



1 Introduction

At the University of Oxford, we use a single-signon system called WebAuth. It was developed and is being maintained at Stanford University. I have not found out how to modify Rails so that it understands WebAuth authentication: instead I have devised a kludge in which a WebAuthed PHP script can be used to sit in front of any Rails application.

The document assumes that the instructions given in the document *Rails HOW-TO: Apache and Basic Authentication* have been followed. The document is available at <http://www.oucs.ox.ac.uk/rails/howtos>.

2 Overview of the kludge

Here is an overview of how this kludge works. A PHP script (called `rails.php`) sits in a directory that is protected by WebAuth:

```
https://www.abcd.ox.ac.uk:8113/restricted/rails.php?
  url=https://www.abcd.ox.ac.uk:8113/apps/contacts/login
```

Having authenticated with WebAuth using an SSO username and password, the PHP script is executed. As you can see, the PHP script is passed a parameter which is the URL of a login page of a Rails application. The only thing the PHP script does is to redirect to that URL passing a parameter that is a base64 encoded string:

```
https://www.abcd.ox.ac.uk:8113/apps/contacts/login?id=base64string
```

The `id` parameter is the base64 encoded version of an encryption of the username and the current date and time.

The login method of the Rails application looks at the `id` parameter, decodes it and then unencrypts it. So it now has the username and the date and time. In order to avoid *replay attacks*, the Rails application checks that the date and time refers to a recent date and time. It also checks that the username is in a list of valid usernames. If both these tests are passed, it sets a session variable (`session[:user_id]`) to "OK"; otherwise, it is set to "BAD".

Having authenticated, subsequent methods of the Rails application are executed in the usual way, i.e., by using the URL of the method, e.g.: <https://www.abcd.ox.ac.uk:8113/apps/contacts/add>

We need to alter the code of the controller of the Rails application so that it has a call of `before_filter`. This call can ensure that a method called `authenticate` is called before any method of the Rails application is called. The `authenticate` method can be similar to the `authenticate` method introduced in the document *Rails HOW-TO: Apache and Basic Authentication*, which is available at

<http://www.oucs.ox.ac.uk/rails/howtos>

There is one change: the method has been altered to deliver `true` if the session variable `session[:user_id]` has the value "OK". Otherwise, it gives a 401 error (meaning *Not authorised*).

3 Installing WebAuth

3.1 Providing WebAuth

OUCS provide a document entitled *Oxford Webauth HOWTO* describing how to protect web pages using WebAuth. It is available at <http://www.oucs.ox.ac.uk/webauth>. Section 3.1 of the document advises you to do each of the following:

- ‘have a working Apache 2 server (with SSL support)’;
- ‘download and compile the WebAuth v3 source’;
- ‘make sure you are running some form of regular system time synchronisation’;
- ‘install a configuration file for the kerberos libraries’;
- ‘create a kerberos keytab for your WebAuth protected service’;
- ‘configure Apache to make use of the WebAuth module’.

As mentioned above, this document assumes that the instructions given in the document *Rails HOW-TO: Apache and Basic Authentication* have been followed. So an Apache web server has been installed and this has been altered to accept https connections. So, we have already done the first step. We will now go through each of the other steps.

3.2 Download and compile the WebAuth v3 source

There is no need for me to compile the WebAuth source as there is a Debian package for WebAuth. It is called `libapache2-webauth`:

```
apt-get install libapache2-webauth
```

We need to ensure that the WebAuth library gets included as part of the Apache 2 server when it is next restarted:

```
cd /etc/apache2/mods-enabled
ln -s /etc/apache2/mods-available/webauth.conf .
ln -s /etc/apache2/mods-available/webauth.load .
```

3.3 Make sure you are running some form of regular system time synchronisation

My version of debian is using `ntpd` from the package `ntp-simple`:

```
ps -ef | grep ntp
```

3.4 Install a configuration file for the kerberos libraries

The *Oxford Webauth HOWTO* document provides a minimal kerberos configuration:

```
cat >/etc/krb5.conf <<'%'
[libdefaults]
    default_realm = OX.AC.UK
[realms]
OX.AC.UK = {
    kdc = kdc0.ox.ac.uk
    kdc = kdc1.ox.ac.uk
    kdc = kdc2.ox.ac.uk
    admin_server = kdc-admin.ox.ac.uk
}
[domain_realm]
    .ox.ac.uk = OX.AC.UK
    ox.ac.uk = OX.AC.UK
%
```

3.5 Create a kerberos keytab for your WebAuth protected service

The *Oxford Webauth HOWTO* document says you should send a message to `sysdev@oucs.ox.ac.uk` for help with creating a kerberos keytab. It says 'we need to know the name of the machine that will be running the service. If you are not the ITSS responsible for the unit in question, please direct your request through them; currently the management of service keytabs is the responsibility of the unit ITSS.'

If your username is `abcd0001` and you are using the machine `www.abcd.ox.ac.uk`, `sysdev` will issue you with a *principal* of `abcd0001/itss` that can be used on that machine.

It is needed when using the `kadmin` command that is used for the management of the kerberos keytab.

```
apt-get install krb5-user
mkdir /etc/apache2/webauth
kadmin -p abcd0001/itss \
-q "ktadd -k /etc/apache2/webauth/keytab "\
" webauth/www.abcd.ox.ac.uk"
ZZZ
cd /etc/apache2/webauth
chown root:www-data keytab
chmod 640 keytab
```

If you are using a more recent version of `kadmin`, you may need to use a `-O` option.

3.6 Configure Apache to make use of the WebAuth module

The *Oxford Webauth HOWTO* document says configuring Apache for WebAuth is in two parts. First, there are 'some general configuration directives'.

```
cd /etc/apache2/mods-available
cat >webauth.conf <<' %'
# Set locations for various files used by mod_webauth
WebAuthKeyring webauth/keyring
WebAuthKeytab webauth/keytab
WebAuthServiceTokenCache webauth/service_token_cache
WebAuthCredCacheDir webauth/cred_cache
# Point to the Oxford Webauth service
WebAuthLoginURL https://webauth.ox.ac.uk/login
WebAuthWebKdcURL https://webauth.ox.ac.uk:8443/webkdc-service/
WebAuthWebKdcPrincipal service/webkdc@OX.AC.UK
# If you're having trouble switch on debugging
WebAuthDebug on
%
```

And then there are 'some per location directives'.

```
cd /etc/apache2/sites-available
ed ssl <<' %'
$i
    <Location /restricted>
        WebAuthExtraRedirect on
        AuthType WebAuth
        require valid-user
    </Location>
.
w
q
%
```

4 Installing PHP

4.1 Adding PHP

As we will be asking the web server to run a PHP script, we need to add the module for PHP to the Apache server. We will also install a package for performing encryption.

```
apt-get install libapache2-mod-php5 php5-mcrypt
```

4.1.1 Testing PHP in an WebAuthed area

We now test both the WebAuth and PHP by creating a PHP script in the WebAuthed area and seeing whether we can execute it. Before executing the script, it should ask us for an SSO username and password.

```
/etc/init.d/apache2 stop
/etc/init.d/apache2 start
mkdir /var/www/restricted
echo '<?php phpinfo(); ?>' >/var/www/restricted/phpinfo.php
https://www.abcd.ox.ac.uk:8113/restricted/phpinfo.php
rm /var/www/restricted/phpinfo.php
```

5 Providing a PHP script fronting a Rails authentication

The PHP script I have devised (that acts as an authentication front to a Rails application) performs the following tasks:

- looks at the `url` parameter to get the URL of the login page of the Rails application;
- obtains the username of the person who authenticated;
- obtains the current date and time;
- forms a 48 character string from the date, the username and some trailing spaces;
- produces an encrypted form of that string;
- converts that into a base64 string;
- redirects to the URL passing the base64 string as a parameter.

```
cd /var/www/restricted
cat >rails.php <<'%'
<?php
  // thanks to:
  // http://jnylund.typepad.com/joels_blog/2007/11/cross-language-.html
  $key = "73267XtGjmQpsAzx";
  $username = $_SERVER['REMOTE_USER'];
  $url = $_GET["url"];
  $date = date("c");
  $input_string =
    substr($date . $username . ' ', 0, 48);
  $cipher_alg = MCRYPT_RIJNDAEL_128;
  $b64Encoded = "";
  for ($chunk_number = 0; $chunk_number < 3; $chunk_number++) {
    $chunk = substr($input_string, $chunk_number*16, 16);
    $encrypted_string =
      @mcrypt_encrypt($cipher_alg, $key, $chunk, MCRYPT_MODE_CBC);
    $b64Encoded .= base64_encode($encrypted_string);
  }
  header('Location: ' . $url . '?id=' . urlencode($b64Encoded));
?>
%
```

6 Modifying the Rails application

6.1 Enabling sessions

As we are going to use a session variable to record the fact that someone has already authenticated, we need to add session support to our Rails application.

```
cd /var/apps/contacts
rake db:sessions:create
cd /var/apps/contacts/db/migrate
cat *_create_sessions.rb
cd /var/apps/contacts
rake db:migrate
ed config/environment.rb <<%
g/# config.action_controller.session_store/s/# //p
w
q
%
ed app/controllers/application.rb <<%
/protect_from_forgery/s/#//p
w
q
%
```

6.2 Modifying the routing table to include login

We are adding a new method to the Rails application. It is one called `login`. So we have to alter the routing code so that a URL ending in `/login` is passed to the `login` method of the controller.

```
cd /var/apps/contacts
head -12 config/routes.rb
ed config/routes.rb <<'%'
g/map.resources :phones/s/$/, :collection => { :login => :get }/p
w
q
%
head -12 config/routes.rb
```

6.3 Ensure that we can execute the login page without having to execute the authenticate method

We need to alter the controller so that we can execute the login page without having to execute the `authenticate` method.

```
cd /var/apps/contacts
ed app/controllers/phones_controller.rb <<'%'
g/ *before_filter/d
la
  before_filter :authenticate, :except => [ :index, :login ]
.
w
q
%
```

6.4 Adding an action for login

We need to provide some code for the `login` method.

```
cd /var/apps/contacts
ed app/controllers/phones_controller.rb <<'%'
/^ *private/, $-1d
w
q
%
ed app/controllers/phones_controller.rb <<'%'
/^ def login/-1
./,/^ end/d
w
q
%
ed app/controllers/phones_controller.rb <<'%'
$i
def login
  valid_usernames = ["abcd0001"]
  max_diff_seconds = 10
  if login_is_OK(valid_usernames, max_diff_seconds)
    session[:user_id] = "OK"
  else
    session[:user_id] = "BAD"
  end
  respond_to do |format|
    format.html # login.rhtml
  end
end
private
def login_is_OK(usernames, max_diff_seconds)
```

```
require 'crypt/rijndael'
key = '73267XtGjmQpsAzx'
rijndael = Crypt::Rijndael.new(key, 128, 128)
b64in = params[:id][ 0..23]
bbyte = Base64.decode64(b64in)
decoded_parameter = rijndael.decrypt_block(bbyte)
b64in = params[:id][24..47]
bbyte = Base64.decode64(b64in)
decoded_parameter += rijndael.decrypt_block(bbyte)
b64in = params[:id][48..71]
bbyte = Base64.decode64(b64in)
decoded_parameter += rijndael.decrypt_block(bbyte)
datetime_string = decoded_parameter[0..24]
username = decoded_parameter[25..-1].strip
require 'parsedate'
datetime_array = ParseDate.parsedate(datetime_string)
datetime_timestamp = Time.local(*datetime_array)
now_timestamp = Time.now()
diff_seconds = now_timestamp.to_i - datetime_timestamp.to_i
return diff_seconds>=0 && diff_seconds<=max_diff_seconds &&
      usernames.include?(username)

end
def authenticate
  if session[:user_id] == "OK"
    true
  else
    render :nothing => true, :status => 401
  end
end
end
.
w
q
%
```

6.5 Adding a view for login

We need to provide a view that shows the result of logging in.

```
cd /var/apps/contacts
cat > app/views/phones/login.html.erb <<'%'
<h1><%= session[:user_id] %></h1>
<%= link_to 'Back', phones_path %>
%
```

7 Using the PHP script

It is time to try out the PHP script. However, we first need to install the RubyGem that contains the code for performing encryption.

```
gem install crypt
/etc/init.d/apache2 stop
/etc/init.d/apache2 start
https://www.abcd.ox.ac.uk:8113/restricted/rails.php
?url=https://www.abcd.ox.ac.uk:8113/apps/contacts/phones/login
```