



Published on Free Software Magazine (<http://www.freesoftwaremagazine.com>)

How to build squid authentication helpers

Build your own authentication helper using the language of your choice

By [Edmundo Carmona](#)

Have you ever tried to figure out how to make Squid authenticate users according to your own exotic rules? Users are in a DB? Are you using an ActiveDirectory? Users/passwords are authenticated by a java class? Everything is possible. Here I intend to explain how to make your own custom authentication helpers so you can develop your own routines for your own requirements.

Squid

Squid is such a wonderful HTTP cache server. It's stable, fast, highly customizable, and you barely notice it when it's working (oh and did I say it's free as in freedom?).

It comes with a number of authentication helpers, but there are times when these helpers are not enough. Sometimes you have authentication requirements exotic enough that make those default helpers useless.

The need

Suppose you have to do a little checking: you have users/passwords in a MySQL DB table. To make it a little more exotic, passwords are not directly stored, but MD5s instead. Suppose that you also want the allowed users to be listed in a text file in your file system and, *finally*, make an LDAP request to see if there's an item in the directory that matches usernames by the field named "thisCrazyField". If all that fails, the user/password can be the pair "foo/bar" (a backdoor... just in case you want to see some things that are better left anonymous in the Squid logs!). I am sure no default helper will be able to pull it off.

Before you waste more brainpower trying to figure out when your boss is going to fire you because you couldn't find a way to make this authentication scheme work with Squid (or any other HTTP cache solution for that matter), you should know that you can make a stand-alone program that can tell if a user is permitted to go through or not. Easy!

Authentication helpers

What a helper does (even a default one) is very simple: it reads username/password pairs from *Standard Input* one pair at a time in a single line of text, and writes a single line of text to *Standard Output* that either says "OK" (for a user that can go through) or "ERR" (in case of problems)—that's it. The helper has to repeat this action in an endless cycle. Username and passwords are encoded using the character encoding described in [RFC 1738](#) (section 2.2) and are separated by a white space.

What a helper does (even default ones) is very simple

How to build squid authentication helpers

Say I want to make a helper in PHP that will check if the user/password is one of the following pairs:

- hello/world
- foo/bar

Here's the PHP code:

```
<?
if (! defined(STDIN)) {
    define("STDIN", fopen("php://stdin", "r"));
}
while (!feof(STDIN)) {
    $line = trim(fgets(STDIN));
    $fields = explode(' ', $line);
    $username = rawurldecode($fields[0]); //1738
    $password = rawurldecode($fields[1]); //1738
    if ($username == 'hello'
        and $password == 'world') {
        fwrite(STDOUT, "OK\n");
    } else if ($username == 'fo'
        and $password == 'bar') {
        fwrite(STDOUT, "OK\n");
    } else {
        // failed miserably
        fwrite(STDOUT, "ERR\n");
    }
}
?>
```

That's it! I've just created a PHP-based Squid helper. Feel free to use any tool you want, be it bash, python, Perl or any other language you like. The only requirement is that the language is able to read from the standard input and write to the standard output (if you want to use bash, be careful to avoid making passwords visible with a `ps ax`).

Testing your masterpiece

Now comes the testing part. You have to act the same way Squid would have to: start the script and interact with it passing username/password pairs. If it outputs “OK” or “ERR” as wanted, then your helper is done. Here's a demonstration of the helper I just made:

```
$ php squid_helper.php
hello world
OK
foo bar
ERR
```

Oops! “foo/bar” is not okay. Go to the source code of the helper. See what's going on? I wrote `== 'foo'` instead of `== 'foo'`. Correct it in the source code and try all over again:

```
$ php squid_helper.php
hello world
OK
fo bar
ERR
foo bar
OK
```

How to build squid authentication helpers

```
other things
ERR
cool!
ERR
```

Great job! Now we know that the script really authenticates the way we want to. Our helper is ready to be used together with Squid.

Configuring Squid

You have to edit `/etc/squid/squid.conf` and add these lines, in the "OPTIONS FOR EXTERNAL SUPPORT PROGRAMS" section:

```
auth_param basic program /bin/php your_script_location
auth_param basic children 20
auth_param basic realm Username and password
auth_param basic credentialsttl 5 hours
```

What these lines tell Squid is:

program: how to run your helper? If it's an executable in itself, just call it; if it needs another interpreter, then provide all the things to make it run. In my case, I need the PHP interpreter (unless I make the script executable and provide the interpreter with a shebang line).

children: this is the number of processes that Squid will have to do concurrent authentications for all the clients. If you set too low a number and authentications are coming in too quickly, Squid will have to wait for a helper to finish an authentication cycle before trying with the next key pair. In my case, I'm creating 20 helpers.

realm: this is the text that will be shown to the user in the authentication window.

credentialsttl: this tells Squid how long an already authenticated user/password pair will be valid without needing to ask a helper to re-authenticate (remember the *children* option? Keep this in mind). In environments that change passwords too often, don't set this parameter too high.

After that, you have to use one access rule to be able to tell authenticated users from "plain" users.

```
acl AuthenticatedUsers proxy_auth REQUIRED
```

And, finally, add a rule to permit these users to go through in the "ACCESS CONTROLS" section:

```
http_access allow AuthenticatedUsers
```

Restart Squid or ask it to reload the new configuration.

Once the new configuration is working, you should be able to see as many children processes of your authentication helper as you told Squid to use (even if no client is using Squid right now) by typing `ps ax`.

In an organization where I previously worked, I created a PHP script that authenticates users against an ActiveDirectory by following this "recipe":

How to build squid authentication helpers

If the user doesn't provide the "realm", the script adds it (so users can write `REALM\username` or plain `username`).

One LDAP connection is created using the provided user/password to "bind".

If the connection is established (which means the user does exist in the directory and the password is correct), then the script checks if the user is a member of one group created in the directory named (strangely enough) `Internet`. For that, it makes a query like this (making sure the `$username` has been stripped of the `REALM\` part before the query, because it's not a part of the `sAMAccountName` field):

```
(& (sAMAccountName=$username)
(memberOf=CN=Internet,DC=OURDOMAIN,DC=ORG))
```

If the query result is empty, unfortunately the user has been rejected (because he/she doesn't belong into the `Internet` group).

That's it.

Any resource you want to use is okay

Conclusion

Any resource you want to use is okay as long as the language you use to make the helper supports it. Just remember that all Squid will provide you with is the username and the password.

Enjoy!

Biography

[Edmundo Carmona](/user/17) (/user/17" title="View user profile.): Edmundo is a Venezuelan Computer Engineer. He is working as a Freelance Java Developer in Colombia since very recently. He has also been a GNU/Linux user and consultant for several years. After years of being retired from music, he's working right now to regain his classical flute skills.

Copyright information

This article is made available under the "Attribution" Creative Commons License 2.5 available from <http://creativecommons.org/licenses/by/2.5/>.

Source URL:

http://www.freesoftwaremagazine.com/articles/authentication_with_squid
