Hacking NodeJS applications for fun and profit

Testing NodeJS Security

by @jmortegac
Agenda

- Introduction nodeJS security
- Npm security packages
- Node Goat project
- Tools
Node JS

- JavaScript in the backend
- Built on Chrome’s Javascript runtime (V8)
- Node.js is based on event loop
- Designed to be asynchronous
- Single Thread
- Node.js is resilient to flooding attacks since there’s no limit on the number of concurrent requests.
Security updates

4.x

- 4.16.0
  - The dependency `forwarded` has been updated to address a vulnerability. This may affect your application if the following APIs are used: `req.host`, `req.hostname`, `req.ip`, `req.ips`, `req.protocol`.
  - The dependency `mime` has been updated to address a vulnerability, but this issue does not impact Express.
  - The dependency `send` has been updated to provide a protection against a Node.js 8.5.0 vulnerability. This only impacts running Express on the specific Node.js version 8.5.0.

- 4.15.5
  - The dependency `debug` has been updated to address a vulnerability, but this issue does not impact Express.
  - The dependency `fresh` has been updated to address a vulnerability. This will affect your application if the following APIs are used: `express.static`, `req.fresh`, `res.json`, `resjsonp`, `res.send`, `res.sendFile` `res.sendFile`, `res.sendStatus`.

<table>
<thead>
<tr>
<th>Advisory</th>
<th>Date of advisory</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensitive Data Exposure</strong></td>
<td>Dec 26th, 2018</td>
<td>not patched</td>
</tr>
<tr>
<td>rails-session-decoder</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cross-Site Scripting</strong></td>
<td>Dec 18th, 2018</td>
<td>patched</td>
</tr>
<tr>
<td>jingo</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Denial of Service</strong></td>
<td>Dec 18th, 2018</td>
<td>patched</td>
</tr>
<tr>
<td>markdown-it-toc-and-anchor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Npm security packages

- Helmet
- express-session
- cookie-session
- csurf
- express-validator
- bcrypt-node
- express-enforces-ssl
Security HTTP Headers

- Strict-Transport-Security
- X-Frame-Options
- X-XSS-Protection
- X-Content-Type-Options
- Content-Security-Policy
Helmet module

```javascript
var helmet = require('helmet');
app.use(helmet.noCache());
app.use(helmet.frameguard());
app.use(helmet.xssFilter());
...  
// .. or use the default initialization
app.use(helmet());
```

- [https://www.npmjs.com/package/helmet](https://www.npmjs.com/package/helmet)
Helmet module

- [https://github.com/helmetjs/helmet](https://github.com/helmetjs/helmet)

<table>
<thead>
<tr>
<th>Module</th>
<th>Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>contentSecurityPolicy</td>
<td></td>
</tr>
<tr>
<td>dnsPrefetchControl</td>
<td>✓</td>
</tr>
<tr>
<td>frameguard</td>
<td>✓</td>
</tr>
<tr>
<td>hidePoweredBy</td>
<td>✓</td>
</tr>
<tr>
<td>hpkp</td>
<td></td>
</tr>
<tr>
<td>hsts</td>
<td>✓</td>
</tr>
<tr>
<td>ieNoOpen</td>
<td>✓</td>
</tr>
<tr>
<td>noCache</td>
<td></td>
</tr>
<tr>
<td>noSniff</td>
<td>✓</td>
</tr>
<tr>
<td>referrerPolicy</td>
<td></td>
</tr>
</tbody>
</table>
Helmet module

- hidePoweredBy
- **Hpkp** → protection MITM
- **Hsts** → forces https connections
- **noCache** → desactive client cache
- **Frameguard** → protection clickjacking
- **xssFilter** → protection XSS
Helmet CSP

```javascript
var policy = {
  defaultPolicy: {
    "default-src": ["'self'"],
    "img-src": ["static.example.com"]
  }
}

helmet.csp.policy(policy);
```
# Security Report Summary

<table>
<thead>
<tr>
<th>Site:</th>
<th><a href="https://github.com/jmortega">https://github.com/jmortega</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address:</td>
<td>192.30.253.112</td>
</tr>
<tr>
<td>Report Time:</td>
<td>10 Oct 2016 11:07:05 UTC</td>
</tr>
<tr>
<td>Report Short URL:</td>
<td><a href="https://schd.io/cW5">https://schd.io/cW5</a></td>
</tr>
</tbody>
</table>

**Headers:**
- ✔️ Content-Security-Policy
- ✔️ Strict-Transport-Security
- ✔️ Public-Key-Pins
- ✔️ X-Content-Type-Options
- ✔️ X-Frame-Options
- ✔️ X-XSS-Protection

- [http://cyh.herokuapp.com/cyh](http://cyh.herokuapp.com/cyh)
- [https://securityheaders.io/](https://securityheaders.io/)
Express versions

- https://www.shodan.io/search?query=express
2 Input and output

2.1 The HTTP header "x-powered-by" is disabled in the responses.

- To avoid the framework fingerprinting.
- **Express**: Enabled by default, two options:
  - Use `app.disable('x-powered-by')`.
  - Use **Helmet** middleware ("hide-powered-by" plugin).
- **Hapi**: Disabled by default.
Disable x-powered-by

- Avoid framework fingerprinting

```javascript
var express = require("express");
var app = express();
...
app.disable("x-powered-by");
```
- Use Helmet and use "hide-powered-by" plugin

```javascript
app.use(helmet.hidePoweredBy({
  setTo: "PHP 4.2.0"
}));
```
Sessions management

- secure
- httpOnly
- domain
- path
- expires

**https://www.npmjs.com/package/cookie-session**

```javascript
var session = require('cookie-session');
var express = require('express');
var app = express();

var expiryDate = new Date(Date.now() + 60 * 60 * 1000); // 1 hour

app.use(session({
  name: 'session',
  keys: ['key1', 'key2'],
  cookie: {
    secure: true,
    httpOnly: true,
    domain: 'domain.com',
    path: '/domain',
    expires: expiryDate
  }
});
```
httpOnly & secure: true

```javascript
var express = require('express');
var session = require('express-session');
var helmet = require('helmet');

var app = express();

app.use(helmet.hsts({
  maxAge: 7776000000,
  includeSubdomains: true
}));

app.set('trust proxy', 1);

app.use(session({
  secret: 'our super secret session secret',
  cookie: {
    maxAge: 3600000,
    secure: true,
    httpOnly: true
  }
}));

app.listen(80);
```

No cookies over HTTP
XSS attacks

An attacker can exploit XSS vulnerability to:

- Steal session cookies/Sesión hijacking
- Redirect user to malicious sites
- Defacing and content manipulation
- Cross Site Request forgery
CSRF attacks

- 2.18 Protection against Cross-Site Request Forgery (CSRF) enabled. (CWE-352)
  - Express: csurf
  - Hapi: crumb
  - Deep explanation and demo included in the section A8 of NodeGoat tutorial.
app.js

var csrf = require("csurf"); //require package
app.use(csrf()); //initialize

//middleware for every http request
app.use(function(req, res, next) {
res.cookie("XSRF-TOKEN", req.csrfToken());
next();
});
app.use(function (request, response, next) {
    response.locals.csrftoken = request.csrfToken();
    next();
});

<form action="/process" method="POST">
    <input type="hidden" name="_csrf" value="{{csrfToken}}">
    <button type="submit">Submit</button>
</form>
CSRF token

```html
<!DOCTYPE html>
<html>
  <head>...
  </head>
  <body>
    <form method="post" action="/login">
      <input type="hidden" name="_csrf" value="<HTrHvVv-h2pQf7q2PK3egLwv911KOM1NNSg"> == $0
      <input type="text" name="email" placeholder="user@domain.com">
      <input type="password" name="password" placeholder="Password">
      <button type="submit">Login</button>
    </form>
  </body>
</html>
```
Filter/sanitize user input

- Fixing XSS attacks
  - https://www.npmjs.com/package/sanitizer

- Module express-validator
  - https://www.npmjs.com/package/express-validator
```javascript
var express = require('express');
var bodyParser = require('body-parser');
var validator = require('express-validator');

var app = express();
app.use(bodyParser.urlencoded());
app.use(validator());
app.use(function(req, res, next) {
  for (var item in req.body) {
    req.sanitize(item).escape();
  }
  next();
});
app.listen(80);
```
app.post('/login', function(request, response){
    request/assert('password', 'Password is required').notEmpty();
    request/assert('email', 'A email is required').notEmpty();
    request/assert('email', 'A valid email is required').isEmail();
    var errors = request/validationErrors();
    if (errors)
        response.render('index', {errors: errors});
    else
        response.render('login', {email: request.email});
});
Bcrypt-node

```javascript
var bcrypt = require('bcrypt-nodejs');

var hashPassword = function(password, callback) {
    bcrypt.genSalt(10, function(err, salt) {
        if (err) return callback(err);

        bcrypt.hash(password, salt, null, function(err, hash) {
            if (err) return callback(err);

            callback(null, hash);
        });
    });
};

var verifyPassword = function(password, hash, callback) {
    bcrypt.compare(password, hash, function(err, isMatch) {
        if (err) return callback(err);

        callback(null, isMatch);
    });
};
```
Welcome user@domain.com

the password hash is $2a$10$siqiJ3zDMZMWyD96QixiSOwHDu22BvIdCbi6n5VQ4XdmMgL.iVqoW

match password and hash

<table>
<thead>
<tr>
<th>Password</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>*********</td>
<td>$2a$10$siqiJ3zDMZMWyD</td>
</tr>
</tbody>
</table>

Verify password and hash

```plaintext
password
$2a$10$siqiJ3zDMZMWyD96QixiSOwHDu22BvIdCbi6n5VQ4XdmMgL.iVqoW
isMatch: true
```

::1 - - [16/Oct/2016:15:26:26 +0000] "POST /validateHash HTTP/1.1"

::1 - - [16/Oct/2016:15:26:26 +0000] "GET /css/style.css HTTP/1.1"
// Generate password hash
var salt = bcrypt.genSaltSync();
var passwordHash = bcrypt.hashSync(password, salt);

// Create user document
var user = {
    userName: userName,
    firstName: firstName,
    lastName: lastName,
    password: passwordHash
};

if (bcrypt.compareSync(password, user.password)) {
    callback(null, user);
} else {
    callback(invalidPasswordError, null);
}
Node Goat

- http://nodegoat.herokuapp.com/tutorial
```
this.handleContributionsUpdate = function(req, res, next) {

    /*!eslint evil: true */
    // Insecure use of eval() to parse inputs
    var preTax = eval(req.body.preTax);
    var afterTax = eval(req.body.afterTax);
    var roth = eval(req.body.roth);

    /*
    //Fix for Al -1 SSJS Injection attacks - uses alternate method to eval
    var preTax = parseInt(req.body.preTax);
    var afterTax = parseInt(req.body.afterTax);
    var roth = parseInt(req.body.roth);
    */
```
EVAL() ATACKS

```javascript
res.end(require('fs').readDirSync('..').toString())
```

![Browser window with URL and directory contents]

```
./forever,.gitignore,.heroku,.jshintrc,.nodemonignore,.profile.d,@1,Dockerfile.js,LICENSE,Procfile,README.md,app,app.json,artifacts,config,docker-compose.yml,node_modules,package.json,server.js,test
```
Insecure Direct Object References

- Use session instead of request param
- `var userId = req.session.userId;

```javascript
var userId = req.params.userId;
allocationDAO.getUserById(userId, function(err, docs) {
  if (err) return next(err);
  docs.userId = userId; // set for nav menu items
  return res.render("allocations", docs);
```
Tools

- KrakenJS
- Lusca middleware
- NodeJsScan
Give your node.js express apps some extra arms

The kraken suite
Kraken is a secure and scalable layer that extends express by providing structure and convention. Though kraken is the main pillar of our framework, the following modules can also be used independently:

- Lusca: Application security
- Kappa: NPM Proxy
- Makara: Dust H8N
- Adaro: Dust Templating

http://krakenjs.com/
app.use(lusca.csrf());
app.use(lusca.csp({ /* ... */}));
app.use(lusca.xframe('SAMEORIGIN'));
app.use(lusca.p3p('ABCDEF'));
app.use(lusca.hsts({ maxAge: 31536000 }));
app.use(lusca.xssProtection(true));
app.use(lusca.nosniff());
NodeJsScan

- https://github.com/ajinabraham/NodeJsScan

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
<th>Line</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Side Injection (SSI) - <code>eval()</code></td>
<td>User controlled data in <code>eval()</code> can result in Server Side Injection (SSI) or Remote Code Execution (RCE).</td>
<td>4</td>
<td>eval_vulnerable.js</td>
</tr>
<tr>
<td>Server Side Injection (SSI) - <code>setTimout()</code></td>
<td>User controlled data in <code>setTimout()</code> can result in Server Side Injection (SSI) or Remote Code Execution (RCE).</td>
<td>40</td>
<td>asap.js</td>
</tr>
<tr>
<td>Server Side Injection (SSI) - <code>setTimout()</code></td>
<td>User controlled data in <code>setTimout()</code> can result in Server Side Injection (SSI) or Remote Code Execution (RCE).</td>
<td>50</td>
<td>asap.js</td>
</tr>
<tr>
<td>Server Side Injection (SSI) - <code>setTimout()</code></td>
<td>User controlled data in <code>setTimout()</code> can result in Server Side Injection (SSI) or Remote Code Execution (RCE).</td>
<td>96</td>
<td>asasync.js</td>
</tr>
<tr>
<td>Server Side Injection (SSI) - <code>setTimout()</code></td>
<td>User controlled data in <code>setTimout()</code> can result in Server Side Injection (SSI) or Remote Code Execution (RCE).</td>
<td>86</td>
<td>async.js</td>
</tr>
<tr>
<td>Server Side Injection (SSI) - <code>setTimout()</code></td>
<td>User controlled data in <code>setTimout()</code> can result in Server Side Injection (SSI) or Remote Code Execution (RCE).</td>
<td>94</td>
<td>bluebird core.js</td>
</tr>
<tr>
<td>Server Side Injection (SSI) - <code>setTimout()</code></td>
<td>User controlled data in <code>setTimout()</code> can result in Server Side Injection (SSI) or Remote Code Execution (RCE).</td>
<td>131</td>
<td>bluebird core.js</td>
</tr>
<tr>
<td>Server Side Injection (SSI) - <code>new Function()</code></td>
<td>User controlled data in <code>new Function()</code> can result in Server Side Injection (SSI) or Remote Code Execution (RCE).</td>
<td>1616</td>
<td>bluebird core.js</td>
</tr>
</tbody>
</table>
https://github.com/jmortega/NodeJsScan/blob/master/rules.xml
<notpresent name="Missing Security Header - X-XSS-Protection:1">
  <signature>helmet.xssFilter(() => usca.xssProtection(true)) | X-XSS-Protection(s=*1, )</signature>
  <description>X-XSS-Protection header set to 1 enables the Cross-site scripting (XSS) filter built into most recent web browsers.</description>
</notpresent>

<notpresent name="Missing Security Header - X-Content-Type-Options">
  <signature>helmet.noSniff(require('!''!''!''!'' dont-sniff-minetype('''!''!''!''!'' )
      | X-Content-Type-Options(s=,'''!''!''!''!'' nosniff)</signature>
  <description>X-Content-Type-Options header prevents Internet Explorer and Google Chrome from MIME-sniffing a response away from the declared content-type.</description>
</notpresent>

<notpresent name="Missing Security Header - X-Download-Options: noopen">
  <signature>require('''!''!''!''!'' ienoppen('''!''!''!''!''
      | ienoppen((helmet.iенoppen(() | X-Download-Options(s=*:1, )
      | nosniff())</signature>
  <description>X-Download-Options header set to noopen prevents IE users from directly opening and executing downloads in your site's context.</description>
</notpresent>

<notpresent name="Missing 'httpOnly' in Cookie">
  <signature>httpOnly(s=true) | httpOnly</signature>
  <description>JavaScript can access Cookies if they are not marked httpOnly.</description>
</notpresent>

<notpresent name="Information Disclosure - X-Powered-By">
  <signature>.disable((''!''!''!''!'' x-powered-by('''!''!''!''!'' )
      | require('''!''!''!''!'' hide-powered-by('''!''!''!''!'' )
      | hidePoweredBy((helmet.hidePoweredBy(() | removeHeader('''!''!''!''!'' )
      | X-Powered-By('''!''!''!''!'' )</signature>
  <description>Remove the X-Powered-By header to prevent information gathering.</description>
</notpresent>
GitHub repositories

- https://github.com/jmortega/testing_nodejs_security
- https://github.com/cr0hn/vulnerable-node
- https://github.com/rdegges/svcc-auth
- https://github.com/strongloop/loopback-getting-started-intermediate
- https://github.com/Feeld/strong-node
Node security learning

Books

Secure Your Node.js Web Application
Keep Attackers Out and Users Happy

Karl Dūūna
edited by Fahmida Y. Rashid
References

- https://blog.risingstack.com/node-js-security-checklist/
- https://blog.risingstack.com/node-js-security-tips/
- https://www.npmjs.com/package/helmet
- http://nodegoat.herokuapp.com/tutorial