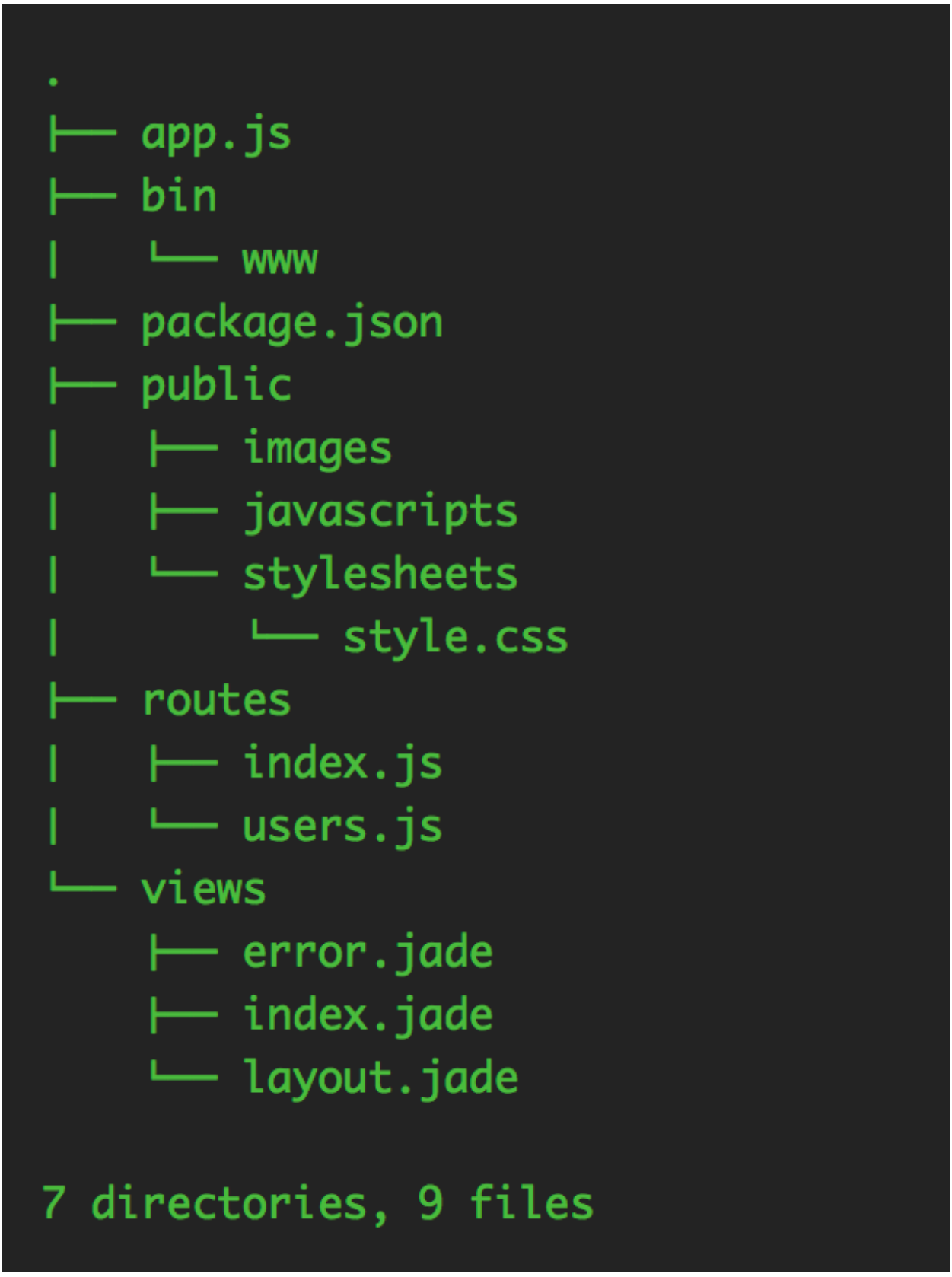
Express黑盒安全测试

Express是一个基于Node.js平台，快速、开放、极简的Web开发框架。

Express应用生成器

通过应用生成器可以快速创建一个express应用的骨架，其目录结构如下：



其中app.js是应用程序入口，运行node ./bin/www即可创建以app.js为入口的Web应用。routes目录下是按照应用业务功能组织的文件结构，在app.js中以如下方式引入：

```
var index = require('./routes/index');
var users = require('./routes/users');
```

与URI关联起来：

```
app.use('/', index);
app.use('/users', users);
```

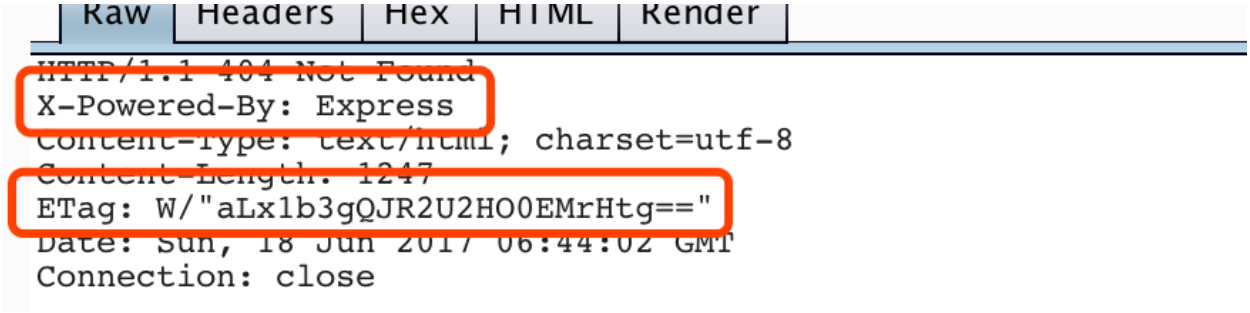
public目录下是应用的静态文件，在app.js中使用express提供的静态文件托管方式进行托管：

```
app.use(express.static(path.join(__dirname, 'public')));
```

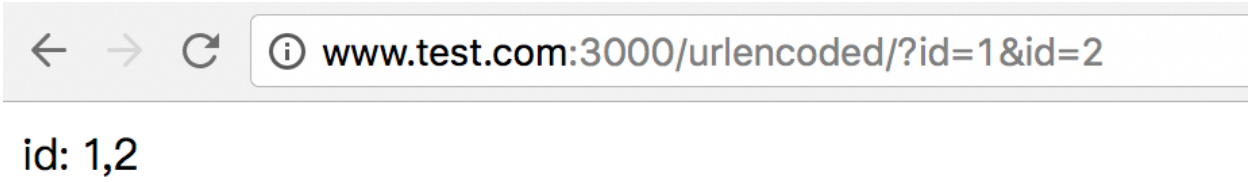
views目录下是jade模版文件。

Express应用识别

响应头

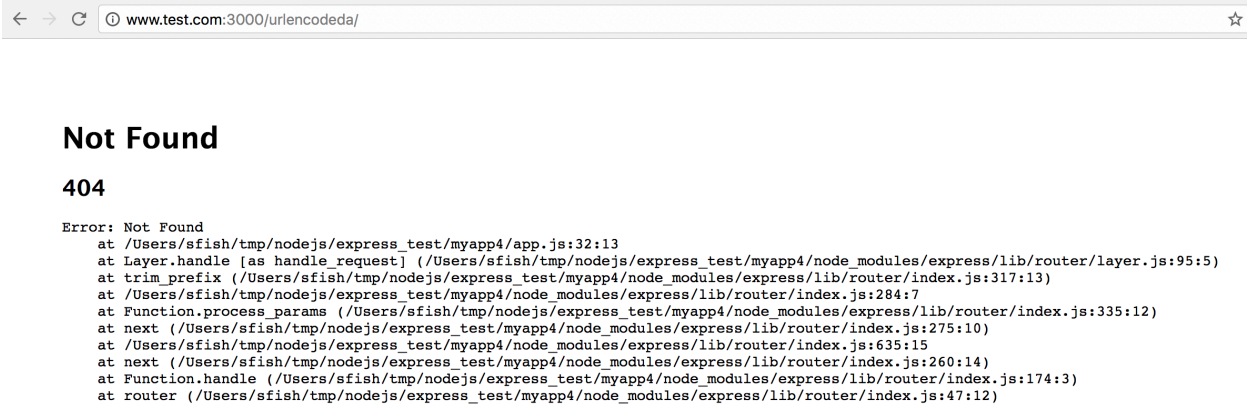


参数解析特征



（这是Node.js的参数解析特征）

报错信息



Jade模版特征

```
<!DOCTYPE html><html><head><meta charset="utf-8"><title>movie list</title><link href="/libs/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet"><script
src="/libs/jquery/dist/jquery.min.js"></script><script src="/libs/bootstrap/dist/js/bootstrap.min.js"></script></head><body><div class="container"><div
class="row"><div class="page-header"><h1>movie list</h1><small>重度科幻迷</small></div></div></div><div class="container"><div class="row"><table class="table
table-hover table-bordered"><thead><tr><th>电影名字</th><th>导演</th><th>国家</th><th>上映年份</th><th>录入时间</th><th>查看</th><th>更新</th><th>删除</th></tr></thead>
<tbody><tr class="item-id-59494b5ee119107cb48a0481"><td>name</td><td>doctor</td><td>country</td><td>1973</td><td>06/21/2017</td><td><a target="_blank"
href="../movie/59494b5ee119107cb48a0481">查看</a></td><td><a target="_blank" href="../admin/update/59494b5ee119107cb48a0481">修改</a></td><td><button type="button"
data-id="59494b5ee119107cb48a0481" class="btn btn-danger del">删除</button></td></tr><tr class="item-id-594952ee5618e7834d274ef0"><td>test2</td><td>test2</td>
<td>test2</td><td>1111</td><td>06/21/2017</td><td><a target="_blank" href="../movie/594952ee5618e7834d274ef0">查看</a></td><td><a target="_blank"
href="../admin/update/594952ee5618e7834d274ef0">修改</a></td><td><button type="button" data-id="594952ee5618e7834d274ef0" class="btn btn-danger del">删除</button>
</td></tr></tbody></table></div></div><h1>movie list</h1></body></html>
```

使用jade模版时输出到前端的代码默认不换行。

Express历史安全问题

根路径披露漏洞

影响版本：

4.x < 4.11.1

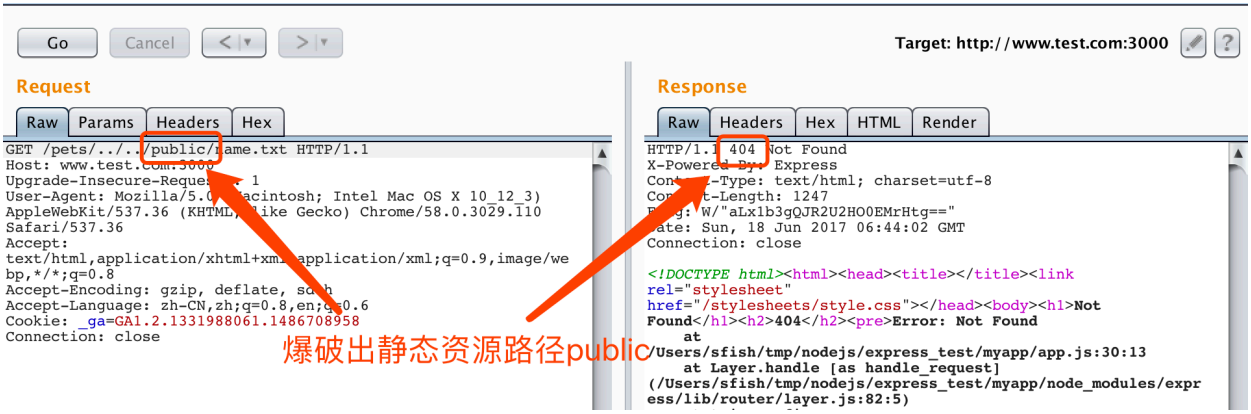
3.x < 3.19.1

漏洞详情：

受该漏洞影响的Express版本中serve-static组件所依赖的send库为0.11.1以下版本，以0.11.0版本为例，关键代码如下：

```
417     if (root !== null) {
418       // join / normalize from optional root dir
419       path = normalize(join(root, path))
420       root = normalize(root + sep)
421
422       // malicious path
423       if ((path + sep).substr(0, root.length) !== root) {
424         debug('malicious path "%s"', path)
425         return this.error(403)
426       }
427
428       // explode path parts
429       parts = path.substr(root.length).split(sep)
430     } else {
```

其中root为服务端配置的静态资源路径，path为客户端请求的文件路径，当path的前root长度字节与root不相等时返回403。这样的判断存在信息泄露风险，攻击者可利用该判断爆破出服务器上静态资源的物理路径，如下所示：



开放重定向漏洞（CVE-2015-1164）

影响版本：

4.x < 4.10.7

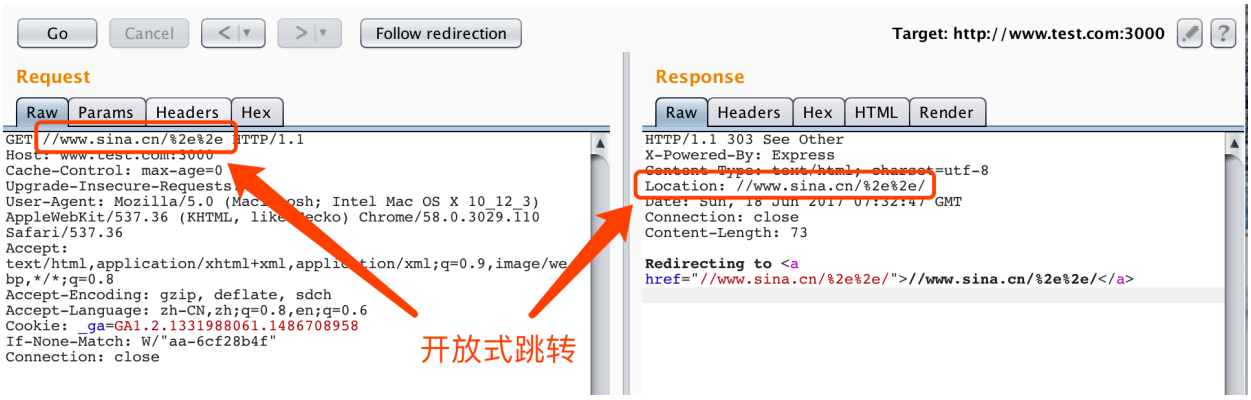
3.x < 3.19.0

漏洞详情：

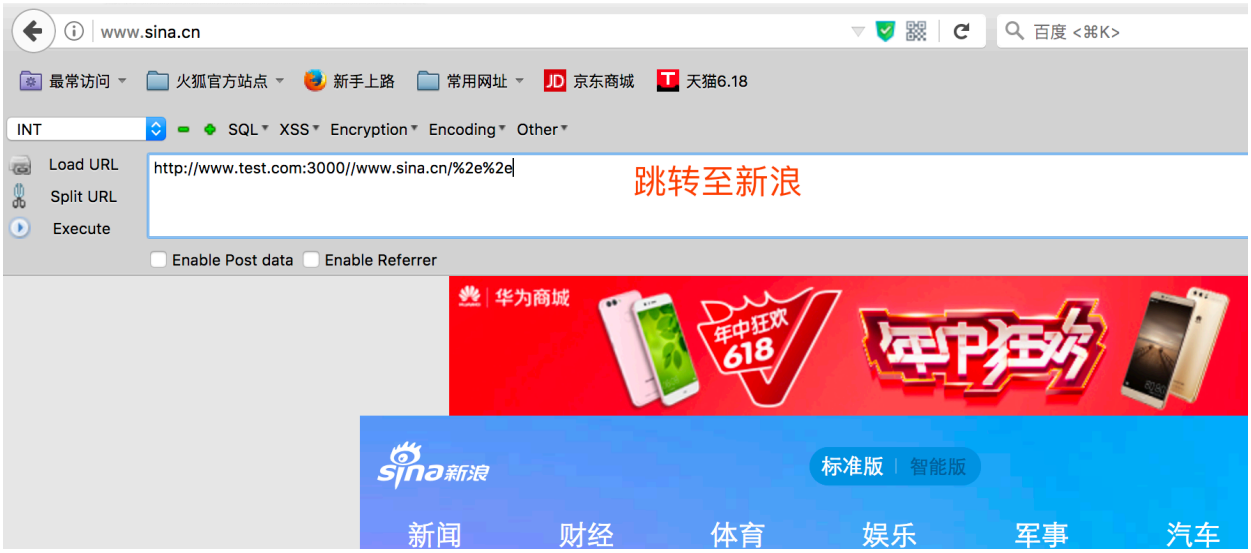
受该漏洞影响的Express版本所依赖的serve-static库为1.7.2以下版本，以1.7.1版本为例，关键代码如下：

```
75      if (redirect) {
76        // redirect relative to originalUrl
77        stream.on('directory', function redirect() {
78          if (hasTrailingSlash) {
79            return next()
80          }
81
82          originalUrl.pathname += '/'
83
84          var target = url.format(originalUrl)
85
86          res.statusCode = 303
87          res.setHeader('Content-Type', 'text/html; charset=utf-8')
88          res.setHeader('Location', target)
89          res.end('Redirecting to <a href="' + escapeHtml(target) + '">' + escapeHtml(target) + '</a>\n')
90        })
91      } else {
```

攻击者请求//domain/%2e%2e可形成开放式跳转，如图：



浏览器效果：



目录遍历漏洞（CVE-2014-6394）

影响版本：

4.x < 4.8.8

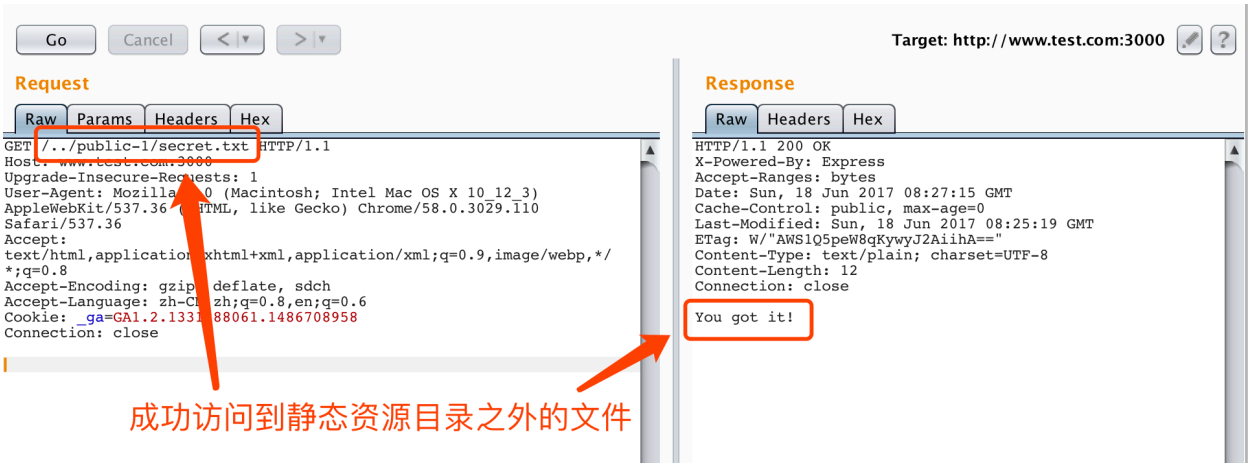
3.x < 3.16.10

漏洞详情：

受该漏洞影响的Express版本中serve-static组件所依赖的send库为0.8.5以下版本，以0.8.3版本为例，关键代码如下：

```
412     if (root !== null) {
413       // join / normalize from optional root dir
414       path = normalize(join(root, path))
415       root = normalize(root)
416
417       // malicious path
418       if (path.substr(0, root.length) !== root) {
419         debug('malicious path "%s"', path)
420         return this.error(403)
421       }
422
423       // explode path parts
424       parts = path.substr(root.length + 1).split(sep)
425     } else {
```

经典的“/”问题！root是静态资源路径，末尾没有这个“/”，假如服务端定义的静态资源路径为“/Users/sfish/tmp/nodejs/express_test/myapp3/public”，用户可以访问到“/Users/sfish/tmp/nodejs/express_test/myapp3/public-1/secret.txt”文件，如图：



常见Web漏洞探究

MongoDB注入

MongoDB经常会作为Node.js应用的数据存储，针对MongoDB的注入在使用Express框架时依然可能存在，使用常见的第三方模块Mongoose进行测试，发现传入的$ne被当作查询符号解析，如图：

使用第三方orm框架绑定参数的特性进行数据库查询可以有效避免SQL注入问题，常用的orm框架如第三方模块node-orm2。

XSS

若目标应用使用了jade模版引擎，则会自动对引号、左右尖括号等字符进行HTML实体编码，可避免部分跨站问题，如图：

CSRF

Express应用常使用第三方模块csurf防止CSRF攻击，该模块在表单中添加_csrf字段（该字段名可修改），在表单提交时与Cookie中的token进行比对来防止CSRF。

文件上传

面对Express应用通过文件上传漏洞获取shell比较困难，需要满足下列前提条件：

1. 服务端使用了supervisor等实现了代码“热部署”
2. 文件上传路径和文件名可控

当满足以上两个条件时，可以尝试通过文件上传覆盖目标应用的某个路由解析文件，将其替换成Webshell（这同时需要知道目标文件名，利用难度较大）。