Securing a site with CouchDB cookie authentication using node.js and nano

For JavaScript developers, the combination of **Node.js** and **CouchDB** has to be the coding equivalent of Beer and Pizza. With the addition of cookie authentication support to nano - a minimalistic driver for CouchDB built on request - the former combination just got a whole deal better.

Up until recently, nano didn't support the use of CouchDB's cookie authentication, which was a limitation for me and my current project. Using this feature greatly simplifies the authentication process of your application, as your authentication logic gets handled by your database, without having to write any additional code.

I'm pleased to say that today nano version 3.0.5 has included my pull request from a few days ago which bakes in support for cookie authentication.

Using cookie authentication in your app

Please note:

> *This run-though assumes that you've already installed Node.js and have CouchDB up and running in the default 'Admin Party' (everyone is an admin) mode on localhost, port 5984.*

Create a project folder, and install our dependencies

```
cd ~
mkdir nanocookie
cd nanocookie
npm install express@3.x nano
```

Note that I'm using express 3.0 alpha - I've found that the cookie parser in this version to be a lot more predictable

Create a file called 'app.js' and paste our simple server code

app.js

```
var express = require('express');
var app = express.createServer();
app.configure(function(){
```

```
    app.use(express.static(__dirname + '/public/'));
    app.use(express.bodyParser());
    app.use(express.cookieParser('secret-string-is-secret'));
});

app.get('/api/foo', function (req, res) {
    res.send('Not yet implemented!');
});

app.listen(3000);
```

You should be able to start up your web server, open a browser window and call **http://localhost:3000/api/foo**

```
node app.js
```

Add in nano

Now we'll update our API method, so that it queries CouchDB. For this example, we'll just do a simple list of all the databases in our Couch instance, but the same principals apply across the board:

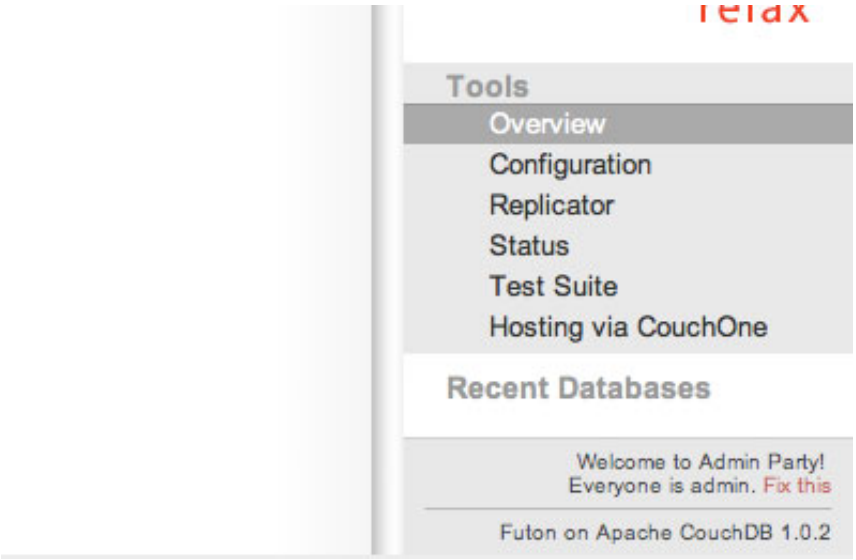
```
app.get('/api/foo', function (req, res) {
    // Nano!
    var nano = require('nano')('http://localhost:5984');

    // Create a database
    nano.db.create('alice', function (err, body) {
        if (err) {
            res.send(err.reason);
        } else {
            // All was OK, destroy the database
            nano.db.destroy('alice', function (err, body) {
                if (err) {
                    res.send(err.reason);
                } else {
                    res.send('All done!');
                }
            });
        }
    });
});
```

If you're in Admin Party mode, visiting **http://localhost:3000/api/foo** should return the message "All done!", which indicates that we successfully created, and then destroyed the database 'alice'.

Secure CouchDB

Now its time to disable Admin Party, and secure CouchDB. To do this, log into the "Futon" GUI at **http://localhost:5984/_utils** and in the bottom-right of the screen click the **Fix this** link (as per the screen shot below):



Type in and administrator username and password, for this example we'll use *admin* and *password*.

Basic Authentication

Now that we have a proper Administrator user on our CouchDB instance, visiting **http://localhost:3000/api/foo** should now return an error of "You are not a server admin."

Now, as always, we could just use Basic HTTP authentication to fix this, by simply replacing...

```
require('nano')('http://localhost:5984');
```

...with...

```
require('nano')('http://admin:password@localhost:5984');
```

Cookie Authentication - Login and Logout

In order to get our cookie set in the browser, for use in secured operations, we'll need to create a login method, so add this to **app.js**

```
app.get('/api/login', function (req, res) {

    var nano = require('nano')('http://localhost:5984'),
        username = 'admin', // your admin credentials
        userpass = 'password';
```

```
    nano.request({
            method: "POST",
            db: "_session",
            form: { name: username, password: userpass },
            content_type: "application/x-www-form-urlencoded; charset=utf-8"
        },
        function (err, body, headers) {
            if (err) { res.send(err.reason); return; }

            // Send CouchDB's cookie right on through to the client
            if (headers && headers['set-cookie']) {
                res.cookie(headers['set-cookie']);
            }

            res.send('Logged in!');
        });
});
```

Of course I'm doing a GET here, plus I'm hard coding the username and password - but I'm sure you can translate this into a real-world POST method! :)

Now we're also going to need a logout method...

```
app.get('/api/logout', function (req, res) {
    // The CouchDB cookie name is AuthSession
    res.clearCookie('AuthSession');
    res.send('Logged out!');
});
```

Cookie Authentication - Sending on the CouchDB cookie

In order to use cookie authentication in our API method, we just need to change our line that pulls in nano...

```
var nano = require('nano')('http://localhost:5984')
```

... becomes ...

```
var auth = req.cookies['AuthSession'],
    nano;

if (!auth) { res.send(401); return; }
nano = require('nano')({ url : 'http://localhost:5984', cookie: 'AuthSession=' + auth
});
```

Thats it! Try out the following steps:

1. Call **http://localhost:3000/api/foo**, you should get a 401 - Unauthorized error
2. Call **http://localhost:3000/api/login**, "Logged in!"
3. Call our /api/foo url again - this time you should get "All done!"
4. Call **http://localhost:3000/api/logout**, "Logged out!"
5. Call /api/foo once more - again you'll get a 401 error

Cookie Authentication - Final step, sliding expiration

You have just one last snippet to add. CouchDB's cookies have a sliding expiry... meaning that you'll be sent a new 'Set-Cookie' header every now and then with a new cookie value for subsequent requests. This is so that you don't get logged out as long as you're regularly calling the DB.

To pass on the new cookies we need to make the following amendment to our secured API method, replace this code...

```
// Create a database
nano.db.create('alice', function (err, body) {
    if (err) {
        res.send(err.reason);
    } else {
        // All was OK, destroy the database
        nano.db.destroy('alice', function (err, body) {
            if (err) {
                res.send(err.reason);
            } else {
                res.send('All done!');
            }
        });
    }
});
```

... with this ...

```
// Create a database
nano.db.create('alice', function (err, body, headers) {
    if (err) {
        res.send(err.reason);
    } else {
        if (headers && headers['set-cookie']) { res.cookie(headers['set-cookie']); }
        // All was OK, destroy the database
        nano.db.destroy('alice', function (err, body, headers) {
            if (err) {
                res.send(err.reason);
            } else {
```

```
                if (headers && headers['set-cookie']) { res.cookie(headers['set-cookie
']); }
                res.send('All done!');
            }
        });
    }
});
```

... and thats it - you're all done!

Jonathan Mahoney (beardtwizzle)
Senior Web Developer at Albany Software

RSS · GitHub · StackOverflow