Advisories - Security vulnerabilities
=====================================

* qs module: < 1.0.0

==

indirect vulnerabilities:

- restler
    * qs used: https://github.com/danwrong/restler/blob/master/lib/restler.js
    * https://github.com/danwrong/restler/issues/186
    * related to the NSP
    * result of discussion: resolved

- rrestjs
    * qs used in multiple places
    * no PR or issue
- hippie
    * qs used: https://github.com/vesln/hippie/blob/408a162154bc0840c4d56348e26e8463e23cb07e/lib/hippie/serializers.js
    * no issue or PR
    * test framework!
- mailchimp
    * qs used: https://github.com/gomfunkel/node-mailchimp/blob/cb918090858242bd33a440520952da704136105a/lib/mailchimp/MailChimpWebhook.js
    * no issue or PR
    * API
- requester
    * A simple network request helper that is geared towards crawling
    * qs used
        - https://github.com/icodeforlove/node-requester/blob/6363cf518e472a3214c5964e0450047a81da0eb0/lib/RequesterHandler.js
        - https://github.com/icodeforlove/node-requester/blob/f9e5c3b6f714030dcf1a4f3cc7f5c06b6bbc37ea/lib/Requester.js
    * no PR or issue
- livewire
    * declared, but not used

- bugsnag-notification-plugins
    * Bugsnag.com can notify you when errors happen in your applications.
    * https://github.com/bugsnag/bugsnag-notification-plugins/blob/28a9f8f5d77bd317938c993ff5640c2da4b89dde/plugins/sprintly/index.coffee
    * no issue or PR
- apostrophe-twitter
    * A twitter widget for Apostrophe webpages.
    * used: https://github.com/punkave/apostrophe-twitter/blob/ce998dd8008a112464c2fdc505a28d366919c0d5/index.js
    * no issue or PR
- https://skimdb.npmjs.com/_utils/document.html?registry/node-linkedin-simple
    * used:
        - https://github.com/paulsamchuk/node-linkedin/blob/29288de434821d88ed7ac4ca56ea33433e8e4fbe/lib/linked_in.js
        - https://github.com/paulsamchuk/node-linkedin/blob/29288de434821d88ed7ac4ca56ea33433e8e4fbe/lib/oauth2.js
    * update on gh but not npm
- okanjo:
    * This library provides easy integration with the Okanjo platform.
    * https://github.com/Okanjo/okanjo-nodejs/blob/1e33eb314095fd15d1c145527492096ee61aef58/lib/provider.js
    * no PR or issue

==

cascading vulnerabilities:

- tesla
    * superagent(0.18)<-qs
    * qs is used: https://github.com/visionmedia/superagent/blob/608cda651db3ac1559a604032878bae3ac16efd7/lib/node/parsers/urlencoded.js, urlencoded is used in server.js
    * superagent not used in the code
- radiatus
    * Multiuser web server for freedom.js
    * body-parser(~1.5.2)<-qs
    * qs code used in urlencoded in body-parser

* radiatus use body-parser:
https://github.com/freedomjs/radiatus/blob/b3c0990df769ead9abbeef27f7cd5c676c3f939d/app.js

* radiatus make use of this function:

```
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}));
```

* update of version required

* no PR or issues

- tarifa

* tarifa is a CLI on top of Apache Cordova. It aims at simplifying the Apache Cordova workflow and adding features to complete cordova toolchain such as:

* restler(3.2.2)->qs(0.6.6)

restler code use:

```
'Should serialize POST body': function(test) {
  rest.post(host, { data: { q: 'balls' } }).on('complete', function(data) {
    test.re(data, /content-type\: application\/x-www-form-urlencoded/, 'should set content-type');
    test.re(data, /content-length\: 7/, 'should set content-length');
    test.re(data, /\r\n\r\nq=balls/, 'should have balls in the body');
    test.done();
  });
```

* restler used:
https://github.com/TarifaTools/tarifa/blob/655b8f6aec452c04e7f149406bf63633e0c6c8dc/actions/watch/index.js

* injection based attack: restler makes a post

- component

* tiny-lr-fork(0.0.5)<-qs(~0.5.2)

* qs code used:

```
Server.prototype.handle = function handle(req, res, next) {
        var url = parse(req.url);
        var middleware = typeof next === 'function';

        req.body = {};
        req.params = {};

        try {
          req.body = JSON.parse(req.data);
        } catch(e) {}

        if(url.query) req.params = qs.parse(url.query);

        // todo: parse Accept header
        res.setHeader('Content-Type', 'application/json');
```

```
  // do the routing
  var route = req.method + ' '  + url.pathname;
  var respond = this.emit(route, req, res);
  if(respond) return;
  if(middleware) return next();

  res.writeHead(404);
  res.write(JSON.stringify({
    error: 'not_found',
    reason: 'no such route'
  }));
  res.end();
};
```

* component in basically a tiny-lr-fork
* not critical vulnerability, its more or less a build process tool
* superagent(0.17.0)<-qs
* superagent used here: https://github.com/componentjs/component/blob/f453af038d526afbf9c80a565d7239ce9edf7cf0/bin/component-crawl
* Component is a vertically integrated frontend solution, handling everything from package management to the build process
* testing!

- juice2
    * Given HTML, juice will inline your CSS properties into the style attribute.
    * dev purpose
    * superagent(~0.18.20->qs
    * this is a fork
    * superagent used: https://github.com/andrewrk/juice/blob/master/lib/juice.js
    * not super vulnerable, its used for convertin to inline html
- ripple-lib
    * superagent(^0.18.0)->qs
    * used in multiple places
    * its an API for ripple, could be a potential
- atomify
    * tiny-lr-fork just declered but not used
    * css related stuff, too much to investigate
- loopback-component-storage
    * pkgcloud(0.9.6)<-qs
    * update version
    * pkgcloud use it in client request: https://github.com/pkgcloud/pkgcloud/blob/8f4bf997a0f6af28b727c3fd7af026c8a4b55368/lib/pkgcloud/core/base/client.js

    * suspect code usage:

```
function getProvider(provider) {
  try {
    // Try to load the provider from providers folder
    return require('./providers/' + provider);
  } catch (err) {
    // Fall back to pkgcloud
    return require('pkgcloud').providers[provider];
  }
}
```

    * possiblity of user input? investigate further

- shout
    * A multi-user web IRC client http://shout-irc.com/
    * superagent(^0.18.2)->qs(0.6.6)
    * code example used here:
https://github.com/erming/shout/blob/1a620e1d54209ed9cc42e49804ac27487e5bd84f/src/plugins/irc-events/link.js
    * user enter url, could lead to a DOS
    * solved now, replaced with request
- open.core
    * Common utility functionality used between multiple applications.
    * restler(>= 0.2.1)<-qs(0.6.6)
    * used here:
https://github.com/philcockfield/open.core/blob/6143df23e4e4937560dc78f8ccd02d54bfe3fc43/lib/src/server/util/converters/pygments.coffee
    * seems like its not vulnerable
- sendwithus
    * restler(~3.2.2)<-qs(0.6.6)
    * used for makin request:

https://github.com/sendwithus/sendwithus_nodejs/blob/8d56212d007341b32963d048b4426b03fc850f71/lib/sendwithus.js
    * could be vulnerable
    * Sendwithus NodeJS Client https://www.sendwithus.com
- taskcluster-base
    * Common modules use in taskcluster components
    * superagent(0.18.2)->qs(0.6.6)
    * used in test cases only
- manta
    * restify(2.8.1)->qs(0.6.6)
    * restify:
https://github.com/mcavage/node-restify/blob/885f1b95cf97a144ad10778d9dfa00fdfca6878c/lib/plugins/query.js

* part of the query string function
* used in multiple places:
- logging: https://github.com/joyent/node-manta/blob/25e6f9c31c49e3b2a3714f20575d3e9e584460b2/test/helper.js
- client: https://github.com/joyent/node-manta/blob/564629a2ae5e04b637efdc3bf420946f33933898/lib/client.js'

```
function createRestifyClient(opts, type) {
    var client = restify.createClient({
        agent: opts.agent,
        ca: opts.ca,
        ciphers: opts.ciphers,
        connectTimeout: opts.connectTimeout,
        headers: opts.headers,
        log: opts.log,
        pooling: opts.pooling,
        rejectUnauthorized: opts.rejectUnauthorized,
        retry: opts.retry,
        type: type,
        url: opts.url,
        socketPath: opts.socketPath,
        version: '~1.0'
    });

    return (client);
}
```

* no userinput it seems, maybe not vulnerable, but nevertheless ask!

- express-admin
* MySQL, MariaDB, SQLite, PostgreSQL admin for NodeJS
* connect-multiparty(1.1.0)<-qs(~0.6.5)
* qs used in connect-multiparty: https://github.com/andrewrk/connect-multiparty/blob/0ba6b09ec0f426e945d3313b6868c3d6ed33da11/index.js
* used here: https://github.com/simov/express-admin/blob/bfbd336c30a1f4fa9f3773e7e1dd8e73d6d58444/app.js

- imager
* A node module to resize, crop and upload images (with different variants and presets) to Rackspace cloudfiles and Amazon S3.
* pkgcloud(0.9.6)<-qs(0.6.x)

    * used here: https://github.com/imagerjs/imager/blob/3220c3ea75099e67150b72ffa32cf8ea5ebfe062/lib/imager.js
    * for rackspace
    * client creation is more or less fixed
- spotify-web-api-node
    * restler(~3.2.0)->qs(0.6.6)
    * used here in the httpmanager: https://github.com/thelinmichael/spotify-web-api-node/blob/af58920b1722423eafa05a3e3efaa1ebc1c4bf61/src/http-manager.js
    * httpmanager: https://github.com/thelinmichael/spotify-web-api-node/blob/0af9409f0460f507e70b7b5fa9ebf45549e89428/src/spotify-web-api.js
    * maybe possible to make attacks:
        - enter trackid or songid

- wordpress-rest-api
    * superagent(^0.18.0)->qs(0.6.6)
    * used to make get and post: https://github.com/kadamwhite/wordpress-rest-api/blob/145b0be6235460eb9e9f3c9ab6f0030316bc48dd/lib/shared/wp-request.js
    * from the testcases its possible to add
        - username and password: querystring!
- stardog
    * restler(~3.2.0)->qs(0.6.6)
    * used in: https://github.com/clarkparsia/stardog.js/blob/4f53f7604e3c0e1d8ffec1babb431b21c4b78a0f/js/stardog.js
    * access to the Stardog DBMS
    * potential vulnerabilties in http requests
- restling
    * restler(^3.2.2)->qs(0.6.6)
    * essentially a promise-based restler
    * same functionaliity as restler, it uses restler for HTTP calls

- clever
    * quest(~0.2.4)<-qs(~0.5.1)
    * qs is used for proessing http requests: https://github.com/Clever/quest/blob/c3fe916b0b3a8b2a8e46af789f2d043699f60477/lib/quest.coffee
    * quest is used in two locations:

  - https://github.com/Clever/clever-js/blob/686d5fcf116d90aa378771f13fb8adc70f61d9bb/lib/querystream.coffee
    only initialised var, but not used
  - https://github.com/Clever/clever-js/blob/master/lib/clever.coffee
    mongoose-like query API for an RESTful HTTP API
* potential vulnerability

- penguin
    * body-parser(~1.5.2)->qs(0.6.6)
    * body-parser is used:
      https://github.com/etabits/node-penguin/blob/c6c79bfdee11f056dc6b05f31ea5f8fb5e606727/src/index.coffee
    * code snippet:
      bodyParser = require('body-parser').urlencoded({ extended: false })
    * vulnerable!
- webdriver-http-sync
    * request-sync (0.0.5)<-qs(0.6.6)
    * qs is used in index.js including several methods such as parse and unescape
    * request-sync is used:
      https://github.com/groupon-testium/webdriver-http-sync/blob/cf660312e67876d0ad73716d51e762372b11a683/src/request.coffee
    * this is a webdriver developed by groupon, perhaps for testing purposes, so its not vital
- cabbie:
    * similar as previous: https://github.com/ForbesLindesay/cabbie
- voicetext
    * superagent(^0.18.1)->qs(0.6.6)
    * used in the code: https://github.com/pchw/node-voicetext/blob/02ddae1503fe2ee71bc878537ff82f8b0fa9e741/lib/voicetext.js
    * REST API
    * vulnerability possible
- node-tvdb:
    * superagent(^0.18.2)->qs(0.6.6)
    * used in the sendRequest function: https://github.com/edwellbrook/node-tvdb/blob/252820c809ec189d04e9f54b9d25d1147913784c/index.js
- hoist-js
    * superagent(^0.18.0)->qs(0.6.6)

    * used in the code: https://github.com/hoist/hoist-js/blob/999a2d3cde5e3e6cd70b4d8cf640c1623a002c2b/src/hoist.js
    * maybe a potential
- sentry-node:
    * quest(~0.2.4)<-qs(~0.5.1)
    * https://github.com/Clever/sentry-node/blob/332923d3107d8afc655cefb504a57b1037c042ba/lib/sentry.coffee
    *project id can be used to invoke the qs vulnerability
- shipshape:
    * restler(3.2.2)<-qs(0.6.6)
    * possibility of DOS attack
    * https://github.com/shipshape-io/shipshape/blob/ae3477a34cb863dd919930856a4ad9406fb3ca92/shipshape.js
    * user input taken

qs, marked, validator, send, connect, syntax-error(don't do, only potential), st

==
Advisories - Security vulnerabilties - st
==

* st module: <0.2.5

Versions prior to 0.2.5 did not properly prevent folder traversal. Literal dots in a path were resolved out, but url encoded dots were not. Thus, a request like /%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd would leak sensitive data from the server.

As of version 0.2.5, any '/../' in the request path, urlencoded or not, will be replaced with '/'. If your application depends on url traversal, then you are encouraged to please refactor so that you do not depend on having .. in url paths, as this tends to expose data that you may be surprised to be exposing.

* This is a critical, as this affects the core functionality
* Investigate usage of this module

Intresting modules using validator (indirect):
- bfydir:
    * discussed on gh: https://github.com/guybrush/bfydir/pull/1
    * for local development
    * st not used in the code, just declered in the dependency list

    * replaced the dependencies with other, not updated on npm
    * could make a request

```
==============================================
Advisories - Security vulnerabilties - send
==============================================
```

* send module: < 0.8.4

Send is Connect's static() extracted for generalized use, a streaming static file server supporting partial responses (Ranges), conditional-GET negotiation, high test coverage, and granular events which may be leveraged to take appropriate actions in your application or framework.

The pipe method is used to pipe the response into the Node.js HTTP response object, typically send(req, path, options).pipe(res).

```
var http = require('http');
var send = require('send');

var app = http.createServer(function(req, res){
  send(req, req.url).pipe(res);
}).listen(3000);
```

- Vulnerable code is in the pipe prototype:

```
/**
  * Pipe to `res.
  *
  * @param {Stream} res
  * @return {Stream} res
  * @api public
  */

 SendStream.prototype.pipe = function(res) {
***
     // malicious path
-    if (path.substr(0, root.length) !== root) {
+    if (path === root || path.substr(0, root.length + 1) !== root + sep) {
       debug('malicious path "%s"', path)
       return this.error(403)
     }
***
 }
```

- test code for this:

```
 it('should not allow root transversal', function(done){
+       var app = http.createServer(function(req, res){
+         send(req, req.url, {root: __dirname + '/fixtures/name.d'})
+         .pipe(res);
+       });
+
+       request(app)
+       .get('/../name.dir/name.txt')
+       .expect(403, done)
+     })
it('should not allow root transversal', function(done){
+       var app = http.createServer(function(req, res){
+         send(req, req.url, {root: __dirname + '/fixtures/name.d'})
+         .pipe(res);
+       });
+
+       request(app)
+       .get('/../name.dir/name.txt')
+       .expect(403, done)
+     })
```

Intresting modules using send (indirect):

- gitbook:
    * send used as root: https://github.com/GitbookIO/gitbook/blob/725510028fc557ae7c1079ed4196cbc4b09218a3/bin/server.js
    * not a serious issue: https://github.com/GitbookIO/gitbook/issues/438
- phpjs:
    * send used as root: https://github.com/kvz/phpjs/blob/98d264640c6a1586a1235d0f1da044c4c28a3752/test/browser/server.js
    * its in a test file and used for test purpose not an serious issue
    * author should put it in the devDependencies section
- shiny-server
    * send used in the directory-router: https://github.com/rstudio/shiny-server/blob/b83dc42867d79d34b934c640705a3e1849c1fc42/lib/router/directory-router.js

    * send used as root

```
*    send(req, suffix)
    .root(self.$root)
    .on('error', onError)
    .on('stream', onStream)
    .pipe(res);
```

  * comments suggest some risks with running as root:
  'Running as root unnecessarily is a security risk! You could be running more securely as non-root.
  * discussions are related to running as root as a problem
  * they could simply update send dependency

- hotplate
    * dependency not used at all, just declered
- xpush
    * used in channel-server:
https://github.com/xpush/node-xpush/blob/309b4ea9b4135a0991d7725524006fe2c2c7160f/lib/server/channel-server.js
    * used as root: send(req, req.params.channel+'/'+req.params.filename, {root: httpRoot})
    * no pull request or discussion
    * active development

Intresting modules using send (cascading):

- roots
    * charge<-pathologist-middleware<-send
    * send only declered in dep and code, but not used
- percolator
    * detour<-send
    * detour not used, just declered, needs clean up
    *
- pesapaljs
    * function used in pesapaljs, not vulnerable
    * send not used in phpjs, only in test

==
Advisories - Security vulnerabilties - marked
===

Content injection attack
* vulnerable <= 0.3.0
* injection possible in to locations
 - gfm codeblocks
 - javascript url's
* regardless of setting vulnerable, also its the core function that is vulnerable, if there is a case of usage, its a problem
* Note: the context of how its used plays a vital role!

===
indirect vulnerabilties:

- cleaver:
  * marked is used here: https://github.com/jdan/cleaver/blob/a8bf0eb6017b3614694ecf8453da8cd9768c5fda/lib/index.js
  * it used to render the slide
  * cleaver is uses markdown for slideshows
  * it is not inheritely vulnerable, if a hacker manages to store content from the document folder, it will be problematic
  * no pull request or issues
- styledocco:
  * marked is used: https://github.com/jacobrask/styledocco/blob/9f12e71a02c5193ed0427aecba7929b230cf0975/styledocco.js
  * the security vulnerability is contextual, same as the previous one
  * no pull request or issues
- handlebar-helpers:
  * marked is used here: https://github.com/assemble/handlebars-helpers/blob/84dcfdd2b42c54ed71ded34b68941466286ead2b/lib/utils/markdown.js
  * this package is used as utility library(general purpose), therefore this should be fixed!
  * discussion in fixing this is already in talks: https://github.com/assemble/handlebars-helpers/pull/125
  * problem is breaking changes
- repload

* marked is used here: https://github.com/thlorenz/replpad/blob/b12448c3a34f0f8fc2cd84c6dc4a7d9a0dc87c1d/lib/dox/pack/open-readme.js
* similar to previous ones
* no dicussion on issues or PR

- glog
  * git push blog server
  * marked is used: https://github.com/substack/glog/blob/b1fb4fd7569498bbfca6d8cff137d748fb6b6a70/index.js
  * unauthorized access to the server, a hacker could potentially publish vulnerable .md files
  * no pr or issue
- marked-toc:
  * marked used: https://github.com/jonschlinkert/marked-toc/blob/8dd8f3bd8006fc3ce29db13a50bd1eab949afb64/index.js
  * the lexer function is used
  * generatres table of content, not so interesting
- gifsockets-server:
  * not used but declered

===

cascading vulnerabilties:

- harp
  * Harp Web Server.
  * harp and terroform same author
  * terraform<-marked
  * marked used in template: https://github.com/sintaxi/terraform/blob/417374b8544b25bb78ff4a0e3bdbdd917c985a3e/lib/template/processors/md.js
  * files are passed in to the function
  * Terraform is the pre-processor engine for the Harp web server. Terraform does not write or serve files. It processes and provides a layout/partial paradigm.
  * provide a path to a public directory
  * used in 3 locations in harp:
    - https://github.com/sintaxi/harp/blob/2f6ef078acc99298c76014282e1121be43891fca/lib/index.js
    - https://github.com/sintaxi/harp/blob/7a09952fb461b7518845062911333f794d70b26/lib/helpers.js
    - https://github.com/sintaxi/harp/blob/06789b862c224a9474de9161955ab0ae56d6ac2e/lib/middleware.js

* not a high risk, dynamic inhjection a difficulty, it uses local files, access to the local files makes it a threat
* https://github.com/sintaxi/harp/issues/352 discussed, but not updated yet
* break things: https://github.com/sintaxi/terraform/pull/65

- yogi
    * dev tool!
    * yuidocjs <- marked
    * action needed: update version!
    * marked used in: https://github.com/yui/yuidoc/blob/843a7d85ef627be9810e6d5941b735c684da74ea/lib/builder.js
    * not a big security problem: https://github.com/yui/yuidoc/issues/254
    * used in yogi, not sure what its used for
    * no PR or Pull Request
- sear
    * brucedown(0.1.1)<-marked
    * action needed: update!
    * brucedown uses marked as a core function, so vulnerable!
    * brucedown used here: https://github.com/Everyplay/sear/blob/a5400daabfbdf0e2a87bf7561aca3d7c0d908d17/lib/htmlutils.js
    * used for converting md to html
    * not sure about the project
    * but seems that its vulnerable
- plupload
    * just declered not used, clean up

===================================

* connect module: <=2.8.0

from test cases:

```
+var connect = require('../');
+
+var app = connect();
+
+app.use(connect.bodyParser());
+app.use(connect.methodOverride());                              <--- use this makes it possible
+
+app.use(function(req, res){
```

```
+  res.end(req.method);
+});
```

==

indirect vulnerabilties:

- fekit
    * connect used: https://github.com/rinh/fekit/blob/0c57cc3f97bf9bfd5e0a9482671793da97bfbbea/src/commands/server.coffee
    * but not methodoverride, so not fully vulnerable
    * no discussion

- socketstream
    * connect used: https://github.com/socketstream/socketstream/blob/58898c5c8879d8fde25798256258bc1587d91a19/lib/session/index.js
    * but not methodoverride
    * no security discussion, version is updated
- everyauth
    * 3000 stars
    * node.js auth package (password, facebook, & more) for Connect and Express apps
    * connect used: https://github.com/bnoguchi/everyauth/blob/59a81236a47f8b6c2511523cb3f608028479a456/index.js
    * no core function used
  * https://github.com/bnoguchi/everyauth/issues/463
  * discussion mentioned and reported, however not solved, the thread halted and closed

- batman.js
    * connect not used
- calipso
    * 1555 stars
    * Calipso is a simple NodeJS content management system based on Express, Connect & Mongoose.
    * https://github.com/cliftonc/calipso/issues/259
    * there are security discussions related to certain functionality
    * but not with NSP

http://calip.so
    * connect used at multiple places
    * mostly utilty features used
- connect-auth
    * must be used with connect, the reason for its apparence on the dep list

* https://github.com/ciaranj/connect-auth/issues/133
* someone reported but no action or reply

- matador
    * an MVC framework for Node
    * connect used: https://github.com/Medium/matador/blob/b5f963c322388b184f9cf5f0f31e949470b8f613/src/matador.js
    * but not methodOverride
    * no security discussion whatsoever
- hem:
    * https://github.com/spine/hem/blob/06fb42d0e2bdabc377c9661e58e9169620395caa/src/hem.coffee
    * no methodoverride
    * Hem is a project for compiling CommonJS modules when building JavaScript web applications
    * dev tool
- rainjs
    * RAIN is a highly scalable frontend platform that orchestrates distributed rendering of loosely coupled components.
    * https://github.com/rainjs/rainjs/blob/1540603e9fad8494fdd36dccbdf7cb602a5f4270/lib/server.js
    * no methodoverride
- amorphic
    * declered but not used in the code

====================================

* Validator Module: <1.1.0 & <2.0.0

* Latest updates: XSS filter removed

Remove the XSS filter: https://github.com/chriso/validator.js/commit/2d5d6999541add350fb396ef02dc42ca3215049e
https://github.com/chriso/validator.js/issues/181

The xss() function was originally a port of the XSS filter from CodeIgniter. I added it to the library because there wasn't an alternative at the time. Unfortunately I don't have the time or expertise to maintain the XSS filter or keep merging upstream changes.

If you need one for your app, I suggest looking at Caja sanitisation engine maintained by Google. (https://code.google.com/p/google-caja/source/browse/trunk/src/com/google/caja/plugin/html-sanitizer.js)

* XSS Filter bypasses on both versions, code functions
    - var str = sanitize(large_input_str).xss();
    - xss() //Remove common XSS attack vectors from user-supplied HTML
    - xss(true) //Remove common XSS attack vectors from images

Intresting modules using validator (indirect):
- pump.io:
    * xss used: https://github.com/e14n/pump.io/blob/1029ac71b94dec2b51f6b2aa461b5b2a514e585a/lib/scrubber.js
    * no pull request or issue
- frontail:
    * log stream
    * xss used: https://github.com/mthenw/frontail/blob/6223033bb0d6bd038ced3ba4ee9f53dc63ddafe2/index.js
    * no pull request or issue
    * not actively maintained
- nodeportal:
    * xss used: https://github.com/saggiyogesh/nodeportal/blob/c0b96f1dad788acd790e222f4091bf22314786e9/plugins/threadComments/ThreadCommentsController.js
    * xss used: https://github.com/saggiyogesh/nodeportal/blob/c0b96f1dad788acd790e222f4091bf22314786e9/lib/FormBuilder/index.js
    * no pull request or issue
    * recent project

Intresting modules using validator (cascading):
- feedparser
    * resantize(latest version and same author)->validator
    * code in feedparser:

```
        resanitize.stripHtml(meta.title);
    resanitize.stripHtml(meta.description);
```

* code in resanitize:

```
    /**
        * Dumbly strip angle brackets
        */
```

```
function stripHtml (str) {
    return str.replace(/<.*?>/g, ");
    }
module.exports.stripHtml = stripHtml;
```

* validator code not used, not security vulnerability
* the core function uses the validator function
* https://github.com/danmactough/node-feedparser/pull/134 -> removed now

- couchtato and datagen
    * github updated but not npm
    * bagofcli->validator
    * code in couchtato:

```
 function exec() {
    var actions = {
    commands: {
        config: { action: _config },
        iterate: { action: _iterate }
    }
};
cli.command(__dirname, actions);
}
```

    * code in bagofcli:
        no use of xss function
    * no vulnerability
    * the mainatiner should only update on npm