late:



Picture 6: Man-in-the-middle attack scenario even IF this is considered secure ...

As picture 6 clearly describes the user must have entered his login and password some time before and the software doesn't check if the login and password are transferred securely.

Now one could argue to secure everything with https and we would say yes to this idea, but please leave the decision at which point only https is allowed as a connection up to the system administrator who likes some liberty in his decisions, because he needs the concept of a proxy or load balancer as shown in picture 3.

3 Related Versions

CIRCABC 2.0 is affected by this problem. Prior versions may also be affected by this problem.

4 Reason

If the tomcat server is configured on other ports than the standard ports for “http” and “https”, this problem will occur.

The URL to the document will be generated internally always by the alfresco methods, which always use a insecure “http” connection. If the document has a security ranking higher than PUBLIC, the CIRCABC extension will exchange the “http” in the generated URL to “https”.

But if the generated URL contains a port which is used for none secure “http” access, the port will be kept in the modified URL. So you get the problem that the “https” URL contains the standard “http” port. This will not work because the tomcat server can not handle different protocols on one port. You need another port for another protocol.

If you are running tomcat with the standard web configuration (http on port 80 and https on port 443), no port declaration in the URL is mandatory and the port is selected by the protocol.

In the source files in class “eu.cec.digit.circabc.repo.template.DirectAccessUrlMethod” at line 49 you find the root of the problem.

```
if (securityRanking != null && !securityRanking.equalsIgnoreCase("PUBLIC"))
{
result = result.replace("http:", "https:");
}
```

Before this lines the variable “result” contains the none secure “http” URL. In this lines the “http” is replaced by “https”.

If your tomcat server does not need a port declaration in the URL because it is running on port 80 (http) and port 443 (https), everything works fine. But if not, you will have your http port declared in the URL, and the port is not replaced in this method. So your secure URL directs to a wrong port on your tomcat server. (The “http” port of your tomcat server.)

5 Solution to the Problem

These are some suggestions how to solve the problem. None of these solutions have been tested yet. Source code modifications will not be tested until the release of CIRCABC 3.0.

5.1 Reconfigure Tomcat

Reconfigure the tomcat server to use the standard ports for “http” and “https”.

5.2 Always use Https

Configure your tomcat server to use the secure “https” protocol for the CIRCABC deployment.

In the configuration file “circa-settings.properties” set the “web.root.url” to the “https” URL.

5.3 Disable Https

To disable https completely you have to modify the source code.

Just remove the check for the security ranking in the class "eu.cec.digit.circabc.repo.template.DirectAccessUrlMethod" at line 49.

Then the security handling can be done completely via a reverse proxy.

5.4 Define the Https URL

Modify the source code of CIRCABC so that it uses a configuration entry for the https URL.

Define a configuration key "web.root.url.ssl" in the file "circa-settings.properties" which contains the https url for your CIRCABC installation. This key should be used when generating a https URL. To fully apply this source code modification you have to check the following classes if they need modifications.

- eu.cec.digit.circabc.repo.template.DirectAccessUrlMethod
- eu.cec.digit.circabc.web.WebClientHelper
- eu.cec.digit.circabc.web.servlet.ExternalAccessServlet

6 Questions

If you have any questions about this document, please contact CIRCA support at ni database solutions. Please refer to this document in the subject.

E-mail address: support@circa-support.eu

7 Disclaimer

You use all information in this document at your own risk. ni database solutions does not guarantee the completeness or correctness of the information in this document. ni database solutions may modify this document at any time without any notification. ni database solutions will only take responsibility for losses if it is required by European and/or German laws.