passport-jwt

build unknown | maintainability C

A Passport strategy for authenticating with a JSON Web Token.

This module lets you authenticate endpoints using a JSON web token. It is intended to be used to secure RESTful endpoints without sessions.

Supported By

If you want to quickly add secure token-based authentication to Node.js apps, feel free to check out Auth0's Node.js SDK and free plan at auth0.com/overview 

Install

```
npm install passport-jwt
```

Usage

Configure Strategy

The JWT authentication strategy is constructed as follows:

```
new JwtStrategy(options, verify)
```

`options` is an object literal containing options to control how the token is extracted from the request or verified.

`secretOrKey` is a string or buffer containing the secret (symmetric) or PEM-encoded public key (asymmetric) for verifying the token's signature. REQUIRED unless `secretOrKeyProvider` is provided.
`secretOrKeyProvider` is a callback in the format `function secretOrKeyProvider(request, rawJwtToken, done)`, which should call `done` with a secret or PEM-encoded public key (asymmetric) for the given key and request combination. `done` accepts arguments in the format `function done(err, secret)`. Note it is up to the implementer to decode rawJwtToken. REQUIRED unless `secretOrKey` is provided.
`jwtFromRequest` (REQUIRED) Function that accepts a request as the only parameter and returns either the JWT as a string or *null*. See Extracting the JWT from the request for more details.
`issuer`: If defined the token issuer (iss) will be verified against this value.
`audience`: If defined, the token audience (aud) will be verified against this value.
`algorithms`: List of strings with the names of the allowed algorithms. For instance, ["HS256", "HS384"].
`ignoreExpiration`: if true do not validate the expiration of the token.
`passReqToCallback`: If true the request will be passed to the verify callback. i.e. verify(request, jwt_payload, done_callback).
`jsonWebTokenOptions`: passport-jwt is verifying the token using jsonwebtoken. Pass here an options object for any other option you can pass the jsonwebtoken verifier. (i.e maxAge)

`verify` is a function with the parameters `verify(jwt_payload, done)`

`jwt_payload` is an object literal containing the decoded JWT payload.
`done` is a passport error first callback accepting arguments done(error, user, info)

An example configuration which reads the JWT from the http Authorization header with the scheme 'bearer':

```
var JwtStrategy = require('passport-jwt').Strategy,
    ExtractJwt = require('passport-jwt').ExtractJwt;
var opts = {}
opts.jwtFromRequest = ExtractJwt.fromAuthHeaderAsBearerToken();
opts.secretOrKey = 'secret';
opts.issuer = 'accounts.examplesoft.com';
opts.audience = 'yoursite.net';
passport.use(new JwtStrategy(opts, function(jwt_payload, done) {
    User.findOne({id: jwt_payload.sub}, function(err, user) {
        if (err) {
            return done(err, false);
        }
        if (user) {
            return done(null, user);
        } else {
            return done(null, false);
            // or you could create a new account
        }
    });
}));
```

Extracting the JWT from the request

There are a number of ways the JWT may be included in a request. In order to remain as flexible as possible the JWT is parsed from the request by a user-supplied callback passed in as the `jwtFromRequest` parameter. This callback, from now on referred to as an extractor, accepts a request object as an argument and returns the encoded JWT string or *null*.

Included extractors

A number of extractor factory functions are provided in passport-jwt.ExtractJwt. These factory functions return a new extractor configured with the given parameters.

`fromHeader(header_name)` creates a new extractor that looks for the JWT in the given http header
`fromBodyField(field_name)` creates a new extractor that looks for the JWT in the given body field. You must have a body parser configured in order to use this method.
`fromUrlQueryParameter(param_name)` creates a new extractor that looks for the JWT in the given URL query parameter.
`fromAuthHeaderWithScheme(auth_scheme)` creates a new extractor that looks for the JWT in the authorization header, expecting the scheme to match auth_scheme.
`fromAuthHeaderAsBearerToken()` creates a new extractor that looks for the JWT in the authorization header with the scheme 'bearer'
`fromExtractors([array of extractor functions])` creates a new extractor using an array of extractors provided. Each extractor is attempted in order until one returns a token.

Writing a custom extractor function

If the supplied extractors don't meet your needs you can easily provide your own callback. For example, if you are using the cookie-parser middleware and want to extract the JWT in a cookie you could use the following function as the argument to the jwtFromRequest option:

```
var cookieExtractor = function(req) {
    var token = null;
    if (req && req.cookies)
    {
        token = req.cookies['jwt'];
    }
    return token;
};
```

Authenticate requests

Use `passport.authenticate()` specifying `'JWT'` as the strategy.

```
app.post('/profile', passport.authenticate('jwt', { session: false }),
    function(req, res) {
        res.send(req.user.profile);
    }
);
```

Include the JWT in requests

The strategy will first check the request for the standard *Authorization* header. If this header is present and the scheme matches `options.authScheme` or 'JWT' if no auth scheme was specified then the token will be retrieved from it. e.g.

```
Authorization: JWT JSON_WEB_TOKEN_STRING.....
```

If the authorization header with the expected scheme is not found, the request body will be checked for a field matching either `options.tokenBodyField` or `auth_token` if the option was not specified.

Finally, the URL query parameters will be checked for a field matching either `options.tokenQueryParameterName` or `auth_token` if the option was not specified.

Migrating

The the Migration Guide for help upgrading to the latest major version of passport-jwt

Tests

```
npm install
npm test
```

To generate test-coverage reports:

```
npm install -g istanbul
npm run-script testcov
istanbul report
```

License

The MIT License

Copyright (c) 2015 Mike Nicholson

```
npm install passport-jwt
```

4.0.0 published a month ago

https://github.com/themikenicholson/passport-jwt

MIT License

119,967 downloads in the last day

620,295 downloads in the last week

2,644,334 downloads in the last month