美团点评技术团队

如何把Android手机变成一个WIFI下载热点? — 报文转发及DNS报文拦截

球哥 · 2014-05-17 11:38

随着wifi的普及，移动运营商的热点也越来越多了，如中国移动的CMCC、中国电信的ChinaNet、中国联通的ChinaUnicom等，一般来说，连上此类的热点，打开浏览器上网时都会自动跳转到一个验证页面，最近有个项目也有类似的需求，Android手机自建热点，别的手机wifi连接此热点，打开浏览器，输入任意内容，自动跳转到一个下载列表页面，点击相应的链接即可下载相应的文件。

分析

考虑如下几种情况:

- 浏览器输入IP地址，请求对应IP地址的80端口的内容
- 浏览器输入域名，先进行DNS解析域名，得到IP地址后，请求对应的80端口的内容
- 浏览器输入任意字符，一般浏览器内部设置一个默认的搜索引擎，此时地址栏的内容会作为搜索的关键字，加在搜索的url中

因此，需要解决如下问题:

- 端口报文转发
- DNS报文拦截
- url重定向

端口报文转发

Android系统本身是Linux内核，1024以下端口都名花有主，如http是80，https是443，dns是53，对于这些1024以下端口的绑定需要root权限，但一般的App是没有root权限的，除非在 AndroidManifest.xml 文件中声明 android:sharedUserId="android.uid.system"，并使用密钥文件进行签名:

```
java -jar signapk.jar platform.x509.pem platform.pk8 your.apk your_signed.apk
```

但问题是密钥文件属于手机厂商，显然不可能拿到这个密钥文件，当然，如果在模拟器里测试倒是可以的，从android源代码 build/target/product/security 里找到密钥文件，platform.pk8 和 platform.x509.pem，签名工具 signapk.jar 在 build/tools/signapk 下。

基于以上原因，一般Web服务器都绑定8080端口，手机浏览器如果输入IP地址，会访问Web服务器的80端口，这样就需要进行端口报文转发，对应dns报文拦截，无法监听53端口，同样需要端口转发，此外，浏览器的搜索引擎如果是google的话，使用https，同样也有这个问题。

iptables是个很好的防火墙管理工具，这里需要做如下配置：

```
)tables -t nat -A PREROUTING -d 0.0.0.0/0 -p tcp --dport 80 -j DNAT --to 192.168.43.1:8080
)tables -t nat -A PREROUTING -d 0.0.0.0/0 -p tcp --dport 443 -j DNAT --to 192.168.43.1:8443
)tables -t nat -A PREROUTING -d 0.0.0.0/0 -p udp --dport 53 -j DNAT --to 192.168.43.1:53530
```

说明：-t nat：指定nat表，-A：添加，PREROUTING：路由前处理，-d 0.0.0.0/0：任意目的地IP，-p tcp：协议，--dport 80：端口，-j DNAT：地址映射跳转，--to 192.168.43.1:8080：转发目的地，总的意思就是，到达防火墙的报文，不管去往那个IP地址，只要是发往80端口的tcp包，都转发到192.168.43.1的8080端口。剩下两条意思类似。

需要注意的是：

1、如果是App中的java代码调用，需要root权限，一般这么写：

```
:ables -t nat -A PREROUTING -d 0.0.0.0/0 -p tcp --dport 80 -j DNAT --to 192.168.43.1:8080";
:(shell);
```

2、Android手机设置为热点模式时，IP地址一般都会固定成192.168.43.1，这是由手机的dhcpcd服务指定的，一般不会去改dhcpcd服务的源代码然后重新编译，但这种写死的做法显然是不太合适的，通用的做法是自动取Ap的IP地址：

```
    public static String getNetworkIpAddress(String name) {
        try {
            Enumeration<NetworkInterface> interfaces = NetworkInterface.getNetworkInterface
            while (interfaces.hasMoreElements()) {
                NetworkInterface networkInterface = (NetworkInterface) interfaces.nextEleme
                Enumeration<InetAddress> enumeration = networkInterface.getInetAddresses();
                while (enumeration.hasMoreElements()) {
                    InetAddress inetAddress = (InetAddress) enumeration.nextElement();
                    if (!inetAddress.isLoopbackAddress()
                            && inetAddress instanceof Inet4Address
                            && TextUtils.equals(name, networkInterface.getDisplayName())){
                        return inetAddress.getHostAddress().toString();
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return "";
    }

    public static String getApName(Context context) {
        try {
            ConnectivityManager connectivityManager = (ConnectivityManager) context.getSyst
            Method method = connectivityManager.getClass().getMethod("getTetheredIfaces");
            String[] names = (String[]) method.invoke(connectivityManager);
            return names[0];
        } catch (Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

看着挺复杂的，因为热点模式和连接到别的热点是完全不同的，取Ap名字时，用到了一个隐藏的方法，需要用反射的方式调用。

DNS报文拦截

DNS意思是域名解析协议，用户打开浏览器浏览网页时，不会记IP地址，而是记某些有含义的网址，DNS就是解决网址到IP地址的对应问题。DNS报文格式参考RFC1035文档，微软的网站上也有介绍：http://technet.microsoft.com/en-us/library/dd197470(v=ws.10).aspx (http://technet.microsoft.com/en-us/library/dd197470(v=ws.10).aspx)，这里主要介绍DNS报文的格式。

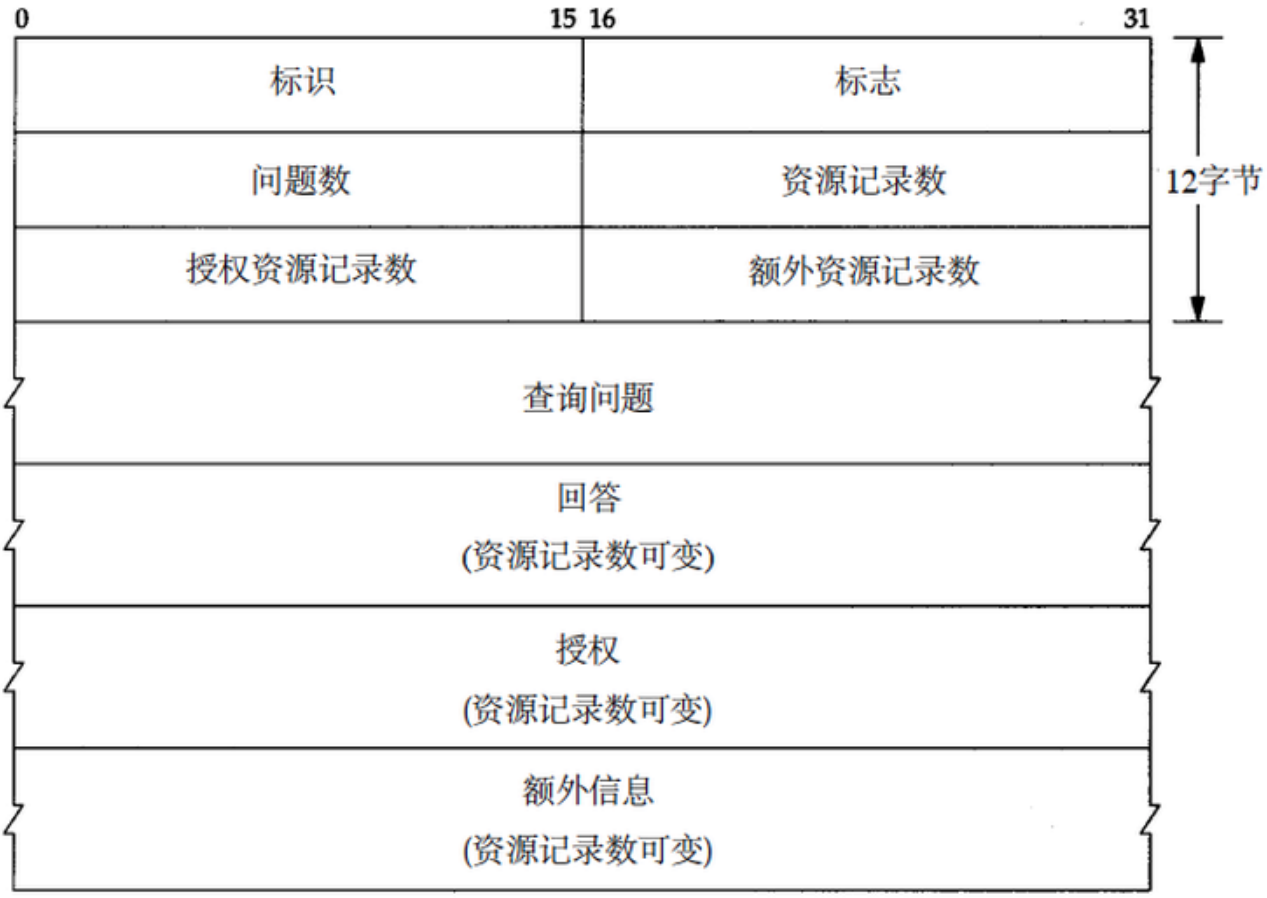
DNS报文格式

DNS报文一般由如下部分组成：

- DNS header (fixed length，12 Bytes)
- Question entries (variable length)

- Answer resource records (variable length)
- Authority resource records (variable length)
- Additional resource records(variable length)

套用《TCPIP详解卷》中的一张图



DNS查询和响应的一般格式

DNS header（固定占12字节）

总共占12字节，结构如下：
-- 标识：报文的标识，占2字节，查询的报文里生成，响应的报文里复制此内容，用来标识是对相应查询的响应
-- 标记：报文的标记位，占2字节，也就是16位，如下：

DNS报文首部中的标志字段

- QR：0标识查询，1标识响应，
- opcode：0为一般查询，1为反向查询（IP地址反查域名），2为查询服务器状态，一般为0
- AA：是否为授权回答（authoritative - answer），可以理解为当前域名服务器是否对结果负责，如果从别的域名服务器查询过来的结果，显然不是当前域名服务器可掌控的，因此设为0
- TC：是否被截断，UDP报文限定512字节（不包含IP及头部信息），如果超出将截断，此标记也被置1

- RD：是否递归查询（Recursion Desired），如果为1，说明DNS服务器如果没有结果，那么DNS服务器会递归地找别的DNS服务器要结果，直到得到结果并返回，如果为0，则在没有结果的情况下，返回DNS列表
- RA：是否支持递归查询，在响应报文中，一般都会支持递归查询
- zero：必须为0
- rcode：错误码，一般为0，表示没有错，如果不为0，表示有问题，错误码可以参考相关文档

--问题资源数：占2字节
--回答资源数：占2字节
--授权资源数：占2字节
--附加资源数：占2字节

Question entries（不定长）

不定长，结构如下：

- 名字：域名的字符串表示，microsoft.com表示为：0x09microsoft0x03com0x00，需要注意的是长度的最高两位必须为0，因此字符长度不能超过63，也就是说最多0x3f，另外，参考rfc1053，为简化起见，域名总长度不能超过255
- 类型：相关含义可查手册，一般是0x0001，表示IP地址类型
- 类：一般是0x0001，表示普通的internet问题

Answer resource records（不定长）

不定长，结构如下：

- 名字：同前面的查询名字，一般会以索引方式表示，引用到前面的字符，比如前面的 microsoft.com 字符在报文中的位置
- 类型：同前面的查询类型
- 类：同前面的查询类
- 生存时间：报文生存时间（TTL），占4字节，单位秒
- 资源长度：占2字节
- 资源内容：资源具体的内容，如IP地址：1.1.1.1 表示为 0x01010101

Authority resource records（不定长）

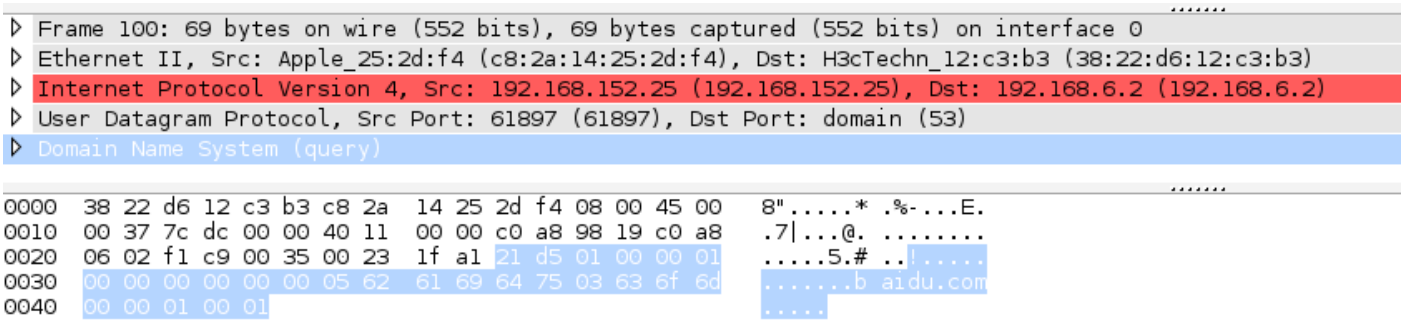
同上。

Additional resource records（不定长）

同上。

例子

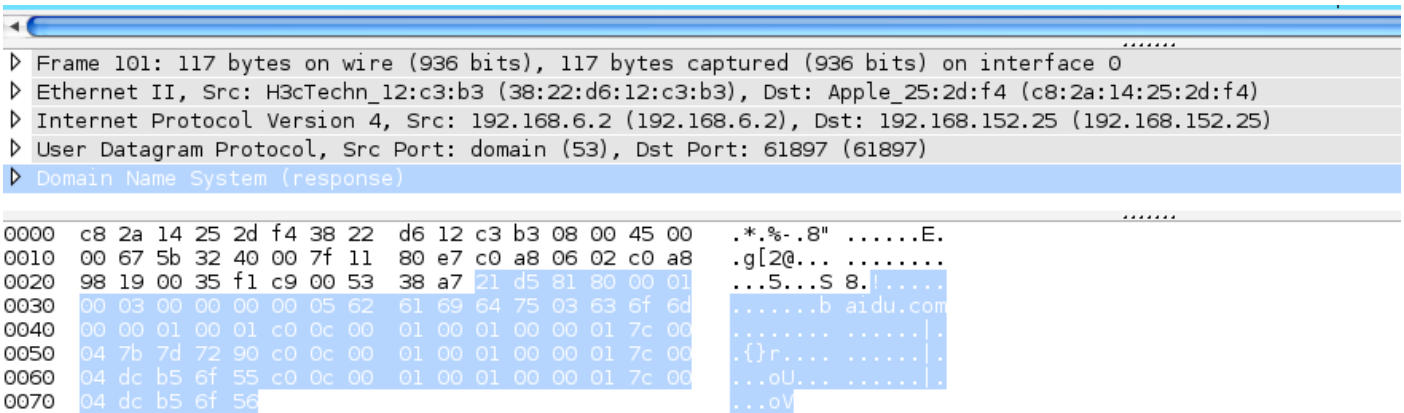
经过以上的分析，来看个例子，打开wireshark抓包工具，监听网卡数据包，打开控制台，输入：host baidu.com，并抓取DNS报文。



上图为DNS查询：

```
21d5010000010000000000000562616964750363 6f6d0000010001
```

- 21 d5：会话标识，应答中用于标识是哪个查询
- 01 00：标记，二进制为0000 0001 0000 0000，参考标记位，RD被置1，标识是一个递归的查询
- 00 01：问题数1
- 00 00：回答资源数0
- 00 00：授权资源数0
- 00 00：额外资源数0
- 05 62 61 69 64 75 03 63 6f 6d 00：域名名字，就是5baidu3com0这种格式，数字标识字符的数量，baidu有5个字符，所以是5baidu，域名字符串最后要跟一个0x00
- 00 01：查询IP地址
- 00 01：普通的internet查询



上图为DNS应答：

```
21d5818000010003000000000562616964750363 6f6d0000010001c00c000100010000017c00047b7d7290c00c
```

- 21 d5：标识，和之前的查询一一对应
- 81 80：标记，二进制为1000 0001 1000 0000，QR、RD、RA被置1，表示支持递归查询的应答
- 00 01：问题数1
- 00 03：回答资源数3
- 00 00：授权资源数0
- 00 00：额外资源数0

- 05 62 61 69 64 75 03 63 6f 6d 00：域名名字，参考前面的查询报文
- 00 01：查询IP地址
- 00 01：普通的internet查询
- c0 0c：域名名字，应该和前面的一致，但是为了节省报文长度，这是个引用，怎么知道的呢? 因为是c0，二进制是1100 0000，最高位两位置1了，056261696475036 36f6d00这串字符前面有12个字节，因此相对位置是0c（从0开始）
- 00 01：查询IP地址
- 00 01：普通的internet查询
- 00 00 01 7c：生存时间（Time to Live，TTL，单位秒），380秒
- 00 04：资源长度，4个字节，表示接下来的4个字节是资源的实际内容
- 7b 7d 72 90：资源内容，其实就是IP地址，123.125.114.144
- c0 0c ...：同上

从上面的DNS应答报文看，关注7b7d7290、dcb56f55、dcb56f56即可，分别对应三个IP地址：123.125.114.144、220.181.111.85、220.181.111.86

DNS报文拦截实现

了解了DNS报文的内容，下面需要做的就是，监听DNS端口，构造自己的报文返回即可，由于权限问题，一般Android的App是无法监听53端口的，这里可以监听53530端口，再通过iptables设置防火墙，将53端口的报文转发到53530端口即可，注意DNS是UDP包，代码参考如下

```
byte[] requestBuffer = new byte[256];
byte[] responseBuffer = new byte[256];
byte[] ipBuffer = { (byte) 0xc0, 0x0c, 0x00, 0x01, 0x00, 0x01, 0x00,
        0x00, 0x01, 0x7c, 0x00, 0x04, 0x01, 0x01, 0x01, 0x01 };
try {
    datagramSocket = new DatagramSocket(53530);
    DatagramPacket requestPacket = new DatagramPacket(requestBuffer,requestBuffer.length);
    while (!Thread.currentThread().isInterrupted()) {
        datagramSocket.receive(requestPacket);
        int requestLength = requestPacket.getLength();
        System.arraycopy(requestBuffer, 0, responseBuffer, 0, requestLength);
        System.arraycopy(ipBuffer, 0, responseBuffer, requestLength, ipBuffer.length);
        // 标志位
        responseBuffer[2] = (byte) 0x81;
        responseBuffer[3] = (byte) 0x80;
        // 响应数
        responseBuffer[6] = (byte) 0x00;
        responseBuffer[7] = (byte) 0x01;
        DatagramPacket response = new DatagramPacket(responseBuffer, requestLength + ipBuf
        datagramSocket.send(response);
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

这样，所有的DNS解析请求都被转到1.1.1.1这个IP地址了。