

逻辑漏洞实战

肖云老师



课程大纲

- 0、逻辑漏洞概要
- 1、身份验证漏洞
- 2、登录验证安全
- 3、登录前端验证漏洞
- 4、任意账号密码
- 5、权限类漏洞
- 6、其他漏洞
- 7、SCR中逻辑漏洞检查总结

0

逻辑漏洞概要

逻辑漏洞概要

逻辑漏洞就是基于开发人员设计程序的时候，逻辑不严密，导致攻击者可以修改、绕过或者中断整个程序，让程序按照开发人员的预料之外去执行。

比如某一网页的登录验证逻辑如下：

输入用户名验证--验证成功后输入密码--输入验证码--数据包前端传到后端处理--数据库匹配--匹配成功回包

由于整体网站可能采用前端验证，黑客可以直接篡改或者绕过某些流程，如下：

BP抓取用户名，密码、验证码特定格式--发送给后端匹配

由于网站采用前端验证，导致攻击者可以直接抓取数据包，从而绕过用户名验证的过程，直接爆破，简单来讲，只要你能修改、绕过、中断整个开发者运行软件的整体逻辑，这个便是逻辑漏洞，只是绝大部分逻辑漏洞的危害性并不高，比如：开发人员需要你先输入账号，在输入密码；但是你改变了这个逻辑，可以先输入密码，在输入账号，其实这个也是逻辑漏洞，这个漏洞没有任何危害；但是也有可能在某些特定情况下可以结合其他的漏洞可能产生新的风险

漏洞的根因：

比如SQL注入：

程序的开发者也没有想过你可以拼接他的语言产生他预料之外的危害，他不过是按照程序逻辑完成了整个查询的动作

那么为什么会产生逻辑漏洞就很容易理解了：研发只负责满足客户的需求，大部分的研发并不懂安全，所以并不会带着黑客的思维考虑这个软件的安全

程序的本质就是按照研发设计的逻辑运行，这个过程中出现的所有漏洞，皆可以为逻辑漏洞

逻辑漏洞的分类：

从漏洞的本质，我习惯将逻辑漏洞分为两类：

软件（系统）设计之初便存在的漏洞

使用者未能安全使用软件所产生的

1、第一点很好理解，比如永恒之蓝，sql注入漏洞、文件上传等等，均为设计的时候未能按照安全设计的方法进行，所产生的，攻击者只需要找到特定的点，执行特性的代码即可产生研发预料之外的现象

2、第二点的关键在于使用者，比如弱口令，比如匿名用户；程序开发者会设计很多便捷的功能，但是由于使用者使用不当，产生了新的问题

但是我们本节课的目的，为web逻辑漏洞，去除常见的漏洞大类，比如：sql注入、文件上传等等；我们也将讲讲部分不好分类，但是在业务测对用户有影响的漏洞，下文中的漏洞

1

身份验证漏洞

暴力破解漏洞-未限制爆破

未限制爆破，即对于用户登录的地方没有做什么限制爆破的策略

因此对于这个地方可以直接使用bp来抓包放进intruder模块爆破，参考dvwalow关卡或者pikachu《基于表单的暴力破解》，不做详述

Vulnerability: Brute Force

Login

Username:

Password:

暴力破解漏洞-限制IP爆破

某些具有安全意识的开发人员或运维人员，针对账户爆破问题就会使用限制攻击者IP的方式，当在短时间内有大量来自该IP的尝试登录现象时就会封锁该IP，导致该IP无法使用，针对这种情况，建议自己写脚本，调用git开源的代理API来爆破，当然也可以直接用别人的轮子

自建轮子的方法：首先需要新建一个代理池：现在好用的一般是收费版本，这里可以自行购买，我这里拿一个代理举例

然后使用python调用代理池进行爆破：

案例讲解：python脚本结合bp抓包

暴力破解漏洞-限制密码错误次数来爆破

有些网站管理员会限制某个账号的登录次数，如果超过限制次数，账号就会被锁定，除非管理员解锁或者设定一段时间过后自动解锁

由于他限制了一个账号，比如只允许输入5次，但是不限制你换个账号又可以输错5次，对于这种情况，通常可以采用弱密码反过来爆破账户的方式，即设置任意的不超过爆破次数的弱密码数量来反过来爆破用户名

如下图：通过12345678弱口令来爆破用户，可以比对长度后发现，test和admin账号存在，如果账号足够多，总会有弱口令的

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
12	test	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	6604	
1	admin	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	6593	
17	'or'='or'	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	6564	
295	WINDOWS_PASSTHRU	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	6551	
129	5rDvZ82Jj4Bm	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	6549	
56	Administrator	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	6548	
202	POWERCARTUSER	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	6548	
272	USER_TEMPLATE	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	6548	
70	CHEY_ARCHSVR	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	6547	
250	SYSTEST_CLUG	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	6547	
372	administrator	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	6547	
773	anne-cortine	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	6547	

Request

Response

Raw

Headers

Hex

HTML

Render

<div class = "errorfield">登录系统失败:LIN106

密码输入错误，现已连续输错1次，如果连续输错超过3次，您的帐号test将被自动锁定</div>

暴力破解漏洞-多字段爆破

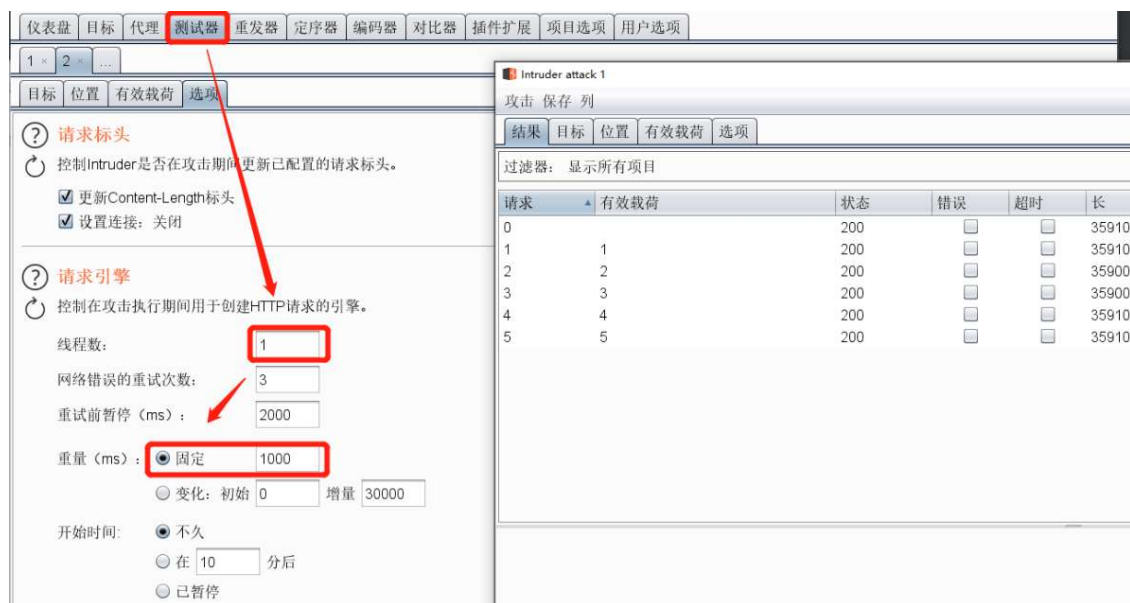
多字段爆破即需要爆破多个字段大于等于2，比如说：需要同时爆破：账号密码验证码，当我们爆破一个网站时返回信息是用户名或密码错误时，大多数时候仍然使用burpsuite的Intruder模块，只是与单个字段爆破选择的模式不同，但是当某几个字段相同的情况下，例如在不仅在post内容中确定还要在cookie或者session或者token中确定的时候，也可以自己写脚本解决问题，如下如为pikachutoken爆破的脚本

实验案列为：pikachu的token爆破

- 1、BP爆破
- 2、脚本爆破

暴力破解漏洞-限制登录频率爆破

限制登录频率爆破即限制在一定的时间内爆破的次数，比如10分钟内只允许爆破10次，对于这种方式可以采用延时爆破的方式，但是可能需要时间比较久，但总比手工爆破舒服，在bp上如下文设置，该时间以ms为单位（1000ms=1S）



暴力破解漏洞-Authorization爆破

Authorization爆破，狭义上单纯指basicbase64爆破，比如tomcat的密码在传输的时候，是采用base64编码的，而广义上可以泛指经过编码过后的用户名与密码下面我们以tomcat的basic爆破举例：
环境可以采用vulhub的tomcat8环境

cdvulhub-master/tomcat/tomcat8#路径

docker-composeup-d#开启环境

#后台路径如下：

<http://192.168.10.223:8080/host-manager/html>

实验演示：BP爆破

暴力破解漏洞-密文传输爆破

渗透测试中，有时会遇到密码从客户端到服务端中间通过前端js代码将密码加密后，在发送给服务器，所以这个时候，我们可以采用bp上自带的加密方法进行加密

常见的加密手法有MD5或者RSA，如果需要JS的复杂加密，也可以读懂JS的加密逻辑自定义Python进行爆破，或者使用python的pyexecjs包来帮助你执行JS文件复现加密方法



Session固定攻击 (未授权)

值 → URL
Session

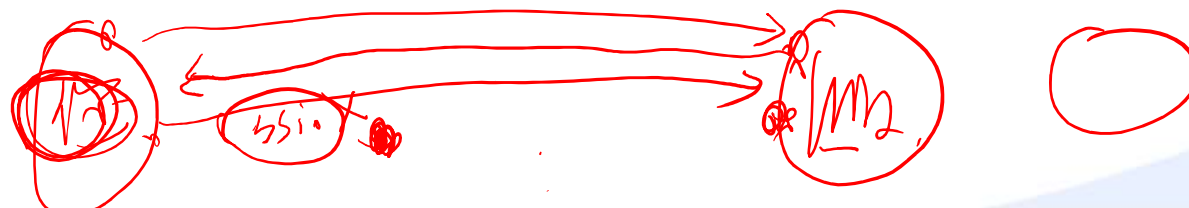
漏洞介绍：会话固定攻击是利用服务器的session不变机制，借他人之手获得认证和授权，然后冒充他人

漏洞原理：在请求登录过程时候，URL带有一个session，登录成功之后会将登录成功的信息绑定到这个session中，攻击者可以发送带有session的URL给相关工作人员诱导其登录，相当于获取了其身份信息

SSes's GET →

漏洞点：在GET方法请求登录时候带有session值

修复思路：只要避免在URL中带入session信息即可比较有效的防御另外也要注意POST请求中带有sessionid进行session固定攻击，虽然可利用性比较低，但是建议修复如下，测试方法可以直接赋值session然后更换浏览器打开或者使用无痕模式打开即可，如果直接访问，不需要登录，说明存在该漏洞



login?us=B4194E5E7C64A7AD5D9396F80170F3E8&ps1=93D6558B01182C26DF0F8D03C033253F&usertype=3|

POST



Firefox

无/良

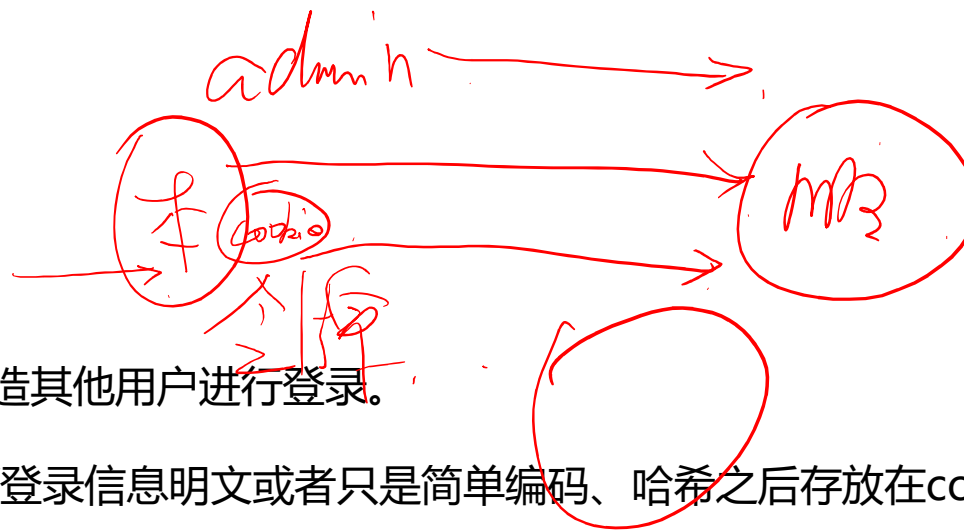


使用百度搜索，或者输入网址

未招

务平台

Cookie欺骗漏洞



漏洞介绍：通过伪造cookie信息能够伪造其他用户进行登录。

漏洞原理：开发者为了方便将身份信息/登录信息明文或者只是简单编码、哈希之后存放在cookies中，网站通过获取到的cookies进行授权或者身份验证

漏洞点：cookie中有明显或者只是简单编码、哈希的字段时候修改IsLogin值为1可以判定为用户已经登录

漏洞修复：Cookie不应该存储可理解的身份信息和登录信息按照规定，cookie对身份信息和登录信息的存储只能通过存储足够长度的随机字符串进行，避免篡改

如果某网站的cookie:asp163=admin，那我们只需要等待管理员登录成功后，网站存在登录信息后即可进行登录，构造报文进行登录，安装editthiscookie可以迅速看到cookie值如下

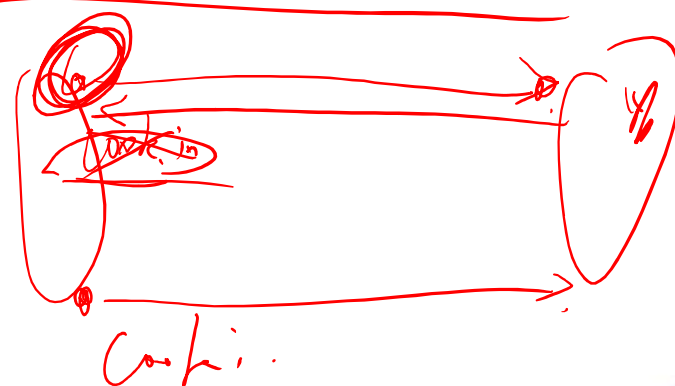
Cookie的效验值过于简单。有些web对于cookie的生成过于单一或者简单，导致黑客可以对cookie的效验值进行一个枚举，如下图所示

```
Cookie:
COOKIE_STORE_SYSTEMNO=7;
CartCookie=;
CustomerLogin=LastLoginTime=2016%2f8%2f11+0%3a53%3a08&ID=xxxx&SysNo=7140658519770066111&key=oSsWl%2b20
```

这家网站对于cookie的较验只单纯的采用了一组数字，并且数值为常量，不会改变，这样非常容易遭到黑客的枚举。

甚至有一些网站做的更简单，直接以用户名，邮箱号或者用户ID等来作为cookie的判断标准。

有一些厂商为了图方便，没有对用户的cookie做太多的加密工作，仅仅是单纯的做一个静态加密就完事了。



比如有个厂家web设置的cookie，解出来后是用户名。因此只要知道一个人的用户名就可以伪造对方的cookie，登陆他人账户

未进行登陆凭证验证

有些业务的接口，因为缺少了对用户的登陆凭证的校验或者是验证存在缺陷，导致黑客可以未经授权访问这些敏感信息甚至是越权操作。

案例一：后台页面访问

某电商后台主页面，直接在管理员web路径后面输入main.php之类的即可进入。

案例二：某航天公司订单ID枚举

2

登录验证安全

图形验证码-验证码可爆破

验证码可爆破，即验证码过于简单，例如验证码中字符数量过少，比如只有四位组成，且只包含0-9的数字还没有干扰点，亦或者验证码可以被猜测到（这种情况很少见）

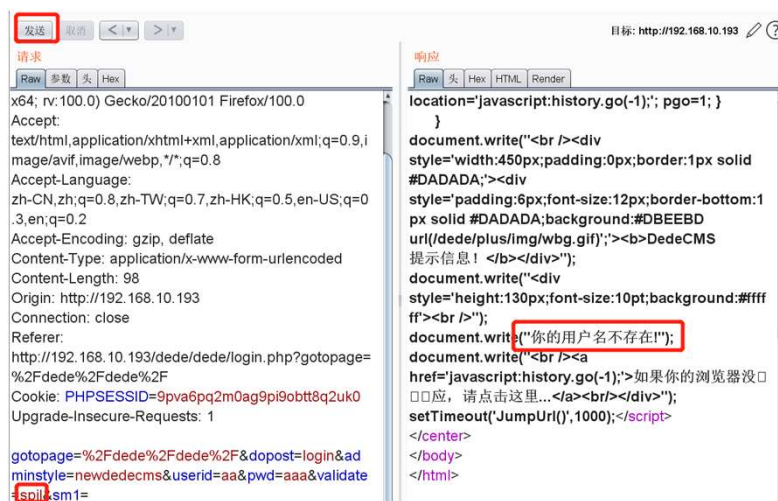
可以使用PKAV来进行带验证码的爆破，先用burpsuite来抓包，再将抓取的数据包放在pkav里，

实验操作：pkav攻击操作如下：PKavHTTPFuzzer爆破带验证码的网站

图形验证码-验证码复用

验证码复用，即登录失败后验证码不刷新，仍然可以使用上一次登录时的验证码且有效，存在爆破风险漏洞，如织梦后台的验证码，测试方法如下，抓包后输入对的验证码后，反复发送查看回包

可以看到，没有出现验证码错误，出现的是用户名不存在，那么这里还出现一个点，用户名爆破，只要输入账号错误和密码错误回显不一样，就可以算成用户名可爆破漏洞



图形验证码-验证码绕过

验证码绕过，即验证码可以通过逻辑漏洞被绕过，通常分为以下情况


- 案例一：验证码验证返回状态值


可BP修改状态值来绕过前端的验证，修改密码页面中存在验证码绕过漏洞：


- 案例二：点击获取验证码时，直接在返回包中返回验证码，通过抓包的来观察response包


图形验证码-客户端验证

通过查看源代码发现验证码是前端验证码，可以通过直接抓包的方式在bp里边爆破，参考pikachu-bf_client.php

 Please Enter Your Information







图形验证码-验证码前端验证漏洞

如下，某网站存在一个链接<http://1XX.XX.XX.XX:9080/setup>访问这个重置链接，点击修改管理员密码，可以重置密码，但是我们不知道旧密码是多少，只能输入新密码



The screenshot shows a web-based dialog box titled "修改系统配置管理员密码：" (Modify System Configuration Administrator Password:). Below the title, it states: "密码字母区分大小写，长度6-16位。限用字母、数字、特殊字符。" (Password letters are case-sensitive, length 6-16 characters. Limited to letters, numbers, and special characters.). There are three input fields: "*输入旧密码：" (Enter old password:), "*输入新密码：" (Enter new password:), and "*新密码确认：" (Confirm new password:). The first field contains six dots. A red rectangle highlights the first field and the instructions above it. A yellow tooltip message is displayed over the second field, stating: "您输入的密码错误，请重新输入。" (Your entered password is incorrect, please re-enter.). At the bottom of the dialog are two buttons: "修改" (Modify) and "取消" (Cancel).

短信验证码-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。

最好在做测试的时候先试一下，有很小的概率可以成功。

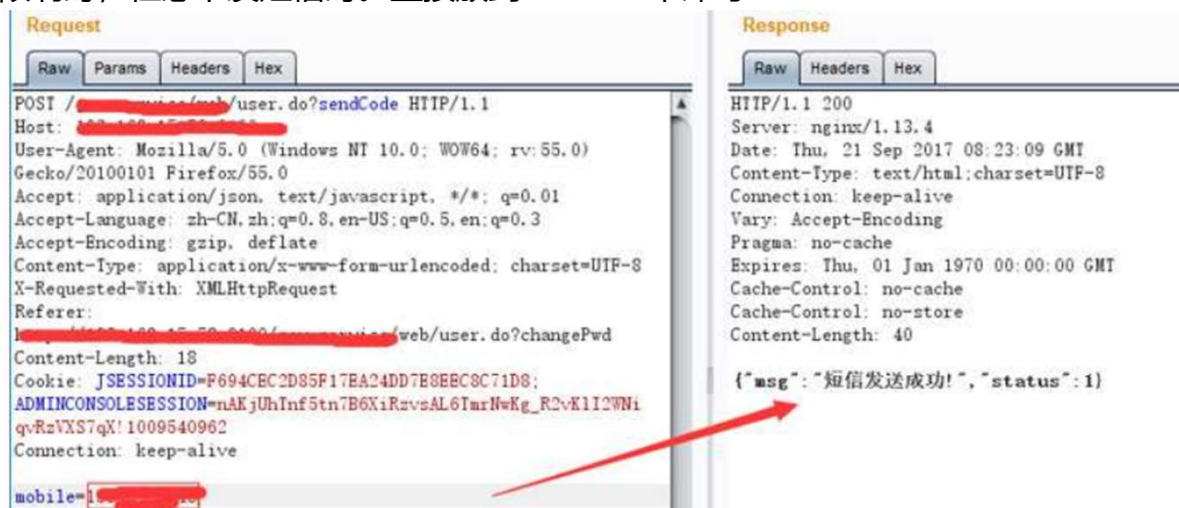
下面用一个案例进行举例说明:获取短信验证码后，随意输入验证码，直接输入两次密码，可成功更改用户密码，没有对短信验证码进行验证。



短信验证码-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有两种情况，一种是完全没限制发短信的频率，另外一种是为60秒发一条短信。

情况一：没有限制时，任意下发短信时。直接放到Intruder中即可



情况二：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。python脚本的编写可以参考上文

3

登录前端验证漏洞

忘记密码-给邮箱/手机发验证码

当在“忘记密码”页面，输入邮箱或者手机号后，网站会给手机或者邮箱发送验证码。用户输入验证码后，服务器会将其与正确的验证码进行对比，若相同则验证成功并进行下一步操作。网站还贴心的考虑到了：万一用户手抖输错怎么办？有两个选择：

第一个：客户重新填验证码。网站本来想着用户这次能输对了吧，没想到用户不停的手抖输错，短信不断的发到手机上（这是短信炸弹漏洞）。一条短信一毛钱啊，网站掏不起啊，那就试试方法2吧。

第二个：验证码再输一遍。在这种情况下，极易产生验证码爆破漏洞。

你可能会觉得输入一万条数据简直太漫长了，但是要知道我们发到网站服务器的都是数据包，验证码就在数据包中。我们只要不断的修改数据包中验证码的参数即可完整验证尝试

攻击方法和图片验证码相似，如果是4位，没有多余的防御措施，直接爆破即可，案列如下

忘记密码-前端验证绕过

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

1. 直接返回明文验证码输入手机号后，点击获取验证码。按F12调出开发者工具，然后在网络中，监听到两条json数据
2. 返回加密后的密码验证加密后，加密报文在应答数据包中，返回客户端，用户解密后即可获取验证码。

案列如下：

忘记密码-设置新密码时改他人密码

修改密码时可否修改参数（用户名/邮箱/手机）达到修改其他用户密码的目的。

首先在一个网站注册一个用户，然后修改这个用户的密码，通过抓取数据包中的参数判断哪里是校验用户，通过修改关键参数，达到任意修改其他用户密码的目的。

下面对一个网站进行模拟攻击，以便于在攻击中讲解能够更深刻的理解。

忘记密码-某网站密码找回功能

因为这种方式较为常见，因此再举一个例子，如下。

在用户更改密码和找回密码时，会发送一个验证码到手机。填写正确验证码后，后台会在返回的报文中通过状态码来进行正确的判断。

这个时候可以通过修改返回报文中的参数，使页面跳转到写一部认证的页面。

案列如下：

忘记密码-链接的形式-链接token参数可逆

通过邮箱找回密码时，邮件中将出现一个含有token的重置URL，该token即为重置凭证。

从经验来看，开发人员习惯以时间戳、递增序号、关键字段（如邮箱地址）等三类信息之一作为因子，采用某种加密算法或编码生成token，攻击者可以基于能收集到的关键字段，用常见加密算法计算一遍，以判断是否可以预测出token。下面举两个案例加以说明。

实验一：基于时间戳生成的token，这是一个靶场：

http://lab1.xseclab.com/password1_dc178aa12e73cfc184676a4100e07dac/

案例二：基于关键字段生成的token某网站密码找回功能

忘记密码-服务端验证逻辑缺陷

1、登陆状态下修改自己的密码时，通过修改截获的数据包，将部分参数替换，从而偷龙换凤的将他人的密码修改为自己指定的密码。下面通过实例讲解。

案例如下：



The diagram illustrates a password modification form with the following fields and annotations:

- 用户名:** zhangwei (Annotated with a red arrow pointing to the text "密码是A987654321")
- 初始密码:** [Redacted]
- 新密码:** [Redacted]
- 重复密码:** [Redacted] (Annotated with a red arrow pointing to the text "密码为AAA123456")
- 修改** (Submit button)

The annotations indicate that the initial password and the new password are being swapped, demonstrating a security flaw where the user's password is modified to the attacker's specified password.

忘记密码-参数带用户名等多阶段验证

密码找回流程一般包括获取短信验证码、校验短信验证码是否有效、设置新密码等三个步骤。

在第二步，校验短信验证码是否有效的结果应保存在服务端，某些网站未在服务端保存而是错误地将结果状态值下发客户端，后续又依靠前端js判断是否可以进入第三步

那么，更改应答包中的状态值，可重置其他用户的密码。下面用两个案例分析说明。

案例一：在密码找回页面<http://www.xx.cn/yy/action/forgot>用攻击者手机号13908081024进入密码找回全流程

案例二：在密码找回页面<http://www.xx.cn/yy/forgot>用攻击者手机号13908081024进入密码找回全流程

重置密码-重置后的默认密码

在一些大型公司或者学校的网站上，会有介绍用户注册的指导手册，里面会介绍用户的用户名和默认密码。

这时可以先用默认密码尝试爆破默认密码，或者寻找重置密码的选项，重置密码后，用默认口令尝试登陆。

因为可能会未经授权修改网站用户的密码

4

任意账号密码

1未验证邮箱/手机号

未验证邮箱/手机号，目前很多应用为了方便用户记录自己的用户名与密码，都可以使用邮箱跟手机号作为登录用户名

因此很多应用在注册的时候就要求用户填写，一般情况下该应用都会发送激活信息，用户在收到激活信息后才能登录

然而有时候开发人员并不去审核邮箱/手机号是否有效，所以可以利用该缺陷，任意注册账号



填写邮箱后，并没有在邮箱发现任何激活信息，且直接把邮箱当作用户登录名使用

2. 批量注册

脚本批量注册造成服务器dos应用层攻击，影响网站的正常使用，通常由于上边无验证码或者验证码不安全导致可以写脚本来批量注册

案列如下：以某站为例，注册页面如下，并无验证码验证，可以使用burpsuite批量注册



M O C K
K I N G
B O T

登录 注册

213er2

4235456345@qq.com

免费注册

```
POST /users HTTP/1.1
Host: modao.cc
Connection: keep-alive
Content-Length: 273
Cache-Control: max-age=0
Origin: https://modao.cc
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: https://modao.cc/signup
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Cookie:
_inmock_session=blDWWJwXNLSVNwUi9CNE1ycT2INDhyMFRPTVhESTFwQ1czUUIJcHNhaEd2Q25rVWZQak1qUxG
ckRPY3hXMG1qVh1sWLNtKzVtOGVqTjkdFzUTRIc1QvL2lFMm1UaWpaWUJldQ2NUpuSVhuc3JCbjBsaaj1MUFQbGN
acn0QYIOrlpJcFBiYjUyVENkb1JGcWhtSU1zbGkzWV25MGNRQk1YbHd2dFheJFzRlNMUpqZUFuUWx3TG1YRFdtNO
NkOVNzRhhSEpqaWhtVzR6bD1vNjA0U0NRRLppSkI3enczbVNjbjBjTt0tLU9vM3JuVWdoSV1WdHcyNFNpS0IzNHc9P
Q%3D%3D--d4316283af5c12de61cf468fa2: _ga=GA1.2.916782811.1534126150;
_gid=GA1.2.151954058.1534126150; _gat=1

utf8=%E2%9C%93&authenticity_token=iSaC7qrlcb%2BH%2BPqVfrzH0VKIgwI%2FcaHKiAgHp0hapL4nWdnq8
Ek%2BuhJWqOvc1h2y56v42jW0WnYvJp78%2BMA%3D%3D&user%5Bname%5D=325423&user%5Bemail%5D=3452321
143%40qq.com&user%5Bpassword%5D=1234567&user%5Bcode%5D=&inapp=0&client_id=916782811.1534126
150
```

```
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://modao.cc
Access-Control-Allow-Methods: GET, POST, PUT, DELETE
Access-Control-Allow-Headers: Origin, Accept, Content-Type, X-Requested-With, X-CSRF-Token
Location: https://modao.cc/workspace/apps
Vary: Accept-Encoding
Cache-Control: max-age=0
Set-Cookie:
_inmock_session=UHJ5ZmU1clFFY2I0Y0RreVhml1Bpc001UXJhenlGd0w4bktmYjUwTG12YidodU9WNEtxMEtNY1F
RYWlzeVhsITd6aUoyVG1wN1U4OXRXkT5mRUJRcXNwbjJlMX11bVdRVUwSSjJlDE11Z1B1ZkM2eVWwQVNWbzBSVUJzN
mE5b0J3SU0xMytLdG40YURGNXQwZ1dDa0MzVnFqUm1aQ1o4L1F0NXdcSaVZlTm5aTjArR2xYd2NzZS9VRU10MDAzQyt
PcnpJaWJ6YU2tZlAxcS90VkhzT3c1eFd5a3FrzBjRW96ZDY0RnBhVz11NUZncXI1azdjUT1Sek5JRmdEd2pRXdkY
UzaZB2JQnFablg0enJNE5JcDgw2nlkNFRKRoazbz12REVRK25xampK0X1qbUhaC25FNF1vHUN5Rw1d0p4NjFqSER
zbVhuZHR1Z292R2RQRjNuZWVhVENBefJ1YURCNDZMa3ZiVY1ESkhFT01YSXkrTVRjYWtvVH100DEyWwtZ11jrQ1h60
Tg9LS012WxneHpES12sWkpCdE1RUHROK253PT0%3D--2f6a2230651929ebe94291ecadd187d5cb3d5fd;
domain=.modao.cc; path=/; expires=Tue, 13 Aug 2019 02:12:41 -0000; HttpOnly
X-Request-Id: efb8af0-b23c-40f7-864a-aee29890bb31
X-Runtime: 0.563283
Expires: Mon, 13 Aug 2018 02:12:41 GMT
Cache-Control: no-cache, private, no-store, must-revalidate, max-stale=0, post-check=0,
pre-check=0
Content-Length: 97

<html><body>You are being <a
href="https://modao.cc/workspace/apps">redirected</a>.</body></html>
```

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
2	1122334455	0123456789	302	<input type="checkbox"/>	<input type="checkbox"/>	1697	
3	11111111	11111111	302	<input type="checkbox"/>	<input type="checkbox"/>	1697	
4	12344321	1111111111	302	<input type="checkbox"/>	<input type="checkbox"/>	1697	
5	147258369	1111111111	302	<input type="checkbox"/>	<input type="checkbox"/>	1697	
6	31415926	11223344	302	<input type="checkbox"/>	<input type="checkbox"/>	1697	
7	12345qwert	12121212	302	<input type="checkbox"/>	<input type="checkbox"/>	1697	
8	1234qwer	1234qwer	302	<input type="checkbox"/>	<input type="checkbox"/>	1697	
9	1a2b3c4d	12344321	302	<input type="checkbox"/>	<input type="checkbox"/>	1697	
10	1a2b3c4d5e	12345qwert	302	<input type="checkbox"/>	<input type="checkbox"/>	1697	
11	a1b2c3d4	12345678	302	<input type="checkbox"/>	<input type="checkbox"/>	1697	
12	a1b2c3d4e5	123456789	302	<input type="checkbox"/>	<input type="checkbox"/>	1697	
13	abcd1234	1234567890	302	<input type="checkbox"/>	<input type="checkbox"/>	1697	

Vary: Accept-Encoding

Set-Cookie:

```
domain=.nodao.cc; path=/; expires=Tue, 13 Aug 2019 02:17:14 -0000; HttpOnly
```

X-Runtime: 0.710889

Cache-Control: no-cache, private, no-store, must-revalidate, max-stale=0, post-check=0, pre-check=0

```
<html><body>You are being <a href="https://modao.cc/workspace/apps">redirected</a>.</body></html>
```

登录

注册

 dds111111152343@163.com



.....

登录

3.个人信息伪造

若出现身份证信息注册，可任意构造绕过身份证与姓名任意填写

注册角色:	场馆注册
登录用户名:	liucy
用户密码:	*****
地址:	吉林长春
姓名:	王天行
身份证号码:	411081199004235955
所属省市:	辽宁省 营口市
<input type="button" value="确定"/>	

basketball.sportreg.org 显示

注册成功,可以登录维护信息!

4.前端验证审核绕过

任意填写注册信息，服务端对注册信息进行审核，例如是否存在恶意标签等恶意信息但是通过返回状态值给前端判断，一旦篡改该值就有可能绕过

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burpsuite或fiddle，之后点击确定，服务器会返回验证码错误之类的信息，

使用正确的信息例如{"MessageHeader":{"MessageID":"RSP036","Description":"成功！"}}进行替换后再执行，注册成功。

具体测试方法和验证码的前端验证同理，这里不再赘述

5. 邮箱/手机号注册激活验证绕过

为防止恶意用户任意注册账户，大多数网站会在用户注册中输入邮箱/手机号后对其真实性进行验证

但是有时候返回的验证信息会直接隐藏在返回包中，只是不在前端显示出来，或者是可以通过抓包改包手机号/邮箱，伪造该信息，劫持到验证信息

以某验证信息返回为例，该漏洞是在发送验证信息时会将验证信息同时发送到返回包中并将其在前端用hidden属性隐藏其值

所以直接前端源代码查看即可

```
if(strlen($username)>10){  
    $username = substr($username, 1, 10);  
    echo "截断后username:". $username. "<br>";  
}  
$sql = "insert into users(username, password) VALUES ($username, $password)";
```

6. 用户名覆盖

渗透测试中，某些不严谨的开发人员，对于注册时，当输入用户名时不会对之前数据库中存在的用户名进行检查，判断是否存在

登录查看时却获取到数据库中同名用户的其他用户信息，导致其他用户信息泄露，或者由于验证用户名存在时，从前端获取到的数据与从数据库获取到的数据不同，但是往数据库中写入的时候却写了相同的部分

而登陆时，当检测到是管理员用户名就给与管理权限。

```
if(strlen($username)>10){  
    $username = substr($username, 1, 10);  
    echo "截断后username:". $username. "<br>";  
}  
$sql = "insert into users(username,password) VALUES ($username,$password)";
```

username	password
admin	admin123
admin	asd

在登陆的时候可以看到他只判断了username（但是实战中出现的几率还是偏少）

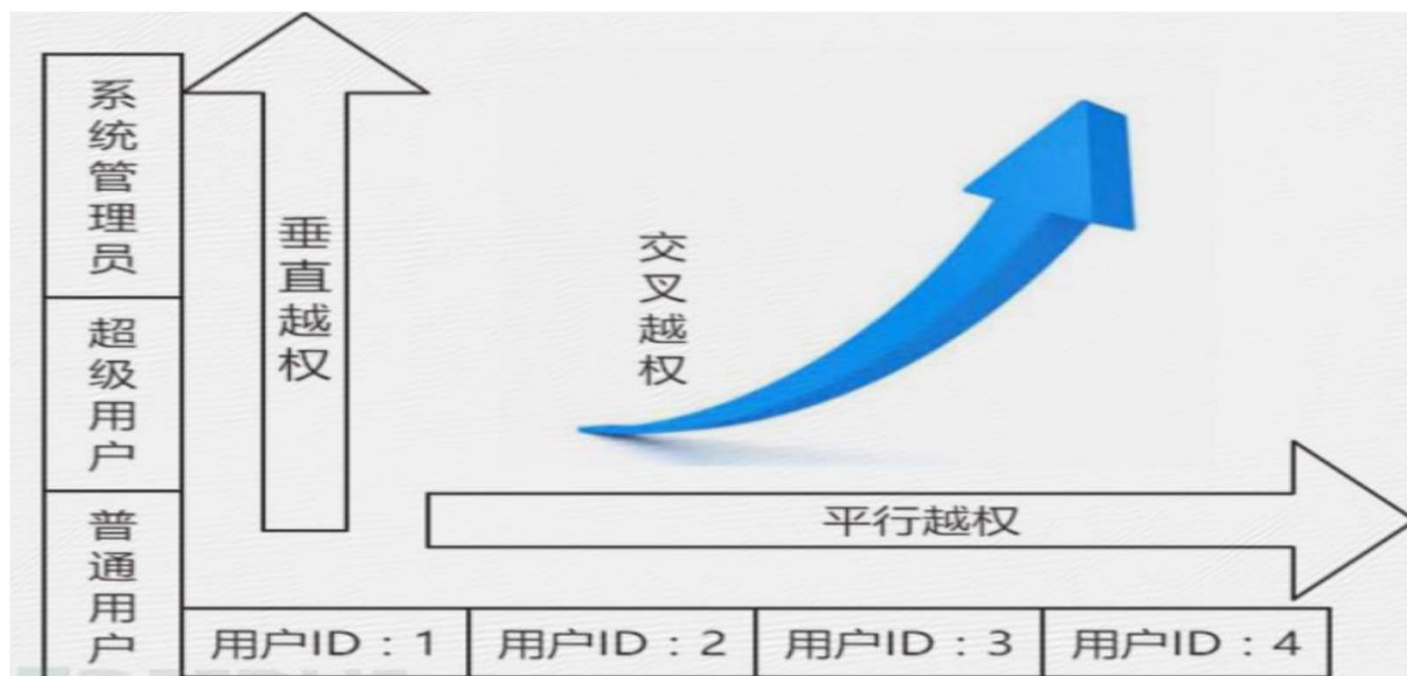
```
while($row = mysqli_fetch_assoc($result)) {  
    if($username === "admin") {  
        echo "flag: flag{2qqqqqqqqqqqq} <br>";  
    }  
}
```

5

权限类漏洞

越权

越权漏洞又分为平行越权，垂直越权和交叉越权。平行越权：权限类型不变，权限ID改变垂直越权：权限ID不变，权限类型改变交叉越权：即改变ID，也改变权限



1 平行权限跨越

水平越权指的是攻击者尝试访问与他拥有相同权限的用户的资源，怎么理解呢？

比如某系统中有个人资料这个功能，A账号和B账号都可以访问这个功能，但是A账号的个人信息和B账号的个人信息不同

可以理解为A账号和B账号个人资料这个功能上具备水平权限的划分。此时，A账号通过攻击手段访问了B账号的个人资料，这就是水平越权漏洞。

系统中所有具备水平权限划分的功能，都存在水平越权的风险，以下是常出现的水平越权的功能的几种场景：

案列如下：

2垂直权限跨越

垂直越权指的是一个低级别攻击者尝试访问高级别用户的资源。比如说某个系统分为普通用户和管理员，管理员有系统管理功能，而普通用户没有，那我们就可以理解管理功能具备垂直权限划分，如果普通用户能利用某种攻击手段访问到管理功能，那我们就称之为垂直越权。

案列如下：

6

其他漏洞

1、数据包重放漏洞

漏洞介绍：通过数据包重放，可以造成短信轰炸、邮件轰炸、重复提交订单等

漏洞原理：后台未进行相关操作的技术导致数据包重放漏洞点：短信验证码、邮件校验、提交订单等功能。

修复方案：修复思路（针对短信、邮件）

构造一个HashMap<String, short>，存放邮箱或电话号码及对应次数

只要某个邮箱或者电话号码次数够了，就不能继续发送了

或者计算两次发送的时间间隔，时间过短就不继续发送了

通用修复方案

需要建立token机制或验证码机制，一次有效

2、条件竞争漏洞

漏洞介绍：条件竞争是指一个系统的运行结果依赖于不受控制的事件的先后顺序。当这些不受控制的事件并没有按照开发者想要的方式运行时，就可能会出现bug。尤其在当前我们的系统中大量对资源进行共享，如果处理不当的话，就会产生条件竞争漏洞。说的通俗一点，条件竞争涉及到的就是操作系统中所提到的进程或者线程同步的问题，当一个程序的运行的结果依赖于线程的顺序，处理不当就会发生条件竞争

漏洞修复-修复思路：

限制同一时间内访问方法的只有单一线程

案列与实操如下：

3、订单金额任意修改

很多中小型的购物网站都存在【订单金额任意修改】漏洞。在提交订单的时候抓取数据包或者直接修改前端代码，然后对订单的金额任意修改。



经常见到的参数大多为：rmb、value、amount、cash、fee、money

关于支付的逻辑漏洞这一块还有很多种思路，比如相同价格增加订单数量，相同订单数量减少产品价格，订单价格设定为负数，无限叠加优惠券等等

4接口无限制枚举

有些关键性的接口因为没有做验证或者其它预防机制，容易遭到爆破攻击。常见账号爆破，密码爆破，验证码爆破，上文都已经提及，还有其他的：

案列如下：

快递公司的优惠券枚举

某电商会员卡卡号枚举

某超市注册用户信息获取

5、支付漏洞

支付漏洞是**高风险漏洞**也属于**逻辑漏洞**，通常是通过篡改价格、数量、状态、接口、用户名等传参，从而造成**小钱够买大物**甚至可能造成**0元购买商品**等等，凡是**涉及购买、资金**等方面的功能处就有可能存在支付漏洞

商户网站接入支付结果，有两种方式，一种是通过**浏览器进行跳转通知**，一种是**服务器端异步通知**。

浏览器跳转通知

基于用户访问的浏览器，如果用户在银行页面支付成功后，直接关闭了页面，并未等待银行跳转到支付结果页面，那么商户网站就收不到支付结果的通知，导致支付结果难以处理。而且浏览器端数据很容易被篡改而降低安全性(**这种方式数据经过了客户端浏览器，极大的可能性被第三方恶意修改**)

服务器端异步通知

该方式是**支付公司服务器后台直接向用户指定的异步通知URI发送参数**，采用POST或者GET的方式。商户网站接受异步参数的URL对应的程序中，要对支付公司返回的支付结果进行签名验证，成功后进行支付逻辑处理，如验证金额、订单信息是否与发起支付时一致，验证正常则对订单进行状态处理或为用户进行网站内入账等。

案列如下：

6、越权支付

通过修改一些**特殊传参(如:id,username,openid)**来达到用他人的资金来干购买自己的商品。

案列如下:

案列：小程序越权积分兑换

7、无限制试用

通过修改特殊传参(如:id,pay,test)来达到无限制试用。

案例：艺龙旅行网严重支付漏洞

支付漏洞总结

支付漏洞总结

1、找到关键的数据包

可能一个支付操作有三四个数据包，我们要对数据包进行挑选。

2、分析数据包

支付数据包中会包含很多的敏感信息（账号，金额，余额，优惠），要尝试对数据包中的各个参数进行分析。

3、不按套路出牌

多去想想开发者没有想到的地方。

4、pc端尝试过，wap端也看看，app也试试

防御方法

1、在后端检查订单的每一个值，包括支付状态；

2、校验价格、数量参数，比如产品数量只能为整数，并限制最大购买数量；

3、与第三方支付平台检查，实际支付的金额是否与订单金额一致；

4、如果给用户退款，要使用原路、原订单退回。比如：退押金，按用户原支付订单原路退回；

5、MD5加密、解密、数字签名及验证，这个可以有效的避免数据修改，重放攻击中的各种问题；

6、金额超过指定值，进行人工审核等。

7

SCR中逻辑漏洞检查总结

注册:

- 短信轰炸
- 验证码安全问题
- 密码爆破
- 邮箱轰炸
- 用户任意注册、批量注册
- 用户名枚举
- XSS (有框的地方就可以尝试插XSS)

登录:

- 短信轰炸、验证码安全问题、密码爆破、邮箱轰炸
- SQL注入
- 撞库
- 抓包把password字段修改为空值发送
- 认证凭证替换、比如返回的数据包中包含账号，修改账号就能登录到其他账号
- Cookie仿冒
- 修改返回包的相关数据，可能会登陆到其他用户

找回密码:

- 短信邮箱轰炸、短信邮箱劫持
- 重置任意用户账户密码、验证码手机用户未统一验证
- 直接跳过验证步骤
- 购买支付、充值（要利用抓包去仔细查看每一个可用的参数）
- 交易金额、数量修改、更换支付模块（比如更换支付的模块金额）
- 交易信息订单编码/导致信息泄露
- 整数溢出，int最大值为2147483647，超过最大值
- 修改充值账户
- 支付绕过
- 抽奖活动
- 刷奖品、积分
- 并发
- 优惠券、代金券
- 并发逻辑漏洞（burp批量获取优惠券）
- 修改优惠券金额、数量
- 订单信息
- 订单信息遍历、泄露
- 订单信息泄露导致用户信息泄露
- 删出他人订单

会员系统:

- 修改个人信息上传文件, 上传带弹窗的html
- 如遇上上传xlsx、docx, 可能存在XXE, 上传恶意的文档盲测
- 图片上传也可能遇到imagereagick命令执行, 上传恶意图片
- 视频上传如果使用ffmpeg<3.2.4 (视频按帧分割成图片), 上传恶意avi盲测ssrf
- 用户横向越权访问、遍历、导致用户信息泄露
- SQL注入、个人简历处存储XSS个人信息注册的名称也可以插入XSS

传输过程、评论

- 明文传输账户密码
- 修改信息处无session/token导致csrf
- POST/COOKIE注入
- POST注入
- 存储型XSS
- 无session/token导致CSRF

验证码问题、短信轰炸

- 万能验证码
- 返回包中存在验证码
- 删除验证码或者cookie中的值可以爆破账号密码
- 一直重放
- 删除修改cookie, 重放数据包
- 遍历参数发送数据包
- 手机号后面加空格或者前面加其他的比如+86或者逗号分号等, 然后重发数据包
- 请求参数修改大小写, 或者添加请求参数比如&id=1
- 一个站的登录处可能做了防护, 但是再找回密码处可能没有安全防护, 或者在注册流程中没有安全防护, 所以说多测试接口
- 如果对手机号一天的次数进行了限制, 还可以再发一次短信, DOIntercept之后修改为成功回显

水平越权、数据泄露、任意用户密码重置

- 主要登陆后还是修改参数，主要找到多个接口不断测试
- 关注网页源代码，有时候会有表单，但被bidden（隐藏标签）给隐藏起来了，可以修改返回包然后尝试获取数据检测
- 多个账号，主要分析请求参数
- 再找回密码处，填写数据后抓包查看返回信息，有可能存在敏感数据返回删除修改cookie，重放数据包
- 目前大部分都是在修改密码处参数修改
- 有些是前端验证

支付逻辑漏洞

- 金额直接传输导致篡改：直接对下单的金额进行修改值，这里可以使用fd或者burp抓包
- 确定支付之后还可以加入购物车：把商品放入购物车点击下单支付，会跳转到微信，支付宝等第三方支付平台。
- 请求重放：购买成功之后，继续重放请求，可以让购买的商品一直增加。
- 请求参数干扰：金钱做了签名认证之后，修改后不通过，但是在里面仍然会有一个参数对金额产生影响导致问题产生。
- 订单替换：订单替换发生在支付之后的事件处理，同时向服务器发起二次支付请求一个多一个少，支付金额少的，然后支付之后进行替换，告知服务器订单支付完成，并且过程可以反复的回放。
- 用户替换：在支付过程中发生用户替换现象，首先登陆自己的账户，然后取得另外一个人的账户名等有效信息，在业务流程中用对方的用户名替换自己的用户名，用对方的余额购买完成后，再替换自己的账户名，这样就形成别人的钱买自己的东西

登录绕过、水平越权、垂直越权

- 部分网站的身份验证放在了前端，因此只需要将response包中的相关字段进行修改，比如0改成1，false改成true，就可以登录任意用户账号
- 遍历ID-在一些请求中，GET和POST中有明显的ID数字参数（手机号、员工号、账单号、银行卡号、订单号等等）
- ID替换如果程序对用户标识进行了hash或者加密，而无法破解用的什么方式的话，就无法通过遍历ID来获取其它用户的信息了，此时可以尝试注册两个账号，通过替换两个ID加密后的值，判断程序是否对权限进行了验证，如果没有，也会存在越权问题
- 观察cookie中的session字段，可能某些字段或者参数代表身份，尝试修改