条件竞争

原创

恋物语战场原 于 2019-06-03 11:23:25 发布 3571 收藏 9

分类专栏： web安全 文章标签： 条件竞争 web安全

本文链接：https://blog.csdn.net/qq_26406447/article/details/90749288

条件竞争

前言

前面强网杯的时候做upload，有大佬提出过条件竞争的思路，虽然最后不是，但看了下条件竞争感觉还是很有意思的，这里来学习一下

下面是从别人博客中引用，我感觉是把条件竞争讲的挺清楚的，举的例子和大佬在比赛中分析的也很像

> 下面以相关操作逻辑顺序设计的不合理为例，具体讨论一下这类问题的成因。在很多系统中都会包含上传文件或者从远端获取文件保存在服务器的功能（如：允许用户使用网络上的图片作为自己的头像的功能），下面是一段简单的上传文件释义代码：

```
<?php
  if(isset($_GET['src'])){
    copy($_GET['src'],$_GET['dst']);
    //...
    //check file
    unlink($_GET['dst']);
    //...
 }
?>
```

> 这段代码看似一切正常，先通过 copy($*GET['src'],$*GET['dst']) 将文件从源地址复制到目的地址，然后检查 $*GET['dst']的安全性，如果发现$*GET['dst'] 不安全就马上通过 unlink($_GET['dst']) 将其删除。但是，当程序在服务端并发处理用户请求时问题就来了。如果在文件上传成功后但是在相关安全检查发现它是不安全文件删除它以前这个文件就被执行了那么会怎样呢？
> 假设攻击者上传了一个用来生成恶意shell的文件，在上传完成和安全检查完成并删除它的间隙，攻击者通过不断地发起访问请求的方法访问了该文件，该文件就会被执行，并且在服务器上生成一个恶意shell的文件。至此，该文件的任务就已全部完成，至于后面发现它是一个不安全的文件并把它删除的问题都已经不重要了，因为攻击者已经成功的在服务器中植入了一个shell文件，后续的一切就都不是问题了。
> 由上述过程我们可以看到这种“先将猛兽放进屋，再杀之”的处理逻辑在并发的情况下是十分危险的，极易导致条件竞争漏洞的发生。

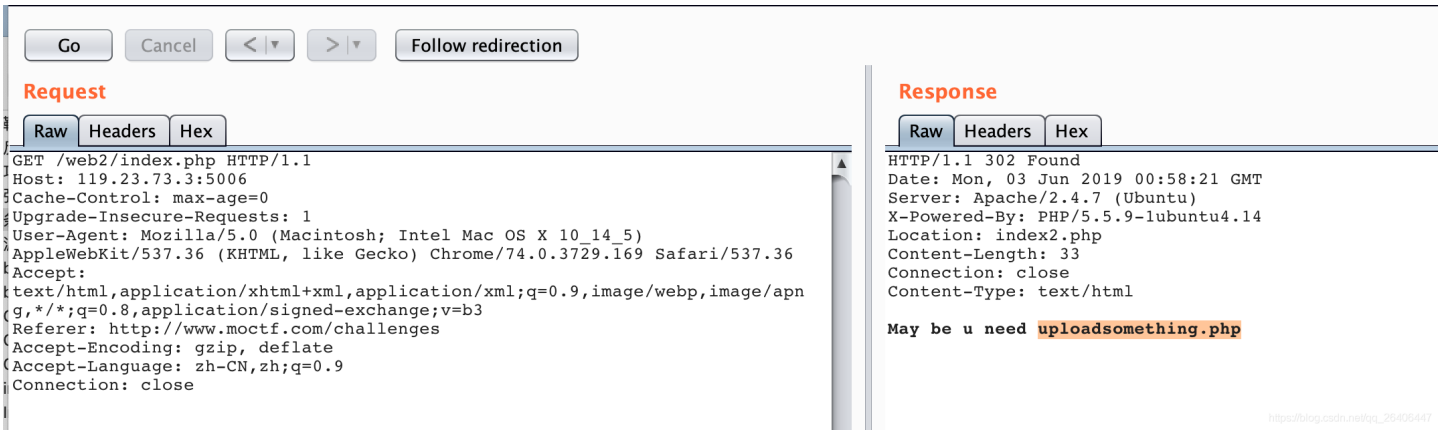
练习

Montf — 没时间解释了

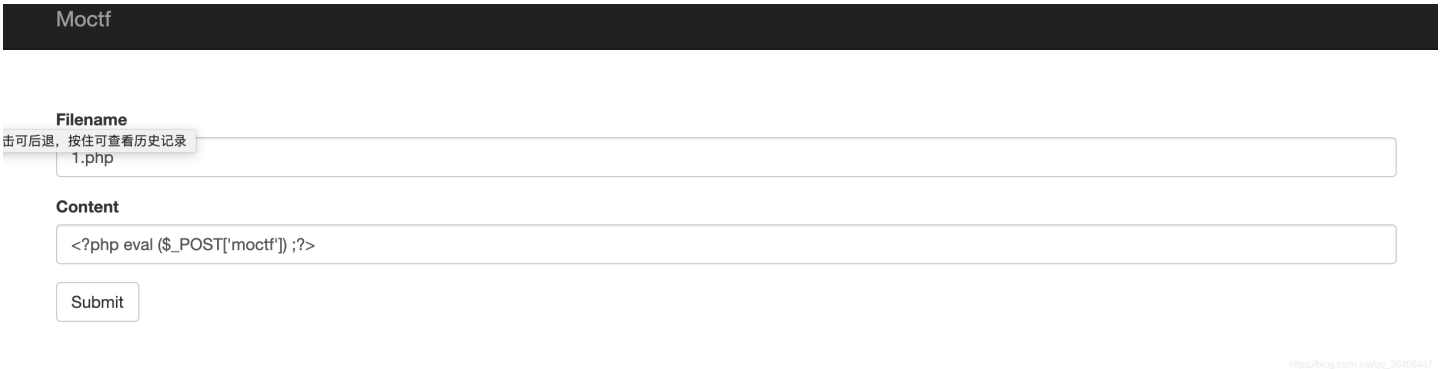
拿到这道题的时候也是很懵逼啊，因为页面和源码都没什么提示，一来就卡住了

后面看了别人的提示说这里是index2，改为index取访问就能得到信息

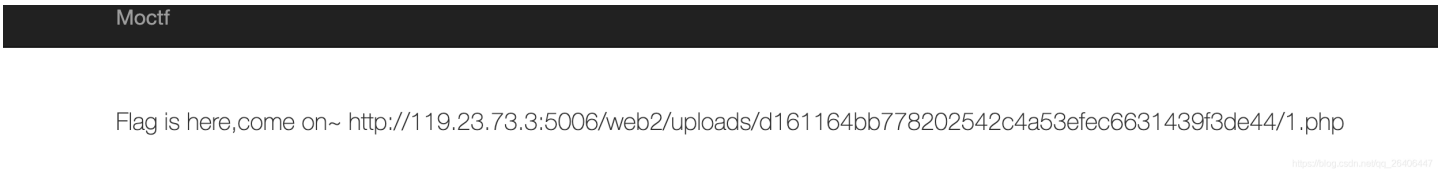
改为index后去访问，发现没有变化，看url发现又变为了index2，说明是发生了302跳转，这时候我们用burp来抓包看下



可以看到burp抓包，抓到了302跳转前的包，看到提示uploadsomething.php，访问



看似是一个登陆界面，但仔细看会发现是个文件上传界面，直接写个一句话木马上传



发现会返回地址，并提示flag就在那里

直接去访问

Too slow!

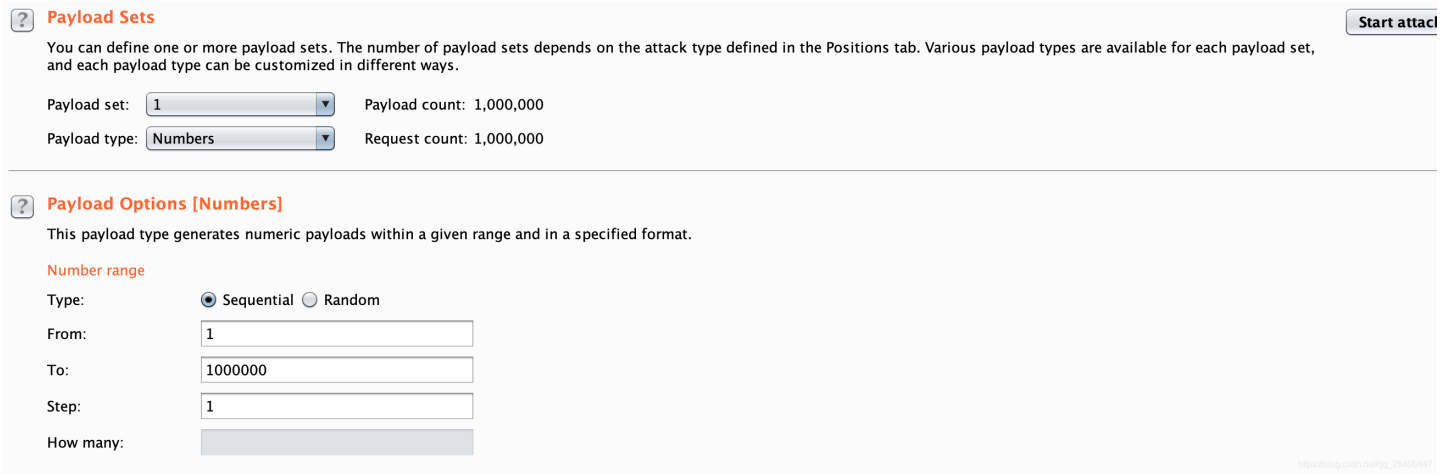
可以看到提示too slow

这里就要用到条件竞争了，我门可以猜测题意，这里应该是我们成功上传了php文件但后端在短时间内将其删除了，所以我们要抢到在它删除之前访问文件，就如我们打开文件的时候去删除它，会提示文件文件已打开一样，这样从而防止文件被删除

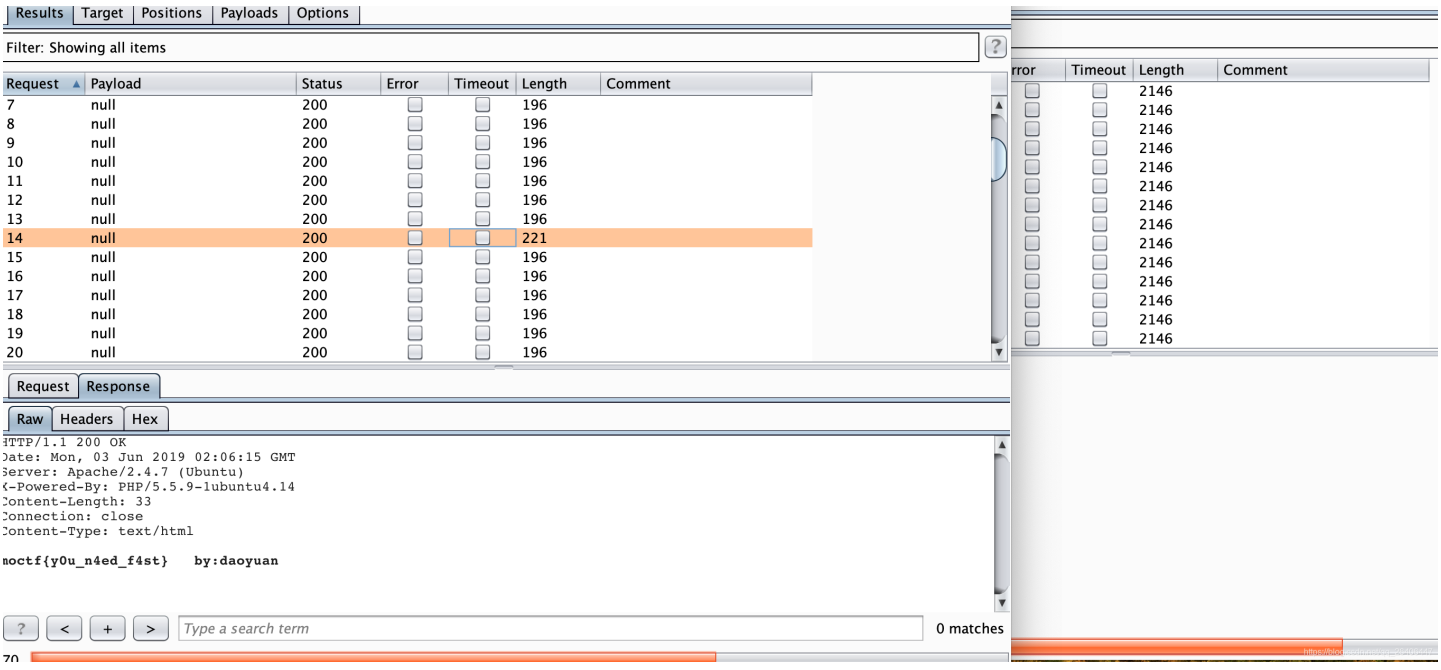
多提交几次发现签名的那串数字是没有变化的，ok这时候我们可以写个脚本，也可以利用burp的爆破模块一直去访问

burp的时候设置爆破模块，选择一个不影响的参数来用，这里我选择的是user-agent

然后payload选择选择数字自动增长，调整时间参数，让其不用又间隔



结果没能成功...还是太慢了？
后面我看了别人的博客，他是开了两个爆破模块，一个是不断上传文件，一个是不断去访问文件（这里我发现别人用的是NUll payload，这个比我的数字应该是要好很多了...）
这样终于成功抓到了包，看到了flag



这道题还是挺有意思的，大部分条件竞争的题是，通过竞争不让文件被删除，然后一句话木马去连接，这里却最后还是被删除了而且这里是需要不断上传和访问相结合，这也是第一次遇到吧...

总结

条件竞争除了上面文件上传的例子还有数据库的漏洞（数据库这个我感觉重视程度一般都会比较高吧，我记得以前看一些开发的视频讲数据库都会涉及到数据安全的问题，讲到原子操作吧，毕竟数据库涉及到一些交易上的问题）

这里也可以参考下面的博客
Web中的条件竞争漏洞
条件竞争

这里对于文件上传的条件竞争我还是有点疑惑的，
我看别的博客给的解决方案是：一定要经过充分完整的检查之后再上传
我不太理解这里的检查之后再上传，毕竟文件检查不在后端检查只靠前端显然是不行的吧，所以这里怎么来防止条件竞争并没有一个好的思路…
然后靶场中条件竞争的题目有些少感觉也没有充分的完成一个锻炼，后面遇到了再补充吧

参考

1. Race Conditions/条件竞争
2. WEB-CTF 条件竞争
3. Moctf–没时间解释了
4. Web中的条件竞争漏洞
5. 条件竞争