SSRF漏洞总结

1. 简介

SSRF(Server-Side Request Forgery:服务器端请求伪造)是一种由攻击者构造形成由服务端发起请求的一个安全漏洞。一般情况下，SSRF攻击的目标是从外网无法访问的内网。（正因为它是由服务端发起的，所以它能够请求到与它相连而与外网隔离的内网）。

SSRF形成的原因大都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和限制。即SSRF漏洞就是通过篡改获取资源的请求发送给服务器，但是服务器并没有发现这个请求是非法的，然后服务器以他的身份来访问其他服务器的资源。

攻击者可以利用 SSRF 实现的攻击主要有 5 种：

1. 可以对外网、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。
2. 攻击运行在内网或本地的应用程序（比如溢出）。
3. 对内网 WEB 应用进行指纹识别，通过访问默认文件实现。
4. 攻击内外网的 web 应用，主要是使用 GET 参数就可以实现的攻击（比如 Struts2，sqli 等）。
5. 利用 file 协议读取本地文件等。
6. SSRF漏洞出现的场景

* 能够对外发起网络请求的地方，就可能存在 SSRF 漏洞
* 从远程服务器请求资源（Upload from URL，Import & Export RSS Feed）
* 数据库内置功能（Oracle、MongoDB、MSSQL、Postgres、CouchDB）
* Webmail 收取其他邮箱邮件（POP3、IMAP、SMTP）
* 文件处理、编码处理、属性信息处理（ffmpeg、ImageMagic、DOCX、PDF、XML）

1. 漏洞挖掘与判断
2. 从WEB功能上寻找

* 分享：通过URL地址分享网页内容
* 转码服务：通过URL地址吧原地址的网页内容调优使其适合手机屏幕浏览
* 在线翻译：通过URL地址翻译对应文本的内容。
* 图片加载与下载：通过URL地址加载或下载图片
* 未公开的api实现以及其他调用URL的功能

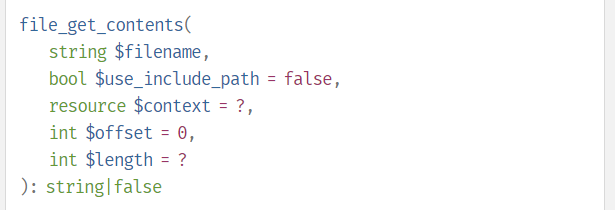
1. 从URL关键字中寻找

* share、wap、url、link、src、source、target、u、3g、display、sourceURL、imageURL、domain。

1. 基本判断

* 回显
* 直接观察
* 抓包
* 用vps监听
* 延时
* DNS请求

1. PHP相关函数
2. file\_get\_contents（PHP 4 >=4.3.0，PHP 5，PHP 7）:将整个文件读入一个字符串。



和file()基本一样，除了file\_get\_contents()把文件读入一个字符串。将在参数offset所指的的位置开始读取长度为maxlen的内容。如果失败，file\_get\_contents()将返回FALSE。

file\_get\_contents()函数是用来将文件的内容读入到一个字符串中的首选方法。如果操作系统支持还会使用内存映射技术来增强性能。

注：如果要打开有特殊字符的URL（比如说有空格），就需要使用urlencode()进行URL编码。

例：



审计代码发现开头必须是 http[s]/ 或者是 file:// ，可以进行SSRF攻击。

首先读取index.php

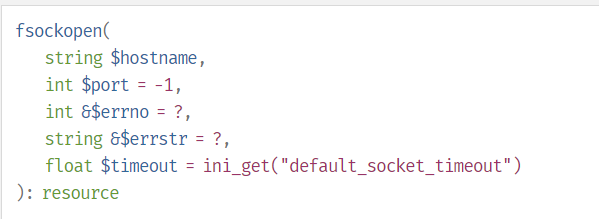
观察到flag在 flag\_chunibyou.php 中，但是index限制读取的文文件不不能包含 \_ ，所以只能找其他方法。

根据提示，数据库文件是可以访问的，从index文件中可以看到数据库结构，因此可以下载ibd文文件：



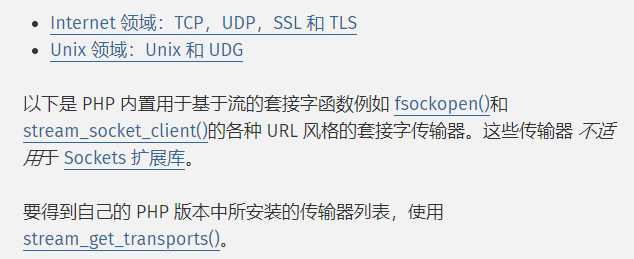
通过 strings user.idb 可以拿到password，进而拿到flag。

1. fsockopen（PHP 4，PHP 5，PHP 7）：打开一个网络连接或者一个Unix套接字连接。



初始化一个套接字连接到指定主机（hostname）。

PHP支持以下的套接字传输器类型列表：



默认情况下将以阻塞模式开启套接字连接。当然你可以通过[stream\_set\_blocking()](https://www.php.net/manual/zh/function.stream-set-blocking.php)将它转换到非阻塞模式。

[stream\_socket\_client()](https://www.php.net/manual/zh/function.stream-socket-client.php)与之非常相似，而且提供了更加丰富的参数设置，包括非阻塞模式和提供上下文的的设置。

fsockopen()将返回一个文件句柄，之后可以被其他文件类函数调用（例如：[fgets()](https://www.php.net/manual/zh/function.fgets.php)，[fgetss()](https://www.php.net/manual/zh/function.fgetss.php)，[fwrite()](https://www.php.net/manual/zh/function.fwrite.php)，[fclose()](https://www.php.net/manual/zh/function.fclose.php)还有[feof()](https://www.php.net/manual/zh/function.feof.php)）。如果调用失败，将返回false。

该函数可以用来实现发送http get请求/post请求等。

如果主机（hostname）不可访问，将会抛出一个警告级别（E\_WARNING）的错误提示。

fsockopen的使用方法：

<https://blog.csdn.net/zjsfdx/article/details/89376176>

1. curl\_exec（php 4 <=4.0.2，PHP 5，PHP 7，PHP 8）：执行cURL会话。



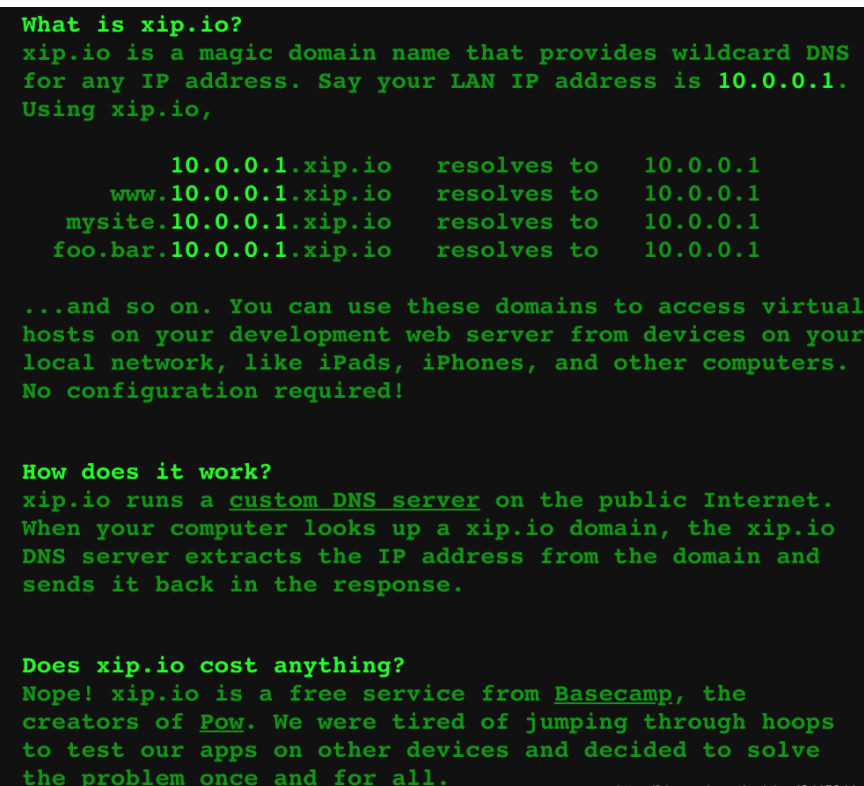
执行给定的 cURL 会话。这个函数应该在初始化一个 cURL 会话并且全部的选项都被设置后被调用。

参数ch：由curl\_init()返回的cURL句柄。

该函数最常用的是用来发送http请求(get请求，post请求等）。除此之外还支持其他应用层协议，比如常用的dict、file、ftp、sftp、gopher、telnet。

1. 绕过方式
2. IP限制绕过：如果ssrf漏洞限定了输入不能为ip地址的话，可以用以下方式绕过正则匹配。

* ip地址之后加端口。例如：192.168.200.9:80
* ip地址前添加用户名。例如：http://root@127.0.0.1
* ip地址之后添加get传参。
* 短网址：使用短网址平台将url转化为短网址（短网址平台实现原理其实非常简单，就是在它的域名下提供一个路由，当你访问这个路由的时候就会向你提供的url发送get请求或者直接跳转到你提供的url）如果curl没有开启跟随跳转选项的话，这种短网址跳转的方式就会得不到回显。
* 使用子域名解析：xip.io。也就是说“ip地址.xip.io” 这个域名会被dns服务器解析为“ip地址”。例如：10.0.0.1.xip.io resolves to 10.0.0.1



* 将ip地址转化为其他进制。例如：目标主机限制了ip地址不能为127.0.0.1,那么就可以将127.0.0.1 -> 8进制 0177.0.0.1->16进制 0x7f.0.0.1

因为常见的ip地址为了人类友好都是点分十进制的表示方法，但是本质上在计算机中ip地址实际存储形式不过是用4个字节的内存去存4个8位的二进制，也就是4个两位的16进制。127.0.0.1 ->0x7f000001浏览器访问http://0x7f000001仍然是可行的。将http://0x7f000001 转化为十进制http://2130706433也是可行的。

也就是说点分十进制的表示方式 和十进制的表示都可行的。点分十进制是将32位的ip地址 8位8位的转化为10进制，所以需要点来分割。十进制是将32位的ip地址32位全部转化为10进制，所以不需要点来分割。

1. gopher协议

gopher是一个互联网上使用过的分布型的文件搜索获取网络协议。

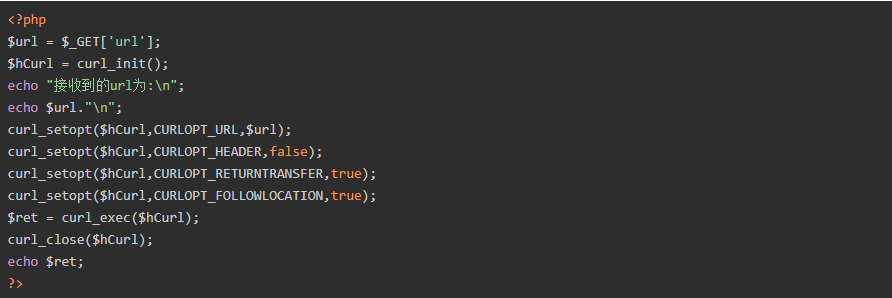
gopher协议的格式：gopher://host:port/\_+数据流

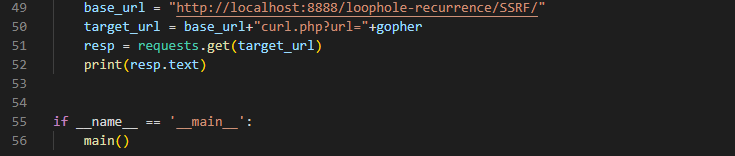
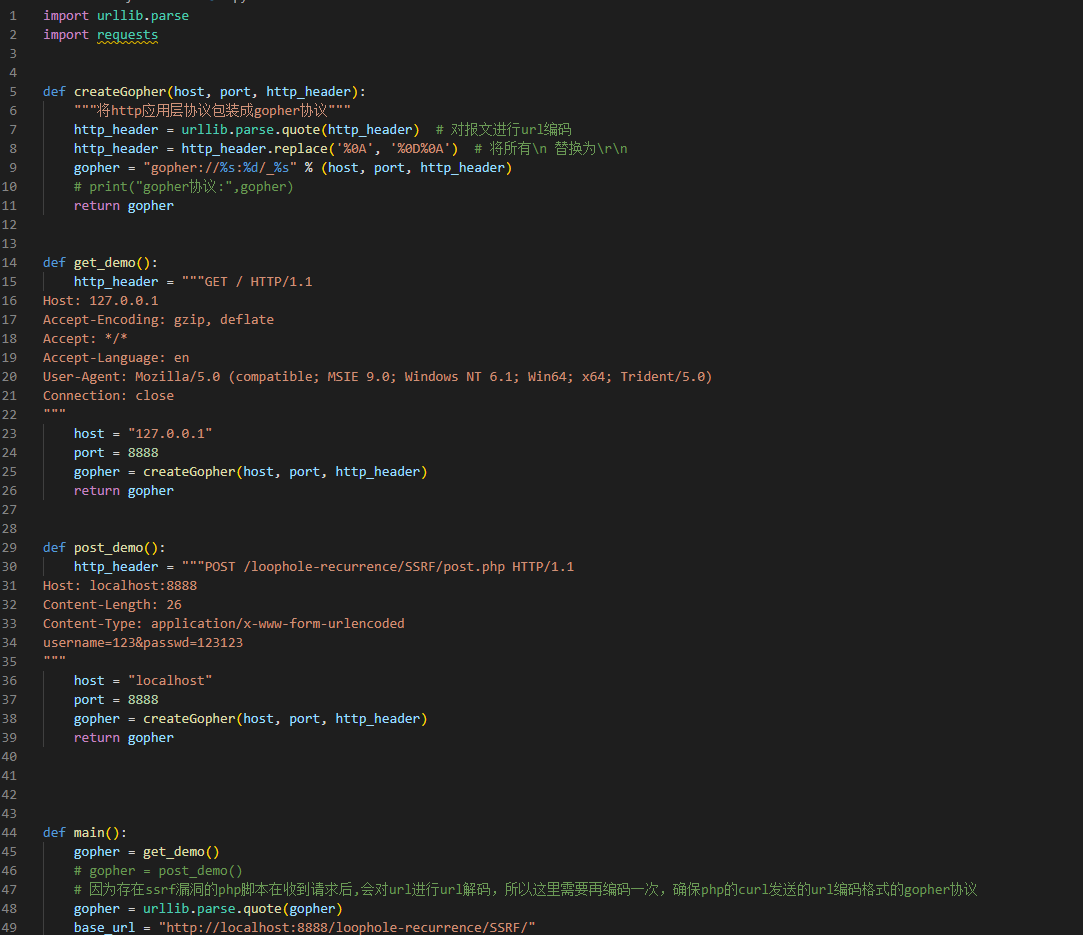
* 如果不指定端口，默认端口为70端口
* 数据流要求全部进行url编码，并且以\r\n换行也就说以%0D%0A换行

gopher协议支持发出GET、POST请求：可以先截获get请求包和post请求包，再构成符合gopher协议的请求，POST包中要注意Content-length，注意url编码的次数。gopher协议是ssrf利用中最强大的协议。

转化思路：将get请求报文/post请求报文先进行url编码，然后将%0A 全部替换为%0D%0A即可。

例如：该页面存在一个ssrf漏洞

利用脚本：



补充：gopher对redis的利用以及gopher对mysql的利用

https://blog.csdn.net/weixin\_43415644/article/details/104115316?utm\_medium=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-3.control&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-3.control

#### 3.DNS重绑定（**DNS Rebinding**）

在网页浏览过程中，用户在地址栏中输入包含域名的网址。浏览器通过DNS服务器将域名解析为IP地址，然后向对应的IP地址请求资源，最后展现给用户。而对于域名所有者，他可以设置域名所对应的IP地址。当用户第一次访问，解析域名获取一个IP地址；然后，域名持有者修改对应的IP地址；用户再次请求该域名，就会获取一个新的IP地址。对于浏览器来说，整个过程访问的都是同一域名，所以认为是安全的。这就造成了DNS Rebinding攻击。

具体步骤：

* 攻击者控制恶意的DNS服务器来回复域的查询,如rebind.network
* 攻击者通过一些方式诱导受害者加载http://rebind.network
* 恶意DNS服务器收到受害者的请求,并使用真实IP地址进行响应,并将TTL值设置为1秒,让受害者的机器缓存很快失效。
* 从http://rebind.network加载的网页包含恶意的js代码,构造恶意的请求到http://rebind.network/index,而受害者的浏览器便在执行恶意请求。
* 一开始的恶意请求当然是发到了攻击者的服务器上,但是随着TTL时间结束,攻击者就可以让http://rebind.network绑定到别的IP,如果能捕获受害者的一些放在内网的应用IP地址,就可以针对这个内网应用构造出对应的恶意请求,然后浏览器执行的恶意请求就发送到了内网应用,达到了攻击的效果。

例题：2020网鼎杯玄武组-Web-SSRFME

参考wp：<https://www.cnblogs.com/20175211lyz/p/13415739.html>

<https://blog.csdn.net/weixin_40422959/article/details/106463823?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_title~default-0.control&spm=1001.2101.3001.4242>