

SECURE Documentation: Institutional Resource Sharing Set Up Guide

Time and skill estimate

Time required: 0.5 day.

Skill required: Some sysadmin experience to install if no modifications to system from those described below; HTML and web server configuration skills; C++ skills if modifications required.

Pre-requisites

- Working CAS server installation for the institution;
- Apache web server version 1.3 with mod_sso and server certificates (there is also a version of mod_cas for Apache 2.0 - these instructions do not cover its installation and it has not been tested by the SECURE project);
- openssl;
- gcc C++ compiler and GNU Make.

Installation procedure

1 Download software

The full software download is available at <http://www.yale.edu/tp/cas/cas-client-2.0.10.tar.gz>, including CAS clients for Apache 2.0, Java, etc. However, the SECURE project has produced a version which contains only the mod_cas software for Apache 1.3, with most of the modifications required for a successful installation already made, and a script to automate much of the setup procedure. To download this, use the wget utility:

```
% wget http://www.angel.ac.uk/SECURE/software/mod_cas.tar.gz
```

Once download is complete, open the archive:

```
% tar zxvf mod_cas.tar.gz
```

2 Configure and compile software

A utility has been provided that will set up the local information needed by mod_cas and run the compilation process. Several pieces of information are needed by this utility (at least when when run for the first time; some items may not be needed if the script is re-run):

- CAS server hostname (the script will continue asking for this until a value is entered);

- CAS server port number (defaults to 443);
- CAS login URL (defaults to "https://CAS_server/cas/login", where CAS_server is the value from the first question, followed by the port number if it is other than 443);
- CAS validation relative URL (defaults to "/cas/validate"; this is assumed always to be on the same machine as the login URL);
- location of GNU make on your machine (script attempts to find it in the standard RedHat location, and if it can will provide a default value);
- location of CA root certificate to use (again, the certificate will ask repeatedly for this, until provided with a file name that exists). If there are intermediate certificates involved in the chain between the root certificate and your server certificate, you will need to create a certificate bundle containing the root certificate and the intermediate certificates:

```
% cat servercert.cert intermediate.cert caroot.cert > cert.bundle
```

(You can of course re-use an existing bundle if you have one.) There are more details on certificate bundles in the [Shibboleth origin installation instructions](#).

Having collected this information, change to the mod_cas directory:

```
% cd mod_cas
```

and run the utility as follows:

```
% ./cas_setup.sh
```

and enter the required values as prompted. This will configure and compile mod_cas.

Once you have carried out this step, if you need to repeat it for any reason (e.g. because you want to run the test software) you will need to run the script with an option to make it recreate the configuration:

```
% ./cas_setup.sh --recreate-config
```

To install the software, you just need to copy the newly created file *mod_cas.so* to the appropriate location. On a standard RedHat machine, with Apache installed via RPM, this will be */usr/lib/apache*; if Apache was compiled from source, it will usually be */usr/local/apache/libexec/*. To copy the file to these locations, you will probably need to be the root user.

```
# cp mod_cas.so /usr/lib/apache
```

3 Configuring mod_cas

The configuration of mod_cas, like other Apache modules, is done through the Apache configuration file. This is located at */etc/httpd/conf/httpd.conf* (on a standard RedHat installation; it may be elsewhere, e.g. */usr/local/apache/conf/httpd.conf*, when Apache has been installed by compiling the source). The first directives to add to the file will load the mod_cas module. This line should be placed in the section of LoadModule

instructions, after the line loading the `ssl_module` (at the end of the list is the safest place to put it):

```
LoadModule cas_module modules/mod_cas.so
```

This version of the directive is appropriate for Apache installed as a RedHat RPM. If installed from source you are likely to need to change "modules" to "libexec". It will be safe to follow the examples given by the other `LoadModule` instructions in the configuration file.

Similarly, the following instruction should be placed at the end of the section of `AddModule` instructions:

```
AddModule mod_cas.c
```

It is also possible to set other configuration parameters for `mod_cas`; these are included as lines in the Apache configuration file - they must be after the `AddModule` directive. These mainly relate to the optional cache directory. Here, details of current CAS tickets for accessors of the system are stored so that `mod_cas` does not need to check validity of access with the central server for every single request from a user. Since the tickets time-out fairly quickly (unless you have configured the server to have longer living tickets), this need not be regarded as a significant security issue, and it speeds up access to the protected resource and reduces the load on the central server. To create the cache file, you should firstly create a directory to contain the file:

```
# mkdir /var/cas_cache
```

and then create the file:

```
# touch /var/cas_cache/cache
```

To make it world read/writable:

```
# chmod a+rw /var/cas_cache/cache
```

You will also need to make it possible to access the directory containing the cache file:

```
# chmod a+x /var/cas_cache
```

The table lists the available parameters that you may want or be required to change. They should go into `httpd.conf` in the format *ParameterName value*, and must come after the `AddModule` directive. Where the default setting is appropriate, the parameter need not be listed (though you might find it helpful to include it in case you want to change the value later).

Parameter	Notes
<i>CASLocalCacheFile</i>	File in which to store tickets. Must be read/writable by the user who runs the web server (this usually implies world read/writable). If this directive is not present, no cache is used and the other cache parameters are ignored.
<i>CASLocalCacheSize</i>	Maximum number of tickets to store in cache. Since tickets are small, this can be a large value. The default is 1000 to produce a file of about 40K.

Parameter	Notes
<i>CASLocalCacheTimeOut</i>	Maximum length of time in seconds for a ticket to live in the cache before mod_cas returns to the CAS server to check credentials. Defaults to 3600 (i.e. 1 hour).
<i>CASLocalCacheInsecure</i>	Cookies stored in a web browser, like the CAS tickets, can be marked as secure or insecure. The only difference is that most web browsers will only send secure cookies via https, as they are assumed to contain sensitive information. This parameter can be set to "on" or "off" (the latter being the default). If the setting is "off", tickets are regarded as being secure cookies and vice versa. The default setting will probably make it impossible to store tickets for a web resource on the server protected by CAS which is not accessed via https. When this option is set to "on", then individual users' sessions can be spoofed by someone with access to the plaintext stream that connects the two servers, thus introducing a security hole which may or may not be considered acceptable - it would not just be the one resource where access would be compromised but all those protected by CAS for the duration of the life of the current ticket. For safety, you should aim to run all services using CAS for authentication with https and leave this parameter at the default value. If you cannot do this, you will need to set this parameter to "on" for those machines that will run insecure services protected by the CAS. (Note that with the parameter set to "off", the redirection back to the resource from the CAS server will change the URL from "http://.." to "https://.." where a none-secure resource is accessed.)

To apply your changes, you need to restart the web server. On a standard RedHat system, type:

```
# /sbin/service httpd restart
```

4 Protecting a Web Resource With mod_cas

To protect a resource, you need to create a directory or location entry in the web server configuration file. Alternatively, you can use a *.htaccess* file in the top directory of the resource you wish to protect. The most appropriate method will depend on the context in which the resource is accessed via the web server.

4.1 Using *.htaccess* to protect a resource

This method is suitable when some directories are to be protected while others are to be left freely accessible. It's major disadvantage is that by using a file placed in the directory to be protected, it is hard to know where all the web server configuration is to be found.

To use the htaccess method, the directory entry which governs access to the resource will need to be modified to allow the file to override what is in *httpd.conf*. This is done with the *AllowOverride* directive. It will be set to one of three options:

To "None", in which case it needs to be changed to:

```
AllowOverride AuthConfig
```

To "All", in which case nothing needs to be done. Or it will be a list of values, which needs to be modified to contain "AuthConfig" if it doesn't already, for example:

```
AllowOverride FileInfo AuthConfig
```

The `.htaccess` file should go in the directory for which authentication is required. A single file will affect every sub-directory as well as the one where the file actually resides. An `.htaccess` file can be very simple. The following two lines are all that are required to protect a directory using CAS:

```
AuthType CAS
Require valid-user
```

4.2 Using a directory or location entry in `httpd.conf` to protect a resource

Essentially, this works exactly like the `.htaccess` method, except that the authorisation directives are included directly in `httpd.conf` rather than in external files. You may need to add a new directory and/or location entry to the configuration to do this (location customises access based on the URL, while directory uses local directory names). An example directory entry would look like:

```
<Directory "/usr/local/application/">
  AllowOverride None
  Options +ExecCGI
  Order allow,deny
  Allow from all
  AuthType CAS
  AuthName "London School of Economics Handle Service"
  require valid-user
</Directory>
```

and an example location entry would look like:

```
<Location /shibboleth/HS>
  AuthType CAS
  AuthName "London School of Economics Handle Service"
  require valid-user
</Location>
```

To apply your changes, you need to restart the web server. On a standard RedHat system, type:

```
# /sbin/service httpd restart
```

Once you have set up a route to a protected resource, you are ready to test `mod_cas`.

5 Testing mod_cas

Testing mod_cas is basically a question of pointing your web browser at a protected resource and seeing whether you get access. What you should see when you do this is a redirection to the login screen for the CAS, followed by access to the resource.

There are various points where the process might fail. First of all, if you do not get forwarded to the CAS server, but to a "404 Response not found" error, you have probably mistyped the CAS server address in the *cas.h* file. You will need to recompile the software (see step 2). If you can get through to the CAS server, you should see the login screen. From this point, there may be problems forwarding back to your resource (though this is unlikely), or in verifying that the ticket provided by the CAS server is valid.

If it fails to work, there is a test suite provided with mod_cas. This too needs to be compiled to run. Return to the directory into which you extracted the mod_cas software archive:

```
% cd ~/mod_cas
```

(assuming you extracted the archive in your home directory), and re-run the configuration script:

```
% ./cas_setup.sh --test-suite
```

(Add the further option "--recreate-config" if you want to check what happens if you alter the configuration settings; note that you will need to do this every time you want to change one of the values in the configuration).

This should create a utility that can be used to test the connection to the CAS server. (This has been modified from the original Yale version, to make it easier to see where an error comes from.) You now need a valid ticket. To get this make another attempt to access your resource with a web browser, and login. You will be redirected to a URL that looks something like "https://secure.institution.ac.uk/cas/validate?ticket=ST-2-jLPFN7aUZ67TrAQrDiea&service=https://webserver.institution.ac.uk/cas-test/"; the ticket is the string "ST-2-jLPFN7aUZ67TrAQrDiea". To use the tester, run:

```
% ./testssl ST-2-jLPFN7aUZ67TrAQrDiea https://webserver.institution.ac.uk/cas-test/
```

This should return an error message, or confirm that the ticket is valid. The errors are as follows:

Error	Meaning
<i>CTX failure</i>	A problem with the SSL libraries. Ensure that your system has an up-to-date set of OpenSSL libraries, and recompile mod_cas.
<i>Unable to connect to CAS host</i>	A networking problem. Check that it is possible to connect to the web server containing the CAS server on the port specified from the command line (to ensure that firewalls do not confuse the issue; you may be able to connect from your desktop but not from the machine where mod_cas is installed). See below for more detailed suggestions on how to do this.

Error	Meaning
<i>Can't establish SSL communication - am I talking to a secured socket?</i>	Although the machine is available, it is not possible to connect to the CAS server using SSL. This may mean that the web server is not running, is not accepting secure connections, or that you mistyped the port in the <i>cas.h</i> file.
<i>Can't get certificate</i>	This is unlikely to occur if you are able to access the verification URL with a web browser. It means that the server certificate belonging to the CAS server installation cannot be retrieved. This suggests that there is a problem with your OpenSSL installation.
<i>Can't validate certificate</i>	<p>This is not necessarily a problem with the server certificate, though this can be checked using:</p> <pre data-bbox="523 685 1254 714">% curl --cacert CAfile https://secure.institution.ac.uk/cas/login</pre> <p>If the certificate validates, you should get back the web page. If not, you will get an error message that starts "curl: (60) SSL certificate problem, verify that the CA cert is OK". (There is a bug with certain versions of openssl, mod_ssl and Apache that appears to prevent some certificates from verifying. If this should affect you, then you will need to use a version of mod_cas which does not carry out the certificate verification. See below for details on how to achieve this.)</p>
<i>Cannot send SSL request to server</i>	This indicates that something has happened between establishing the connection to the server and making the request. It probably indicates that the machine suddenly went down, that the network connection failed, or that there is something wrong in the web server.
<i>unexpected read error or response too large</i>	This means that the response from the CAS server could not be understood, which probably means that the connection is being made to the wrong URL (e.g. to one of the other CAS server functions) or that there is some kind of mismatch between the versions of mod_cas and the CAS server.
<i>No http header returned</i>	This is very unlikely to occur; it would indicate serious problems with the web server.
<i>Response is negative (no "yes")</i>	This probably means that everything is working correctly, but that your ticket has now timed out. Try again via the web browser.
<i>Unexpectedly short body in response</i>	Suggestive of some sort of connectivity problem. You should try again.

5.1 Diagnosing Connectivity Problems

To check the connection to the CAS server, try the following (assuming you have the [curl](#) software installed):

```
% curl https://secure.institution.ac.uk/cas/validate?ticket=ST-2-
jLPFN7aUZ67TrAQrDiea&service=https://webserver.institution.ac.uk/cas-test/
```

A working connection should return either "yes" or "no". If this fails, then ensure that you can connect to the local network by trying:

```
% curl http://www.institution.ac.uk/
```

which should return the institutional home page. If this works, try accessing the validation URL with a web browser from another location. If you can get through, then the problem is may concern your institutional firewall, or possibly with your institutional DNS server (are you using a name for the server which your DNS knows about?) If you are unable to get through to the CAS server, there is probably a problem with the CAS server machine. You can also try using traceroute, to see where the network is having problems, but you will almost certainly need to consult your network administrator to get the CAS working. Problems with the CAS server machine may include:

- Machine down or disconnected from the network;
- Web server not running;
- Web server not receiving https connections;
- Tomcat not running.

See the [CAS server document](#) for more details on how to deal with each of these issues.

5.2 Setting up mod_cas Without Certificate Validation

To do this, you need to recompile the software with some changes, replacing one file with an equivalent that ignores the certificate validation. Return to the directory into which you originally extracted the mod_cas software archive:

```
% cd ~/mod_cas
```

and rerun the configuration utility with an option to remove certificate checking:

```
% ./cas_setup.sh --no-certificate-validation
```

(add the further option "--recreate-config" if you want to alter the configuration settings). Copy the new *mod_cas.so* file as before:

```
# cp mod_cas.so /usr/lib/apache
```

(you will need to type "y" to replace the existing file) and restart the web server to commit the changes.

Copyright © SECURE Project Team, 2004

Document last updated: 28/02/04

This document is also available at:

<http://www.angel.ac.uk/SECURE/deliverables/documentation/modcasinstall.html>