CAF For Developers Application Note

Introduction

This application note describes how federated authentication through the Canadian Access Federation (CAF) can be incorporated into a web service. Software developers and architects are the primary audience for this app note.

This introductory note describes incorporating CAF into an environment in which the user accesses target web services through a web browser. The cases where there is no web browser involved, such as a research platform at one institution accessing CAF-enabled web services from another institution, are more complex and will be described in subsequent application notes.

What is the Canadian Access Federation (CAF)?

The Canadian Access Federation provides trusted authentication for users at universities and colleges across Canada. Managed by CANARIE, CAF was originally created to allow members of the academic community within Canada to access web-based resources at institutions other than their home institution. You can likewise make use of CAF to control access to your web service.

Terminology used in this document

The following terms are used in this document. These are all standard terms in the federated authentication field.

- User Agent – software that acts on behalf of a user – typically a browser, but could also be an application or a web service.
- Service Provider (SP) – a web application that provides a service to the User Agent. Specifically, this would be the service that you want to enable for CAF authentication.
- Identity Provider (IdP) – a service that verifies (authenticates) the identity of a user. This capability is typically implemented by the user's home institution.
- User – a person whose identify is known to CAF

What is Federated Authentication?

Federated Authentication is a mechanism where a user can use one set of credentials, typically from the user's home institution, to gain access to resources and services at other institutions.

For example, suppose you are a researcher at the University of Victoria and you need access to certain online resources at the University of New Brunswick. Without federated authentication,

you would have to arrange for a separate user account with its own name and password to be set up with the University of New Brunswick IT department.

This isn't too bad if only two institutions are involved, but what if you need access to resources at 5 or even 10 other universities? To complicate matters, suppose you retire or change jobs. Your home institution will no doubt be aware of this and will disable your accounts accordingly. However, communicating this fact to other institutions at which you have credentials requires complex agreements and user tracking.

With federated authentication, multiple institutions participate in a "trust fabric" where each participant trusts that the others will authenticate their local users properly. This trust is established through guidelines to which all participating institutions must agree and by public disclosure of each participant institution's policies on user accounts and credential management.

So in our example above, if both the University of Victoria and the University of New Brunswick are members of a federated identity trust fabric (ie. CAF), when you attempt to access an online resource at the University of New Brunswick, you are authenticated through the University of Victoria, your home institution. The University of New Brunswick then trusts that you are who you say you are, based on the trust fabric.

Authentication versus Authorization

Note that the purpose of federated authentication is authentication, not authorization. In our example above, the trust fabric tells the University of New Brunswick that you are affiliated with the University of Victoria – that you are who you say you are - but you would not likely be granted unrestricted access to all University of New Brunswick online services. To help determine what you can and cannot access, your home institution will release a number of user "attributes" to the institution from which you are requesting access.

Attributes could include your name, email address, job title, department, job classification and so on. It is entirely up to your home institution to decide which attributes to release. Similarly, it is up to the institutions providing services to determine which attributes they require for authorization. If there is no agreement between the institutions, you will not be granted access. Assuming that the user's home institution does release the appropriate attributes, your service can use these attributes to make authorization decisions.

Access Denied

Let's consider some situations where federated authentication can prevent access to certain online resources. In our example if you retire, the University of Victoria would typically revoke your user credentials. Subsequent attempts to access a web service at the University of New Brunswick would result in you being redirected to the University of Victoria for authentication. Since your credentials have been rescinded, the authentication would be unsuccessful and access to the University of New Brunswick service would be denied.

Suppose there is a web service at the University of New Brunswick that allows online control of a piece of equipment. Let's assume that this equipment is both expensive and complex and requires that users must have a valid training certificate in order to use it. If you attempt to access the service that runs this piece of equipment, you will be redirected to the University of Victoria for authentication. Assuming that you supply the correct credentials, the authentication approval will be sent from the University of Victoria to the University of New Brunswick's service. This approval could contain a list of attributes describing you. If one such attribute indicates that you have the appropriate training, the University of New Brunswick service will allow you to access the equipment. If that attribute indicates you do not have up-to-date training, or is missing all together, the service could use this information to deny access.

Benefits of using CAF

So, as a provider of web services, what advantages would you get from using CAF for authentication? Integrating CAF support into your web service or portal allows you to avoid implementing and maintaining a complex user authentication management. When a user attempts to access your service, the user's home institution is entirely responsible for determining whether or not that user is who they say they are. Your service receives a token from the IdP confirming that the user is authenticated. At this point, you may choose to further authorize the user for specific operations according to user attributes (name, home institution, title, etc.).

CAF is based on a freely available open source package called Shibboleth (www.shibboleth.net). For communicating authentication information between software components and platforms, Shibboleth uses Security Assertion Markup Language (SAML).

When you implement CAF support for your service through the Apache web server with the Shibboleth plugin, you can get a complete solution, including a useful level of attribute filtering, without having to write a single line of code. This not only speeds up development of new services, but also allows you to add CAF support to existing services without having to make any design changes.

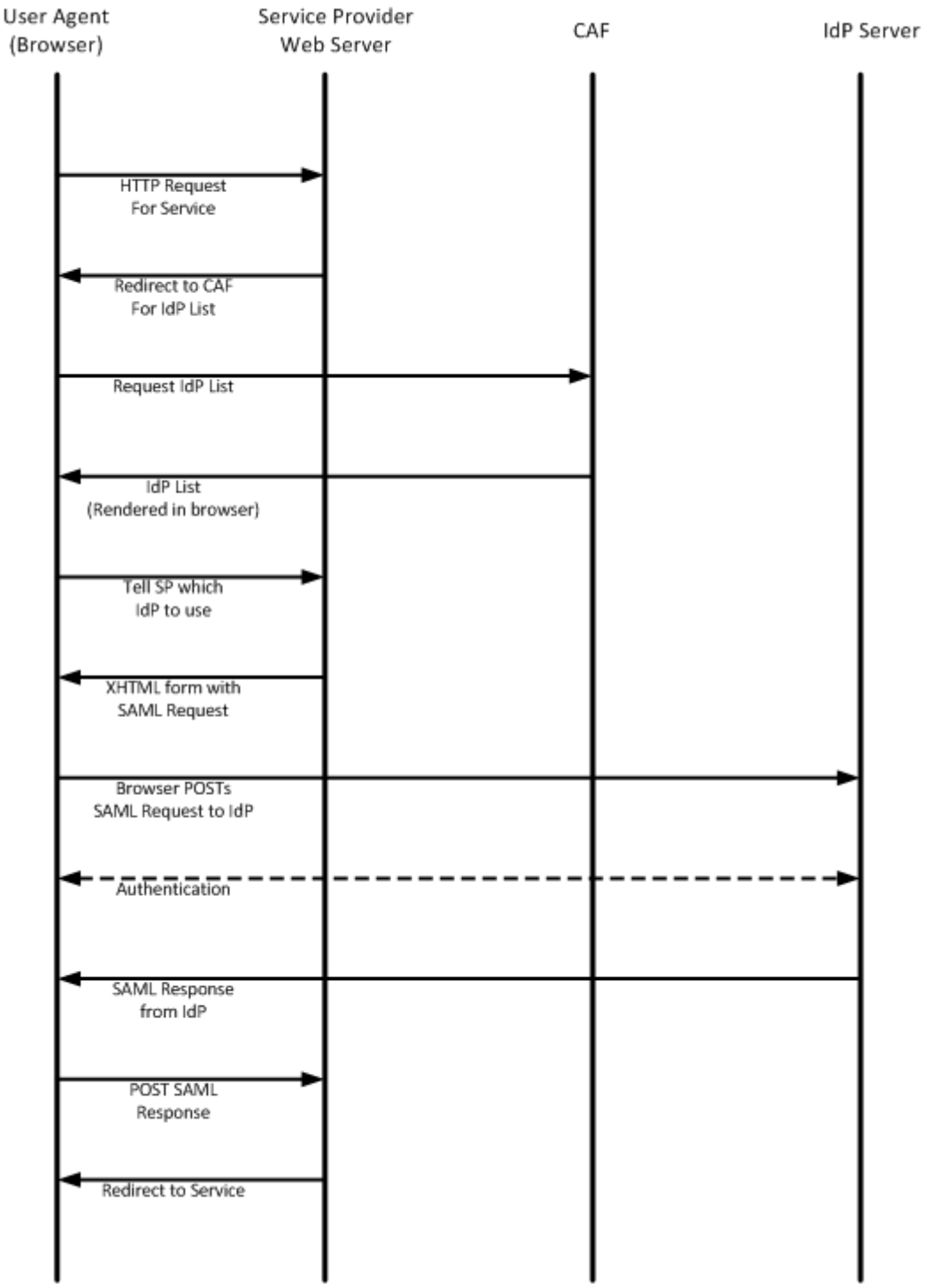
Do I need to join CAF as a service provider?

Yes. As CAF implements a trust fabric, all institutions making use of CAF must submit a "Participation Application" and provide a "Trust Assertion Document" which will be shared with all other CAF participants. Additional information and templates for these documents can be found at http://www.canarie.ca/en/caf/join. For institutions that do not implement identity provider (IdP) services, there is no fee for participating in CAF.

How Does CAF Work?

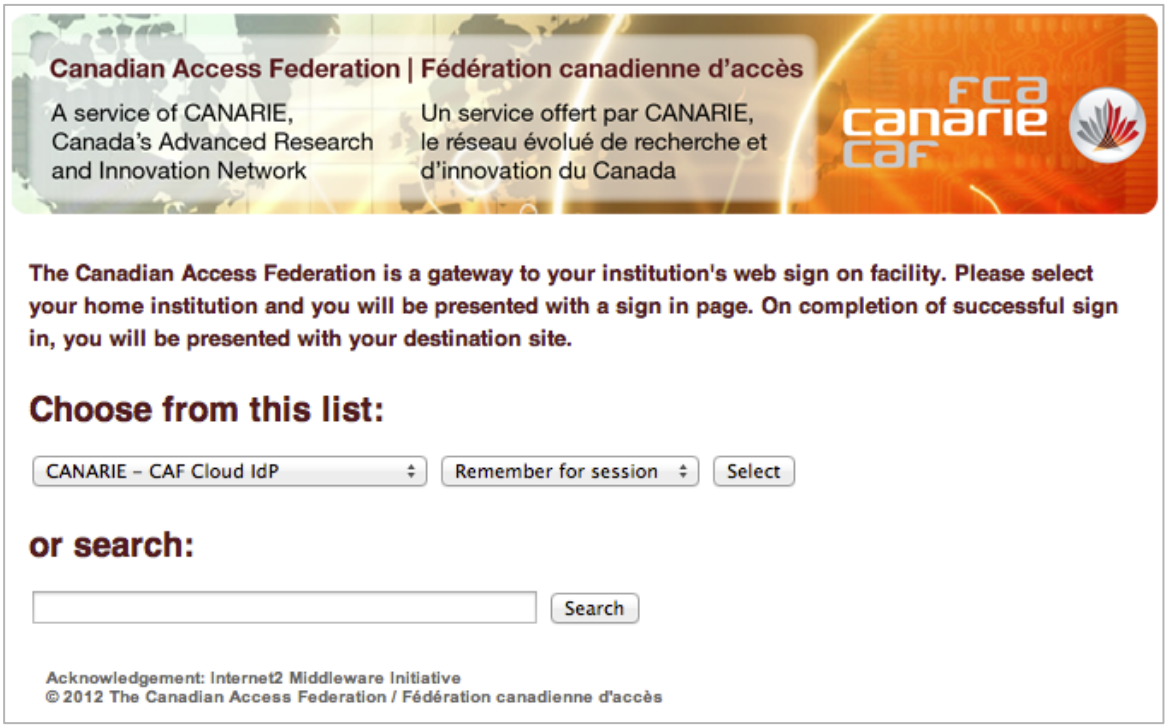
This section provides a brief overview of the protocol exchanges that happen when a browser requests access to a CAF-enabled (ie.Shibboleth-enabled) service. The following diagram

shows the flow of information between the various software components during a CAF authentication.

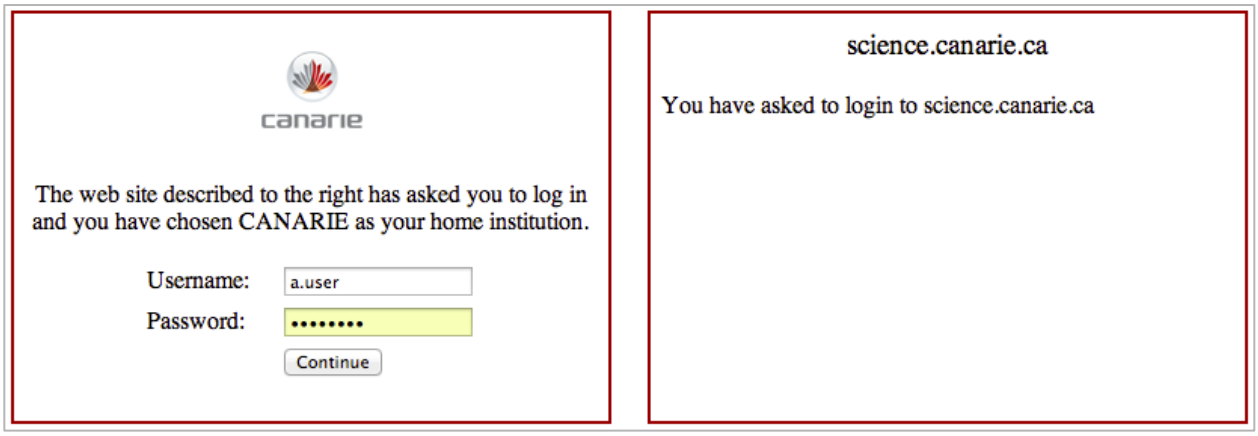


To explain the protocol flow during a CAF authentication, we follow the arrows in the above diagram in order from top to bottom. Note that all communications are encrypted.

1. Via a browser, a user makes a request to access a CAF-enabled service by navigating to this service's URL.
2. Since the service is CAF enabled and there is not an existing session for this user, the service's Shibboleth component (managed by Apache) redirects the browser to CAF for an IdP list. The browser is redirected to the URL of this list (pictured below).



3. The user selects their home institution from the list and then clicks on the "Select" button.
4. A message is sent to the service's Shibboleth component indicating which IdP to use for authentication.
5. The service's Shibboleth component returns a SAML request to the browser, which is then forwarded via an HTTP POST to the appropriate IdP.
6. The browser is then redirected to the IdP login page. CANARIE's IdP login page is shown below. Yours will, of course, look different.



7. The user provides their home institution credentials and logs on. The mechanism used here is IdP dependent and is outside the scope of Shibboleth.
8. A SAML response is sent from the IdP to the browser.
9. The browser sends this response to the service's Shibboleth component using an HTTP POST operation.
10. Assuming the response contains the attributes expected by the service and that these attributes are valid, the Shibboleth component of the service redirects the browser back to the target resource.

Remember, the protocol exchange described above is all done via the browser, the IdP, CAF and the Shibboleth installation at the service provider site. You do not need to write even a single line of code for this to work.

A Note on Architecture

CANARIE recommends that the core of your CAF-enabled service be implemented with Apache and Shibboleth. The Shibboleth distribution includes an Apache plugin as well as a stand-alone service. Both are required to interact with CAF.

Suppose you are using a servlet container that can also act as a web server, such as Tomcat or JBoss. Would you still use Apache in this case? Absolutely, because:

1. The Shibboleth support for Apache is complete and mature. This is not the case for Tomcat or JBoss as of this writing. Implementing Shibboleth in Tomcat would require that you do significant development.
2. This approach isolates your authentication mechanism from your service implementation. Maybe you're using Tomcat now, but if you later decide to implement some services using a different technology, like simple PHP scripting, a Shibboleth implementation within a Java application server would be problematic.

3. To continue with the previous point, a single Apache server with Shibboleth support can provide CAF authentication to a number of services independent of the service implementation technology, even if these services reside on different physical hardware.

Creating a CAF-Enabled Web Service

In this section, we'll describe step-by-step how to set up a web service that uses CAF for authentication. We'll set up this service using CANARIE's DAIR platform under the CentOS 6.0 Linux distribution.

The Approach

As the software components required to implement federated authentication are complex with many configuration options, we'll describe a multi-step approach in which you'll start by configuring your service to use federated authentication with a publicly available test IdP. This will allow you to test the basic setup without requiring the involvement of your IT department. This test IdP (testshib.org) has pre-defined test users that you can use immediately without the need to register or create an account. The IdP logs are also accessible in case something goes wrong.

Once that is working, you'll switch to using your institution's IdP directly. Even when CAF support is enabled, you'll still be authenticating through your institution's IdP, so it's worthwhile verifying that this works in a simpler configuration.

Finally, you'll add CAF functionality so that you can choose your insitution's IdP from the CAF IdP list.

To assist in troubleshooting, we recommend that the steps below be carried out using the Firefox browser with the "SAML tracer" plugin installed. This plugin is available at:

https://addons.mozilla.org/en-US/firefox/addon/saml-tracer/

When "SAML tracer" is activated by selecting "SAML tracer" from the Firefox tools menu, all authentication-related messages transmitted or received by the browser are displayed in decoded format in a separate window.

Step 1 – Setting up a test service

In this first step, we'll create a virtual machine on DAIR and install the additional packages required to support Shibboleth-based authentication. For this example, we'll assume CentOS 6.0.

1. Create a small CentOS image on DAIR according to the DAIR training instructions (available once you register for DAIR access). Ensure to enable access to TCP ports 22, 80 and 443 when completing the security portion of the setup.

2. SSH in to your instance as root.
3. Disable SELinux by typing:

   ```
   #setenforce 0
   ```

   at the prompt. In a production environment, you may want to leave SELinux enabled, but adjust the policy to allow access to Shibboleth and Apache.

4. Install Apache:

   ```
   # yum –y install httpd
   ```

5. Specify that http://<your-host>/secure is protected by Shibboleth. Edit /etc/httpd/conf/httpd.conf and add:

   ```
   <Location /secure>
                AuthType shibboleth
               ShibRequestSetting requireSession 1
               Require valid-user
     </Location>
   ```

6. We'll be using SSL to communicate with our Apache server, so install Open SSL:

   ```
   # yum –y install openssl

   # yum –y install mod_ssl
   ```

7. Generate the SSL key and certificate for Apache and move them into place. For this document, we'll use a fake machine name of your.host.edu. If your host does not have a DNS entry, you can use an IP address in place of your.host.edu.

   ```
   # openssl req -new -x509 -days 365 -sha1 -newkey rsa:1024 -nodes -keyout
   server.key  -out server.crt -subj '/O=<your org>/OU=<your SP>/CN=your.host.edu'

   # mv server.key /etc/httpd/conf/
   # mv server.crt /etc/httpd/conf/
   ```

8. Add the following to /etc/httpd/conf/httpd.conf to configure SSL.

   ```
   SSLEngine on
   SSLCertificateFile /etc/httpd/conf/server.crt
   SSLCertificateKeyFile /etc/httpd/conf/server.key
   ```

9. Next, add the repository that contains the Shibboleth components to the yum database by typing:

```
# wget download.opensuse.org/repositories/security://shibboleth/\
CentOS_CentOS6/security:shibboleth.repo

# mv security:shibboleth.repo /etc/yum.repos.d
# yum -y install shibboleth
```

10. Create the key and certificate used by Shibboleth. We recommend creating new keys here rather than using the ones already created for Apache.

```
# cd /etc/shibboleth
# ./keygen.sh -h your.host.edu -e https://your.host.edu/shibboleth
# chown shibd:shibd sp-key.pem sp-cert.pem
```

Note that http://your.host.edu/shibboleth is a unique identifier for your Shibboleth installation. There does not need to be any content at this URI.

11. Edit /etc/shibboleth/shibboleth2.xml and specify your own IP address or machine name in the entityID entry:

```
<ApplicationDefaults entityID="http://your.host.edu/shibboleth"\
REMOTE_USER="eppn">
```

12. Then, restart the Apache and Shibboleth servers

```
# /etc/init.d/httpd restart
# /etc/init.d/shibd restart
```

13. Test your Apache setup by starting a browser and going to “https://your.host.edu”

Step 2 – Creating a test service

For the purposes of this document, we’ll create a simple PHP service that just dumps the environment supplied by Apache, including SAML attributes to your browser. This service will be accessed via URL http://your.host.edu/secure.

1. Install PHP

```
# yum –y install php
```

2. Create the service

```
# mkdir /var/www/html/secure
```

And then create file /var/www/html/secure/index.php with the following content:

```
<?php echo '<pre>'; print_r($_SERVER); echo '</pre>'; ?>
```

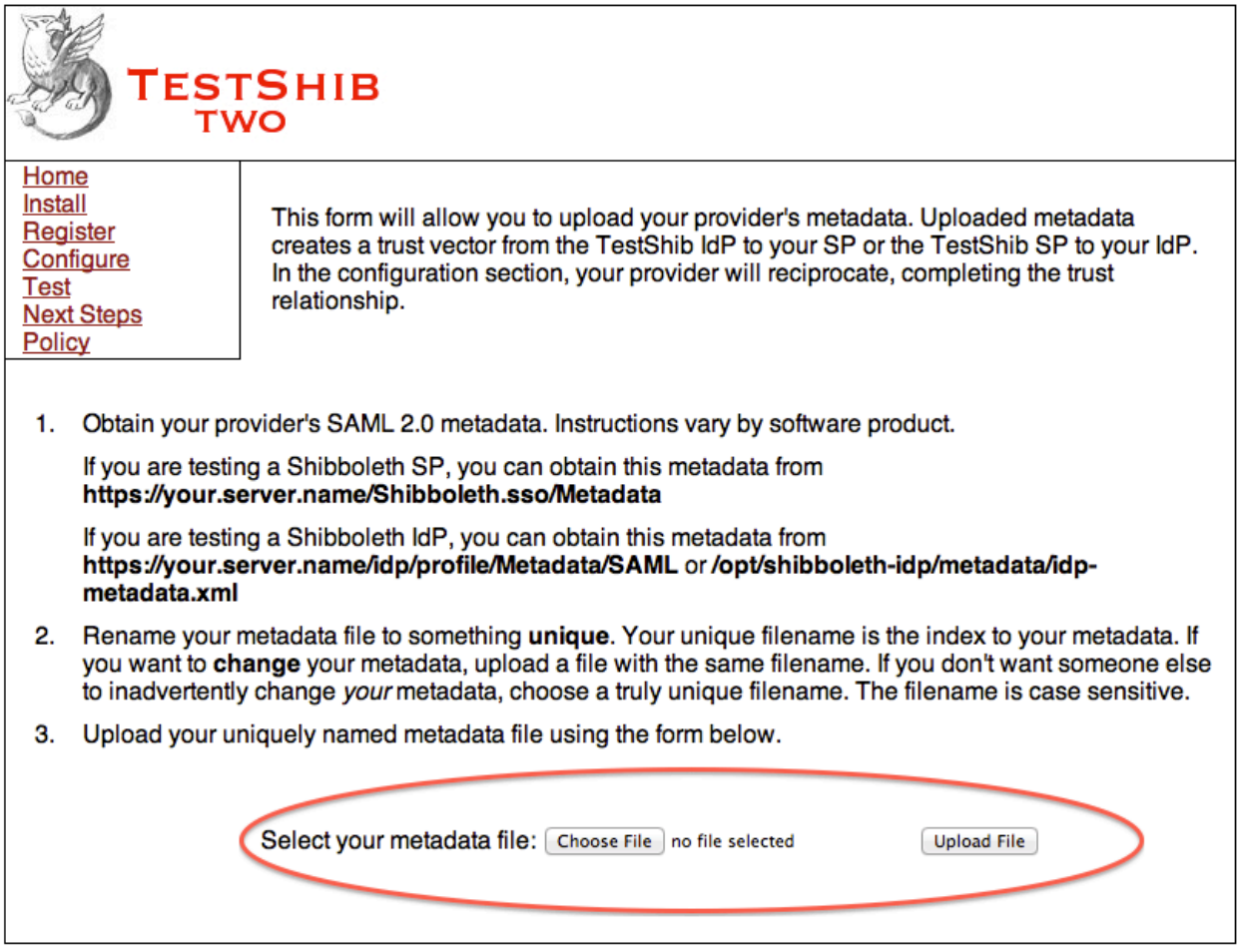
And finally, type:

```
# chmod a+x /var/www/html/secure/index.php
```

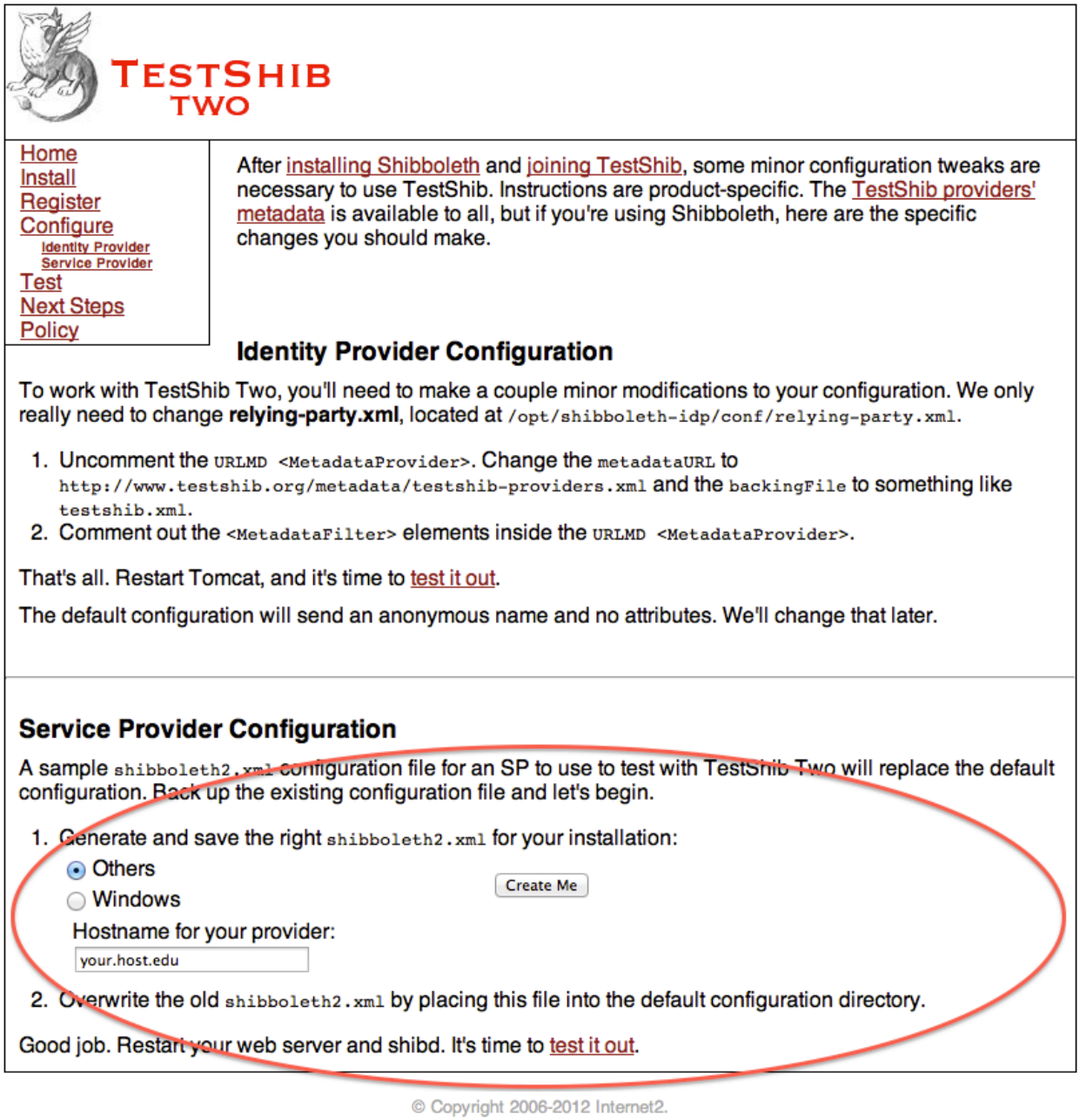
Step 3– Using your test service with the TestShib.org IdP

In this step, we'll attach your service to the test identity provider (IdP) hosted by testshib.org. As noted on the web site, this IdP is for configuration testing only and should not be used for a production system.

1. Navigate to http://your.host.edu/Shibboleth.sso/Metadata with a browser. A file will automatically be downloaded to your client machine. Give this file a meaningful name. For this document, we'll assume your_host_edu_auth_test.
2. Now, navigate to http://testshib.org/metadata.html and upload the file created in step 1 above.



3. Next, navigate to http:// https://www.testshib.org/configure.html. In the “Service Provider” section at the bottom of the page, select “Other”, enter “your.host.edu” into the “Hostname for your provider” text box and press the “Create Me” button.



4. After pressing “Create Me”, a Shibboleth configuration file will appear in your browser. Replace the contents of /etc/shibboleth/shibboleth2.xml with this new configuration.
5. Restart the Shibboleth daemon and Apache by typing:

```
# /etc/init.d/shibd restart
```

/etc/init.d/httpd restart

6. Now it is time to test your service. With your browser, visit https://your.host.edu/secure

7. You will be redirected to the testshib.org IdP login page. Enter the credentials displayed at the bottom of this page and log in.
8. If the login was successful, you will be redirected back to https://your.host.edu/secure and the output of the PHP script should appear in your browser. It will look something like:

```
Array
(
    [Shib-Application-ID] => default
    [Shib-Session-ID] => _be20bccfc5580e47b7b47ee8222e0377
    [Shib-Identity-Provider] => https://idp.testshib.org/shibboleth
    [Shib-Authentication-Instant] => 2013-03-25T14:50:45.154Z
    [Shib-Authentication-Method] => urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
    [Shib-AuthnContext-Class] => urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
    [Shib-Session-Index] => 3a59772281e896102c1de6686f6cb9904c8f1908762be5bcd43aaf59229027d0
    [affiliation] => myself@testshib.org
                                        :
                                        :
```

Step 4– Using your test service with your organization's IdP

Now that you have a service that supports Shibboleth authentication implemented and running correctly, the next step is to move from the testshib.org IdP to your organization's production IdP. This is a fairly straight forward process from your perspective.

1. Provide the metadata file created earlier (ie. your_host_edu_auth_test) to the administrator of your local IdP.
2. Edit /etc/shibboleth/shibboleth2.xml and change the entityID entry. Originally, this line looks like:

   <SSO entityID="https://idp.testshib.org/idp/shibboleth">SAML2 SAML1</SSO>

   Your IdP administrator can provide the entityID appropriate to your organization.

   Also in /etc/shibboleth/shibboleth2.xml, update the MetadataProvider entry to match your institution. The original entry looks like:

   <MetadataProvider type="XML" uri="http://www.testshib.org/metadata/testshib-providers.xml" backingFilePath="testshib-two-idp-metadata.xml" reloadInterval="180000"/>

   Again, your IdP administrator will provide the correct value.

3. Restart the shibd daemon by typing:

```
# /etc/init.d/shibd restart
```

4. Now it is time to test your service against your organization's IdP. Once again navigate to https://your.host.edu/secure with your browser.
5. This time, you should be re-directed to your organization's login page instead of the one presented by testshib.org.
6. Log in using your institutional credentials.
7. Your browser should be redirected to https://your.host.edu/secure and the output of the PHP script should be displayed.

Step – 5 Using CAF

In this final step, we're going to enable CAF so that instead of going directly to your institution's hard-coded IdP for authentication, you select the appropriate institution from a list of CAF members. Please make the following changes to file /etc/shibboleth/shibboleth2.xml. For your convenience, a complete version of this file is included in Appendix B.

1. Delete the <SSO> tag.
2. Delete the <MetadataProvider> tag.
3. Add the following line within the <Sessions> tag.

```
<SSO discoveryProtocol="SAMLDS" discoveryURL="https://caf-
shib2ops.ca/DS/CAF.ds">SAML2 SAML1</SSO>
```

4. Add the following block, also within the <Sessions> tag.

```
<SessionInitiator type="Chaining" Location="/DS" id="DS" relayState="cookie">

        <SessionInitiator type="SAML2" acsIndex="1" template="bindingTemplate.html"/>

        <SessionInitiator type="Shib1" acsIndex="5"/>

        <SessionInitiator type="SAMLDS" URL="https://caf-shib2ops.ca/DS/CAF.ds"/>

</SessionInitiator>
```

5. Add the following tags. These should be outside the <Sessions> tag but within the <SPConfig> tag.

```
<MetadataProvider type="XML" uri="https://caf-
shib2ops.ca/CoreServices/caf_metadata_signed.xml"
backingFilePath="caf_metadata_signed.xml" reloadInterval="180000"/>
```

6. Then, restart the Apache and Shibboleth servers

   ```
   # /etc/init.d/httpd restart
   # /etc/init.d/shibd restart
   ```

7. Note that before you can use CAF, you must provide your Service Provider metadata to the CAF administrator. As above, you can retrieve the metatdata file by visiting

   https://your.domain.edu/Shibboleth.sso/Metadata

   with a browser. Send the information returned to caf@canarie.ca. You will be informed via return email when your metadata has been added to the system.

8. Now navigate to https://your.host.edu /secure with your browser. You should be presented with the CAF member list.
9. Select your institution from the list.
10. You should be now be presented with your institution's IdP login window. Supply your credentials and log in.
11. If all went well, the output of the PHP script created in Step 1 will appear in your browser.

Appendix A – Further Reading

Wikipedia's article on SAML

http://en.wikipedia.org/wiki/SAML_2.0

Alex Shulgin's article on adding Shibboleth authentication to a JBOSS application –

http://www.commandprompt.com/blogs/alex_shulgin/2011/07/real-world_example_of_adding_saml_authentication_to_a_jboss_application/

Shibboleth.net – the ultimate Shibboleth authority

https://wiki.shibboleth.net/confluence/display/SHIB2/Home

Appendix B– CAF-compliant shibboleth2.xml file

Before using this file, please remember to change “your.host.edu” to the name of the machine running your service in the ApplicationsDefault tag.

```
<SPConfig xmlns="urn:mace:shibboleth:2.0:native:sp:config" xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
clockSkew="1800">

<ApplicationDefaults entityID="https://your.host.edu/shibboleth" REMOTE_USER="eppn">

<Sessions lifetime="28800" timeout="3600" checkAddress="false" relayState="ss:mem" handlerSSL="false">

<!-- Added for CAF -->

<SSO discoveryProtocol="SAMLDS" discoveryURL="https://caf-shib2ops.ca/DS/CAF.ds">SAML2 SAML1</SSO>

<!-- End of Added for CAF -->

<Logout>SAML2 Local</Logout>

<Handler type="MetadataGenerator" Location="/Metadata" signing="false"/>

<Handler type="Status" Location="/Status" acl="127.0.0.1"/>

<Handler type="Session" Location="/Session" showAttributeValues="true"/>

<Handler type="DiscoveryFeed" Location="/DiscoFeed"/>



<!-- Added for CAF - has to go inside <Sessions> -->

<SessionInitiator type="Chaining" Location="/DS" id="DS" relayState="cookie">

        <SessionInitiator type="SAML2" acsIndex="1" template="bindingTemplate.html"/>

        <SessionInitiator type="Shib1" acsIndex="5"/>

        <SessionInitiator type="SAMLDS" URL="https://caf-shib2ops.ca/DS/CAF.ds"/>
```

```
</SessionInitiator>

</Sessions>

<Errors supportContact="root@localhost" logoLocation="/shibboleth-sp/logo.jpg" styleSheet="/shibboleth-sp/main.css"/>

<MetadataProvider type="XML" uri="https://caf-shib2ops.ca/CoreServices/caf_metadata_signed.xml" backingFilePath="caf_metadata_signed.xml" reloadInterval="180000"/>

<AttributeExtractor type="XML" validate="true" path="attribute-map.xml"/>

<AttributeResolver type="Query" subjectMatch="true"/>

<AttributeFilter type="XML" validate="true" path="attribute-policy.xml"/>

<CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem"/>

</ApplicationDefaults>


<SecurityPolicyProvider type="XML" validate="true" path="security-policy.xml"/>

<ProtocolProvider type="XML" validate="true" reloadChanges="false" path="protocols.xml"/>

</SPConfig>
```