初探 Relative Path Overwrite

[+] Author: math1as
[+] Team: L team

#1 引言

RPO(relative path overwrite)是一类由于浏览器和服务器中间件或web server本身,对用户传入的url本身进行解析时,产生了理解差异而导致的漏洞。

#1.1 背景

RPO漏洞最早由Gareth Heyes发表的文章中的提出,这种漏洞本质上
是利用前端源码中加载css/js的路径,来加载其他文件实现XSS等攻击。
而还可以做更多的推广

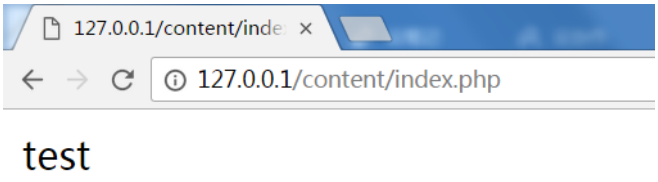
#1.2 测试环境

在本文所描述的漏洞场景下,用到的环境是
apache/nginx + php7-fpm
chrome/firefox 浏览器 最新版

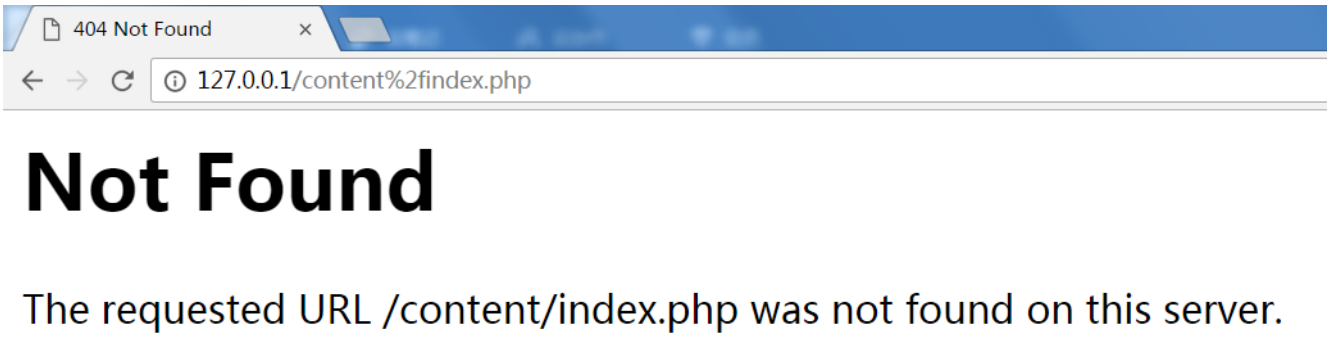
#2 原理剖析

#2.1 web server的url解码功能

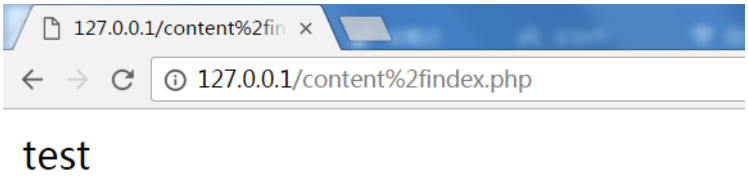
在我们的测试环境,如nginx下,它会自动的把我们提交的url中的一些被编码后的参数解码
而且,还会按照正常的逻辑去进行处理
比如我们在/content/ 目录下有index.php文件
一般的访问方式是这样的



那么,如果我们把目录符号/编码呢?
在apache的环境下
访问127.0.0.1/content%2findex.php
虽然解码了requested url
但是并没有访问到我们的文件



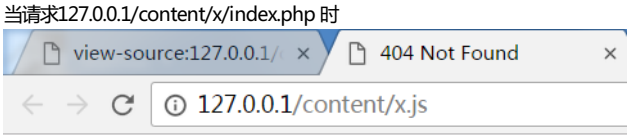
切换到nginx
可以发现,成功的解码了url,并且访问到了指定的文件



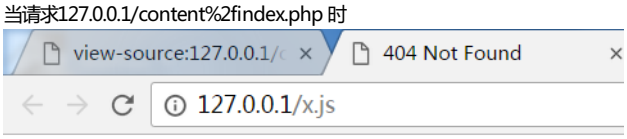
也就是说,nginx等web server是接受这种请求方式的。

#2.2 浏览器加载文件的相对路径

在编写前端页面时,程序员往往需要考虑到多层目录的问题,有时候为了方便
就会去加载类似../xxx.js 或者../xxx.css这样的静态资源
而这个../是相对于浏览器的当前路径的,浏览器是怎么识别当前路径的呢?
我们再增加一个子目录x
里面有一个内容为<script src='../x.js' ></script>的php文件
我们来测试一下2.1种两种不同的请求方式所分别加载的文件
当请求127.0.0.1/content/x/index.php 时



当请求127.0.0.1/content%2findex.php 时



也就是说,浏览器的认知中,并没有解码%2f
而是以最后一个 / 作为了前一个目录结束的标志

#2.3 差异性的利用

我们发现,虽然用两个不同的url都能请求到服务器上相同的文件
也就是我们的/content/x/index.php
但是最终这个index.php中所加载的资源文件却是不同的
那么这个资源文件是否可控呢?答案是肯定的
如果我们构造/content/fake/..%2fx/index.php
仍然会加载到这个index.php
但是这时,浏览器的当前目录是/content/fake/..%2fx/
向上跳一层 也就是/content/fake/
最终加载到了/content/fake/ 这个目录下的文件
这也就是rpo这类漏洞的原理所在
利用了服务器和浏览器理解url的差异性

#3 实践利用

#3.1 url加载问题

在一部分使用了url_rewrite的php框架,和一部分python框架以及jsp中
会存在url加载的一个小问题
比如code igniter中会把/index.php/x/some buffer
的some buffer当作传入的第一个参数
最终访问/index.php/x/some buffer
得到了/index.php/x/ 相同的结果
当然,更加广泛的情况是,web server里可以对? 进行编码
这样最后请求的后面内容被当作queryString而忽略
这样的情况下,类似于前者拼接的x.js就不会对我们最终加载的内容产生干扰
而可以包含任意的文件

#3.2 漏洞测试

实际的利用环境如下

/content/xxx/index.php中加载了../x.js

而 /content/yyy/x.php 是一个把用户传入的参数进行htmlentities转义并输出的模块

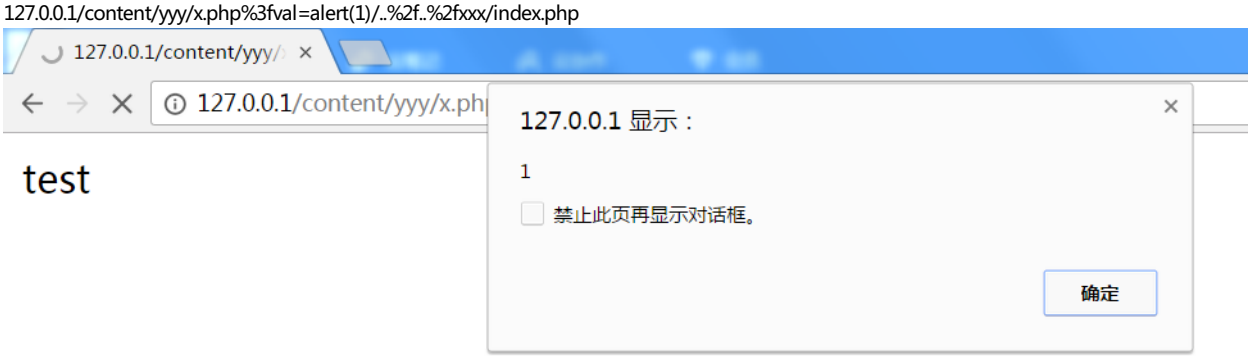
```
<?php
$j=$_GET['val'];
echo htmlentites($j);
?>
```

这样我们一般情况下是没有办法直接对x.php进行xss的,但是通过传入$val=alert(1)

我们获得了一个加载点

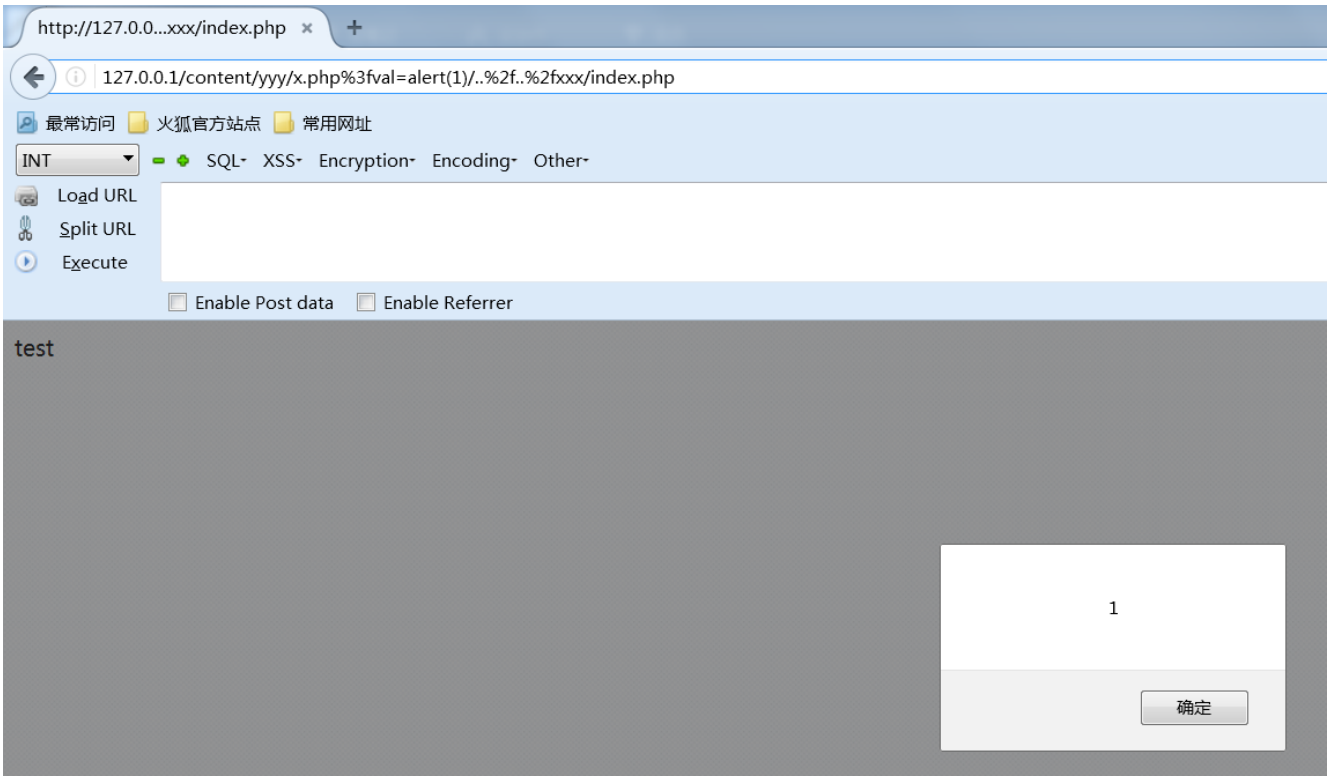
对 127.0.0.1/content/xxx/index.php 进行构造

127.0.0.1/content/yyy/x.php%3fval=alert(1)/..%2f..%2fxxx/index.php



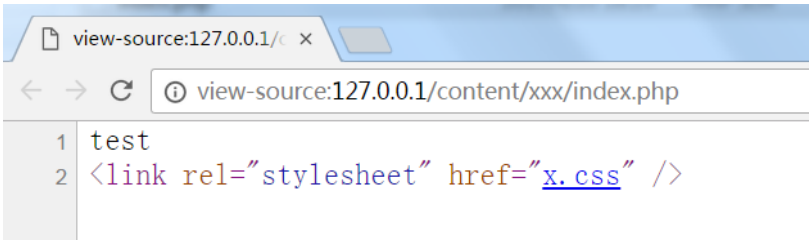
成功的进行了xss

当然firefox浏览器下的情况也允许它进行这样的攻击



#3.3 加载css

我们把场景做一下小的改变



如果现在加载的是一个css的话

针对ie仍然可以做xss攻击

比如传入如下代码

#header {

background:url(javascript:alert('1'));

}

即可得到执行

而也可以通过@import 加载一个远程地址

进而进行信息的泄露或者其他攻击

当然还有更多的通用攻击方式

比如劫持加载某个swf来引发flash-xss

#4 结语

通过我们对Relative Path Overwrite这类漏洞的简单的探讨

我们发现它可以引发一些类似xss的漏洞,对正常的业务产生危害

因此,需要避免直接使用相对路径进行静态文件的加载

#5 参考

[1] http://www.tuicool.com/articles/eIf6Vje