Packetizer®

OpenID Provider Server Software

December 30, 2010

PUBLIC INFORMATION

OpenID

Contents

1 What is this software? 3

2 Limitations 3

3 Installation Procedures 3

3.1 Create Database Tables 3

3.2 Decide Where to Put the Software 3

3.3 Install the Software 3

3.4 Configure the Software 4

3.4.1 Database Connection Configuration 4

3.4.2 OpenID Server Configuration 4

3.4.3 Configuring Apache 5

3.5 Ensure Scripts are Executable 6

3.6 Create Users 6

3.7 Customizing Templates 7

3.8 Database Management 7

4 Miscellaneous 7

Legal Information

1 What is this software?

This software package implements an OpenID Provider. It is written in Perl, uses MySQL, and is designed to work with the Apache web server. Since it is free, open source software, you are welcome to modify it as desired to work in other environments.

2 Limitations

This software implements only the OpenID 2.0 specification without any extensions. Support for 1.x is limited. In particular, there is a "dumb mode" specified in the 1.1 specification that may not supported: it has not been tested.

3 Installation Procedures

3.1 Create Database Tables

The first step is to create the required database tables. This software uses four tables: nonce, openid_assoc and openid_users, and openid_sigs. The database tables may be created inside any existing MySQL database, or you may create a new database just for these tables. You will find a script in the misc directory that contains the SQL statements to create the required tables.

3.2 Decide Where to Put the Software

You need to decide where to place the files. As provided, the software assumes that the software will be rooted somewhere like http://openid.example.com, with user identifiers of the form http://openid.example.com/auser. However, you could use any kind of structure, such as http://www.example.com/openid/auser. It is important to pick something you can live with, since OpenID identifiers are intended to be permanent identifiers for your domain.

3.3 Install the Software

The software contains several important directories. The software should be installed as suggested below:

- bin – This is a directory where you will place programs to run via cron and should not be accessible via a web browser
- lib – This directory contains common library routines used by the various Perl scripts.
- htdocs – This script contains the files that will reside in the directory with all other web content. The files in this directory should be stored at the root of whatever prefix you choose for your OpenID identifiers. A good place is at the root of http://openid.example.com/. Note that one does not need to use HTTPS with this software, but it's strongly advised that access to the checkid script be secured with HTTPS since this routine will prompt the user for a password, which will then be transmitted from the browser to the checkid script. That script will also validate users based on a cookie that should not be sent in the clear. If you host multiple domains on the same physical server, you may put the checkid script on one virtual server and the other software on a different virtual server. For example, this script might be stored at

https://secure.example.com/checkid/ while the rest of the software is at http://openid.example.com/.

- templates – This directory contains HTML files that are used as templates when generating pages via the CGI scripts. These may be modified as you see fit and located anywhere. We'll discuss customizations below.
- misc – This directory contains miscellaneous files that are used as a part of the installation process, but are not used once installed.

3.4 Configure the Software

The next step is to configure the software. We have tried to isolate all of the configuration information into two different files. One is used to configure the database connection and the other is used to configure the OpenID software itself.

3.4.1 Database Connection Configuration

The lib directory contains a file called config.pl. Inside, you will see four global variables defined. They each need to be populated with the proper value so as to enable a connection to the MySQL database. The variables are:

- database_user_id – This is the user ID associated with the database connection.
- database_password – This is the password used to connect to the database. Since this file contains the password in plaintext form, the config.pl file should be stored in a location that is not directly accessible via a web browser.
- database_name – This is the name of the database that contains the database tables discussed in section 3.1
- database_server – This is the name of the database server, which is often localhost in smaller installations.

3.4.2 OpenID Server Configuration

The lib directory contains a file called openid_config.pl which global variables used by the various scripts. Each of these values should be replaced with appropriate values. The variables are:

- openid_site_name – This is the name of your site (e.g., we use "Packetizer" here, obviously)
- op_endpoint – This is the URL to the login routines. We locate this at http://openid.packetizer.com/login/, but since this is a CGI script, you might prefer http://www.mysite.com/cgi-bin/openid_login. You can move the login script and rename it as you wish.
- contact – This is a URL used to direct remote parties to a web page that contains a means of contacting the domain owner. It is provided in certain as recommended by the OpenID specification.
- process_login – This is the URL to the checkid script we discussed above. This should be an HTTPS URL, else you will risk transmitting passwords over the wire in the clear!
- openid_url_prefix – This is the URL used to prefix user identities. So, if this value is http://openid.example.com/, then a valid user might be http://openid.example.com/auser.

- openid_xrds_url_prefix – This is the prefix to the XRDS documents for users. Assuming this value is http://openid.example.com/xrds/, then an XRDS document for "auser" would be accessible via http://openid.example.com/xrds/auser.
- openid_identity_template – This is a pathname to the HTML template file used to provide user identity information. This page is used by the user to initiate a logoff operation after previously requesting the OpenID server to keep the user logged in. This variable should point to the file found in templates/ identity_template.html.
- openid_not_found_template – This is the pathname to a 404.shtml file that will be shown to anybody that tries to access a user identity page that does not exist. While this is a template, we actually use the same 404.shtml file normally used on our web server, so the file is found in htdocs/404.shtml, but you are free to use a different file or move it elsewhere.
- openid_login_template – This is the pathname to the HTML page that serves as the user login page.
- openid_insecure_cookie_domain – This is the domain the administrator wishes to associate with the cookie "openid_user". This should be defined as a sub-domain of the administrative domain, but the cookie is not considered secure and may be passed over HTTP or HTTPS connections.
- openid_secure_cookie_domain – This is the domain the administrator wishes to associate with the cookie "openid_user_key", a secret key established between the browser and OpenID server when the user wishes to be logged in automatically. Since this is a sensitive piece of data, the administrator should ensure that cookies passed over this sub-domain use only HTTPS. Cookies will be marked as "secure". If the administrator wishes to disable the "remember this computer" option, they may do so by setting this domain value to an empty string.
- openid_cookie_expiration – This value indicates how long the above "openid_user" and "openid_user_key" cookies will remain valid. The expiration time of the cookies and the associated entry in the database is increased upon each successful login.
- session_type – This might have one of two values, but unless you use HTTPS for everything, we strongly suggest you leave this as it is, else you risk sending encryption keys in the clear! Comments in the code explain this a little more.
- assoc_expiration – This indicates how long the an association should remain valid before a new association must be established between the server and a relying party.
- assoc_expiration_grace – How many seconds will we honor an association beyond the expected expiration time.
- Others variables are not intended to be modified and doing so might break things

3.4.3 Configuring Apache

If you leave the CGI scripts in the htdocs/ directory, you need to ensure that files ending in .cgi will be executed. That probably goes without saying, but if you try to access a file like user.cgi and see code, then you have a configuration error.

If you wish to use the 404.shtml document, add this to your Apache configuration:

```
ErrorDocument 404 /404.shtml
```

You need to ensure that the Perl libraries are accessible. To do this, add these lines to your Apache configuration:

```
SetEnv PERL5LIB /path/to/lib/
```

You might have noticed that there are two CGI scripts in htdocs, once called “user.cgi” and one called “xrds.cgi”. Those may be located anywhere, but wherever you place them, you need to ensure that Apache performs the correct URL re-writing to access them. Assuming you leave these in the default location, the following configuration should be added to Apache:

```
RewriteEngine On
RewriteRule ^/([A-Za-z0-9]+)$          /user.cgi?username=$1 [L]
RewriteRule ^/xrds/([A-Za-z0-9]+)$     /xrds.cgi?username=$1 [L]
```

If you use other re-writing rules, you may already have the re-write engine turned on and the first line may be unnecessary. You will notice that these rules assume that any name that contains the characters [A-Xa-z0-9] at the root of the OpenID directory is a possible OpenID user ID.

3.5 Ensure Scripts are Executable

Make sure that all of the CGI scripts are executable (chmod 755). The scripts to examine are login/index.cgi, user.cgi, xrds.cgi, and checkid/index.cgi.

3.6 Create Users

The software does not contain web pages to create users. However, adding users to the system is trivial and you may add users by hand or you might right your own software to create users. The reasons we do not provide a means to add users to the database are that you probably already have software that creates user accounts in your network and this would be largely redundant. If you do not, then creating users by hand is sufficient.

To create a new OpenID user ID, simply add a row to the openid_users table. The only required fields are “serial” (which should be assigned automatically by MySQL if you use a value of 0), “username”, “password”, and “name”. The field called “homepage” is optional and, if present, will be shown on the user’s identity page.

The only field that warrants mention is the password field. We do not store passwords in the clear in the database. Rather, the password field contains the SHA-1 hash of the user’s password. To create the password, you can use any SHA-1 tool (software available on Packetizer). The simplest way is from the Linux command-line:

```
echo –n user_password | sha1sum
```

It is important not to forget the -n flag to prevent the LF character from being used as a part of the password hash.

3.7 Customizing Templates

There are several HTML files and templates that you may want to customize for your environment.

The first file is htdocs/index.html. This file is the file that will be shown to visitors who visit the root directory of the OpenID server, but do not indicate a user ID. This file is of no significance to the software and is only there to present some content to the user accessing the site.

The second file to consider is templates/identity_template.html. This file contains several tags that are replaced by the Perl script(s) that process this file, including OPENID_SITE_NAME, OPENID_NAME, CONTENT_TAG, etc. You could change these, but the headers in the file are important. You should not change these headers:

```
<link rel="openid2.provider" href="OPENID_OP_ENDPOINT"/>
<link rel="openid2.local_id" href="OPENID_URL_PREFIXOPENID_ID"/>
```

The one exception to the above rule is if you are integrating this page into an HTML 4.0 site. Note that this page is an XHTML-conformant page and, as such, as the closing "/>" bracked. The important thing is that these headers exist in some proper form so that remote sites acting as a Relying Party can find these values. The Perl script(s) will properly fill in the values. Note that two tags are side-by-side here: OPENID_URL_PREFIXOPENID_ID. It is assumed that the OPENID_URL_PREFIX will have a '/' at the end.

The last template that you will want to consider modifying is in templates/login_template.html. When a user is on a remote site (say, slashdot.org) and tries to log in using his/her identity, this is the template file that will be presented asking for login credentials.

3.8 Database Management

The only management that needs to be done is removal of old nonce values and removal of old signatures and associations. There two scripts that you may execute periodically from cron to do that for you: bin/expire_associations and bin/nonce_expiration. Of course, you should also ensure that those scripts are executable.

When you run those scripts is entirely up to you. You could run the scripts every few minutes or daily. There should be no security concerns if the scripts are run only once each day, as the software should ensure that a nonce is not used more than once and it should ensure that associations are not validated more than once.

4 Miscellaneous

As noted elsewhere, this software is absolutely free. You can do whatever you wish with it, including using for your own domain, starting a new open source project, incorporating it in other software (including commercial software), or whatever. We recognize that this package is not complete, but we

believe it implements everything required for OpenID 2.0. It was not intended to be a commercial product, but we have found it to be sufficient for our own needs and felt it would be useful to share.

This software is not supported, though we welcome any feedback and would be happy to collaborate with anybody who has the copious spare time to manage an open source project. We do not have the time to manage such projects.

If you see performance problems where it takes several seconds for the software to respond, it might be due to the use of less efficient big integer arithmetic routines. We found that installing the Perl module called perl-Math-BigInt-GMP makes it run much faster. The time required to perform the Diffie-Hellman calculations went from several seconds to a fraction of a second.