第一届合天杯河北科技大学网络安全技术大赛 web6 writeup

转载

weixin_30332241 于 2018-12-10 09:48:00 发布 63 收藏
原文链接：http://www.cnblogs.com/EEEE1/p/10094883.html

预备知识

竞争条件漏洞

“竞争条件”发生在多个线程同时访问同一个共享代码、变量、文件等没有进行锁操作或者同步操作的场景中。

实验目的

通过该实验了解php的竞争条件漏洞。

实验环境

服务器：Windows 10，IP地址：随机分配

辅助工具：burp，phpstudy （网站根目录C:\phpStudy\WWW\）

脚本下载地址：http://tools.hetianlab.com/tools/T006.zip

实验步骤与内容

本次实验的上传处理代码如下：

upload_file.php

```php
<?php

$filename = $_FILES['file']['name'];
$ext = substr($filename, strrpos($filename, '.') + 1);

$path = 'uploads/' . $filename;
$tmp = $_FILES['file']['tmp_name'];
if(move_uploaded_file($tmp, $path)){
    if(!preg_match('/php/i', $ext)){
        echo 'upload success, file in ' . $path;
    } else {
        unlink($path);
        die("can't upload php file!");
    }
} else {
    die('upload error!');
}

```

很简单的一个上传代码，从代码中可以看到，会先把上传文件保存到上传目录，然后会判断上传文件的后缀是否包含php，如果包含，则会把上传上去的文件删除。

看起来似乎没有问题，但是这里实际上存在一个竞争条件漏洞。

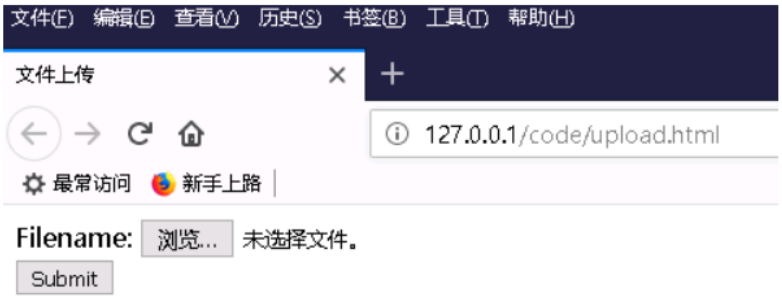
虽然php代码在执行的时候是线性执行代码的，但是执行的时候可以有多个线程。比如a线程执行a.php的代码，b线程执行b.php的代码，当然也可以2个线程都在执行同一个文件。像上图中的代码，如果我们上传一个php文件，在执行完move_uploaded_file之后，执行unlink之前，此时这个php文件是

果我们上传一个php文件，在执行完move_uploaded_file之后，执行unlink之前，此时这个php文件是已经保存到了web服务器上的，并且我们能够访问。所以只要我们在删除该文件之前，访问上传的文件，就可以获得一次执行php代码的机会。如果上传的php的功能是写一句话到一个php文件，这样我们在删除之前访问该文件，就会生成一个一句话木马，就可以得到webshell。

但是这里有一个难题，就是如何做到在保存文件之后，删除文件之前访问这个文件呢？很明显手工速度跟不上，我们这里考虑用工具或者自己编程解决，用多个线程上传文件，同时用多个线程访问上传后的文件，这样就存在竞争，就有可能在删除之前执行我们上传的文件。本次实验选择burp这个工具。

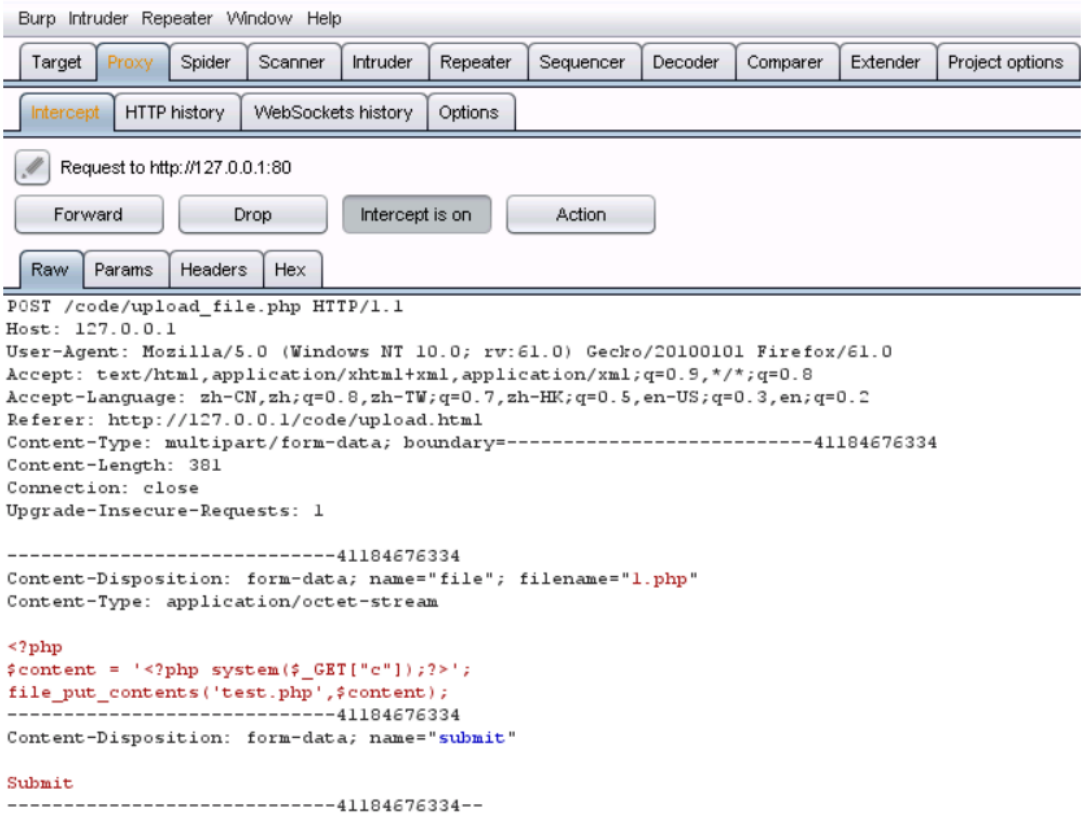
打开burp，配置好代理，开启burp拦截。

然后新建一个php文件上传上去



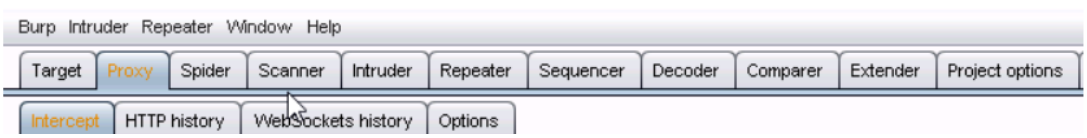
该文件内容如下：

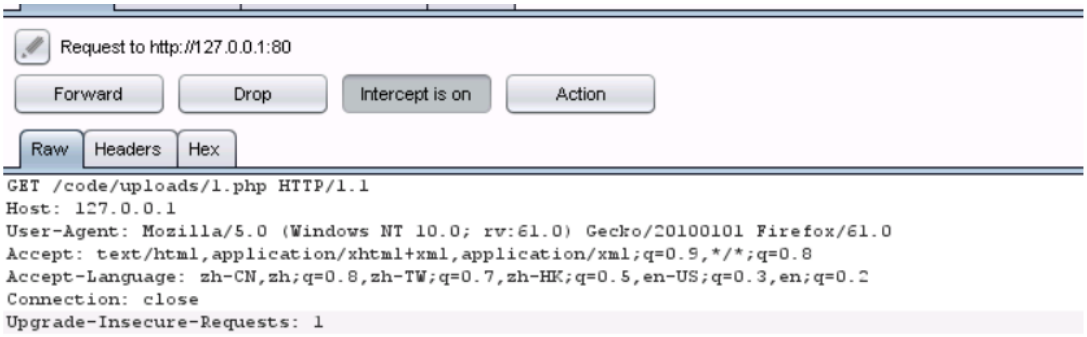
点击提交后，被burp拦截



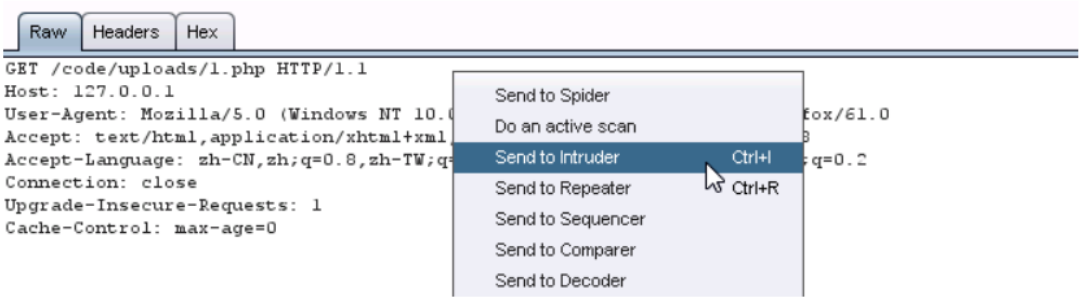
右击选择send to intruder后，放行该数据包。

然后回到浏览器，直接访问http://127.0.0.1/code/uploads/1.php，同样被burp拦截



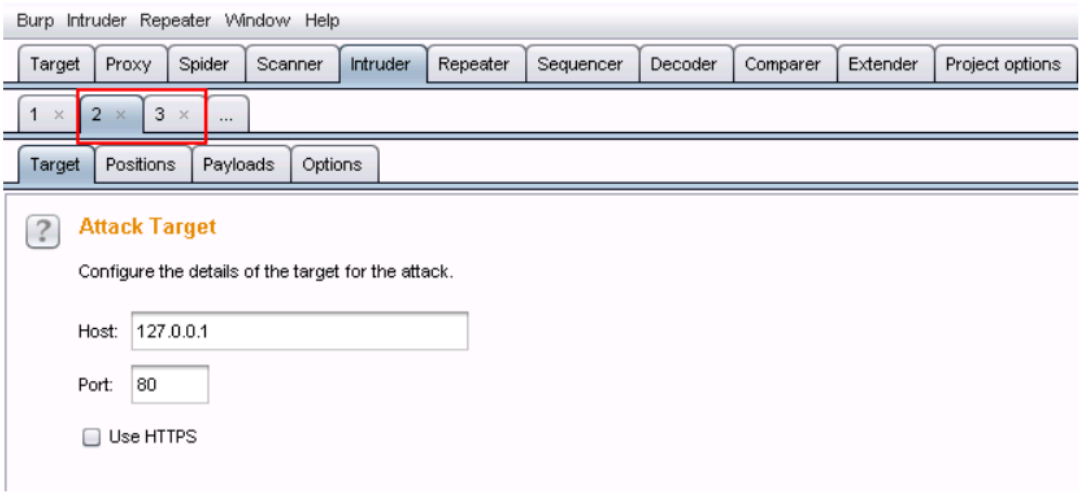


右击选择send to intruder

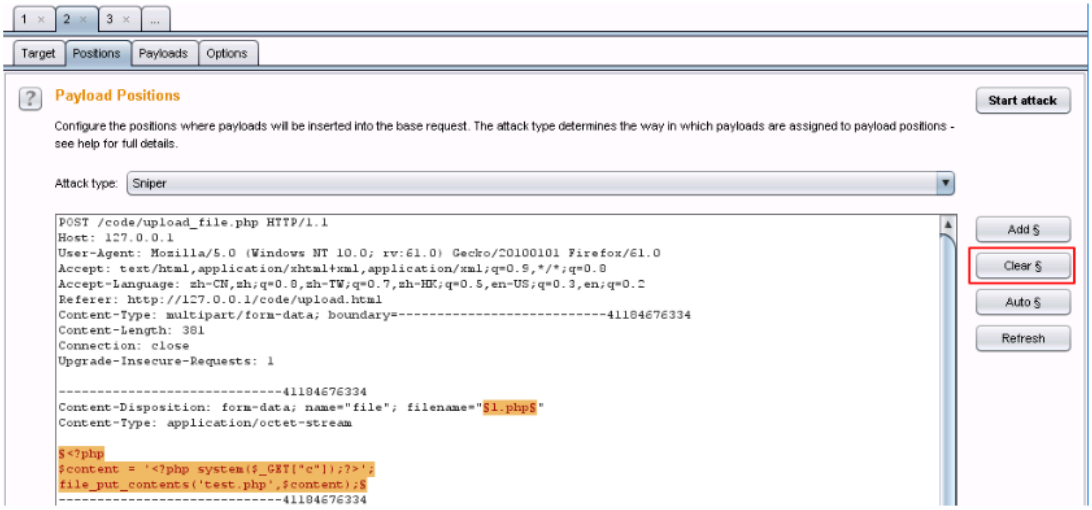


然后放行该数据包。

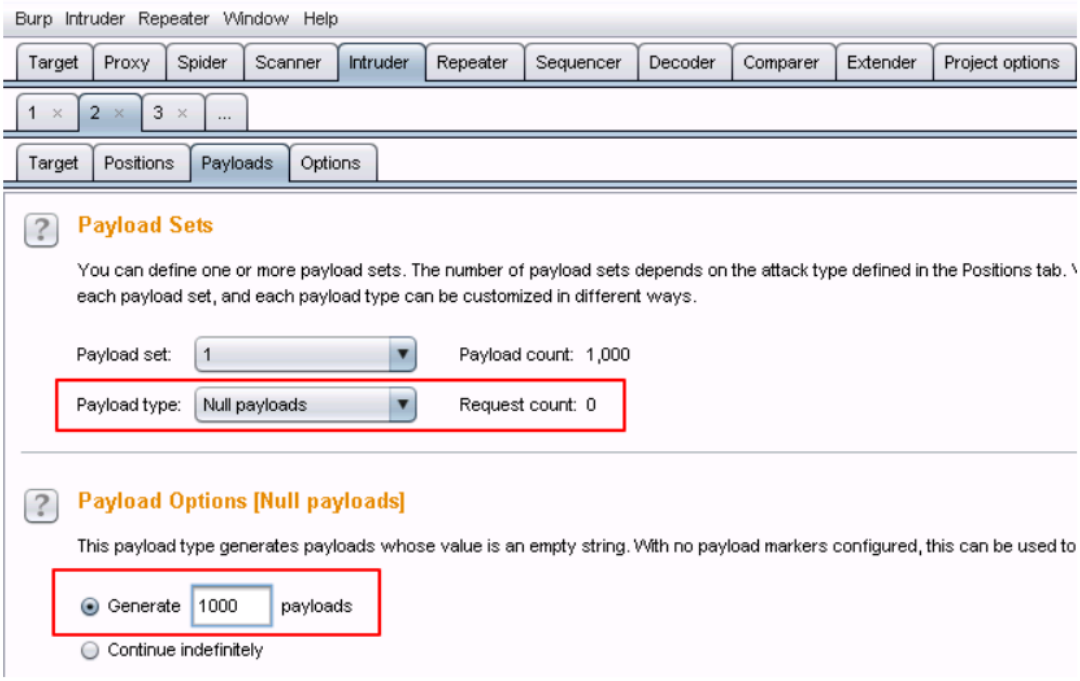
然后在burp工具中，进入Intruder选项卡，找到刚才我们发送过来的2个数据包，我这里的序号为2、3。



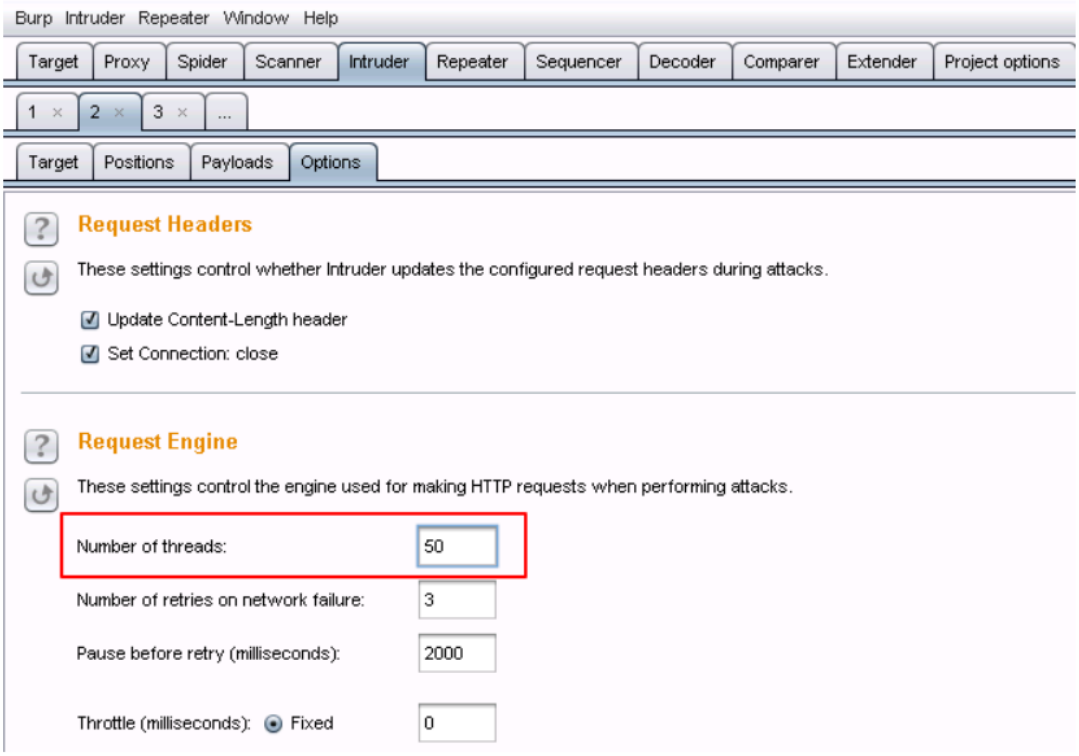
首先选择序号为2的包，进入Positions选项卡，点击右侧的Clear §



进入Payloads选项卡，在Pyaload type 选择Null payloads，并在下方配置Generate 1000 payloads，这里设置为1000表示发1000个请求，如果跑完1000个请求后，还没有生成一句话木马，可以调整这个数值和线程数。



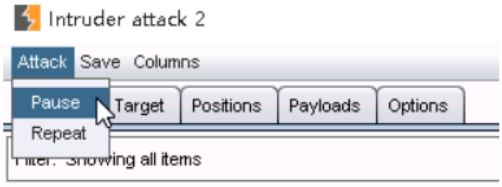
然后在Options选项中，设置线程为50(可以视机器配置调整)，如下图：



然后选择序号为3的包，做上面同样的操作，在Positions选项卡，点击右侧的Clear §，设置Payload type 为Null payloads，配置Generate 1000 payloads，并设置线程为50。

对两个数据包做了如上设置后，先选择序号为2的包，大家做实验的时候选择对应的就行了，然后点击Intruder，选择Start attack。然后选择序号为3的包，执行同样的操作。

当开始后，我们就可以访问test.php，如果不是提示404，说明成功生成了一句话木马，注意这里的test.php要修改成上传文件中写的文件名。



如果此时burp还没跑完，可以停止burp的线程，可以直接关了burp，或者在弹出来的窗口选择Attack - Pause

Notice: Undefined index: c in **C:\phpStudy\WWW\code\uploads\test.php** on line **1**

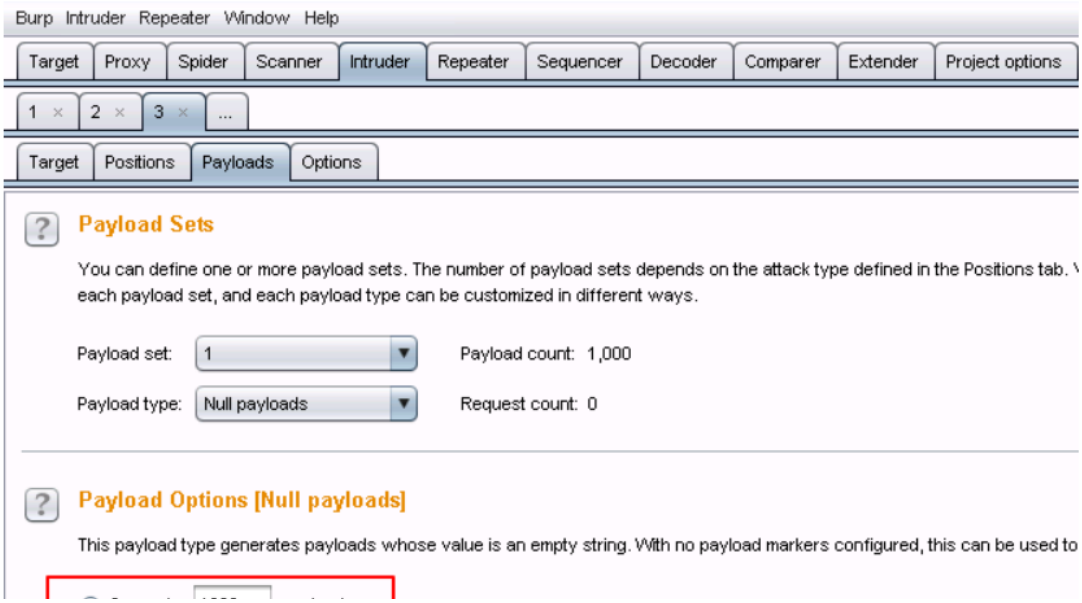
Warning: system(): Cannot execute a blank command in **C:\phpStudy\WWW\code\uploads\test.php** on line **1**

当然也可以等他跑完1000个请求，然后再检查是否生成了一句话木马，当成功生成后就可以构造数据来执行命令了。

http://127.0.0.1/code/uploads/test.php?c=whoami

nt authority\system

如果burp跑完后还没有生成，则可以调整线程数和请求数，这里说的请求数指在Payloads选项中的Generate payloads，如下图：



实验报告要求

参考实验原理与相关介绍，完成实验任务，并对实验结果进行分析，完成思考题目，总结实验的心得体会，并提出实验的改进意见。

分析与思考

如果不用burp，是否能自己编程解决?

配套学习资源

参考资料:

http://www.freebuf.com/articles/network/107077.html

https://www.0dayhack.com/post-666.html

转载于:https://www.cnblogs.com/EEEE1/p/10094883.html