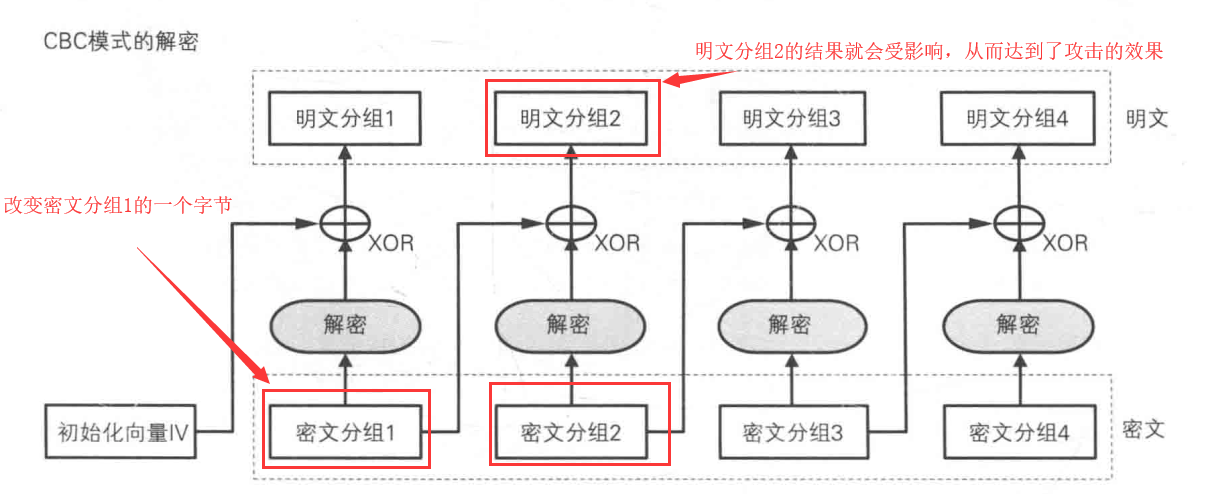
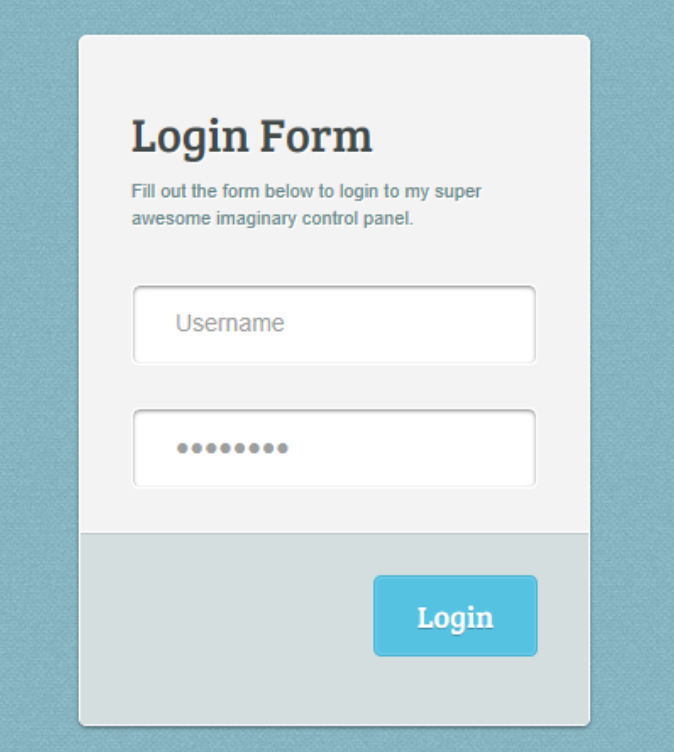
CBC字节翻转攻击

1.cbc字节翻转攻击的原理



2. bugku-login4

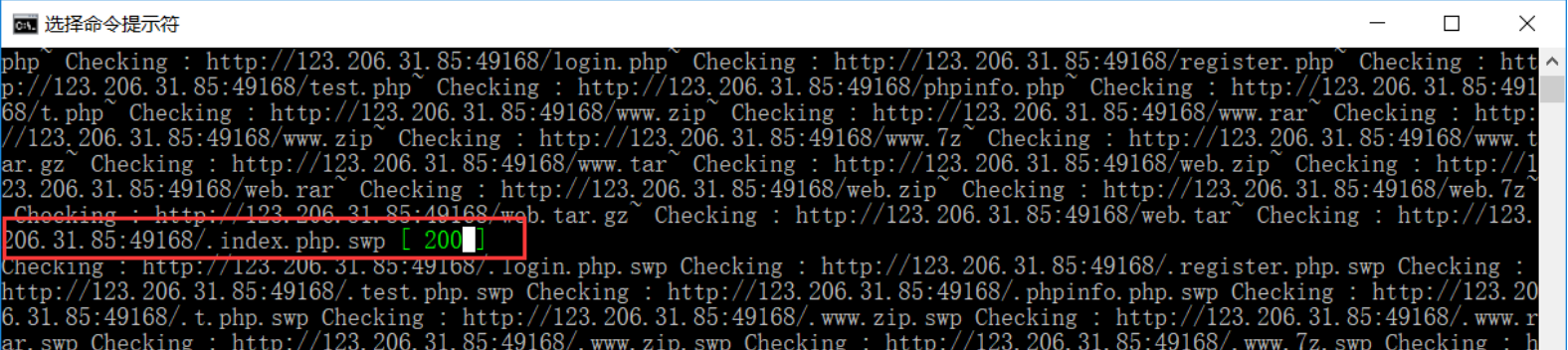
打开是一个登陆界面，随便输入一个账号密码就能登陆，但是admin不能登陆



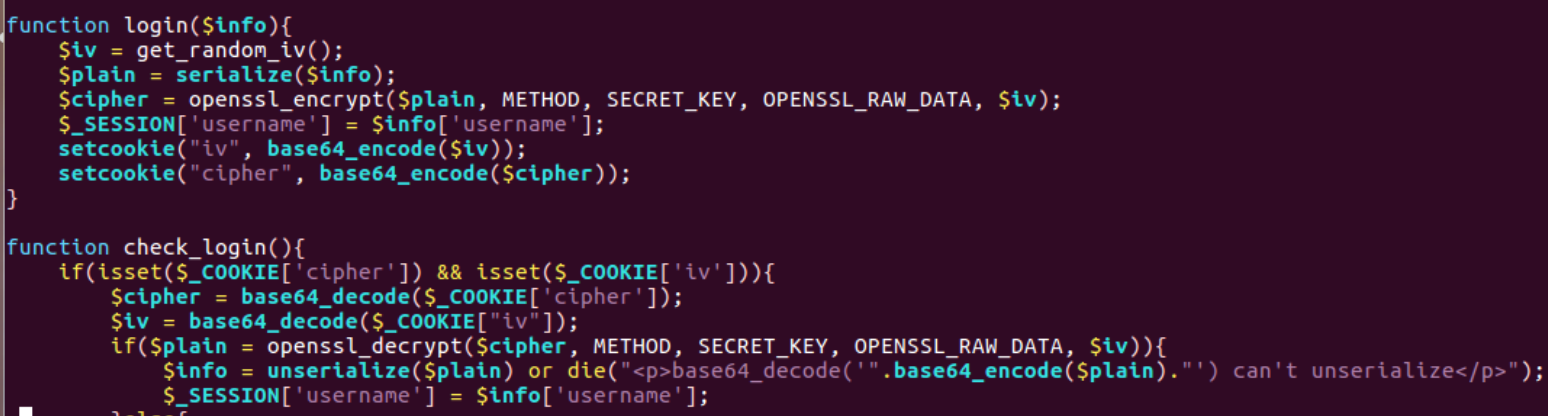
抓包发现，有个iv和cipher，猜测是CBC模式的加密

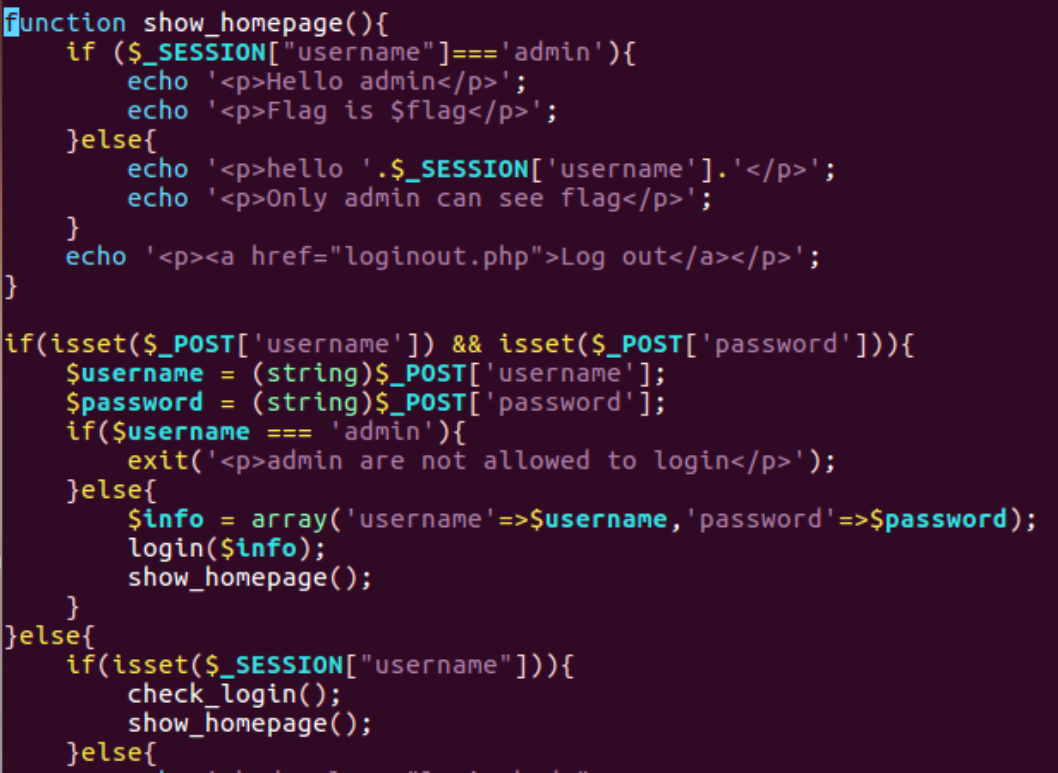


扫描一波发现有swp源码泄露



vim -r index.php.swp恢复后得到源代码，审计关键代码





进行代码审计，可以看到要想获得 flag 需要以 admin 的身份登陆，但是又禁止了直接用 admin 登录，但是是通过获取 cookie 中的值来判断是否为 admin，所以就需要利用 cookie 伪造登录。这里把登录的用户名及其密码存入数组，序列化后进行 AES-CBC 模式的加密，其中 iv 和 cipher 以 cookie 储存，可以控制，导致存在攻击的可能，即利用 CBC 字节翻转攻击。然后以账号为admia，密码为12345登陆得到的明文是：a:2:{s:8:"username";s:5:"admia";s:8:"password";s:5:"12345"}然后16个字节分组得到

1. s:2:{s:8:"userna
2. me";s:5:"admia";
3. s:8:"password";s

④:3:"12345";}

所以我们要翻转的是第二组的“a”翻转为“n”，所以要改变第一组的密文从而达到攻击的效果.

利用cbc字节翻转代码如下：

import base64

cipher="yQQeUDxlzRvPToe631KV1vcy8DyI4e0kz7Knb9K6GIH4yP8Q32kufQvWoD7oN3hzi2EpiBxx6t/7sfIH1pCExg=="

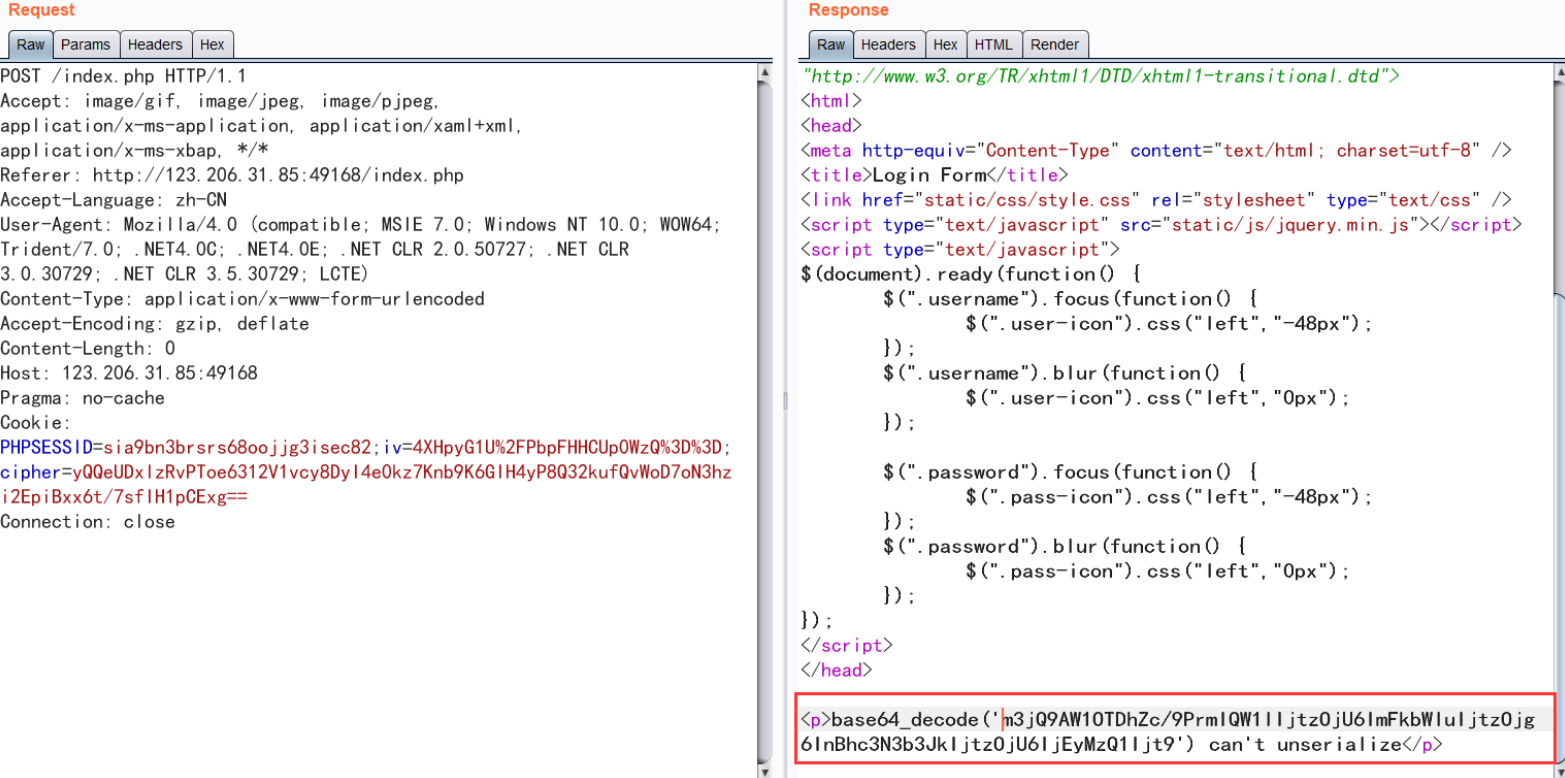
plain=base64.b64decode(cipher).decode('unicode\_escape')

result=plain[0:13]+chr(ord(plain[13])^ord("n")^ord("a"))+plain[14:]

print(base64.b64encode(result.encode('latin-1')).decode())

#print yQQeUDxlzRvPToe6312V1vcy8DyI4e0kz7Knb9K6GIH4yP8Q32kufQvWoD7oN3hzi2EpiBxx6t/7sfIH1pCExg==

POST过去提示不能正常反序列化，因为修改了第一组的密文，导致第一组的密文与iv xor会出错，从而导致了第一组明文不能正常解密，所以我们还要对iv进行修改



代码如下

import base64

cipher="wRPT3VONV2zFV6D2PbHjIm1lIjtzOjU6ImFkbWluIjtzOjg6InBhc3N3b3JkIjtzOjU6IjEyMzQ1Ijt9"

plain=base64.b64decode(cipher).decode('unicode\_escape')

oldiv=base64.b64decode("uxrq4TtskqrNJh7JUZV9rg==").decode('unicode\_escape')

one='a:2:{s:8:"userna'

iv=""

for i in range(0,16):

iv=iv+chr(ord(one[i])^ord(plain[i])^ord(oldiv[i]))

print(base64.b64encode(iv.encode('latin-1')).decode())#GzMLBhOS//4yU8tMCVbw7Q==

将得到的值修改为 iv，刷新页面得到 flag

