

## 1. 进入正题

相信各位的学习、生活中都遇到过这样的页面



此处我以某厂商的云服务购买为例，由图可知，需要我们输入姓名、身份证、联系电话等。如果按照我们普通的挖掘思路，此处可能存在的漏洞是不是有 SQL、XSS、越权查看他人提交信息、CSRF 等等，其实此处可以利用一种新的思路，我称之为不受限制的资源调用。

## 2. 漏洞测试

此处我们先输入自己的真实姓名+身份证号，然后把身份证号的最后一位 7，改成 5，进行提交，此时可以发现，提示我们需要输入正确的身份证号码，同时 Burp 没有任何数据包请求，判断此处是前端做了校验，校验用户输入的身份证号是否能够与规则匹配。同时可在 JS 文件中找到相应规则，此处校验不通过会返回 false

阻止我们进行提交。

请输入正确的身份证号

姓名

身份证

2335

联系电话

19999999999



console 有如下结果：



所以此时，我们需要把身份证号改成一个正确的身份证号，把姓名也改成正确的姓名，同时进行提交，此时可见，在我们的 Burp 中出现了我们想要的数据包，包含了我们的姓名、身份证号、联系电话等等。此时我们再将数据包中的 6 改成 5，也就是把真实身份证号又改回去一个不存在的身份证号，然后抓取返回包，可见，此时后端又做了一个验证，告知我们：身份证验证错误。

```

19 Cookie: JSESSIONID=
20
21 {
  "cip": "",
  "x": "",
  "y": "",
  "pid": "",
  "cid": "",
  "ccid": "",
  "pname": "",
  "cname": "",
  "page": "",
  "logs_type": "唐旭",
  "logs_type2": "32336",
  "timestamp": "",
  "u_id": "",
  "sub_channel": "",
  "productName": "",
  "serial_number": ""
}

```





先知社区

错误图:

```

17 {
  "result": 1001,
  "msg": "\u8eab\u4efd\u8bc1\u683c\u5f0f\u4e0d\u6b63\u786e\u76f1",
  "data": [
  ],
  "currtime": 1636363000
}

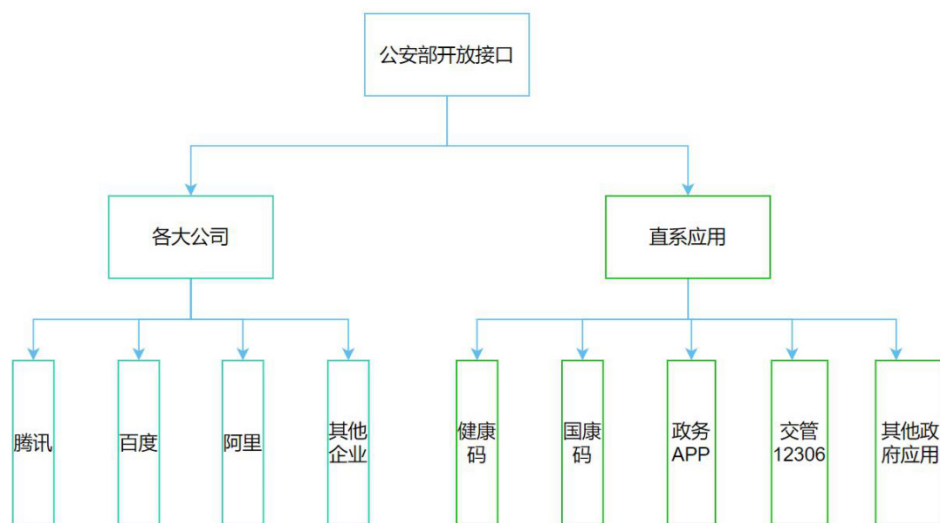
```



先知社区

### 3. 原理剖析

此时先不着急往下进行测试，我们先来了解一下身份证验证的原理：



先知社区

这里我做了一张流程图，假如我此时是一名开发者，我需要给我的 APP 加上实名验证功能，那么我可以直接去向最上层的那个机构申请接口吗？不能，因为我不是企业，而且我也不是属于它直系应用的开发者。我只能向他的下级，也就是腾讯、阿里、百度这样的企业去申请 API 接口，同时这些公司会把我们提交的数据，提交给最上层的那个机构，并且根据返回的数据，给我们返回的数据。也就是身份证号验证成功，或者二要素验证不一致。

我们再来说一下直系应用与企业的区别，直系应用去申请二要素验证，一般是不用花钱的。而我们作为个人开发者，或者企业，去调用那个接口，其实是要钱的。

我们在网上随便找一些关键字，可以看到，价格其实还是蛮高的。



APILink / API集市 / 身份证二要素验证【实名认证API】



44490 8933 243658750

### 身份证二要素验证【实名认证API】

身份核验

服务简介 输入姓名和身份证号，即可快速验证姓名与身份证号的真实性和一致性。可用于金融、保险、在线教育、电商、租赁、物流、旅游等需要实名认证场景。  
直连自公安部和银联中心接口，实时查询，权威可靠！【购买前可人工咨询并申请免费测试次数，18612273012（微信同号）】

更新时间 2020-06-10 20:03:57

金额 **¥180** (约0.180元/次 有效期1年)

用户评分: ★★★★★

总共交易量: 3065 笔

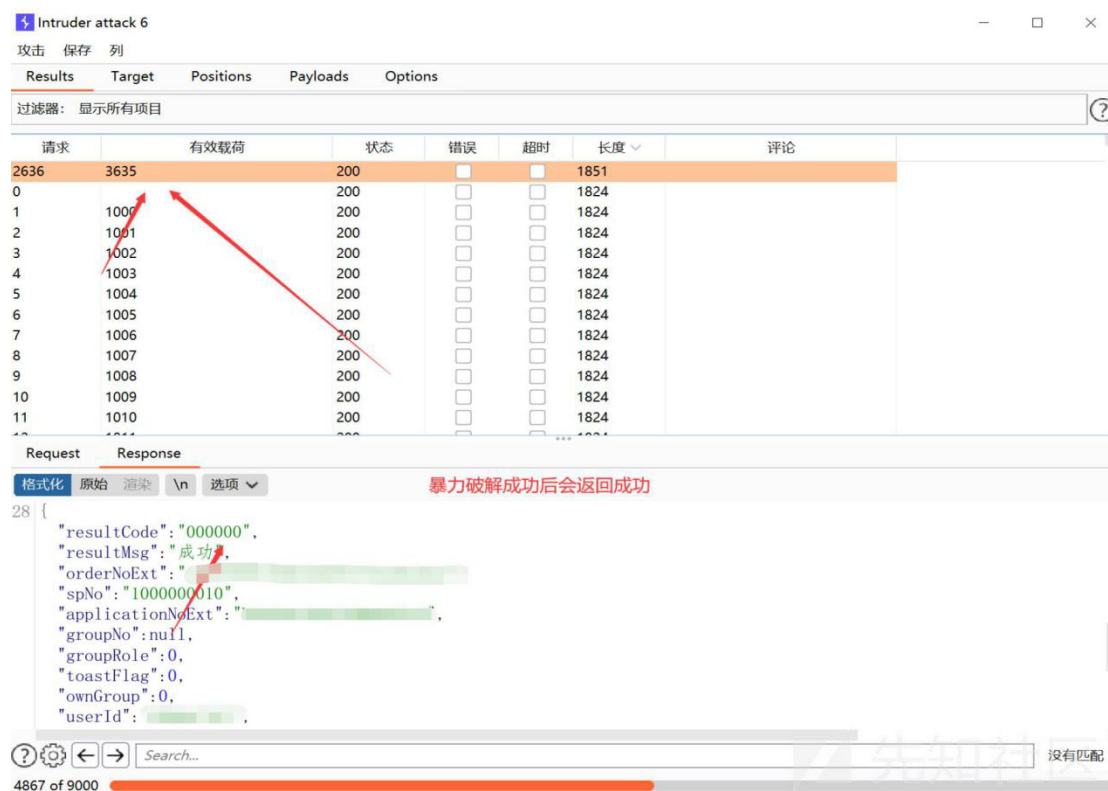
规格 10次 1000次 5000次 20000次 100000次 200000次

购买 收藏

先知社区

#### 4. 深入理解

那么此时是不是可以利用楼上所示的接口？去做一些事情呢，我这里假设要对别人进行社工，那么他的姓名是 **XXX**，身份证号的后四位或者后六位我不知道，就可以对他进行一个爆破。此时我们勾选上最后四位，然后把数值调整到 **0000-9999** 之间，此时根据返回包的长度大小、可判断身份证号码是否正确。



Intruder attack 6

攻击 保存 列

Results Target Positions Payloads Options

过滤器: 显示所有项目

请求	有效载荷	状态	错误	超时	长度	评论
2636	3635	200	<input type="checkbox"/>	<input type="checkbox"/>	1851	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	1824	
1	1000	200	<input type="checkbox"/>	<input type="checkbox"/>	1824	
2	1001	200	<input type="checkbox"/>	<input type="checkbox"/>	1824	
3	1002	200	<input type="checkbox"/>	<input type="checkbox"/>	1824	
4	1003	200	<input type="checkbox"/>	<input type="checkbox"/>	1824	
5	1004	200	<input type="checkbox"/>	<input type="checkbox"/>	1824	
6	1005	200	<input type="checkbox"/>	<input type="checkbox"/>	1824	
7	1006	200	<input type="checkbox"/>	<input type="checkbox"/>	1824	
8	1007	200	<input type="checkbox"/>	<input type="checkbox"/>	1824	
9	1008	200	<input type="checkbox"/>	<input type="checkbox"/>	1824	
10	1009	200	<input type="checkbox"/>	<input type="checkbox"/>	1824	
11	1010	200	<input type="checkbox"/>	<input type="checkbox"/>	1824	

Request Response

格式化 原始 渲染 \n 选项

暴力破解成功后会返回成功

```

28 {
  "resultCode": "000000",
  "resultMsg": "成功",
  "orderNoExt": "",
  "spNo": "1000000010",
  "applicationNoExt": "",
  "groupNo": null,
  "groupRole": 0,
  "toastFlag": 0,
  "ownGroup": 0,
  "userId": ""
}

```

4867 of 9000

此处可见，我们利用某平台开放的实名认证接口，可以完成我们自己想做的身份证二要素验证，同时由于厂商没有做限制，便可以无限消耗此厂商的资源，从而达到我们的目的。像本文中的二要素验证，以及短信验证，还有活人检测，其实都是基于 **Money** 的，在我们的 **SRC** 挖掘过程中，也可以去尝试一下这些点。