Blockade Documentation

Release 1.0.4

Caleb Kinney

Apr 15, 2019

Contents:

Blockade is a lightweight package that adds optional security headers and cookie attributes for Node.js web frameworks.

Security HTTP headers and cookie attributes help enhance the security of your web application by enabling built-in browser security mechanisms.

CHAPTER 1

Supported Node.js web frameworks:

AdonisJs, Express, Fastify, hapi, Koa, Meteor, Nest, Polka, restify, Sails, Total.js

CHAPTER 2

Install

```
$ npm i blockade
```

After installing Blockade:

```
const blockade = require("blockade");

const secureHeaders = new blockade.SecureHeaders();
const secureCookie = new blockade.SecureCookie();
```

CHAPTER 3

Documentation

3.1 Secure Headers

Security Headers are HTTP response headers that, when set, can enhance the security of your web application by enabling browser security policies.

You can assess the security of your HTTP response headers at securityheaders.com

Recommendations used by secure,py and more information regarding security headers can be found at the OWASP Secure Headers Project .

3.1.1 Server

Contain information about server software
Default Value: `NULL` *(obfuscate server information, not included by default)*

3.1.2 Strict-Transport-Security (HSTS)

Ensure application communication is sent over HTTPS
Default Value: `max-age=63072000; includeSubdomains`

3.1.3 X-Frame-Options (XFO)

Disable framing from different origins (clickjacking defense)
Default Value: `SAMEORIGIN`

3.1.4 X-XSS-Protection

Enable browser cross-site scripting filters
Default Value: `1; mode=block`

3.1.5 X-Content-Type-Options

Prevent MIME-sniffing
Default Value: `nosniff`

3.1.6 Content-Security-Policy (CSP)

Prevent cross-site injections
Default Value: `script-src 'self'; object-src 'self'` *(not included by default)**

3.1.7 Referrer-Policy

Enable full referrer if same origin, remove path for cross origin and disable referrer in unsupported browsers
Default Value: `no-referrer, strict-origin-when-cross-origin`

3.1.8 Cache-control / Pragma / Expires

Prevent cacheable HTTPS response
Default Value: `no-cache, no-store, must-revalidate, max-age=0` / `no-cache` / `0`

3.1.9 Feature-Policy

Disable browser features and APIs
Default Value: `accelerometer 'none'; ambient-light-sensor 'none'; autoplay 'none'; camera 'none'; encrypted-media 'none'; fullscreen 'none'; geolocation 'none'; gyroscope 'none'; magnetometer 'none'; microphone 'none'; midi 'none'; payment 'none'; picture-in-picture 'none'; speaker 'none'; sync-xhr 'none'; usb 'none'; vr 'none';` *(not included by default)*

3.1.10 Additional information

- The `Strict-Transport-Security` (HSTS) header will tell the browser to **only** utilize secure HTTPS connections for the domain, and in the default configuration, including all subdomains. The HSTS header requires trusted certificates and users will unable to connect to the site if using self-signed or expired certificates. The browser will honor the HSTS header for the time directed in the `max-age` attribute *(default = 2 years)*, and setting the `max-age` to `0` will disable an already set HSTS header. Use the `{ hsts: false }` option to not include the HSTS header in Secure Headers.
- The `Content-Security-Policy` (CSP) header can break functionality and can (and should) be carefully constructed, use the `{ csp : true }` option to enable default values.

3.1.11 Usage

```
const secureHeaders = new blockade.SecureHeaders();
secureHeaders.framework(response);
```

Default HTTP response headers:

```
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Referrer-Policy: no-referrer, strict-origin-when-cross-origin
Cache-control: no-cache, no-store, must-revalidate, max-age=0
Pragma: no-cache
Expires: 0
```

3.1.12 Options

You can toggle the setting of headers with default values by passing an object with `new blockade.Header().default()` or `new blockade.Header().notSet()` and override default values by passing `new blockade.Header().set("custom")` or policy to the following options:

- `server` - set the Server header, e.g. `new blockade.Server().set("Blockade")` - (default= `default=Server().notSet()`)
- `hsts` - set the Strict-Transport-Security header - (default= `HSTS().default()`)
- `xfo` - set the X-Frame-Options header - (default= `XFO().default()`)
- `xxp` - set the X-XSS-Protection header - (default= `XXP().default()`)
- `content` - set the X-Content-Type-Options header - (default= `Content().default()`)
- `csp` - set the Content-Security-Policy - (default= `CSP().notSet()`)
- `referrer` - set the Referrer-Policy header - (default= `Referrer().default()`)
- `cache` - set the Cache-control and Pragma headers - (default= `Cache().default()`)
- `feature` - set the Feature-Policy header - (default= `Feature().notSet()`)

Example:

```
const blockade = require("blockade");

const secureHeaders = new blockade.SecureHeaders({
  server: "Blockade",
  csp: true,
  hsts: false
});

. . .

secureHeaders.framework(response)

  . . .
```

3.2 Secure Cookies

3.2.1 Path

The Path directive instructs the browser to only send the cookie if provided path exists in the URL.

3.2.2 Secure

The Secure flag instructs the browser to only send the cookie via HTTPS.

3.2.3 HttpOnly

The HttpOnly flag instructs the browser to not allow any client side code to access the cookie's contents.

3.2.4 SameSite

The SameSite flag directs the browser not to include cookies on certain cross-site requests. There are two values that can be set for the same-site attribute, lax or strict. The lax value allows the cookie to be sent via certain cross-site GET requests, but disallows the cookie on all POST requests. For example cookies are still sent on links `<a href="x">`, prerendering `<link rel="prerender" href="x"` and forms sent by GET requests `<form-method="get"...`, but cookies will not be sent via POST requests `<form-method="post"...`, images `<img src="x">` or iframes `<iframe src="x">`. The strict value prevents the cookie from being sent cross-site in any context. Strict offers greater security but may impede functionality. This approach makes authenticated CSRF attacks impossible with the strict flag and only possible via state changing GET requests with the lax flag.

3.2.5 Expires

The Expires attribute sets an expiration date for persistent cookies.

3.2.6 Usage

```
const secureCookie = new blockade.SecureCookie();
secureCookie.framework(response, "foo", "bar");
```

Default Set-Cookie HTTP response header:

```
Set-Cookie: foo=bar; Path=/; secure; HttpOnly; SameSite=lax
```

3.2.7 Options

You can modify default cookie attribute values by passing the following options:

- `name` - set the cookie name *(string, No default value)*
- `value` - set the cookie value *(string, No default value)*
- `path` - set the Path attribute, e.g. `path="/blockade"` *(string, default="/")*

- `secure` - set the Secure flag *(bool, default=True)*
- `httpOnly` - set the HttpOnly flag *(bool, default=True)*
- `sameSite` - set the SameSite attribute, e.g. `{value: "Strict" }`, `{value: "Lax" }` *or* `False` *default={value: "Lax" }*
- `expires` - set the Expires attribute with the cookie expiration in hours, e.g. `expires=1` *(number / bool, default=False)*

Example:

```
const blockade = require("blockade");

const secureCookie = new blockade.SecureCookie({
  sameSite: { value: "Strict" },
  expires: 1
});

secureCookie.framework(response, "foo", "bar");
```

3.3 Policy Builder

3.3.1 CSP()

Directives: `baseUri(sources)`, `blockAllMixedContent()`, `connectSrc(sources)`, `defaultSrc(sources)`, `fontSrc(sources)`, `formAction(sources)`, `frameAncestors(sources)`, `frameSrc(sources)`, `imgSrc(sources)`, `manifestSrc(sources)`, `mediaSrc(sources)`, `objectSrc(sources)`, `pluginTypes(types)`, `reportTo(json_object)`, `reportUri(uri)`, `requireSriFor(values)`, `sandbox(values)`, `scriptSrc(sources)`, `styleSrc(sources)`, `upgradeInsecureRequests()`, `workerSrc(sources)`

Example:

```
const cspValue = new blockade.CSP()
  .defaultSrc(blockade.values.none)
  .baseUri(blockade.values.self)
  .blockAllMixedContent()
  .connectSrc(blockade.values.self, "api.spam.com")
  .frameSrc(blockade.values.none)
  .imgSrc(blockade.values, "static.spam.com").value;

// default-src 'none'; base-uri 'self'; block-all-mixed-content; connect-src 'self'␣
↪api.spam.com; frame-src 'none'; img-src [object Object] static.spam.com
```

You can check the effectiveness of your CSP Policy at the CSP Evaluator

3.3.2 HSTS()

Directives: `includeSubDomains()`, `maxAge(seconds)`, `preload()`

Example:

```
const hstsValue = new blockade.HSTS()
  .includeSubdomains()
  .preload()
  .maxAge(blockade.seconds.oneMonth).value;

// includeSubDomains; preload; max-age=2592000
```

3.3.3 XXP()

Directives: `disabled()` = 0, `enabled()` = 1, `enabledBlock()` = 1; mode=block, `enabledReport(uri)` = 1; report=uri

Example:

```
 const xxpValue = new blockade.XXP().enabledBlock().value;

// 1; mode=block
```

3.3.4 XFO()

Directives: `allow_from(uri)`, `deny()`, `sameorigin()`

Example:

```
 const xfoValue = new blockade.XFO().deny().value;

// deny
```

3.3.5 Referrer()

Directives: `noReferrer()`, `noReferrerWhenDowngrade()`, `origin()`, `originWhenCrossOrigin()`, `sameOrigin()`, `strictOrigin()`, `strictOriginWhenCrossOrigin()`, `unsafeUrl()`

Example:

```
 const referrerValue = new blockade.Referrer().noReferrer().value;

// no-referrer
```

3.3.6 Feature()

Directives: `accelerometer(allowlist)`, `ambient_light_sensor(allowlist)`, `autoplay(allowlist)`, `camera(allowlist)`, `document_domain(allowlist)`, `encrypted_media(allowlist)`, `fullscreen(allowlist)`, `geolocation(allowlist)`, `gyroscope(allowlist)`, `magnetometer(allowlist)`, `microphone(allowlist)`, `midi(allowlist)`, `payment(allowlist)`, `picture_in_picture(allowlist)`, `speaker(allowlist)`, `sync_xhr(allowlist)`, `usb(allowlist)`, `Values(allowlist)`, `vr(allowlist)`

Example:

```
const featureValue = new blockade.Feature()
  .geolocation(blockade.values.self, "spam.com")
  .vibrate(blockade.values.none).value;

// geolocation 'self' spam.com; vibrate 'none'
```

3.3.7 Cache()

Directives: `immutable()`, `maxAge(seconds)`, `maxStale(seconds)`, `minFresh(seconds)`, `mustRevalidate()`, `noCache()`, `noStore()`, `noTransform()`, `only_if_cached()`, `private()`, `proxyRevalidate()`, `public()`, `sMaxage(seconds)`, `staleIfError(seconds)`, `staleWhileRevalidate(seconds)`,

Example:

```
const cacheValue = new blockade.Cache()
 .noStore()
 .mustRevalidate()
 .proxyRevalidate().value;

 // no-store, must-revalidate, proxy-revalidate
```

3.3.8 seconds

Values: `fiveMinutes` = “300”, `oneWeek` = “604800”, `oneMonth` = “2592000”, `oneYear` = “31536000”, `twoYears` = “63072000”

3.3.9 values

Values: `all` = “*”, `none` = “‘none’”, `self` = “‘self’”, `src` = “‘src’”, `strictDynamic` = “‘strict-dynamic’”, `unsafeEval` = “‘unsafe-eval’”, `unsafeInline` = “‘unsafe-inline’”

3.3.10 Usage

Example:

```
const express = require("express");
const blockade = require("blockade");
const app = express();
const port = 3000;

const cspValue = new blockade.CSP()
  .defaultSrc(blockade.values.none)
  .baseUri(blockade.values.self)
  .blockAllMixedContent()
  .connectSrc(blockade.values.self, "api.spam.com")
  .frameSrc(blockade.values.none)
  .imgSrc(blockade.values, "static.spam.com").value;

const hstsValue = new blockade.HSTS()
  .includeSubdomains()
```

(continues on next page)

(continued from previous page)

```
  .preload()
  .maxAge(blockade.seconds.oneMonth).value;

const xxpValue = new blockade.XXP().enabledBlock().value;

const xfoValue = new blockade.XFO().deny().value;

const referrerValue = new blockade.Referrer().noReferrer().value;

const featureValue = new blockade.Feature()
  .geolocation(blockade.values.self, "spam.com")
  .vibrate(blockade.values.none).value;

const cacheValue = new blockade.Cache()
  .noStore()
  .mustRevalidate()
  .proxyRevalidate().value;

const secureHeaders = new blockade.SecureHeaders({
  csp: cspValue,
  hsts: hstsValue,
  xxp: xxpValue,
  xfo: xfoValue,
  referrer: referrerValue,
  feature: featureValue,
  cache: cacheValue
});

app.use(function(req, res, next) {
  secureHeaders.express(res);
  next();
});

...
```

Response Headers:

```
Strict-Transport-Security: includeSubDomains; preload; max-age=2592000
X-Frame-Options: deny
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'none'; base-uri 'self'; block-all-mixed-content;
↪ connect-src 'self' api.spam.com; frame-src 'none'; img-src [object Object] static.
↪spam.com
Referrer-Policy: no-referrer
Cache-control: no-store, must-revalidate, proxy-revalidate
Feature-Policy: geolocation 'self' spam.com; vibrate 'none'
```

3.4 Supported Frameworks

3.4.1 Framework Agnostic

Return Dictionary of Headers:

`secureHeaders.headers()`

Example:

```
const secureHeaders = new blockade.SecureHeaders({ csp: true, feature: true });
return secureHeaders.headers()
```

Return Value:

```
{
  'Strict-Transport-Security': 'max-age=63072000; includeSubdomains',
  'X-Frame-Options': 'SAMEORIGIN',
  'X-XSS-Protection': '1; mode=block',
  'X-Content-Type-Options': 'nosniff',
  'Content-Security-Policy': "script-src 'self'; object-src 'self'",
  'Referrer-Policy': 'no-referrer, strict-origin-when-cross-origin',
  Pragma: 'no-cache',
  Expires: '0',
  'Cache-control': 'no-cache, no-store, must-revalidate, max-age=0',
  'Feature-Policy':
   "accelerometer 'none'; ambient-light-sensor 'none'; autoplay 'none'; camera 'none';
↪ encrypted-media 'none'; fullscreen 'none'; geolocation 'none'; gyroscope 'none';␣
↪magnetometer 'none'; microphone 'none'; midi 'none'; payment 'none'; picture-in-
↪picture 'none'; speaker 'none'; sync-xhr 'none'; usb 'none'; vr 'none'"
}
```

3.4.2 AdonisJs

Headers

`secureHeaders.adonis(response)`

Example:

```
const blockade = require("blockade");
const secureHeaders = new blockade.SecureHeaders();

class Blockade {
  async handle({ response }, next) {
    secureHeaders.adonis(response);
    await next();
  }
}
module.exports = Blockade;
```

Cookies

Coookies

`ecureCookie.adonis(response, name, value)`

Example:

```
const blockade = require("blockade");
const secureCookie = new blockade.SecureCookie();

. . .
```

(continues on next page)

(continued from previous page)

```
Route.get("/blockade", ({ response }) => {
  secureCookie.adonis(response, "foo", "bar");
  response.send("Blockade");
});

. . .
```

3.4.3 Express

Headers

`secureHeaders.express(res)`

Example:

```
const express = require("express");
const blockade = require("blockade");
const app = express();
const port = 3000;

const secureHeaders = new blockade.SecureHeaders();
. . .

app.use(function(req, res, next) {
  secureHeaders.express(res);
  next();
});

. . .
```

Cookies

`secureCookie.express(res, name, value)`

Example:

```
const express = require("express");
const blockade = require("blockade");
const app = express();
const port = 3000;

const secureCookie = new blockade.SecureCookie();

. . .

app.get("/blockade", function(req, res) {
  secureCookie.express(res, "foo", "bar");
  res.send("blockade");
});

. . .
```

3.4.4 Fastify

Headers

`secureHeaders.fastify(reply)`

Example:

```
const fastify = require("fastify")();

const blockade = require("blockade");
const secureHeaders = new blockade.SecureHeaders();

. . .

fastify.addHook("preHandler", async (request, reply) => {
  secureHeaders.fastify(reply);
});

. . .
```

Cookies

`secureCookie.fastify(reply, name, value)`

Example:

```
const fastify = require("fastify")();

const blockade = require("blockade");
const secureCookie = new blockade.SecureCookie();

. . .

fastify.get("/", function(request, reply) {
  secureCookie.fastify(reply, "foo", "bar");
  reply.send({ blockade: true });
});

. . .
```

3.4.5 hapi

Headers

`secureHeaders.hapi(response)`

Example:

```
const Hapi = require("hapi");
const blockade = require("blockade");

const secureHeaders = new blockade.SecureHeaders();

. . .
```

(continues on next page)

(continued from previous page)

```
server.ext("onPreResponse", (request, h) => {
  const response = request.response;
  secureHeaders.hapi(response);
  return response;
});

. . .
```

Cookies

`secureCookie.hapi(h, name, value)`

Example:

```
const Hapi = require("hapi");
const blockade = require("blockade");

const secureCookie = new blockade.SecureCookie();

. . .

server.route({
  method: "GET",
  path: "/blockade",
  handler: function(request, h) {
    secureCookie.hapi(h, "foo", "bar");
    const response = h.response("blockade");
    return response;
  }
});

. . .
```

3.4.6 Koa

Headers

`secureHeaders.koa(ctx)`

Example:

```
const Koa = require("koa");
const blockade = require("blockade");

const secureHeaders = new blockade.SecureHeaders();
. . .

app.use(async (ctx, next) => {
  await next();
  secureHeaders.koa(ctx);
});

. . .
```

Cookies

`secureCookie.koa(ctx, name, value)`

Example:

```
const Koa = require("koa");
const app = new Koa();
const blockade = require("blockade");

const secureCookie = new blockade.SecureCookie();

. . .

app.use(async ctx => {
  ctx.body = "Blockade";
  secureCookie.koa(ctx, "foo", "bar");
});

. . .
```

3.4.7 Meteor

Headers

`secureHeaders.meteor(res)`

Example:

```
import { Meteor } from "meteor/meteor";
import { SecureHeaders } from "blockade";

const secureHeaders = new SecureHeaders({});

var connectHandler = WebApp.connectHandlers;

Meteor.startup(function() {
  connectHandler.use(function(req, res, next) {
    secureHeaders.meteor(res);
    return next();
  });
});

. . .
```

Cookies

Meteor does not support cookies naively, please see https://atmospherejs.com/?q=cookie for cookie support packages.

3.4.8 Nest

Headers

`secureHeaders.nest(res)`

Example:

```
import { SecureHeaders } from 'blockade';

const secureHeaders = new SecureHeaders({});

export function blockade(req, res, next) {
  secureHeaders.nest(res);
  next();
}
```

Cookies

`secureCookie.nest(res, name, value)`

Example:

```
import { Controller, Get, Post, Res, HttpStatus } from '@nestjs/common';
import { AppService } from './app.service';
import { SecureCookie, SameSite } from 'blockade';
const secureCookie = new SecureCookie({});

@Controller()
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get('blockade')
  getHello(@Res() res): string {
    secureCookie.nest(res, 'foo', 'bar');
    return res.status(HttpStatus.OK).json([]);
  }
}
```

3.4.9 Polka

Headers

`secureHeaders.polka(res)`

Example:

```
const polka = require("polka");
const blockade = require("blockade");
const secureHeaders = new blockade.SecureHeaders();

function headers(req, res, next) {
  secureHeaders.polka(res);
  next();
}

polka()
  .use(headers)
  .get("/", (req, res) => {
    res.end(`Blockade`);
  })
```

(continues on next page)

(continued from previous page)

```
  .listen(3000, err => {
    if (err) throw err;
    console.log(`> Running on localhost:3000`);
  });
```

Cookies

`secureCookie.polka(res, name, value)`

Example:

```
const polka = require("polka");
const blockade = require("blockade");
const secureCookie = new blockade.SecureCookie();

polka()
  .get("/", (req, res) => {
    secureCookie.polka(res, "foo", "bar");
    res.end(`Blockade`);
  })
  .listen(3000, err => {
    if (err) throw err;
    console.log(`> Running on localhost:3000`);
  });
```

3.4.10 restify

Headers

`secureHeaders.restify(res)`

Example:

```
var restify = require("restify");
const blockade = require("blockade");
const secureHeaders = new blockade.SecureHeaders();

function respond(req, res, next) {
  res.send("Blockade");
  next();
}

function headers(req, res, next) {
  secureHeaders.restify(res);
  next();
}

. . .

var server = restify.createServer();
server.pre(headers);
server.get("/", respond);
```

Cookies

`secureCookie.restify(res, name, value)`

Example:

```
var restify = require("restify");
const blockade = require("blockade");
const secureCookie = new blockade.SecureCookie();

function respond(req, res, next) {
  secureCookie.restify(res, "foo", "bar");
  res.send("Blockade");
  next();
}

. . .

var server = restify.createServer();
server.get("/", respond);
```

3.4.11 Sails

Headers

`secureHeaders.sails(res)`

Example:

```
const blockade = require("blockade");
const secureHeaders = new blockade.SecureHeaders();

module.exports.http = {
  middleware: {
    order: ["blockade"],

    blockade: (function() {
      return function(req, res, next) {
        secureHeaders.sails(res);
        return next();
      };
    })()
  }
};
```

Cookies

`secureCookie.sails(res, name, value)`

Example:

```
const blockade = require("blockade");
const secureCookie = new blockade.SecureCookie();

module.exports = {
```

(continues on next page)

(continued from previous page)

```
  blockade: function(req, res) {
    secureCookie.sails(res, "foo", "bar");
    return res.send("Blockade");
  }
};
```

3.4.12 Total.js

Headers

`secureHeaders.total(response)`

Example:

```
const blockade = require("blockade");
const secureHeaders = new blockade.SecureHeaders();

exports.install = function() {
  ROUTE("/", view_index);
};

function view_index() {
  var response = this;
  secureHeaders.total(response);
  response.view("index");
}
```

Cookies

`secureCookie.total(response, name, value)`

Example:

```
const blockade = require("blockade");
const secureCookie = new blockade.SecureCookie();

exports.install = function() {
  ROUTE("/", view_index);
};

function view_index() {
  var response = this;
  secureCookie.total(response, "foo", "bar");
  response.view("index");
}
```

3.5 Resources

3.5.1 Frameworks

- AdonisJs - NodeJs Web Application Framework. Makes it easy for you to write webapps with less code

- Express - Fast, unopinionated, minimalist web framework for node.
- Fastify - Fast and low overhead web framework, for Node.js
- hapi - Server Framework for Node.js
- Koa.js - Next generation web framework for Node.js
- Meteor, - Meteor, the JavaScript App Platform
- Nest, - A progressive Node.js framework for building efficient and scalable server-side applications on top of TypeScript & JavaScript (ES6, ES7, ES8) heavily inspired by Angular
- Polka - A micro web server so fast, it'll make you dance!
- restify - The future of Node.js REST development
- Sails, - Realtime MVC Framework for Node.js
- Total.js - Node.js framework

3.5.2 General

- OWASP - Secure Headers Project
- OWASP - Session Management Cheat Sheet
- Mozilla Web Security
- securityheaders.com

3.5.3 Policies

- **CSP:** CSP Cheat Sheet | Scott Helme, Content-Security-Policy | MDN, Content Security Policy Cheat Sheet | OWASP, Content Security Policy CSP Reference & Examples
- **XXP:** X-XSS-Protection | MDN
- **XFO:** X-Frame-Options | MDN
- **HSTS:** Strict-Transport-Security | MDN, HTTP Strict Transport Security Cheat Sheet | OWASP
- **Referrer:** A new security header: Referrer Policy | Scott Helme, Referrer-Policy | MDN
- **Feature:** A new security header: Feature Policy | Scott Helme, Feature-Policy | MDN, Introduction to Feature Policy | Google Developers
- **Cache:** Cache-Control | MDN

CHAPTER 4

Indices and tables

- genindex
- modindex
- search