Overview

The following method describes how to reroute all user interaction for EZproxy authentication to your own script. With CGI authentication, you are responsible for all user interaction, including collecting credentials, validating them, and returning the user to EZproxy. See also External Script Authentication for a simpler method that allows you to provide your own script for validating credentials without being required to handle all user interaction.

To provide your own custom user authentication handling, follow these steps:

1. Edit your user.txt file and add a line like:

   ::CGI=http://auth.yourlib.org/ezpauth.cgi?url=^R

   OR

   ::CGI=http://auth.yourlib.org/ezpauth.cgi?url=^U

   This line indicates that when a user needs to be authenticated, EZproxy should redirect the user to a script that is available at http://auth.yourlib.org/ezpauth.cgi.

   If you choose the form that ends in url=^R, then EZproxy will provide an opaque string that represents the original URL requested. This is the simplest form to use as it avoids the use of special characters such as ampersands (&).

   If you choose the form that ends in url=^U, then EZproxy will provide the original URL requested in a URL escaped format, where special characters such as ampersands (&) will be replaced by a %xx format such as %26.

   After CGI=, you may specify a URL that starts with https instead of http if your authentication script is installed on a secure web server.

   User.txt will need additional entries to allow the Ticket URL to be recognized and accepted.

2. Once the user is redirected to the custom script, the script can authenticate the user in any manner you choose. During authentication the value provided by the url query string variable must be preserved so it can be returned to EZproxy when the user has been authenticated.

3. Assuming that the script authenticates the user, the script then construct a return URL using Ticket Authentication to return the user back to the EZproxy server. The Ticket URL must be constructed to include the url value originally provided when the user was redirected to the custom script.
Data flow

You may want to print a copy of this diagram as it is referenced by the following.

The diagram describes the flow of data during an EZproxy login that involves external CGI authentication. In this document and on the diagram, "ezproxy" is used as the host name for your EZproxy server, "CGI" is the name of your CGI server, "db" is the remote database vendor, "script" is your CGI script, and "s" (as in http://db/s) is the starting page portion of the URL the user is trying to reach.

Also, ezproxy:2 represents the login port on EZproxy (normally ezproxy:2048), and ezproxy:3 represents the virtual web server created for the "db" host.

In EZproxy, you would set up this link by editing ezproxy.usr and adding a line like this:

```plaintext
::CGI=http://cgi/script?url=^R
::Ticket
TimeValid 1
MD5 verysecret
Expired; Deny expiredticket.htm
/Ticket
```

The first line tells EZproxy that authentication requests should be rerouted to http://cgi/script with a query string parameter named url which will receive an opaque representation of the destination URL (e.g., http://db/s is not passed to the CGI script, but instead a string of letters, digits, numbers, and select special characters which represents the destination URL).

The subsequent lines are typical of configuration for a ticket URL. A ticket URL is the mechanism used to return an authenticated user to EZproxy for access. More information on ticket URLs appears at Ticket Authentication.

Your CGI script is involved in lines 3 - 6. It is automatically brought into the loop by EZproxy when unauthenticated users attempt access. After a user has been authenticated, it is no longer involved in the process.

Here is a detailed description of what happens during each step.

1. We start with the assumption that the user just clicked a link to http://ezproxy:2/login?url=http://db/s. This link may have been on any web server anywhere, and would typically come from a page that has links to all of your database vendors from somewhere in your libraries home page.
2. EZproxy has identified that the user is not yet authenticated, so it reroutes the user to your CGI script, sending an opaque string of letters, digits, hyphen (-), underscore (_), and/or periods (_) that represents the URL the user was trying to reach.
3. The user's browser goes straight to your CGI script, with the user not having noticed anything has really happened.
4. Your CGI script sends back a form for the user to complete for validation. This form contains a hidden field url that your CGI program will use later as it completes the authentication process, allowing your user to continue on to the resource originally requested.
5. The user has filled out the form, and now the user submits the form for processing.
6. Your CGI script has successfully authenticated the user (if not, you might go back to line 4 to send the CGI form again, along with appropriate error message). Since the user authenticated, your CGI script generates a ticket URL to authenticate the user into EZproxy. The ticket URL contains the username to associate with the user, a
cryptographic signature to prove authentication, and the opaque version of the URL of the database the user is trying to reach. More information on generating ticket URLs can be found at Ticket Authentication.

7. The browser sends the ticket URL to the EZproxy server.
8. EZproxy creates a new session and routes the user back to complete session setup.
9. The browser returns to confirm session setup.
10. EZproxy sends back the right response to embed a cookie in the user’s browser as an ongoing indication that the user has validated, along with a redirect to come back to EZproxy to verify that the cookie embedded correctly.
11. EZproxy verifies that the cookie is present. If it is missing, the file cookie.htm is sent from the docs subdirectory.
12. The user's cookie was present, so EZproxy now redirects the user to the EZproxy virtual web server to begin remote access.
13. The user's browser requests the "/s" page from the virtual web server.
14. EZproxy forwards the request on to the real server.
15. EZproxy receives back the response from the remote database server.
16. EZproxy edits the response, then sends it back to the user.

Groups

Normally, when your custom CGI script sends the user back to EZproxy with a Ticket URL, the Ticket URL should indicate all groups for which the user should be authorized for access. If you want to handle inadequate group access separately, your entry in user.txt can look like this:

```bash
::CGI=http://auth.yourlib.org/ezpauth.cgi?url=^U&logup=^L
```

The ^L will be replaced by "true" if the user is trying to access a resource outside of group assignment and "false" during an initial authentication. This version demonstrates the use of url=^U to provide the original URL requested. Your script may be able to use this information along with a value of "true" for logup to determine a particular course of action when giving the user feedback that access is unavailable.