kali_wifihack-抓包平台

参考：

doc_201119_移动安全-APP逆向-针对常见反抓包手段对抗.md： 非常骚的操作思路，动用安卓的iptables

doc_201119_kali搭建钓鱼WiFi_hostapd_dnsmasg.md

doc_201119_Kali_linux搭建钓鱼wifi_airbase-ng.md

doc_201119_kali使用Fluxion钓鱼WiFi.md

共享wifi热点-实现基础的无线联网需求

以下参考： doc_201119_kali搭建钓鱼WiFi_hostapd_dnsmasg.md

1. 安装必要工具包：

```
root@mykali:~# apt install hostapd dnsmasq apache2 aircrack-ng tcpdump
burpsuite ssh chromium firefox-esr
```

用service sshd start开启SSH服务时提示：

```
Failed to start sshd.service: Unit sshd.service not found.
```

解决方案：

添加ssh.service

```
systemctl enable ssh.service
```

2. 查看网卡：

```
root@mykali:~# ip addr
这个tplink的小的，用来做出口
3: wlx30b49e2da741: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq
state UP group default qlen 1000
    link/ether 30:b4:9e:2d:a7:41 brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.156/24 brd 192.168.20.255 scope global dynamic
wlx30b49e2da741
       valid_lft 80127sec preferred_lft 80127sec
    inet6 fe80::32b4:9eff:fe2d:a741/64 scope link
       valid_lft forever preferred_lft forever
这个小雷达，用来做热点
5: wlx00e05c30a11f: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
group default qlen 1000
    link/ether 00:e0:5c:30:a1:1f brd ff:ff:ff:ff:ff:ff
```

3. 启动雷达的监听模式：

```
root@mykali:~# airmon-ng start wlx00e05c30a11f
```

使用双wifi方案，可能会存在问题：

```
Found 3 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

    PID Name
    476 wpa_supplicant
    976 wpa_supplicant
    978 dhclient
```

但是应该还是开了：

```
PHY     Interface       Driver          Chipset

phy0    wlx00e05c30a11f rt2800usb       Ralink Technology, Corp.
RT2870/RT3070
Interface wlx00e05c30a11fmon is too long for linux so it will be renamed to
the old style (wlan#) name.

                (mac80211 monitor mode vif enabled on [phy0]wlan0mon
                (mac80211 station mode vif disabled for
[phy0]wlx00e05c30a11f)
null    wlx30b49e2da741 r8188eu         Realtek Semiconductor Corp.
RTL8188EUS 802.11n Wireless Network Adapter
```

验证确实还是开启了：

```
root@mykali:~# ip addr
6: wlan0mon: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group
default qlen 1000
    link/ieee802.11/radiotap 00:e0:5c:30:a1:1f brd ff:ff:ff:ff:ff:ff
```

4. 配置hostapd：

直接新建一个hostapd.conf就可以：

```
root@mykali:~/wifi_phish# pwd
/root/wifi_phish

root@mykali:~/wifi_phish# cat hostapd.conf
interface=wlan0mon
driver=nl80211
# driver=wext
ssid=yuntian00
#无线名称 随意即可
hw_mode=g
channel=6
macaddr_acl=0
ignore_broadcast_ssid=0
```

插曲：之前错误配置：(wext驱动invalid； ssid 的注释不管用！)

```
root@mykali:~/wifi_phish# cat hostapd.conf
interface=wlan0mon
#driver=nl80211
driver=wext
ssid=yuntian00 #无线名称 随意即可
hw_mode=g
channel=6
macaddr_acl=0
ignore_broadcast_ssid=0
```

5. 配置dnsmasq文件：

```
root@mykali:~/wifi_phish# vim dnsmasq.conf
interface=wlan0mon
dhcp-range=192.168.100.2, 192.168.100.30, 255.255.255.0, 12h
dhcp-option=3, 192.168.100.1
dhcp-option=6, 192.168.100.1
#server=8.8.8.8
server=114.114.114.114
log-queries
log-dhcp
listen-address=127.0.0.1
```

6. 配置防火墙和端口转发:

先看看原始iptables配置：注意，有docker的东西在.

```
root@mykali:~/wifi_phish# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination
DOCKER-USER  all  --  anywhere             anywhere
DOCKER-ISOLATION-STAGE-1  all  --  anywhere             anywhere

ACCEPT     all  --  anywhere             anywhere             ctstate
RELATED,ESTABLISHED
DOCKER     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain DOCKER (1 references)
target     prot opt source               destination

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
target     prot opt source               destination
DOCKER-ISOLATION-STAGE-2  all  --  anywhere             anywhere

RETURN     all  --  anywhere             anywhere
```

```
Chain DOCKER-ISOLATION-STAGE-2 (1 references)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain DOCKER-USER (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
```

配置路由转发和iptables：

```
root@mykali:~/wifi_phish# cat config_iptables.sh
#!/bin/bash

#config route enable
echo 1 > /proc/sys/net/ipv4/ip_forward
#config nat
#iptables --table nat --append POSTROUTING --out-interface eth0 -j
MASQUERADE
iptables --table nat --append POSTROUTING --out-interface  wlx30b49e2da741
-j MASQUERADE
iptables --append FORWARD --in-interface wlan0mon -j ACCEPT
#实现源nat共享ip，需要改iptables的"nat表的postrouting链"和"filter表的forward链"
```

注意： iptables有表和链两重概念！每个表若干链！

查看iptables规则：

```
root@mykali:~/wifi_phish# iptables -L   # 这个只显示 input output forword
查看nat表，要单独加表名
root@mykali:~/wifi_phish# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  anywhere             anywhere             ADDRTYPE
match dst-type LOCAL

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  172.17.0.0/16        anywhere
MASQUERADE  all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  anywhere            !127.0.0.0/8          ADDRTYPE
match dst-type LOCAL

Chain DOCKER (2 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
```

7. 配置雷达接口地址：

```
root@mykali:~/wifi_phish# ip link set wlan0mon up
root@mykali:~/wifi_phish# ip addr add 192.168.100.1/24 dev wlan0mon
```

查看下：

```
root@mykali:~/wifi_phish# ip addr
6: wlan0mon: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state
UNKNOWN group default qlen 1000
    link/ieee802.11/radiotap 00:e0:5c:30:a1:1f brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.1/24 scope global wlan0mon
       valid_lft forever preferred_lft forever
```

配置一个到内网路由：(出口早有dhcp的网关，这个是回程路由)

```
root@mykali:~/wifi_phish# route add -net 192.168.100.0 netmask
255.255.255.0 gw 192.168.100.1
```

查看路由表：

```
root@mykali:~/wifi_phish# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt
Iface
default         192.168.20.1    0.0.0.0         UG        0 0          0
wlx30b49e2da741
172.17.0.0      0.0.0.0         255.255.0.0     U         0 0          0
docker0
192.168.20.0    0.0.0.0         255.255.255.0   U         0 0          0
wlx30b49e2da741
192.168.100.0   192.168.100.1   255.255.255.0   UG        0 0          0
wlan0mon
192.168.100.0   0.0.0.0         255.255.255.0   U         0 0          0
wlan0mon
```

8. 启动钓鱼热点：

   启动热点：

```
root@mykali:~/wifi_phish# hostapd hostapd.conf &
[1] 4341
root@mykali:~/wifi_phish# Configuration file: hostapd.conf
Using interface wlan0mon with hwaddr 00:e0:5c:30:a1:1f and ssid "yuntian00"
wlan0mon: interface state UNINITIALIZED->ENABLED
wlan0mon: AP-ENABLED
```

   启动dns服务：

```
root@mykali:~/wifi_phish# dnsmasq -C dnsmasq.conf -d &
[2] 4357
root@mykali:~/wifi_phish# dnsmasq: started, version 2.82 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP
DHCPv6 no-Lua TFTP conntrack ipset auth DNSSEC loop-detect inotify dumpfile
dnsmasq-dhcp: DHCP, IP range 192.168.100.2 -- 192.168.100.30, lease time
12h
dnsmasq: using nameserver 114.114.114.114#53
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 114.114.114.114#53
dnsmasq: using nameserver 211.137.191.26#53
dnsmasq: read /etc/hosts - 5 addresses
```

9. 故障排查：

   ssid: yuntian00

   wifi密码：无

   使用手机连接wifi，可以正常获取dhcp地址分配，ip：192.168.100.9

   kali可以ping手机

   手机可以正常访问，kali起python3 http server：`http://192.168.100.1`！

   以上说明，整个wifi的内网是可以的，没问题！

   手机也可以访问，`http://192.168.20.156`kali的pyton server！

   说明，linux的ip route 没问题

   手机访问arch的python server：`http://192.168.20.100`，用arch 抓包调试下！

   不能访问！

   关闭docker的iptables规则

```
systemctl disable docker
reboot
```

   重启之后，iptable就清爽多了

```
root@mykali:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

   重新启动热点：

```
root@mykali:~/wifi_phish# pwd
/root/wifi_phish

以下可以直接打包，做启动脚本。
cd /root/wifi_phish
airmon-ng start wlx00e05c30a11f
ip link set wlan0mon up
ip addr add 192.168.100.1/24 dev wlan0mon
route add -net 192.168.100.0 netmask 255.255.255.0 gw 192.168.100.1

hostapd hostapd.conf &
dnsmasq -C dnsmasq.conf -d &
bash config_iptables.sh
```

iptables规则生效后：（没docker清爽多了）

iptables -L 只显示3个

```
root@mykali:~/wifi_phish# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

iptables -t nat -L 显示5个：

```
root@mykali:~/wifi_phish# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

iptables -t nat -L 如上显示，手机连接wifi可以顺利访问百度啦！

最后确认，故障的原因就是docker自动启动的iptables有冲突！！

PS：网上还有一种iptables的nat规则做法：

```
利用iptable的路由点Hook功能，将at0(伪AP)上的流量和网口(eth0)做一个NAT，将流量牵引出去
iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o eth0 -j SNAT -to-source
192.168.159.254
iptables -t nat -A PREROUTING -d 192.168.159.254 -i eth0 -j DNAT -to
10.0.0.100
```

以上，参考：doc_201119_Kali_linux搭建钓鱼wifi_airbase-ng.md

实现burpsuite的透明代理http和https

最近在搞汽车ivi设备的渗透测试，发现有些车辆ivi设备的网络设置没有配置代理服务器的地方（基本手机都有），同样针对https导入证书，同样厂家也是把安卓"设置"中基本的证书导入功能阉割了。于是有个想法，那笔记本做一个wifi网关，然后能不能实现透明http代理进行抓包。这样笔记本、汽车、手机等连接wifi的移动设备，不用设置代理就可以进行wedb抓包的分析。岂不是很爽，参考几个大佬文章后，感觉可行度比较高。

另外，为方便外出抓包，经常抓车，放不到办公室。所以外网口和内网口都用无线网卡，用起来方便啊。

先看看之前大佬的智慧：

- https://blog.bbskali.cn/2179.html：主要学习实现wifi网关
- https://blog.bbskali.cn/2179.html：绝对大佬，后续按照hook思路，继续深入解析数据包。安卓端数据包重定向，burp透明代理功能
- https://blog.csdn.net/hulifox007/article/details/3706729：可行性论证，但没方法，我也不熟悉iptables
- https://blog.csdn.net/weixin_33963189/article/details/92352944：大佬直接干货，没废话

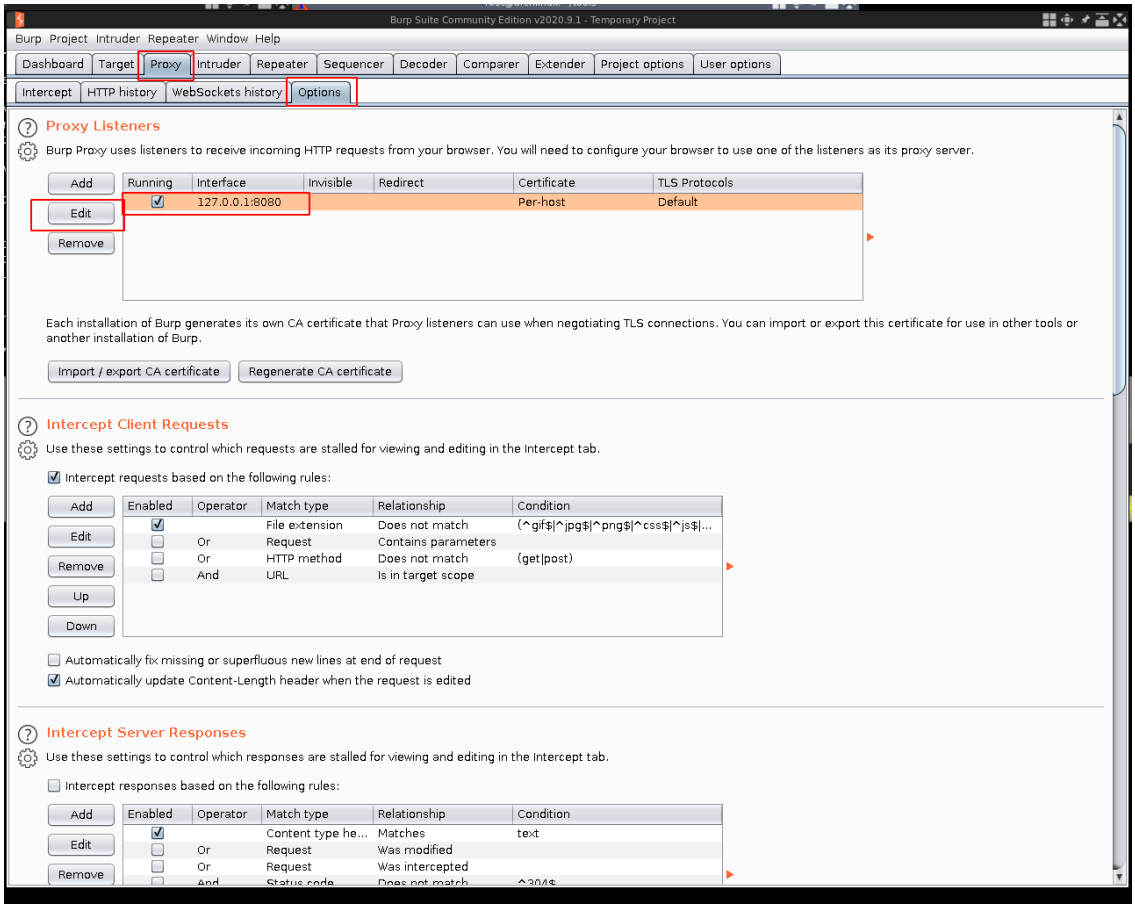
总结下：

- 实现wifi共享，做nat共享上网
- 了解透明代理，恰好burp有透明代理姿势
- iptables神器和网络知识，关于路由表和数据包修改（nat，redirect）的先后，数据包路由原理
- 安卓https证书adb导入（阉割功能的）

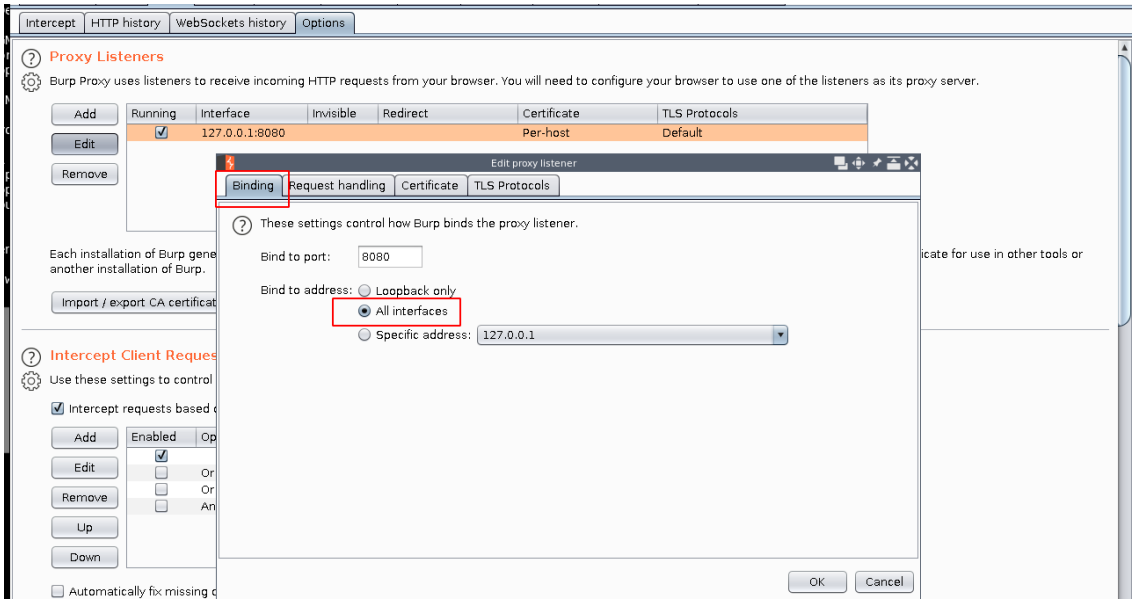
共享wifi参考说的很明白，很好不在多说了。

1. burpsuite的透明代理配置：（大佬原文也讲很清晰）

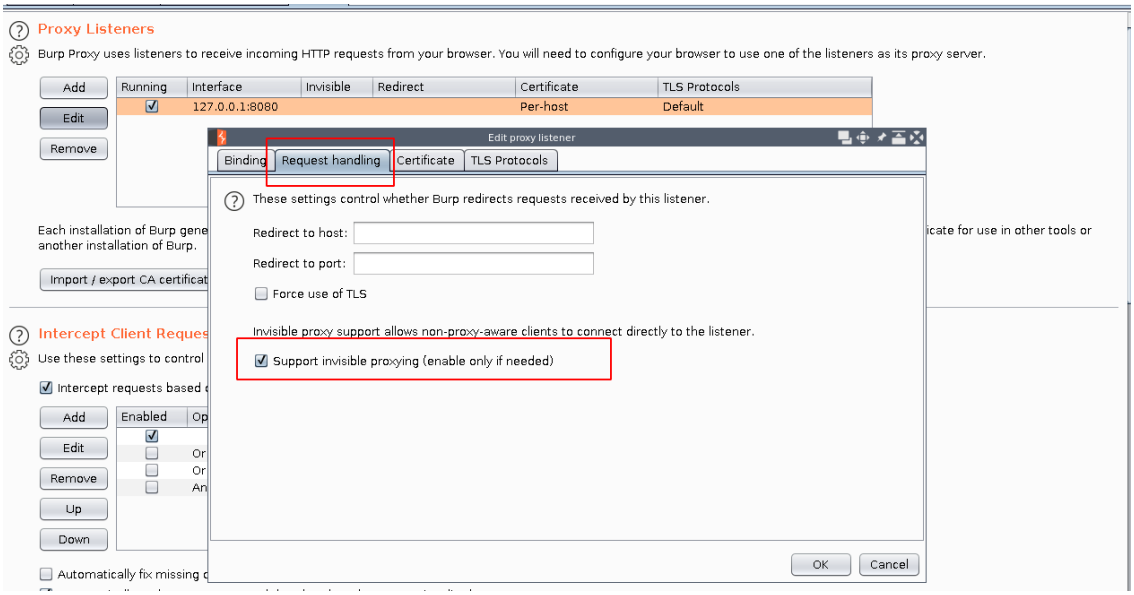
   修改burp代理设置，原配置只监听127.0.0.1接口，所以明白的



可以具体设置内网端wifi接口ip，这里我监听所有口了：



然后，很重要，开启透明（invisiable英文重要）代理模式，可以试试，如果这个模式不开，能访问burp，得不到response：



burp配置比较简单。

2. 启动wifi共享，参照自己网络参数配置：

```
封装一个脚本：start_wifi.sh
airmon-ng start wlx00e05c30a11f
ip link set wlan0mon up
ip addr add 192.168.100.1/24 dev wlan0mon
route add -net 192.168.100.0 netmask 255.255.255.0 gw 192.168.100.1
hostapd hostapd.conf &
dnsmasq -C dnsmasq.conf -d &
bash config_iptables.sh
#注意查看防火墙规则，尤其设备有docker要注意。docker规则会影响nat，具体还没时间分析
```

3. 配置iptables把wifi内网网段80和443的流量重定向到burp的8080口，当然也可以另外一台机器的burp。（to-port 8080改to ip:8080)

```
这个也可以封装到脚本：
iptables -t nat -A PREROUTING -s 192.168.100.0/24 -i wlan0mon -p tcp -m tcp
--dport 80 -j REDIRECT --to-ports 8080
iptables -t nat -A PREROUTING -s 192.168.100.0/24 -i wlan0mon -p tcp -m tcp
--dport 443 -j REDIRECT --to-ports 8080
```

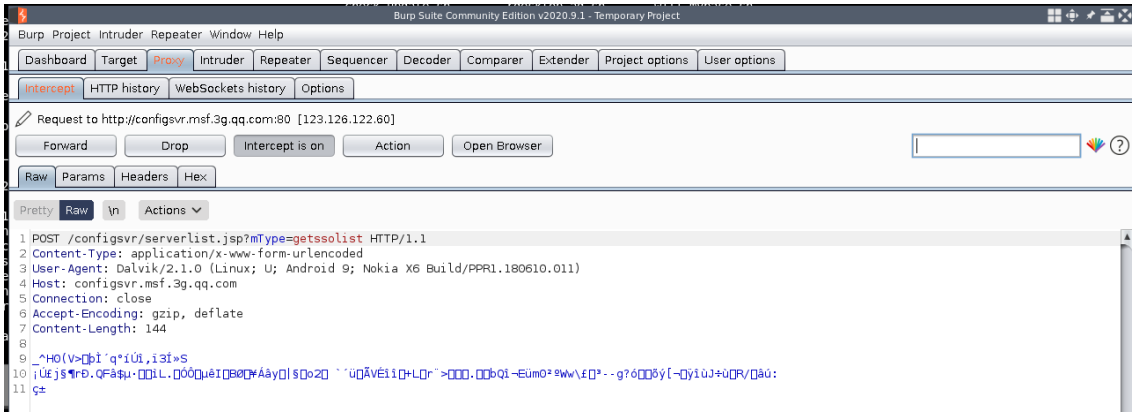
然后检查下iptables：(主要看filter表和nat表)

```
root@mykali:~/wifi_phish# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  192.168.100.0/24     anywhere             tcp dpt:http
redir ports 8080
REDIRECT   tcp  --  192.168.100.0/24     anywhere             tcp dpt:https
redir ports 8080

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  anywhere             anywhere
```

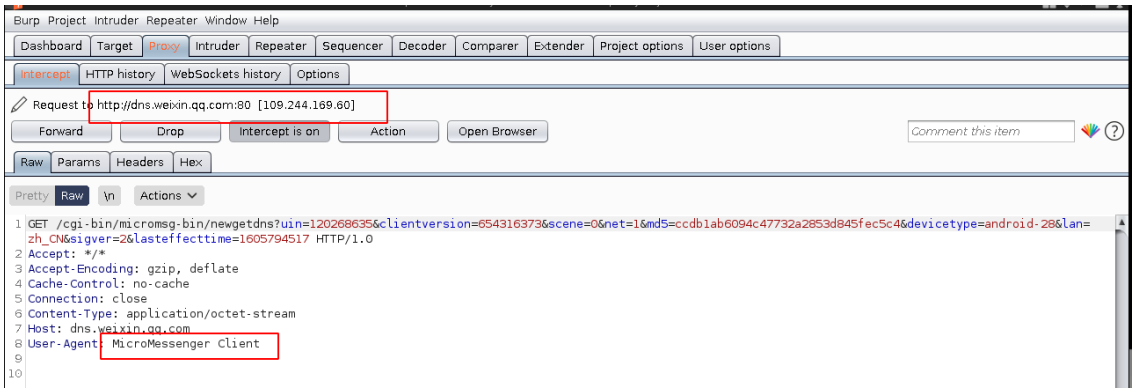
```
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

root@mykali:~/wifi_phish# iptables -t filter -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

iptables不熟悉，理解iptables机制和操作花费不少时间，菜鸡。

4. 可以看到burp上好多http的包，浏览器的、app的开心啊，相当全局代理。



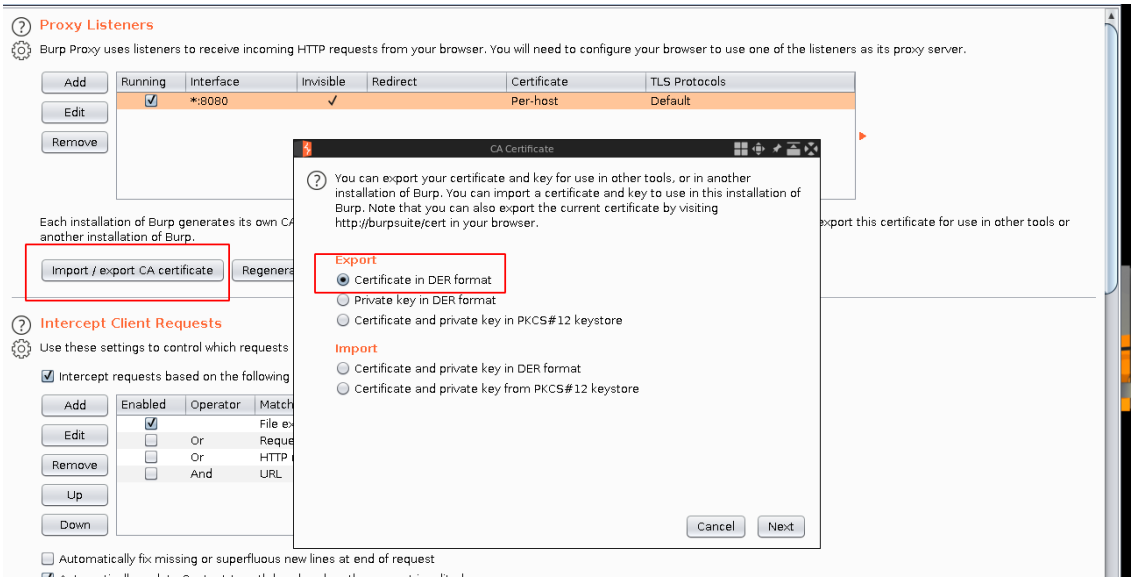
用的全新干净的debian做的，没处理burp字体。看手机qq通信包被抓。我的Nokia



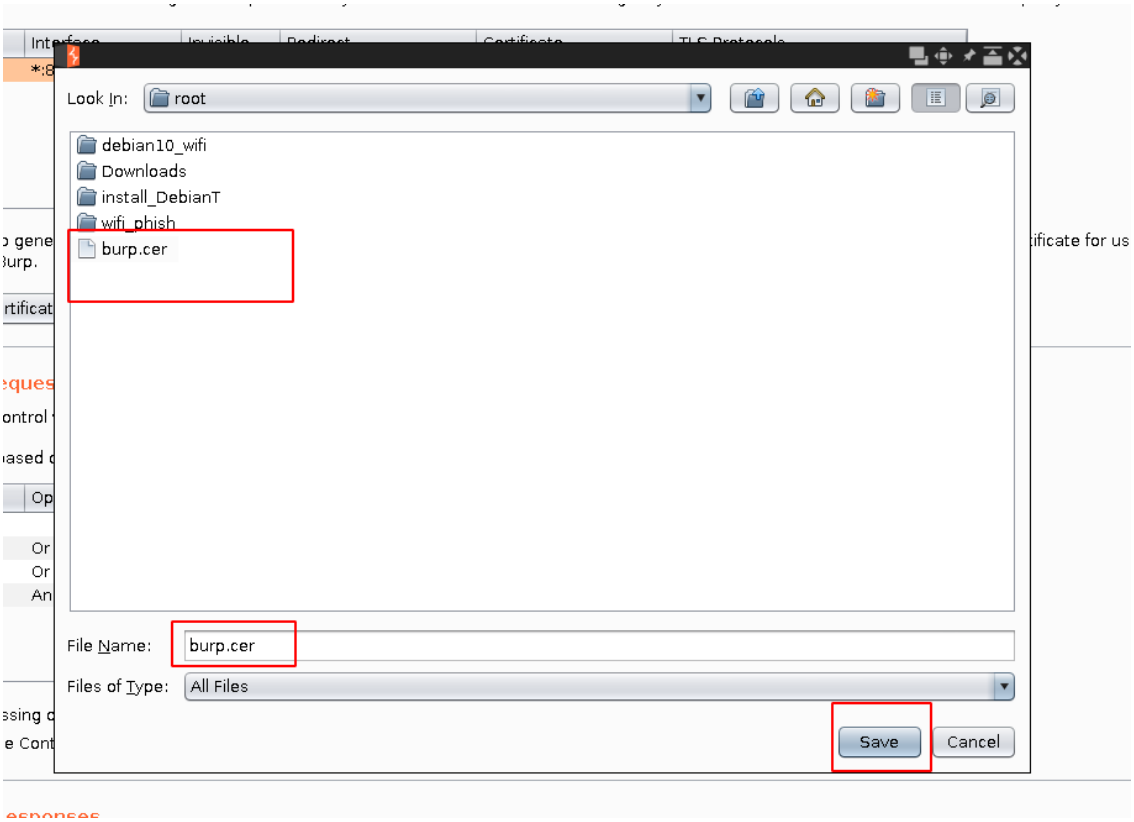
微信的通信（后边连扫码登录都拿到了）

5. 配置https：（吐槽现在chromium导入证书真费劲，firefox很友好）

   burp导出证书，本地导出和浏览器下载都行：



导出：



6. 使用adb导入安卓，一般方式adb push 到安卓的/sdcard，然后用“设置”的证书导入。但是我的车机阉割版，没有证书导入。

参考：

- https://blog.csdn.net/winceos/article/details/32130655
- https://blog.csdn.net/u011975363/article/details/83654074
- https://www.cnblogs.com/liuqiyun/p/12488845.html

希望：

- https://blog.csdn.net/jlvsjp/article/details/78018393

准备试试adb启动证书安装：

```
adb shell am start -n com.android.certinstaller/.CertInstallerMain -a
android.intent.action.VIEW -t application/x-x509-ca-cert
file:///sdcard/cacert.cer
```

证书在：`AOSP Android系统中CA证书文件的位置在：/ system/etc/security/cacerts/一系列的以数字命名的.0文件`

参卡：

> 首先看Android 4.x 系统的证书存放位置：
> AOSP Android系统中CA证书文件的位置在：/ system/etc/security/cacerts/一系列的以数字命名的.0文件
>
> 方法一：
> Android 4.0 已经支持用户安装根证书了，只需要将根证书放到sdcard根目录，然后到设置(Settings) - 安全(Security) - 从存储设备安装(Install from storage)就可以了，但是这样安装需要设置锁屏PIN或密码才可以。
> 但是，该操作需要每次打开手机输入锁屏PIN或密码，为用户带来很大的麻烦。
>
> 方法二：（注意：需要Root 权限才可以）
> 手机获取Root权限后，直接把Base64文本格式的根证书文件复制到etc/security/cacerts文件夹里，然后到设置(Settings) - 安全(Security) - 受信任的凭据(Trusted credentials)里面，此时你要安装的根证书应该会显示已经安装好了。这样安装之后根证书是作为系统证书使用的，而不是按照方法一安装方式的用户证书。
> 如果要删除就把文件夹里面的根证书文件删掉或者直接把证书后面的勾去掉就行了。

但是，直接强制adb push过去没用的，readonly，或许能搞到ssh不知道行不行?

```
[root@archlinux ~]# adb push burp.cer   /system/etc/security/cacerts/
burp.cer: 1 file pushed, 0 skipped. 3.1 MB/s (940 bytes in 0.000s)
adb: error: failed to copy 'burp.cer' to
'/system/etc/security/cacerts/burp.cer': remote Read-only file system
```

所以，证书只能push到/sdcard/：(常规操作，我的车机奇葩，只能访问/sdcard，内部/dada什么的通过屏幕访问不到！！)

```
[root@archlinux ~]# adb push burp.cer   /sdcard
burp.cer: 1 file pushed, 0 skipped. 2.0 MB/s (940 bytes in 0.000s)
```

明天去单位试试 adb启动证书安装的大招，看看是不是这个也被阉割

PS：chromium 导入cert最靠谱一个，国外大佬：https://www.youtube.com/watch?v=rPz9uYmt_xo