

iOS,Android网络抓包教程之tcpdump

现在的移动端应用几乎都会通过网络请求来和服务器交互，通过抓包来诊断和网络相关的bug是程序员的重要技能之一。抓包的手段有很多：针对http和https可以使用Charles设置代理来做，对于更广泛的协议可以使用tcpdump或者wireshark。wireshark提供GUI，方便做深入全面的数据分析。tcpdump则输出原始的包内容，好处是快速高效，之前写过一篇简单的微信红包图片的破解教程 (<http://www.mrpeak.cn/ios/2016/01/26/tcpdump-wc>)，就是用tcpdump来操作的。这篇文章主要介绍tcpdump的基本使用方法，阅读目标是能基本掌握并运用tcpdump解决网络相关的问题。阅读前提是对TCP/IP (https://en.wikipedia.org/wiki/Internet_protocol_suite) 有初步的了解。

1.启动tcpdump

1.1 iOS上启动tcpdump

iOS设备上启动tcpdump比较方便。apple在mac上有个叫rvictl的程序，可以通过iOS设备的udid创建一个虚拟网卡，然后通过这个虚拟网卡监听设备上所有的网络流量。步骤如下：

获取itunes获取设备udid

iPhone 6

容量： 11.91 GB
电话号码： +86 137-5813-
UDID： 418D838461A811A05A163888C9F10778
1

拷贝

打开终端（terminal），创建虚拟网卡

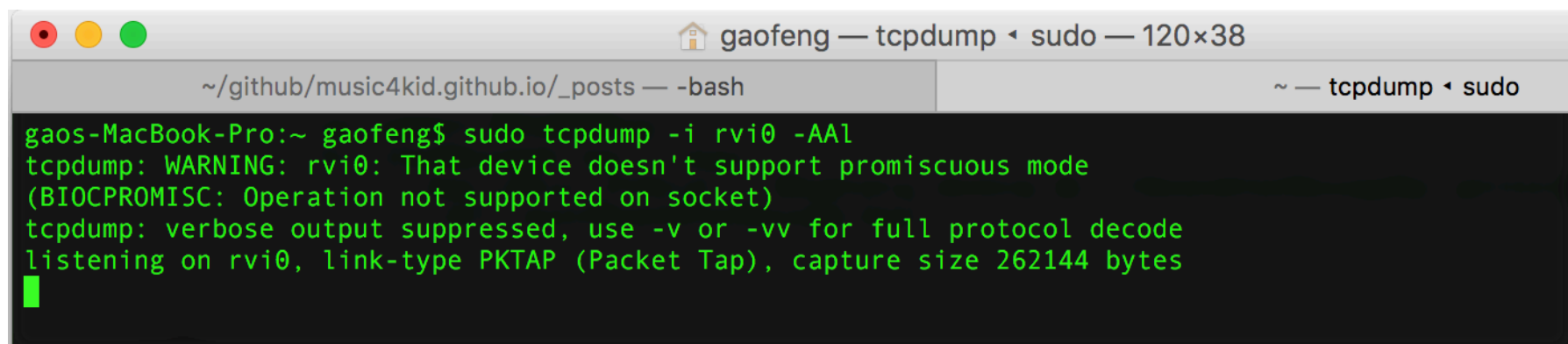
在终端输入`rvictl -s udid`，创建虚拟网卡。

A terminal window titled 'gaofeng — -bash — 120x38' with a tab labeled '~/github/music4kid.github.io/_posts — -bash'. The terminal shows the command 'rvictl -s 418d838461a891b5d7161a05a163888c9f107781' being executed. The output is 'Starting device 418d838461a891b5d7161a05a163888c9f107781 [SUCCEEDED] with interface rvi0'. The prompt returns to 'gaofeng\$'.

```
gaofeng$ rvictl -s 418d838461a891b5d7161a05a163888c9f107781
Starting device 418d838461a891b5d7161a05a163888c9f107781 [SUCCEEDED] with interface rvi0
gaofeng$
```

启动tcpdump监控流量

在终端继续输入`sudo tcpdump -i rvi0 -AAI`，启动tcpdump监控。

A terminal window titled 'gaofeng — tcpdump ◀ sudo — 120x38' with a tab labeled '~/github/music4kid.github.io/_posts — -bash'. The terminal shows the command 'sudo tcpdump -i rvi0 -AAI' being executed. The output is 'tcpdump: WARNING: rvi0: That device doesn't support promiscuous mode (BIOCPROMISC: Operation not supported on socket) tcpdump: verbose output suppressed, use -v or -vv for full protocol decode listening on rvi0, link-type PKTAP (Packet Tap), capture size 262144 bytes'.

```
gaofeng$ sudo tcpdump -i rvi0 -AAI
tcpdump: WARNING: rvi0: That device doesn't support promiscuous mode
(BIOCPROMISC: Operation not supported on socket)
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on rvi0, link-type PKTAP (Packet Tap), capture size 262144 bytes
```

2.1 Android上启动tcpdump

Android设备没办法通过`rvictl`创建虚拟网卡，但是可以把`tcpdump`的可执行文件上传到android设备上，然后通过mac远程登录android设备运行`tcpdump`，前提是这台android设备必须已经root过。步骤如下：

下载android版本的tcpdump

从这个链接 (<http://www.androidtcpdump.com/android-tcpdump/downloads>)可以下载到专门为android系统编译的`tcpdump`版本。

通过adb将tcpdump上传到android设备

通过`adb push`将`tcpdump`文件上传到特定的目录，这里我们选择`/sdcard/data`目录。

```
gaos-MacBook-Pro:~ gaofeng$ adb push ~/Desktop/tcpdump /sdcard/data
6161 KB/s (1893728 bytes in 0.300s)
gaos-MacBook-Pro:~ gaofeng$
```

在android设备上运行tcpdump

通过adb shell登陆设备，并执行tcpdump，最后一步执行./tcpdump即可。

```
gaos-MacBook-Pro:~ gaofeng$ adb shell
shell@kltevzw:/ $ cd /sdcard/data
shell@kltevzw:/sdcard/data $ chmod 777 tcpdump
```

2. 分析tcpdump输出

经过上面的步骤成功运行tcpdump之后，接下来就可以分析输出的网络包内容了，iOS设备和Android设备的输出是一致的。我们先来解析下几个基本的格式：

```
14:37:41.615018 IP 17.143.164.37.5223 > 10.29.44.240.58036: Flags [P.], seq 1:54, ack 101, win 255, options [nop,nop,TS val 2381386761 ecr 427050796], length 53
.....pdp_ip0.....g...apsd.....N.V
jb.....E..i.x@.'...Q...%
...g...XI.O.B.....
...tG,...0.6.0)pZy....zZ.....|. (.L{.Te..$P...Q..f.e....e
14:37:41.615234 IP 10.29.44.240.58036 > 17.143.164.37.5223: Flags [.], ack 54, win 4094, options [nop,nop,TS val 427051182 ecr 2381386761], length 0
.....pdp_ip0.....g...apsd.....N.V
Bc.....E..4e$..@.(.
...%...gO.B..XIN.....
.tH....
14:37:41.856203 IP 10.29.44.240.58407 > 113.31.17.122.redwood-broker: Flags [S], seq 1017270493, win 65535, options [mss 1410,nop,wscale 5,nop,nop,TS val 427051422 ecr 0,sackOK,eol], length 0
.....pdp_ip0.....Toutiao.....N.V
.....E...@yV...@.G.
...q..z..'...<.P.....
.tI.....
14:37:41.951247 IP 113.31.17.122.redwood-broker > 10.29.44.240.58407: Flags [S.], seq 2023343145, ack 1017270494, win 14600, options [mss 1400,nop,nop,sackOK,nop,wscale 7], length 0
.....pdp_ip0.....Toutiao.....N.V
.....E..4..@.-...q..z
...x..>.<.P...9..j.....x.....
14:37:41.951415 IP 10.29.44.240.58407 > 113.31.17.122.redwood-broker: Flags [.], ack 1, win 8192, length 0
.....pdp_ip0.....Toutiao.....N.V
w.....E..(.P..@...
...q..z..'...<.P.x..*P. ..
```

图中红色方框内的部分是一个ip包的详细记录，类似的纪录还有好几条。这里我们着重分析第一条的各部分字段含义。

14:37:41.615018 很简单，是该包接收到的时间。

17.143.164.37.5223 是发送方的ip地址及端口号（5223是端口号）。

10.29.44.140.58036 是我iphone的ip地址及端口号。

Flags [P.] 是tcp包header部分的第14个字节的P位。这个字节所包含的几个flag很重要，后面我会单独详细讲解。这里P位表示接受方需要马上将包push到应用层。

seq 1:54 tcp包的seq号，1是起始值，54结束值。tcp之所以被认为是流，是因为tcp包所携带的每一个字节都有标号（seq号）。1:54表明总共有54个字节被接受，其中一个字节是三次握手阶段所使用，所以一共发送的长度是53字节。

ack 101 tcp包的ack号，ack 101表明seq号为100的字节已被确认收到，下一个期望接收的seq号从101开始。

win 255 win表示的是tcp包发送方，作为接受方还可以接受的字节数。这里win 255表明ip为17.143.164.37的主机还可以接受255个字节。

options [nop,nop,...] options[...]表示的是该tcp包的options区域，nop是no operation的缩写，没什么实际用途，主要是用做padding，因为options区域按协议规定必须是4字节的倍数。

options[... TS val 2381386761] ts val这个值是tcp包的时间戳，不过这个时间戳和设备的系统时间没啥关系，刚开始是随机值，后面随着系统时钟自增长。这个时间戳主要用处是seq序列号越界从0重新开始后，可以确认包的顺序。

options[... ecr 427050796] ts ecr这个值主要用来计算RTT。比如A发送一个tcp包给B，A会在包里带上TS val，B收到之后在ack包里再把这个值原样返回，A收到B的ack包之后再根据本地时钟就可以计算出RTT了。这个值只在ack包里有效，非ack包ecr的值就为0。

length 53 这个length是应用层传过来的数据大小，不包括tcp的header。这个值和我们上面分析的seq 1:54是一致的。

以上就是一个基本的tcp包结构，大家可以按照上面的分析再把其他几个包理解下。我们在做应用的时候面对的更多是http协议，但对一个http请求是怎么通过tcp/ip分解成一个个的packet，然后怎么在网络上稳定可靠的传输，要有个基本的印象。下面我们再看下tcpdump更多的功

能，这些功能都是基于对tcp/ip协议的理解，遇到不理解的建议多google下相关的技术概念。

3. tcpdump知识拓展

再继续深入tcpdump之前，先贴上一张tcp header格式图，常看常新。

TCP Header																																		
Offsets	Octet	0								1								2								3								
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0	0	Source port																Destination port																
4	32	Sequence number																																
8	64	Acknowledgment number (if ACK set)																																
12	96	Data offset				Reserved 0 0 0			N S	C W R	E C E	U R G	A C K	P S H	R S T	S Y N	F I N	Window Size																
16	128	Checksum																Urgent pointer (if URG set)																
20	160	Options (if <i>data offset</i> > 5. Padded at the end with "0" bytes if necessary.)																																
...																																

3.1 TCP Flags （tcp header第十四个字节）

我们再仔细看下上面提到的flags概念， flags位于tcp header的第十四个字节， 包含8个比特位， 也就是上图的CWR到FIN。这8个比特位都有特定的功能用途， 分别是： CWR， ECE， URG， ACK， PSH， RST， SYN， FIN。

CWR ， **ECE** 两个flag是用来配合做congestion control的， 一般情况下和应用层关系不大。发送方的包ECE（ECN－Echo）为0的时候表示出现了congestion， 接收方回的包里CWR（Congestion Window Reduced）为1表明收到congestion信息并做了处理。我们重点看其他六个flag。

URG URG代表Urgent， 表明包的优先级高， 需要优先传送对方并处理。像我们平时使用terminal的时候经常ctrl+c来结束某个任务， 这种命令产生的网络数据包就需要urgent。

ACK 也就是我们所熟悉的ack包， 用来告诉对方上一个数据包已经成功收到。不过一般不会为了ack单独发送一个包， 都是在下一个要发送的packet里设置ack位， 这属于tcp的优化机制， 参见delayed ack (https://en.wikipedia.org/wiki/TCP_delayed_acknowledgment)。

PSH Push我们上面解释过，接收方接收到P位的flag包需要马上将包交给应用层处理，一般我们在http request的最后一个包里都能看到P位被设置。

RST Reset位，表明packet的发送方马上就要断开当前连接了。在http请求结束的时候一般可以看到一个数据包设置了RST位。

SYN SYN位在发送建立连接请求的时候会设置，我们所熟悉的tcp三次握手就是syn和ack位的配合：syn->syn+ack->ack。

FIN Finish位设置了就表示发送方没有更多的数据要发送了，之后就要单向关闭连接了，接收方一般会回一个ack包。接收方再同理发送一个FIN就可以双向关闭连接了。

这8个flag首字母分别是：C E U A P R S F。初看难以记忆，我脑洞了下，把它们组合成suprcafe，当然少了super少了个e，我可以将就下。我们在使用tcpdump的时候会经常看到这几个flag，[S],[P],[R],[F],[.]。其他几个都好理解，[.]特殊点，是个占位符，没有其他flag被设置的时候就显示这个占位符，一般表示ack。

3.2 tcpdump 更多使用参数

这部分我们来看下tcpdump常用的一些命令参数。文章最开始部分的tcpdump命令是这样的：
sudo tcpdump -i rvi0 -AAI。 -i rvi0 -AAI都是属于参数部分。常见的有这些：

- -i, 要监听的网卡名称， -i rvi0监听虚拟网卡。不设置的时候默认监听所有网卡流量。
- -A, 用ASCII码展示所截取的流量，一般用于网页或者app里http请求。-AA可以获取更多的信息。
- -X, 用ASCII码和hex来展示包的内容，和上面的-A比较像。-XX可以展示更多的信息（比如link layer的header）。
- -n, 不解析hostname,tcpdump会优先暂时主机的名字。-nn则不展示主机名和端口名（比如443端口会被展示成https）。
- -s, 截取的包字节长度，默认情况下tcpdump会展示96字节的长度，要获取完整的长度可以用-s0或者-s1600。
- -c, 只截取指定数目的包，然后退出。
- -v, 展示更多的有用信息，还可以用-vv -vvv增加信息的展示量。
- src, 指明ip包的发送方地址。
- dst, 指明ip包的接收方地址。
- port, 指明tcp包发送方或者接收方的端口号。
- and,or,not,操作法，字面意思。

上面几个是我个人比较常用的，更多的参数可以参考这个详细文档

(<http://manpages.debian.org/cgi-bin/man.cgi?query=tcpdump>)。有兴趣的可以分析下面几个例子练习下：

```
tcpdump 'tcp[13] & 16!=0'
```

```
tcpdump src port 80 and tcp
```

```
tcpdump -vv src baidu and not dst port 23
```

```
tcpdump -nnvvS src 192.0.1.100 and dst port 443
```

4. 用tcpdump分析http完整请求

说了这么多，我们再来实战下，看一个完整的http请求流程。下面截图里的流量是我监听的 知乎App点赞之后发送的一个https请求。我之前先分析过server的ip地址了，tcpdump命令是：

```
sudo tcpdump -i rvi0 -AAI src 60.28.215.123 or dst 60.28.215.123
```



```
music4kid.github.io — tcpdump — sudo — 125x60
16:32:12.349875 IP 10.29.44.240.49769 > 60.28.215.123.https: Flags [S], seq 3807915317, win 65535, options [mss 1410,nop,wsc
le 5,nop,nop,TS val 475309629 ecr 0,sack0K,eol], length 0
.....pdp_ip0.....k...osee2unifiedRel.....V.V..
.....E..@...@.k(
...<...{.i....-5.....U.....
T=
16:32:12.509230 IP 60.28.215.123.https > 10.29.44.240.49769: Flags [S.], seq 995125905, ack 3807915318, win 28960, options [h
ss 1400,sack0K,TS val 2656263653 ecr 475309629,nop,wscale 9], length 0
.....pdp_ip0.....k...osee2unifiedRel.....V....
.....E..<...@...C.<...{
.....i;Pj...-6..q Se.....x...
S1..T=
16:32:12.509422 IP 10.29.44.240.49769 > 60.28.215.123.https: Flags [.], ack 1, win 4120, options [nop,nop,TS val 475309788 e
r 2656263653], length 0
.....pdp_ip0.....k...osee2unifiedRel.....V....
.....E..4;...@...
...<...{.i....-6;Pj.....`.....
T..S1
16:32:12.515426 IP 10.29.44.240.49769 > 60.28.215.123.https: Flags [F.], seq 1.241, ack 1, win 4120, options [nop,nop,TS val
475309792 ecr 2656263653], length 240
.....pdp_ip0.....k...osee2unifiedRel.....Vb...
.....E..$....@...{.
...<...{.i....-6;Pj.....-.....
T..S].....V...7...un..J..f=.9P!..'.i..uu...~ II.t....R.#...-&.5.....=.....4.....+.$.#.
...0../.(.'.....=.<.5./..
.....j.....api.zhihu.com.
.....3t.....spdy/3.1.spdy/3.http/1.1.....
16:32:12.585516 IP 60.28.215.123.https > 10.29.44.240.49769: Flags [.], ack 241, win 59, options [nop,nop,TS val 2656263674
ecr 475309792], length 0
.....pdp_ip0.....k...osee2unifiedRel.....Vd...
.....E..47a@...
...<...{
.....i;Pj....&....;4.....
S1..T=
16:32:12.594225 IP 60.28.215.123.https > 10.29.44.240.49769: Flags [.], seq 1.1309, ack 241, win 59, options [nop,nop,TS val
2656263674 ecr 475309792], length 1388
.....pdp_ip0.....k...osee2unifiedRel.....V1.
.....E...7b@...M<...{
.....i;Pj....&....;Y.....
S]..T.....j...f..V.....*A\..h...{..tx.1H.Uf..D...$L ..<gX.4..*a..
t...^H.....e.6...0.....3t.. http/1.1... k.. g. d...0...0.....c.....j.0.. *.H.
...0D1.0 ..U...US1.0...U.
..GeoTrust Inc.1.0...U...GeoTrust SSL CA - G30...140927000000Z..180506235959Z0..1.0 ..U...CN1.0...1.0...U...
beijing1)0'.U.
..ZHIZHE SIHAI(BEIJING) TECHNOLOGY1.0...U...Infrastructure1.0...U...*.zhihu.com0.."0.. *.H..0...0..
.....%.^)4(6...<.....;N.@4'.<.3z9>.`...b.0..
y..[f....2o.K-...r.....Kr3...(T.D<.....+P(.CF...l...t..1..L.>+u..p.....].....3.....N.....Uk..^....eB...*W..
..N...HQI...'.|..kw.....`)i..c..^...5.K..#Fh.a.g...E/2..h..$[-.....b...[....p'.....0...0!..U...0...*.zhihu.com.
zhihu.com0 ..U...0.0...U.....0+..U...$0"0 .....http://gn.symcb.com/gn.crl0...U. ...0..0...
..H...E..60..0?..+.....3https://www.geotrust.com/resources/repository/legal0A..+.....05.3https://www.geotrust.com/resour
ces/repository/legal0...U.%..0...+.....0...U.#..0....o...?r<0}#...x...|Z|0W...+.....K0I0...+.....0...http://gn.s
ymcd.com0&..+.....0...http://gn.symcb.com/gn.crt0.. *.H.....Y.....).xU...h.]j...K..7S.a...e&..">....4|*
...l..$.F0l...V.4....\RA....Nq"P..@u..Po.A9.....BW}..@....$.....{w#.(E...m(...R.p>,W.....o>.....AR...:
U.%'%.=F0..[.....W.2.t...*.U.....?
dM..T<
```

图中列出了6个前面的packet，10.29.44.240是我iphone的ip地址，60.28.215.123是知乎server的ip地址，红色方框内是iphone发出的packet，白色方框内是server发出的packet。packet1是iphone三次握手的第一个syn包，packet2是server ack + syn的包，packet3是iphone ack的包。这3个packet之后tcp的三次握手就完成了。

packet4是iphone发出的http request。长度只有240个字节，所以一个packet就发过去了，当然还设置了flags的P位，request需要马上被应用层处理。包里面出现了spdy，点赞。

packet5是server ack刚收到的包，长度位0，所以这仅仅是一个ack包。

packet6是server返回http的response了，1388个字节。packet5和packet6都ack了seq为241的包，当然是为了增加ack的成功率。

中间还有好几个packet就不仔细分析了，最后再看下请求完成的最后几个包：


```
16:38:55.546655 IP 10.29.44.240.49773 > 60.28.215.123.https: Flags [F.], seq 2465, ack 3873, win 4095, options [nop,nop,TS va
l 475712667 ecr 2656305347], length 0
.....pdp_ip0.....k...osee2unifiedRel.....V_w...
.....E..4f...@...
...<..{.m...S.$.....
.Z...T..
16:38:55.791730 IP 60.28.215.123.https > 10.29.44.240.49773: Flags [R], seq 449228190, win 0, length 0
.....pdp_ip0.....k...osee2unifiedRel.....V.....
.....E..(5.@....2<..{
.....m.....P.....
```

最后两个packet比较简单，iphone发送个FIN + ACK的包就断开连接了，server直接发送了一个RST包后也断开连接了。

这篇教程到这里就结束了，建议大家自己多练习下，遇到不懂的参数或关键字多google。最好能系统的学习下tcp/ip协议😊。

上一篇

微信iOS朋友圈红包图片抓包教程
([/ios/2016/01/26/tcpdump-wc](https://ios.2016/01/26/tcpdump-wc))

下一篇

iOS应用层架构之CDD ([/blog/cdd/](https://blog.cdd/))

Hosted by **Coding Pages** (<https://pages.coding.me>)