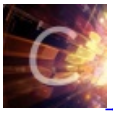
实验吧Writeup

转载

千尺浪 于 2017-10-18 18:04:25 发布 732 收藏
分类专栏： CTF竞赛 文章标签： 实验吧 Writeup


CTF竞赛 专栏收录该内容
3 篇文章 0 订阅
订阅专栏

自以为sql注入掌握的还是比较系统的，然而，做了这些题之后才发现，大千世界无奇不有，真是各种猥琐的思路...还是要多学习学习姿势跟上节奏

登录一下好吗？？

http://ctf5.shiyanbar.com/web/wonderkun/web/index.html

试了一下发现他过滤了/ or union select - #

虽然and ' " = + %没有被过滤，但用%0b和%23都没效，于是还在想这是用了多麻烦的WAF，越想越复杂我竟然还去用xss，真是跑题了...

然而最后的payload是这样的...跪了，果然题刷的太少吗

whoare you?

http://ctf5.shiyanbar.com/web/wonderkun/index.php

进入这个页面，发现他获取了我的ip，首先想到的就是user agent注入，用updatexml试一下，什么错的没有报...看来这个思路不行

百度了才知道这是一道伪造IP的题，相关知识可以看这里：http://www.cnblogs.com/x2048/articles/1794020.html

总结下，伪造IP的HTTP头都有这些：

```
X-Forwarded-For
Client-IP
x-remote-IP
x-originating-IP
x-remote-addr
```

一般用的最多的就是前两个，这里我们用X-Forwarded-For来伪造

但是很奇怪，不管构造的是什么语句，都会返回到页面，看了下别人的writeup发现，他的后台处理过程大致是这样的，首先获取到HTTP-X-Forwarded-For，对他进行字符串的处理，只截取逗号前的内容，然后直接将其输出到页面，再插入到数据库，但应该没有对插入结果做处理，即没有输出数据库的报错仅输出空，所以想从数据库的报错获取信息应该是不行了，返回页面也是不具判断性的，那么可以考虑时间型的盲注

这里解决的方法可以有三种:

1、写个盲注脚本跑

这是跑出来的库名：

表名：

字段：

于是，我们知道了flag存储在flag表的flag字符里，且长度为32，接下来直接跑就行了

2、用burp进行时间盲注

3、用sqlmap进行http头的盲注

然后，要注意的是跑出来的结果要加在ctf{xxx}里....被坑的很惨，跑出来flag一直提交不对，最后发现花括号用的中文字符输入的...

因缺思汀的绕过

http://ctf5.shiyanbar.com/web/pcat/index.php

查看网页的源码，发现登录的源码路径是source.txt

于是拿到源码如下：

```
1 <?php
2 error_reporting(0);
3
4 if (!isset($_POST['uname']) || !isset($_POST['pwd'])) {
5     echo'<form action=""method="post">'."<br/>";
6     echo'<input name="uname"type="text"/>'."<br/>";
7     echo'<input name="pwd"type="text"/>'."<br/>";
8     echo'<input type="submit" />'."<br/>";
9     echo'</form>'."<br/>";
10     echo '<!--source: source.txt-->'."<br/>";
```

```
    die;
}

function AttackFilter($StrKey,$StrValue,$ArrReq){
    if (is_array($StrValue)){
        $StrValue=implode($StrValue);
    }
    if (preg_match("/".$ArrReq."/is",$StrValue)==1){
        print "姘村彨杞借埅锛屼害鍙□濈鐨勫紒";
        exit();
    }
}

$filter ="and|select|from|where|union|join|sleep|benchmark|,|\(|\)";
foreach($_POST as $key=>$value){
   AttackFilter($key,$value,$filter);
}

$con = mysql_connect("XXXXXX","XXXXXX","XXXXXX");
if (!$con){
    die('Could not connect: ' . mysql_error());
}
$db="XXXXXX";
mysql_select_db($db, $con);
$sql="SELECT * FROMinterest WHERE uname = '{$_POST['uname']}'";
$query = mysql_query($sql);
if (mysql_num_rows($query) == 1) {
    $key = mysql_fetch_array($query);
    if($key['pwd'] == $_POST['pwd']){
        print "CTF{XXXXXX}";
    }else{
```

```
42        print "浜〜鬆璧涜堵锛€";
43    }
44 }else{
45    print "涓€棰楠滋鑶囷紒";
46 }
47 mysql_close($con);
48 ?>
```

从源码中可以知道这些全都被过滤了：and|select|from|where|union|join|sleep|benchmark|,|\(|\)

并且数据库中只有一条数据

```
if (mysql_num_rows($query) == 1) {
```

最核心的部分是这里

```
$sql="SELECT * FROM interest WHERE uname ='{$_POST['uname']}'";
```

大致的执行过程是先将用户输入的uname作为查询条件，在数据库中查询uname和pwd，然后将查询到的pwd与用户输入的pwd进行比较，内容一致才输出flag

这里的思路是，利用group by pwd with rollup在查询中的一个特点，他可以返回pwd所在的那一条记录，通过limit控制返回哪一条，因此他不可以返回多条，一旦返回2条及以上，pwd就会为空，但同一条记录中的其他字段则是正常的

那么利用这一点令查询结果为空，我们输入的pwd也为空值，则构成了if(null==null)为true

简单的sql注入之三

http://ctf5.shiyanbar.com/web/index_3.php

看源码，是一个用get方式传参数id的查询，于是手工注入一波，发现单引号引起报错，加上注释后回显正常，id在1~3的范围都可以正常回显“Hello!”，否则返回空，尝试逻辑运算，可以正常返回，那这就很好办了

然后，我就开始入了手注的坑...

首先来试试有几个字段

2报错了，说明只查询出一个字段，那么用union试试

没有想要的结果回显，很奇怪，看来这个Hello不是输出的位置，应该是后台在获取到id的值后，执行sql语句，如果存在查询的结果，那么就输出Hello，但不输出结果

如果是这样的话，那么只能从报错回显入手了，先试试双注入

很尴尬，做了报错处理？那试试xpath的报错

还是不行...好吧，报错这条路也被堵了，那就考虑盲注吧

，

根据上面页面中的不同返回，我们可以写一个bool型的盲注脚本来爆数据库，当然也可以用burp搞定

这里偷个懒，因为已经知道ctf的套路了，所以可以直接来猜表名和字段

ViewCode

这是我参考了下别人的wp写的，其中最重要的是这里：

```
poc = "1'and+ascii(substr((select+flag+from+flag)%%2C%d%%2C1))%%3D%d%%23" % (x, i)
```

x是正在匹配的flag的字符位置，i是这个字符可能对应的ascii码整数，%%是Python翻译转义字符%时用的，因为是直接用的ctf的套路，也有可能不准，所以在这之前需要进行一些验证，像这样：

页面返回正常，说明存在flag表，用同样的方法可以验证flag表里存在flag字段

还有这段POC中，验证页面是否返回正常是用的这句：

```
  if res.headers['Content-Length'] == '471':
      return 1
  else:
      return 0
```

因为当语句合法时（flag字符猜解正确），页面始终返回的是Hello!那么他的返回报文长度就是一定的，我们可以通过拦截返回包查看报文长度：

这里的长度是471，所以当Content-Length等于471时返回1，否则返回0

运行结果：

简单的sql注入之二

http://ctf5.shiyanbar.com/web/index_2.php

首先尝试单引号：

根据以上报错，可以知道这里单引号没有被过滤，他可以影响到sql语句，继续尝试其他的发现，他过滤了空格，当然，空格可以用%20，%0a，/**/，/*!*/，/*!50000*/，+，()替换，其中%20，+和()均被过滤了，尝试/**/替换空格可以绕过，对于其他的字符都没有做什么特别的过滤，于是构造如下：

这里就是这道题很奇怪的地方了，用hackbar输入的结果和直接输入效果完全不同...上面是用hackbar的结果，下面看看直接输入：

竟然就这么给我爆flag了，此时我想做一个捂脸笑哭的表情...简直就是神奇，如果有人知道为什么会这样，请一定要告诉我

还有，为什么flag和上一道题是一样的啊，出题的也太懒了吧...后台肯定是用的一张表，BCTF也出现过这种[摊手]

简单的sql注入

http://ctf5.shiyanbar.com/423/web/

首先还是尝试单引号：

没有被过滤，然后尝试逻辑运算，and被过滤了但还好or没有被过滤：

下面尝试爆表名和字段，发现他过滤了很多关键字：and，select，from，union，where，这里绕过关键字的方法有：关键字中间加/**/隔断，/*!关键字*/，关键字中间加%0b隔断，关键字重写（关关键字键字），大小写混合等等，尝试后/*!*/可以绕过，还是按套路，先拿flag做表名和字段名试试：

说明存在flag表，那基本就是老套路了，可以继续用这个方法验证，下面我就直接爆flag了：

上面直接用之前的方式会报错，后来注意到应该是后面还有个查询的判断条件，所以就算注释后还是有报错

ps:这里还有一种检验表名是否存在的方式：

天下武功唯快不破

http://ctf5.shiyanbar.com/web/10/10.php

首先看源码发现他给的提示：

和post的参数有关，那就看看报文吧

发现了一个FLAG参数是用base64编码的，解码后内容是：P0ST_THIS_T0_CH4NGE_FL4G:Hrd54sRiP，试验了几次发现：后的字符是随机的，既然他提示的是快速post提交参数key，首先想到的就是写py脚本构造请求头，脚本如下：

ViewCode

大致的执行过程就是获取报头中FLAG字段的内容，用base64解码，然后作为key的值构造post请求，最后打印请求响应的内容，即flag，运行结果如下：

但这道题很有意思的是，他其实可以不用session，貌似出题人在后台没有用session控制访问，应该是通过ip的判断实现的用户区分，我做了一个小测试：当获取到一个key值后构造post请求最大的延时时间是3s

想了想他的后台操作应该是这样的：客户端get请求该地址，服务端收到请求会随机生成一个的字符串，加密后作为返回报文的FLAG字段，即base64加密的key值，同时，后台会将加密前的该值存入数据库，这里会有一个时间判断，超过3s会从数据库中删除，若客户端在3s内获取key值并将他作为参数post后，后台会从数据库中取出进行匹配，匹配成功将会获取flag，但是，从数据库中取出时需要分辨是哪个用户在什么时候存入的（同一个用户可以做连续多次的get请求），所以这里在数据库中的一定有一个ip字段做索引，即做请求操作的ip与对应生成的随机字符串存入数据库，当用户post后会根据该用户的ip来取对应的字符串，与post的字符串做匹配，这样，就可以不用session来控制访问了

还有一道类似的题，但需要使用session来访问：http://web.sniperoj.cn:10003/

首先还是抓包看看:

可以看到报文里有一个hint和get_flag字段，意思就是将get_flag解密后作为sinperOJ的值，提交post请求获取返回的内容，把get_flag解密后也是随机字符串，但是，他和上面那道题有一个明显的不同，他的最大时间限制大约是24小时，所以，可以推测应该是用了session来控制访问，于是还是和之前的套路一样，写一个脚本来跑，把之前的稍微改一下：

ViewCode

运行结果：

让我进去

http://ctf5.shiyanbar.com/web/kzhan.php

这道题的突破口很奇怪，抓包后发现cookie里有一个可疑的参数source=0

尝试把source改成1后，就爆源码了，所以说ctf里的姿势都是千奇百怪的...下面是源码：

ViewCode

研究一下他的源码，看看他是怎样实现的，简单来说就是cookie中的参数getmein不为空时，传入经过url编码的参数username和password，然后将两者解码后与后台一个参数secret拼接成一个字符串，然后做一个md5加密，如果和cookie中参数getmein的值相同，则会爆出flag，但问题是cookie中没有这个参数getmein，也就是说需要我们自己构造，但是可以根据cookie中存在的参数sample-hash，其生成的原理进行构造，sample-hash是一个md5值，解码该值后可以获得后台的secret，然后就可以根据这个值构造正确的md5值了，但是md5加密是单项不可逆的，要解密的话一般方法是用暴力破解，暴力肯定是不可能了，毕竟secret有15位，这里要用到的一个知识点就是hash长度扩展攻击，这里有一篇文章就是讲解这个知识点的：http://www.freebuf.com/articles/web/69264.html

思路是这样，但这道题还没吃透，先过

拐弯抹角

http://ctf5.shiyanbar.com/10/indirection/

这道题一来就给了源码，大致思路就是安照他的源码，按要求构造请求的url，先放源码：

ViewCode

于是根据这些要求最后构造出来的url如下：

很奇怪的是，如果url是这样的，也能返回正常页面：

在第七条要求中不是说不能和他相同吗...不过，这道题确实提供了不错的解题思路

安女神之名

http://ctf5.shiyanbar.com/10/an.php

这道题根据题意先拿安女神试试手，确实是被过滤了，这时意识到需要汉字转码，直接把安女神转成unicode编码安女神

但是这里有一个坑，之前上传的参数被记到浏览器的cookie中去了，所以还要先清理一些浏览器的cookie，然后再构造如下url，查看源码发现flag

Forms

http://ctf5.shiyanbar.com/10/main.php

对于这类题已经总结出一些经验了，一般页面简单且没有特别提示的，那么不是源码里有东西，就是发送请求的包里有东西，这道题就是在源码里，而且多半是提供特别信息或者获取源码之类的

可以看到在源码中有一个隐藏表单，名字是showsource，value是0，相似的题之前已经遇到过了，不过之前是在发送的包里，同样的我们把这里的value改成1，即可获取源码：

之后的就简单了，直接根据源码post数据，结果如下：

天网管理系统

http://ctf5.shiyanbar.com/10/web1/

这道题还是看源码：

看到注释里面有一个提示，当传入的username值经md5加密后等于0，就会返回某样东西，多半就是flag或者源码

在某些情况下，PHP会把类数值数据（如含有数字的字符串等）转换成数值处理，== 运算符就是其中之一。在使用 == 运算符对两个字符串进行松散比较时，PHP会把类数值的字符串转换为数值进行比较，如果参数是字符串，则返回字符串中第一个不是数字的字符之前的数字串所代表的整数值。比如: '3' =='3ascasd'结果为true。

推荐看看这篇writeup:http://www.cnblogs.com/ssooking/p/5877086.html

因此只要找到一个字串加密后第一个字符为0即可，总体结构就是e+数字，这里提供几个：240610708，aabg7XSs，aabC9RqS，s878926199a、QNKCDZ

这里我们获取到一个路径，多半就是源码的路径了：

源码中关键的一个函数unserialize()，这是一个和php序列化有关的函数：

serialize() 对输入的数据进行序列化转换
unserialize()恢复原先变量，还原已经序列化的对象。

他的大致意思是，将接收到的password经过序列化还原成数组，其中user和pass都要等于后台的某个值，然后会返回一个flag，但关键是我们并不知带后台的这两个值是什么，于是这里考察的就是php的弱类型了，bool类型的true跟任意字符串都可以弱类型相等，因此我们可以构造bool类型的序列化数据：a:2:{s:4:"user";b:1;s:4:"pass";b:1;}这句话的意思是password是一个长度为2的数组（a指数组，s指字符string，数字是长度），他的一个元素是user，长度为4，bool值为1，另一个元素是pass，长度为4，bool值为1（b指bool值），于是运行后结果如下：

参考文献：

http://www.jianshu.com/p/5d34b3722128

http://blog.csdn.net/qq_34841823/article/details/54313457#