Misc300 抓包 官方writeup

tags: `scapy` `adb` `android`

author: flanker

0x01 数据包分析

将数据包http://pan.baidu.com/s/1ntrzThB 密码：cbf2 下载下来，在wireshark中打开，看一下statistics和conversations，会看到一大坨http和personal-agent(5555)端口。http看过一遍，基本都是新浪新闻、google搜索ACTF这种，似乎没有什么有价值信息,(OS X里的X wireshark太难看了)。

Ethernet: 4 | Fibre Channel | FDDI | IPv4: 31 | IPv6 | IPX | JXTA | NCP | RSVP | SCTP | TCP: 170 | Tok

TCP Conversations

Address A	Port A	Address B	Port B	Packets	Bytes	Packets
10.0.2.2	51917	10.0.2.15	personal-agent	1 473	519 459	
10.0.2.15	55186	74.125.128.106	http	11	2 149	
10.0.2.15	46952	74.125.128.199	http	65	40 192	
10.0.2.15	50516	74.125.128.199	http	15	5 770	
10.0.2.15	40768	74.125.128.199	http	140	126 961	
10.0.2.15	53809	173.194.127.239	http	35	20 188	
10.0.2.15	43069	173.194.127.239	http	64	45 199	
10.0.2.15	53928	173.194.127.239	http	11	1 989	
10.0.2.15	46858	74.125.128.105	http	8	613	
10.0.2.15	35685	74.125.128.105	http	8	613	
10.0.2.15	52502	74.125.128.105	http	8	613	
10.0.2.15	34390	74.125.128.105	http	8	613	
10.0.2.15	44507	74.125.128.199	http	99	67 116	
10.0.2.15	59193	74.125.128.105	http	8	622	
10.0.2.15	42442	74.125.128.105	http	9	686	
10.0.2.15	39408	74.125.128.105	http	9	686	
10.0.2.15	55432	74.125.128.105	http	8	622	

☑ Name resolution ☐ Limit to display filter

那么5555端口会是什么？大概follow stream一下，

```
CNXN.....2.....host::.CNXN.....device::.OPEN
k.....track-jdwp.OKAY....k.....WRTE....k.....4.
OKAYk.....OPENl.....S.....shell:getprop.OPENm.....
dwp:61.OKAY....l.....OKAY....m.....WRTEm.....
HandshakeOKAY....m.....WRTEm.....@..T...READ.
m.....JDWP-
HandshakeOKAY....m.....OKAYm.....WRTE....m.....
@..T...READ.....WRTEm.....@..U...HELO.....OKAYm....
OKAY....m.....WRTE....m...Y....
.....Y@..U...HELO...F.....=...
.....D.a.l.v.i.k. .v.1...4...0.s.y.s.t.e.m._.p.r.o.c.e.s.sWRTEm.....&..
@..V...FEAT.....@..W...MPRQ....OKAYm.....OKAY....m.....
E....m.....
+.....@..V...FEAT.....h.p.r.o.f.-.h.e.a.p.-.d.u.m.p.-.s.t.r.
g.....h.p.r.o.f.-.h.e.a.p.-.d.u.m.p... .m.e.t.h.o.d.-.t.r.a.c.e.-.p.r.o.f
```

玩过android的人应该会意识到这是adb的协议。从数据流大小来看，普通的adb shell命令很难会产生这么多数据，那么要么

是adb pull从设备中拖取了什么信息，要么是adb push了什么东西。

再往下翻：

```
stdio.OKAY.....................WRTE.....................
OKAY.....................CLSE.....................CLSE.....................OPE
N.....................sync:.OKAY.....................WRTE........!...g
......STAT..../data/local/tmp/
Weibo.apkOKAY.....................WRTE............<.......STAT............OKA
Y.....................WRTE........'...........SEND..../data/local/tmp/
Weibo.apk,33188OKAY.....................WRTE............T`......DATA....PK........d..
D...r?...........res/drawable/add_account.xml.....R.N.@.....A.....QQQ.-B.|.
...D.-..%|.?..":~....!s.sr.$B...Z.;;..B.O...1.`..8....S..../..."..mj..BcD...5*.x W.
a.....1^Y..0.sm.k...\.*
.0..s..+...pk.#vh..'.yZ..gg.A.?`.b-'_sj.6.....g6..
4+
..um..x..o.x.... ^E.....=.o..0......z.....<].....
.....e....zs}...N...D....7.4..../...]~..8.Wy.........|....nv.+......:..|
o......PK.....r?........PK........d..D..j.?...........res/drawable/
btn_auth.xml.R.N.@.......B...QQQ.-B.|.
```

，

就会发现有意思的东西，安装流量，也就是说流量里是一个完整的安装APK的过程！

0x02 协议分析

adb安装APK的流程如下：

- adb push Weibo.apk，分解为如下几步：
  - STAT /data/local/tmp/Weibo.apk 检查目标文件是否已存在，能否写入
  - SEND weibo.apk
    - WRTE....(DATA....)(DATA....)(DATA....)DONE
  - adb shell pm install Weibo.apk

其中adb数据包本身协议可以参考adb代码中的protocol.txt，通过wireshark-adb插件辅助分析，可知adb数据包基本结构如下：

- adb头结构
  - Command (4bytes)
  - Arg0 (4bytes)
  - Arg1 (4bytes)
  - Data Length (4bytes)
  - Data Checksum (4bytes)
  - Magic (4bytes)
  - Payload

Payload中还会承载二级协议，例如DATA等。比如现在要写入一个大文件，那么command field会填为WRTE，随后跟随DATA(4byte)Length(4byte)，然后将剩余数据tcp stream发送出去。

0x03 quick and dirty hack script

在了解了adb协议之后，现在需要写一个脚本来自动化提取出里面的APK。我们选用scapy。注意TCP有流重组现象。首先要通过SEND /data/local/tmp/Weibo.apk定位到传输起始位置，然后通过pm install 定位到APK传输结束位置。脚本如下：

```
from scapy import *
from scapy.all import *
import io
import struct
import sys
```

```
b = io.BytesIO()

rawpcap = rdpcap(sys.argv[1])
rawpcap = [_ for _ in rawpcap if TCP in _ and  _[TCP].dport == 5555 and _[IP].src == "10.0.2.2
" and Raw in _[IP]]
rawpcap = next( rawpcap[i+1:] for i,p in enumerate(rawpcap) if Raw in p and p[Raw].load.find('
/data/local/tmp')  != -1 and p[Raw].load.find('SEND') != -1)
rawpcap = next( rawpcap[:i] for i,p in enumerate(rawpcap) if Raw in p and p[Raw].load.find('pm
 install') != -1)
print(len(rawpcap))
#b.write(''.join([p[Raw].load[24:] for p in rawpcap if Raw in p ]))
for p in rawpcap:
    if Raw in p:
        data = p[Raw].load
        if data.startswith('WRTE'):
            data = data[24:]
        b.write(data)
b.seek(0)

#print b.read()
a = open('out.apk','wb')
header = b.read(8)
while header != "":
    tag, datalen = struct.unpack('<4sI', header)
    if tag == "DATA":
        a.write(b.read(datalen))
    else:
        break
    header = b.read(8)
a.close()
```

这样就可以提取出原始apk，安装或者直接逆向就可以看到flag以toast的形式显示了出来。最后的FLAG是ACTF{ANDR01D_15_REALLLLLLLY_FUN}。

0x04 其他

这道题还是有一定难度的，比赛的时候做出来的人也不多，大家都纠结在http流量中了。这道题的出题灵感来源于一些应用在线沙盒检测系统，因为这些系统会提供pcap流量下载，通过模拟器启动时的-tcpdump选项，但这个流量往往把要检测的APK也给包含进去了，如上述分析。