

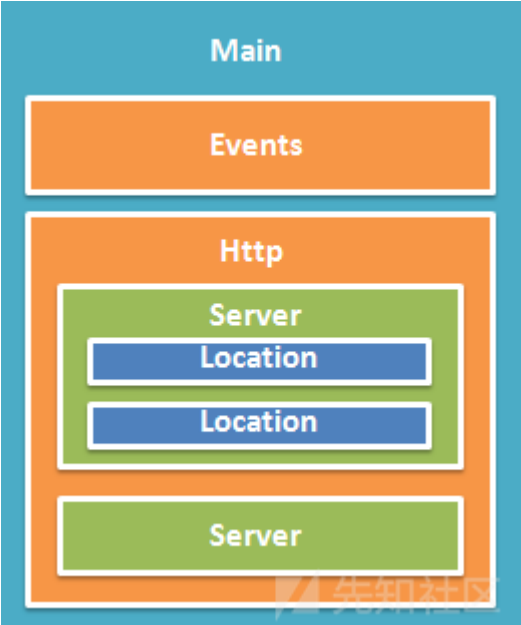
Nginx配置不当可能导致的安全问题

Author: Spark1e

目前很多网站使用了nginx或者tengine（淘宝基于Nginx研发的web服务器）来做反向代理和静态服务器，nginx的配置文件nginx.conf的一些错误配置可能引发一些安全问题

0x00 Nginx的配置文件的格式

Nginx的主配置文件非常简短，是由一些模块构成的。在任何情况下Nginx都会加载其主配置文件。
一个主配置文件nginx.conf的结构如下：



```
...          #■■■■  -->main

events {      #events■
    ...
}

http         #http■
{
    ...      #http■■■
    server   #server■
    {
        ...   #server■■■
        location [PATTERN] #location■
        {
            ...
        }
        location [PATTERN] #■■■■location■
        {
            ...
        }
    }
}
```

Nginx是分层级组织的，每个层级可以有指令，并且子块会继承父块的配置，但是如果子块配置了与父块不同的指令，则会覆盖掉父块的配置。指令的格式是：
■■■■ ■■■1 ■■■2 ■■■3；

也可以在配置文件中包含其他配置文件。如include /etc/nginx/mime.types;就包含了各种支持的Content-type.

一个server块表示一个host，可以在server块中添加或者更改nginx服务监听的端口、存放网页文件的位置、以及虚拟主机配置（开反向代理）

一个location块代表一个路由映射规则

0x01 反向代理配置不当导致的ssrf漏洞

Nginx经常拿来做反向代理服务器。反向代理服务器其实就是一台负责转发的代理服务器，实现了转发的作用，然后从真正的服务器获取数据并转发给客户端。比如，我们让nginx监听一个端口（假设我们监听了80端口），然后通过配置反向代理转发给另一个应用端口或者服务器，由它来执行真正的请求。请求处理完成后数据

```
server {
    listen      80;
    server_name 192.168.1.2:8080;

    location / {
        proxy_pass http://192.168.1.2:8080
    }
    #.....
}
```

SSRF漏洞通常出现在不正确的反向代理配置中
如果nginx.conf进行了如下配置

```
location /([a-zA-Z0-9-_%]+) {
    proxy_pass http://$1
}
```

此时url是可控的。如果攻击者修改url, 将其修改成内网IP即可导致SSRF漏洞

0x02 alias导致的目录遍历/目录穿越/部分文件下载漏洞

修改nginx.conf文件,在server块加入autoindex on;可以添加目录浏览功能，但是也会导致安全问题

```
server {
    autoindex on;
    ...
}
```

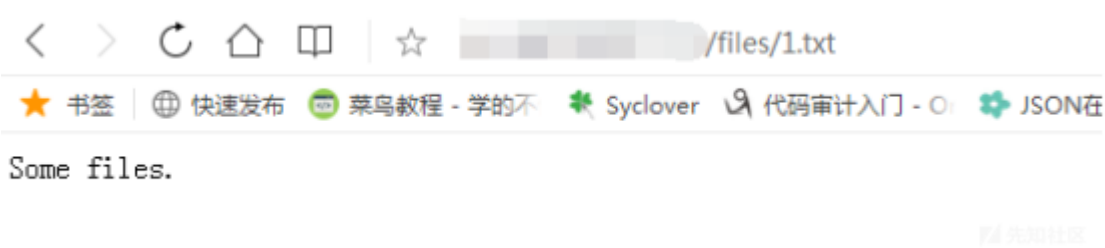
即可达成目录遍历



在nginx做反向代理的时候，我们通常会把动态部分传递给后方解析的服务器，由nginx来处理静态文件
当使用alias来对文件路径进行配置时，有可能会造成目录穿越漏洞
假设配置文件中的配置如下：

```
location /files/ {
    alias      /etc/nginx/txtpath/;
}
```

正常用户访问http://your_ip/files/1.txt时，就可以读取/etc/nginx/txtpath/1.txt这个文件

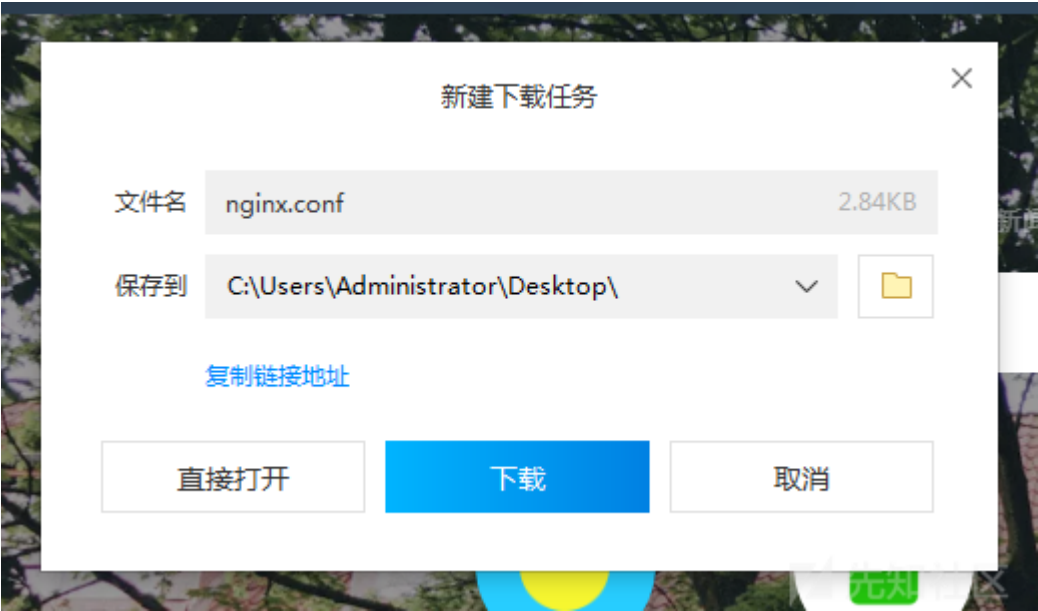


但是如果配置错误，files后面没有/，如下

```
location /files {
    alias      /etc/nginx/txtpath/;
}
```

那么攻击者有可能读到目标文件夹之外的文件。

但是因为在files后面没有/，当我们访问http://your_ip/files../nginx.conf，会返回/etc/nginx/nginx.conf



导致我们可以通过对目录进行爆破扫描等方法，获取到指定文件夹之外的文件

当我们能同时达成以上两个漏洞的条件时，我们就能够读取到部分文件。



Index of /files../

../		
conf.d/	06-Mar-2018 09:30	-
default.d/	06-Mar-2018 09:30	-
txtpath/	16-Apr-2019 00:38	-
fastcgi.conf	06-Mar-2018 09:27	1077
fastcgi.conf.default	06-Mar-2018 09:27	1077
fastcgi_params	06-Mar-2018 09:27	1007
fastcgi_params.default	06-Mar-2018 09:27	1007
koi-utf	06-Mar-2018 09:27	2837
koi-win	06-Mar-2018 09:27	2223
mime.types	06-Mar-2018 09:27	3957
mime.types.default	06-Mar-2018 09:27	3957
nginx.conf	16-Apr-2019 01:10	2923
nginx.conf.default	06-Mar-2018 09:27	2656
scgi_params	06-Mar-2018 09:27	636
scgi_params.default	06-Mar-2018 09:27	636
uwsgi_params	06-Mar-2018 09:27	664
uwsgi_params.default	06-Mar-2018 09:27	664
win-utf	06-Mar-2018 09:27	3610

当alias指定的文件目录足够上层（例如在/home,/usr等）时，我们就可以穿梭到根目录，读取到所有文件。因为配置错误而导致了任意文件读取漏洞

Index of /files../		
../		
bin/	11-Apr-2019 08:02	-
boot/	12-Sep-2017 03:57	-
data/	24-Aug-2017 03:59	-
dev/	11-Apr-2019 12:51	-
etc/	11-Apr-2019 20:51	-
home/	11-Apr-2019 08:09	-
lib/	10-Apr-2019 10:59	-
lib64/	11-Apr-2019 08:02	-
lost+found/	18-Aug-2017 03:51	-
media/	05-Nov-2016 15:38	-
mnt/	05-Nov-2016 15:38	-
opt/	05-Nov-2016 15:38	-
proc/	11-Apr-2019 12:51	-
root/	11-Apr-2019 08:10	-
run/	16-Apr-2019 00:42	-
sbin/	11-Apr-2019 08:02	-
srv/	05-Nov-2016 15:38	-
sys/	11-Apr-2019 20:51	-
tmp/	15-Apr-2019 19:11	-
usr/	16-Apr-2019 00:52	-
var/	11-Apr-2019 20:51	-

0x03 uri导致的CRLF注入漏洞

当一个网站使用https协议的时候，很多站点会强制用户使用https进行访问。当用户访问http的时候会302跳转到https页面。如果使用了 `\$uri` 来进行配置，可能会导致CRLF注入漏洞

```
location /302 {  
    return 302 https://$host$uri;  
}
```

nginx中 `\$uri` 指的是请求的文件和路径，不会包含后面请求的数据（即?和#后面的数据）

nginx服务器会对 `\$uri` 进行解码。当我们在传入的参数后面加入urlencode之后的换行符 `%0d%0a`，我们就可以污染HTTP头的数据

例如，访问 `http://your_ip/302/123` 会302跳转到 `https://your_ip/302/123`。这是正常的跳转。

但是由于配置文件里面使用的是 `\$uri`，会对我们传入的参数进行转码，当我们访问 `http://your_ip/302/123%0d%0a%0d%0atest=1` 时，302跳转会指向 `https://your_ip/302/123%0d%0a%0d%0atest=1`

GoCancel<>Follow redirection

Request

RawHeadersHex

GET /302/123%0d%0a%0d%0atest=1 HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Connection: close
Upgrade-Insecure-Requests: 1

Response

RawHeadersHex

HTTP/1.1 302 Moved Temporarily
Server: nginx/1.12.2
Date: Tue, 16 Apr 2019 14:02:50 GMT
Content-Type: text/html
Content-Length: 161
Connection: close
Location: https:// /302/123

test=1

<html>
<head><title>302 Found</title></head>
<body bgcolor="white">
<center><h1>302 Found</h1></center>
<hr><center>nginx/1.12.2</center>
</body>
</html>

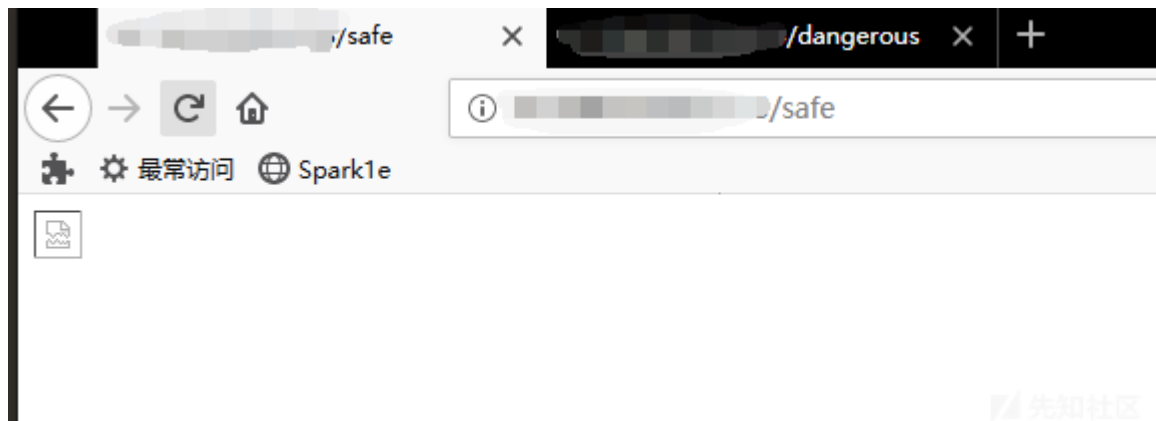
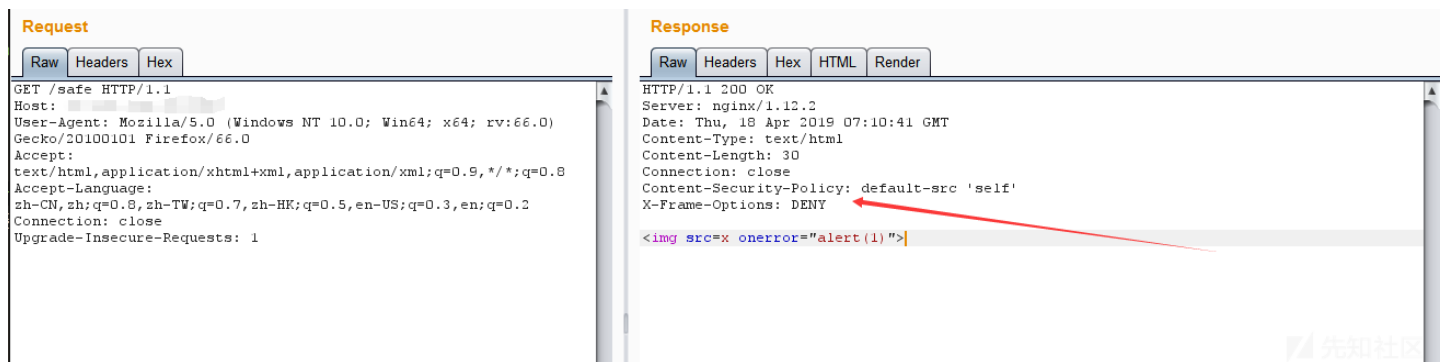
导致了CSRF注入漏洞

0x04 子块覆盖父块HTTP头

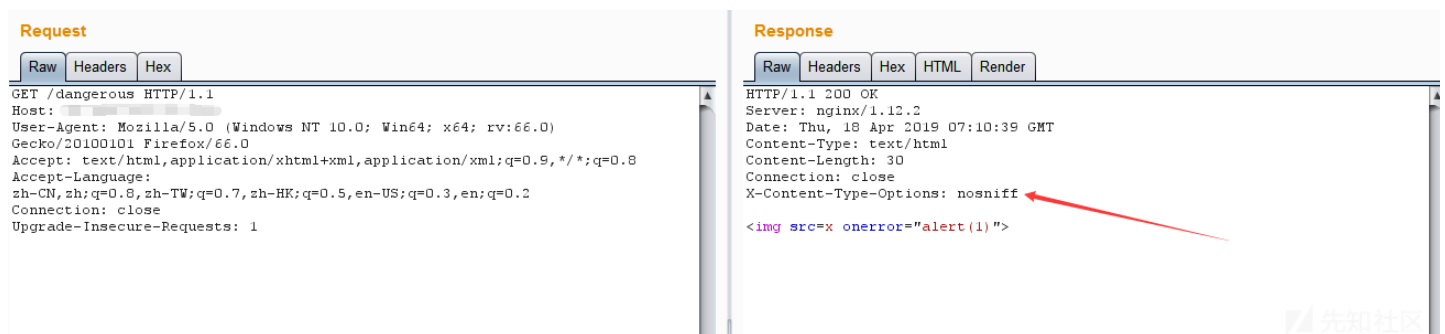
在nginx配置文件中子块是可以继承父块的配置的。但是当我们在父块中设置了add_header头，然后再在子块中设置另一个add_header头时，子块会覆盖掉父块中的add_header配置。假如配置文件是这么设置的

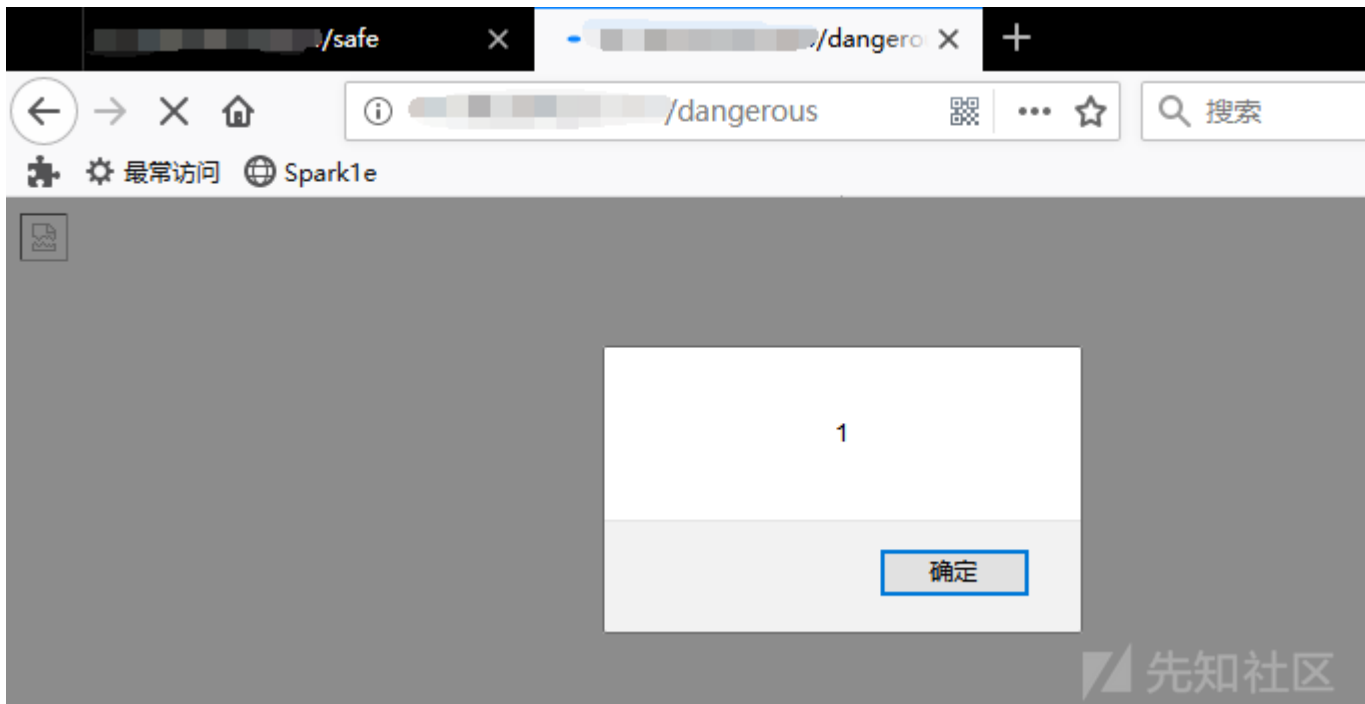
```
server {  
    ...  
    add_header X-Frame-Options DENY;  
    add_header Content-Security-Policy "default-src 'self'";  
  
    location = /safe {  
        return /xss.html;  
    }  
  
    location = /dangerous {  
        add_header X-Content-Type-Options nosniff;  
        return /xss.html;  
    }  
}
```

其中X-Frame-Options DENY和Content-Security-Policy "default-src 'self'"是用来抵御一般的XSS攻击的。当我们访问http://your_ip/safe时，因为我们设置了这两个文件头，所以并不会触发xss。



但是当我们访问http://your_ip/dangerous时，因为我们在子模块添加了add_header X-Content-Type-Options nosniff，父级的模块add_header的被子级的覆盖了，导致对xss的防御不再生效，成功触发xss。





Reference

<http://nginx.org/en/docs/>
<https://www.leavesongs.com/PENETRATION/nginx-insecure-configuration.html>

点击收藏 | 7 关注 | 3
[上一篇：我是如何获得数千家Shopify商...](#) [下一篇：PlaidCTF 2019 CRY...](#)
1. 1 条回复



[blackwolf](#) 2019-04-21 15:01:02

alias那个目录遍历:只能跳到上一层目录，读取上一层目录及其任意子目录的文件，不能任意目录遍历读取任意文件,有局限性。
一种场景是：很多网站会把备份文件放置在网页目录的上一层路径下，利用遍历读取备份文件；另一种场景是：路由限制到上传或静态图片路径，利用遍历读取上一层路
http://blackwolfsec.cc/2018/05/23/Nginx_alias_misconfig_path_traverse/

4 回复Ta

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)