Google

Can You Hear Me Now?

Remote Eavesdropping Vulnerabilities in Mobile Messaging Applications

About Me

- Natalie Silvanovich AKA natashenka
- Project Zero member
- Previously did mobile security on Android and BlackBerry
- Messaging enthusiast

Jan 29, 2019, 07:20am EST | 109,633 views

Apple Confirms iPhone FaceTime Eavesdropping Exploit -- Here's What To Do

Davey Winder Senior Contributor

Cybersecurity

I report and analyse breaking cybersecurity and privacy stories

This article is more than 2 years old.

Group FaceTime Bug

- Allowed call to be connected without user interaction
- Available through user interface
- Completely unprecedented

Group FaceTime Bug

- Allowed call to be connected without user interaction
- ~~Available through user interface~~
- Completely unprecedented

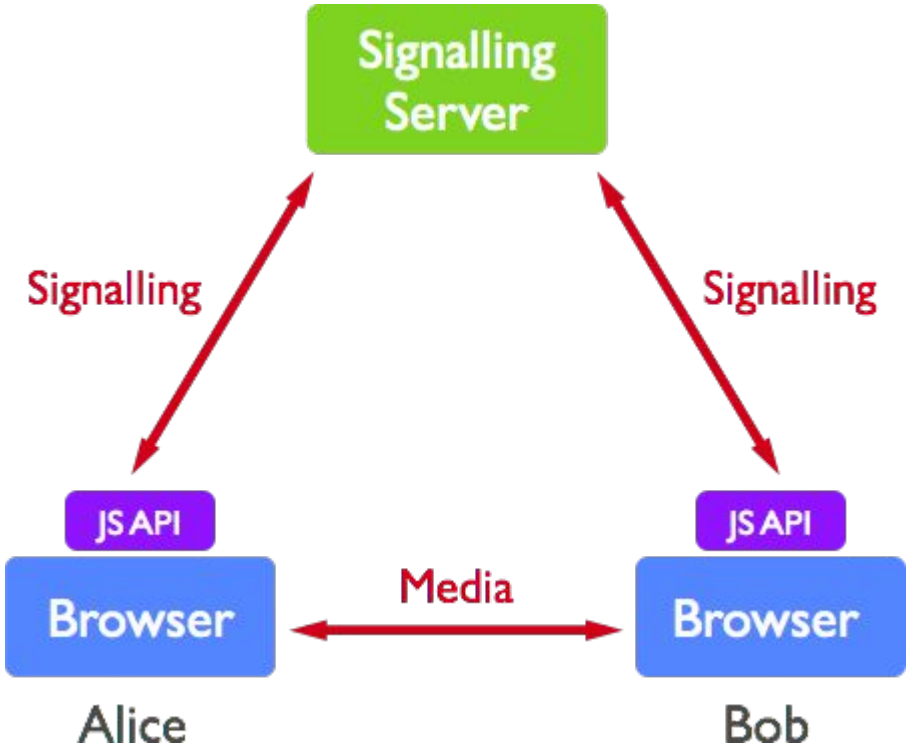
Goals

- Determine how this bug class works
- Investigate apps for similar bugs

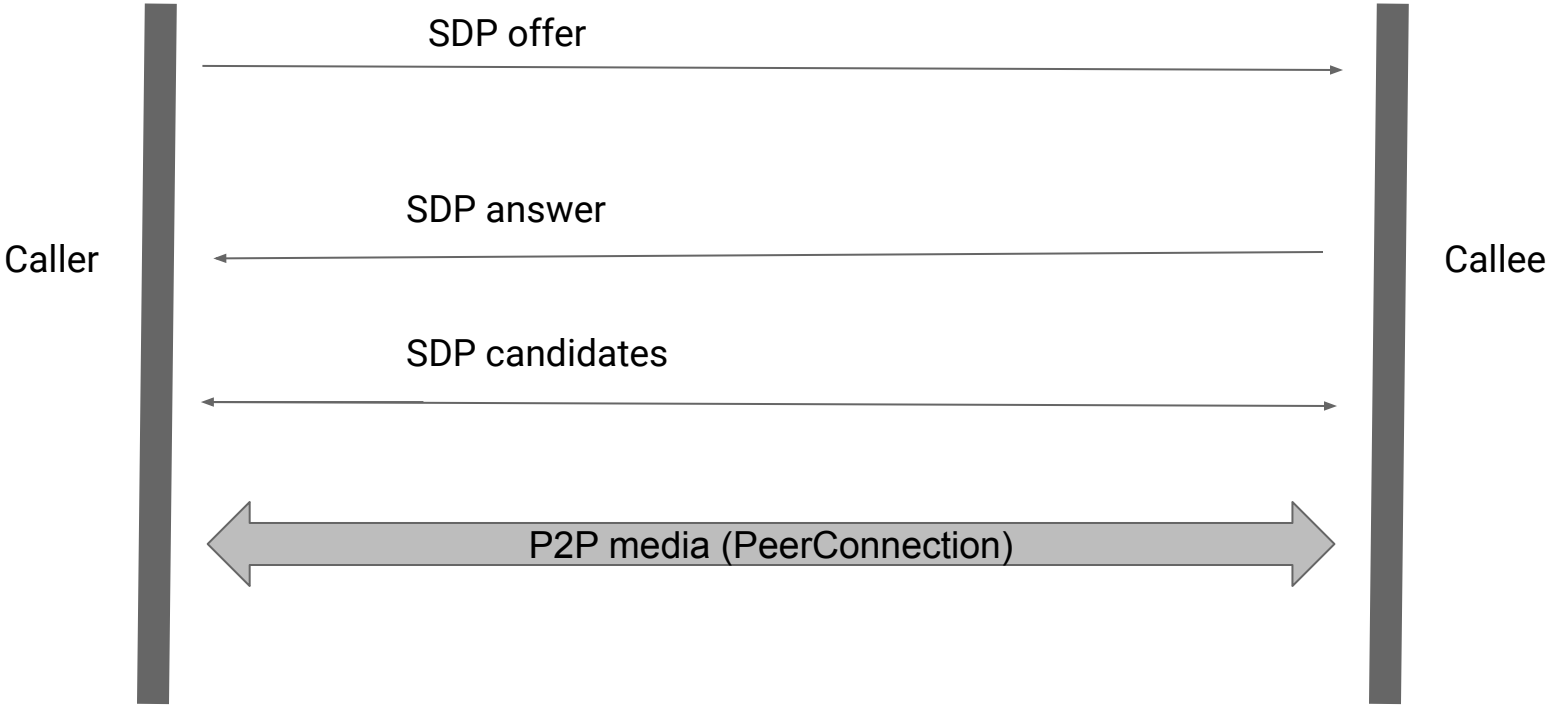
What is WebRTC?

- RTC = Real Time Communication
- Audio and video conferencing library maintained by Google
  - Also a protocol with a specification
- Used by all major browsers
- Used by many mobile applications
- Alternatives have similar design

WebRTC Architecture



Call Signaling Flow



Tracks

- Tracks are input devices that can be streamed to a peer
  - Camera
  - Microphone
- Tracks need to be added to a PeerConnection and enabled before input is streamed
- Can be done at any time during a call, but transmission won’t work until a P2P connection has been established

What causes call connection vulnerabilities?

- Video conferencing applications require a state machine to manage offers, answers, candidates and tracks
  - Sometimes they have implementation bugs
  - Sometimes developers misunderstand these constructs
  - Sometimes WebRTC has bugs*

*I haven’t seen an example of a state machine bug caused by this yet

Finding calling state machine vulnerabilities

- Understand state machine
- Think about possible problems
- Test problems

Understanding State Machines

- Some projects (Signal/Telegram) document their state machines well
  - All projects should do this
- Otherwise used Frida to hook signalling on an Android device
  - Logged offers, answers, candidates and tracks
  - Manipulated user interface

Understanding State Machines

- Occasional decompiled app with apktool to see when WebRTC natives were called
  - Necessary for apps with threading

```
setLocalDescription
setRemoteDescription
addIceCandidate
addTrack
removeTrack
setEnabled
```

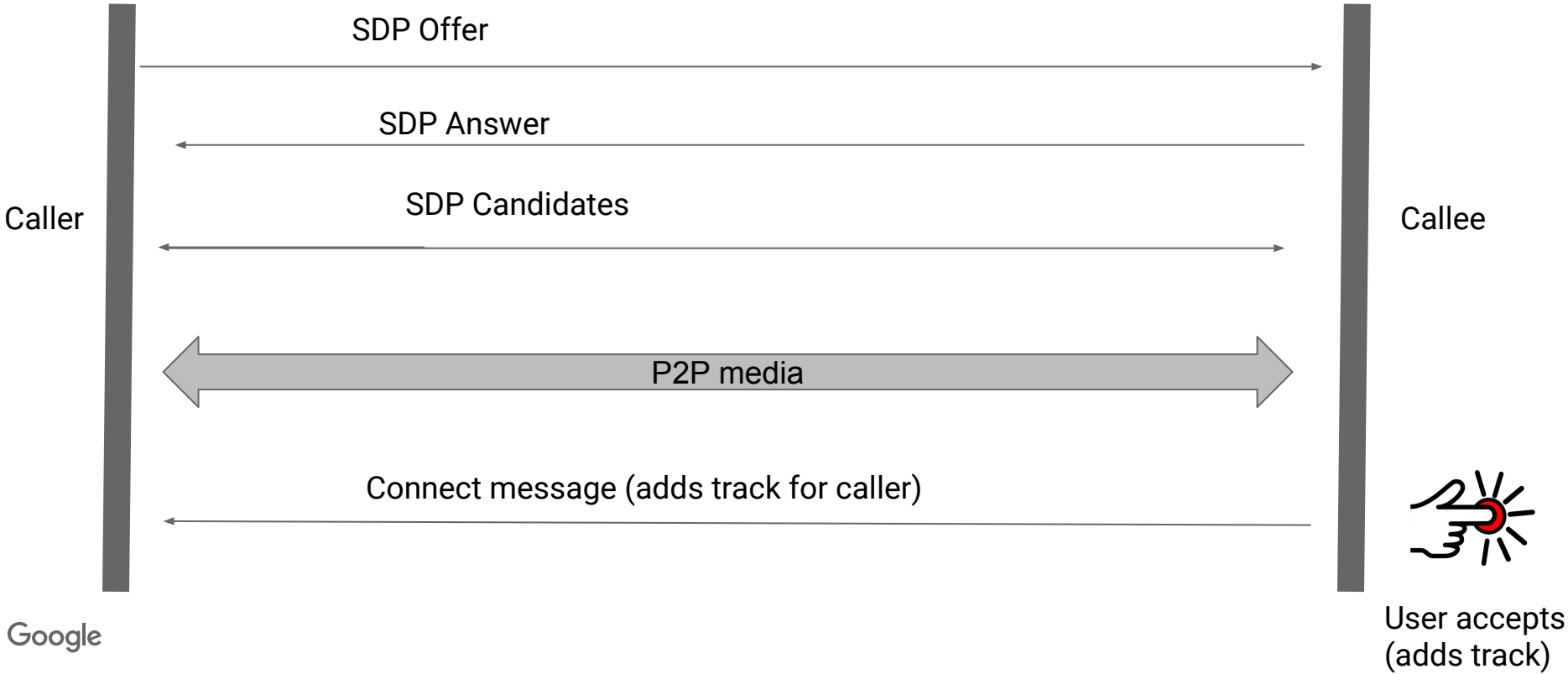
Possible Problems

- Send extra messages
- Drop messages
- Send messages in wrong order
- Send messages in wrong direction
- ‘Secret’ message types

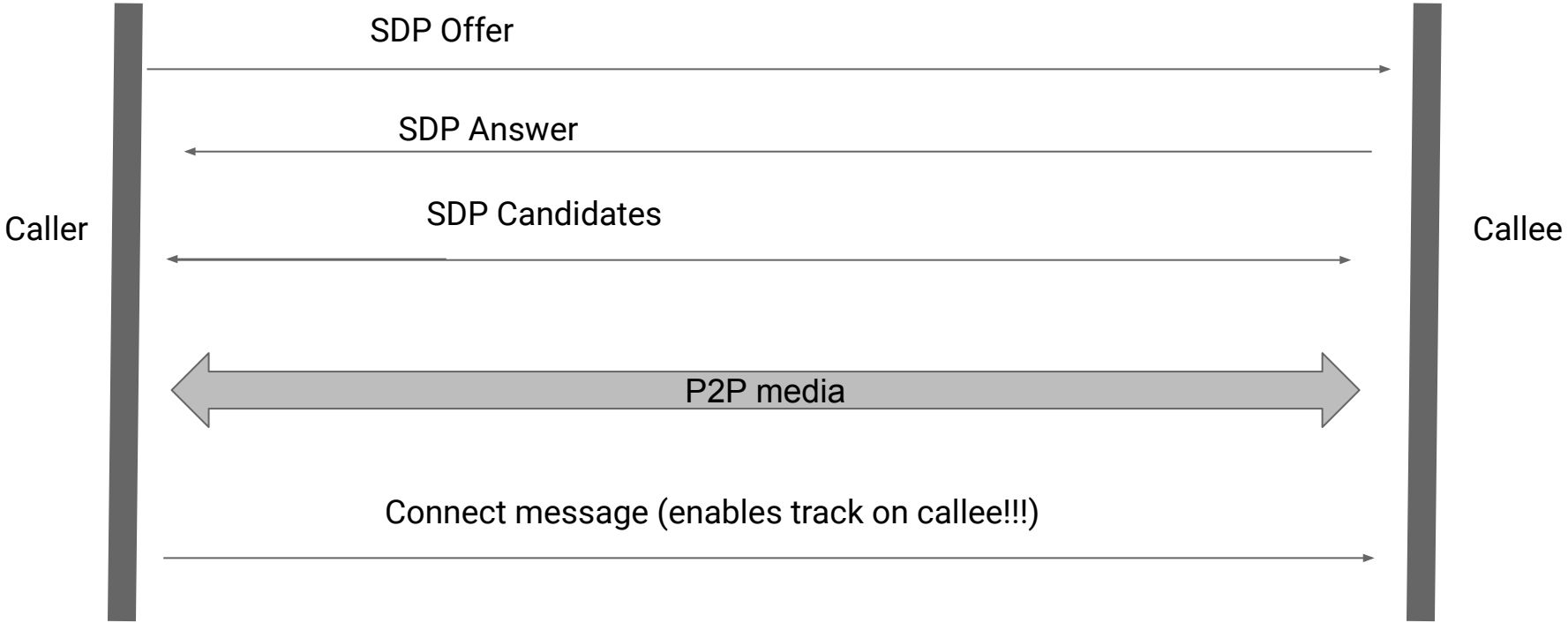
Signal Messenger and Facebook Messenger Vulnerabilities

- Signal vulnerability reported and fixed in 2019
- Root cause is confusion between caller and callee state
- Facebook Messenger vulnerability reported and fixed in 2020
- Similar root cause involving state mismanagement
- Both allow audio to be transmitted without consent

Logic Vulnerability Example (Signal)



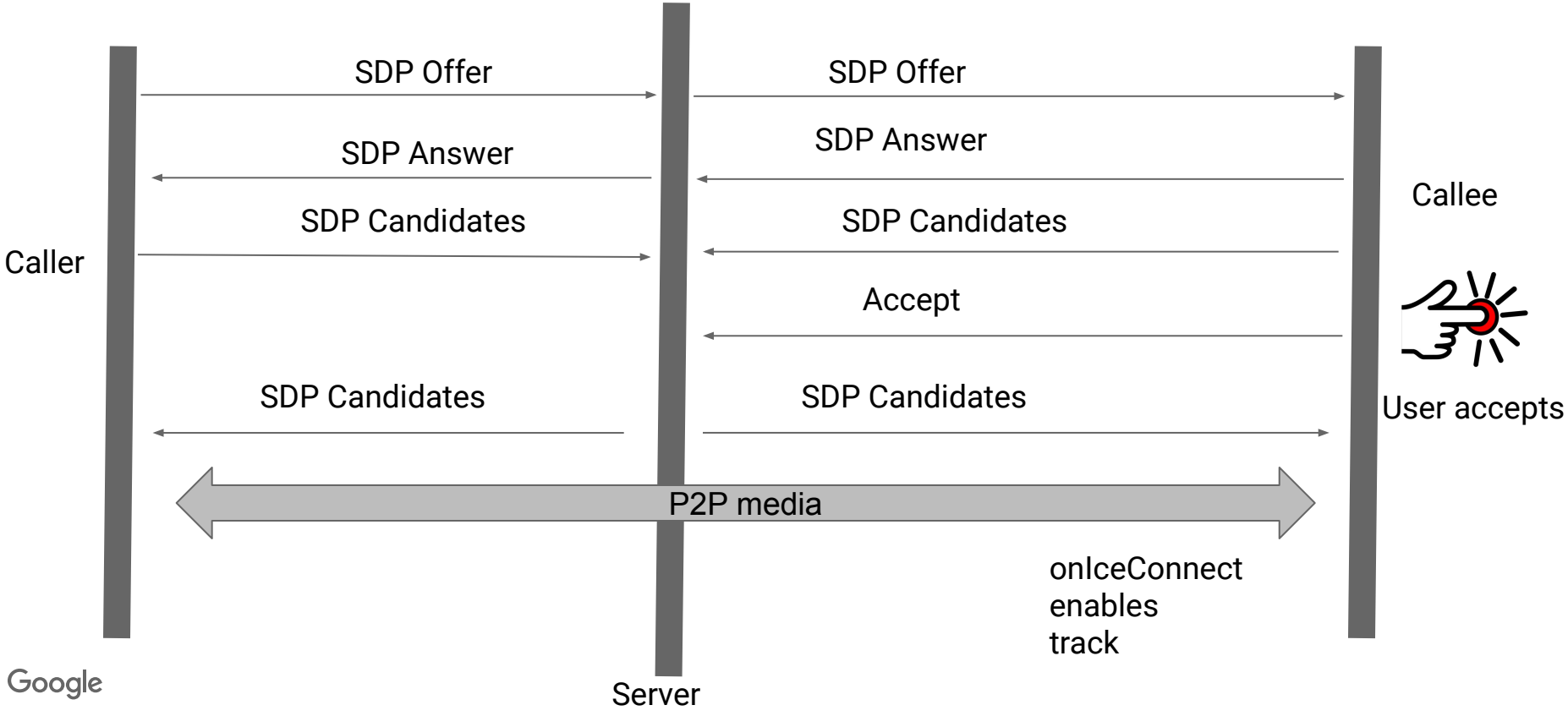
Logic Vulnerability Example (Signal)



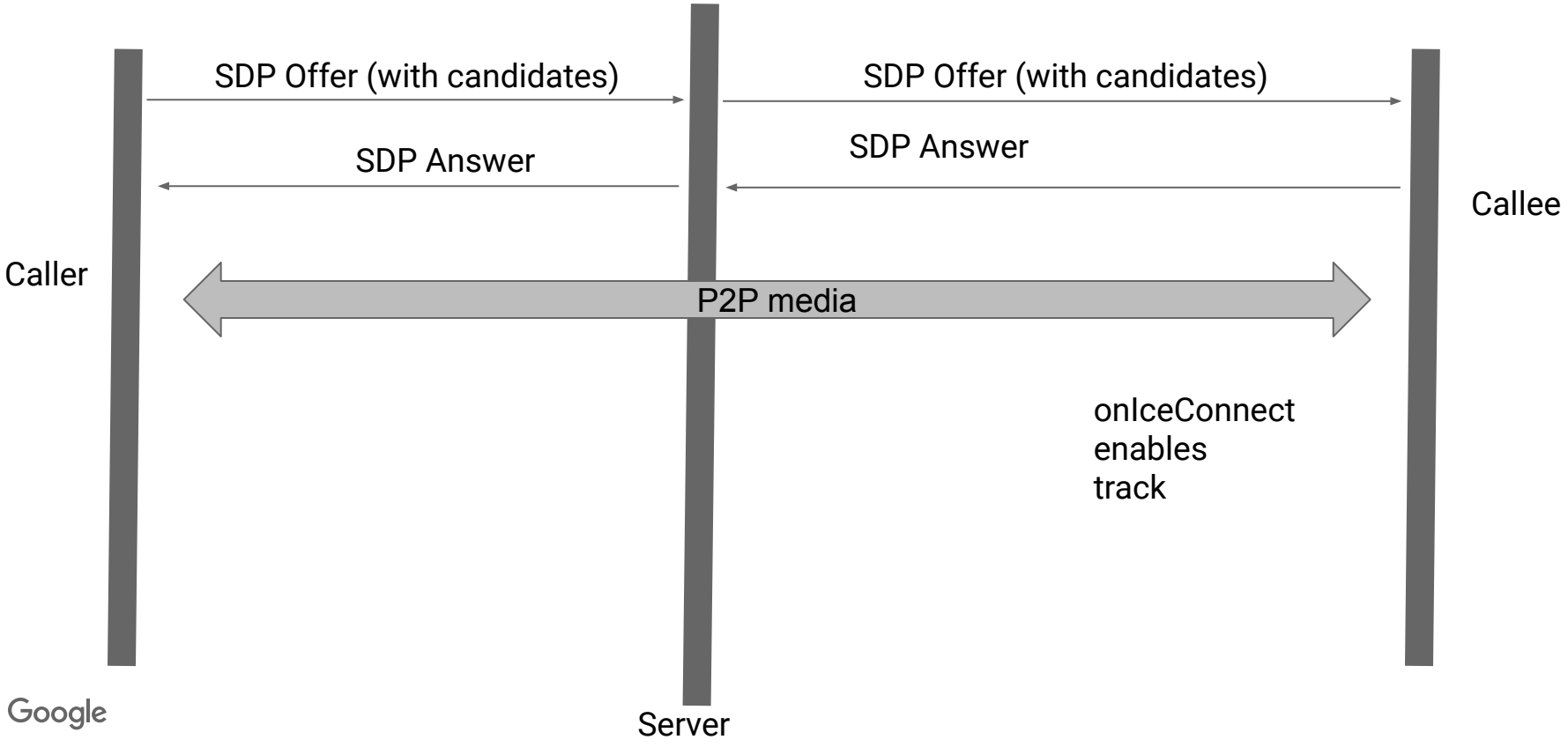
JioChat/Mocha Vulnerability

- JioChat and Mocha had very similar vulnerabilities, reported and fixed in 2020
- Root cause is not understanding that offers and answers can contain candidates
- Allowed audio and video to be transmitted without consent

Logic Vulnerability Example (JioChat)



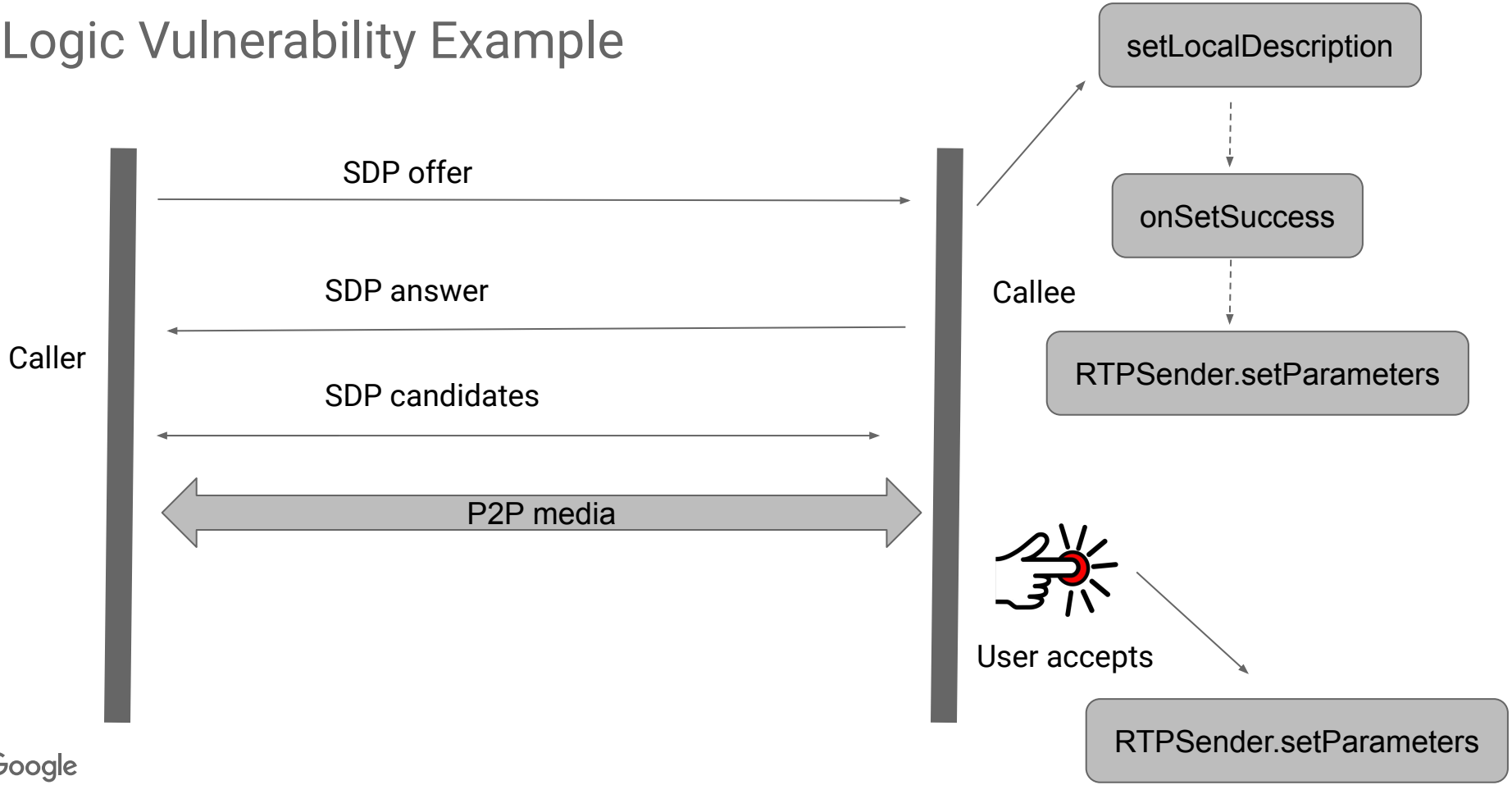
Logic Vulnerability Example (JioChat)



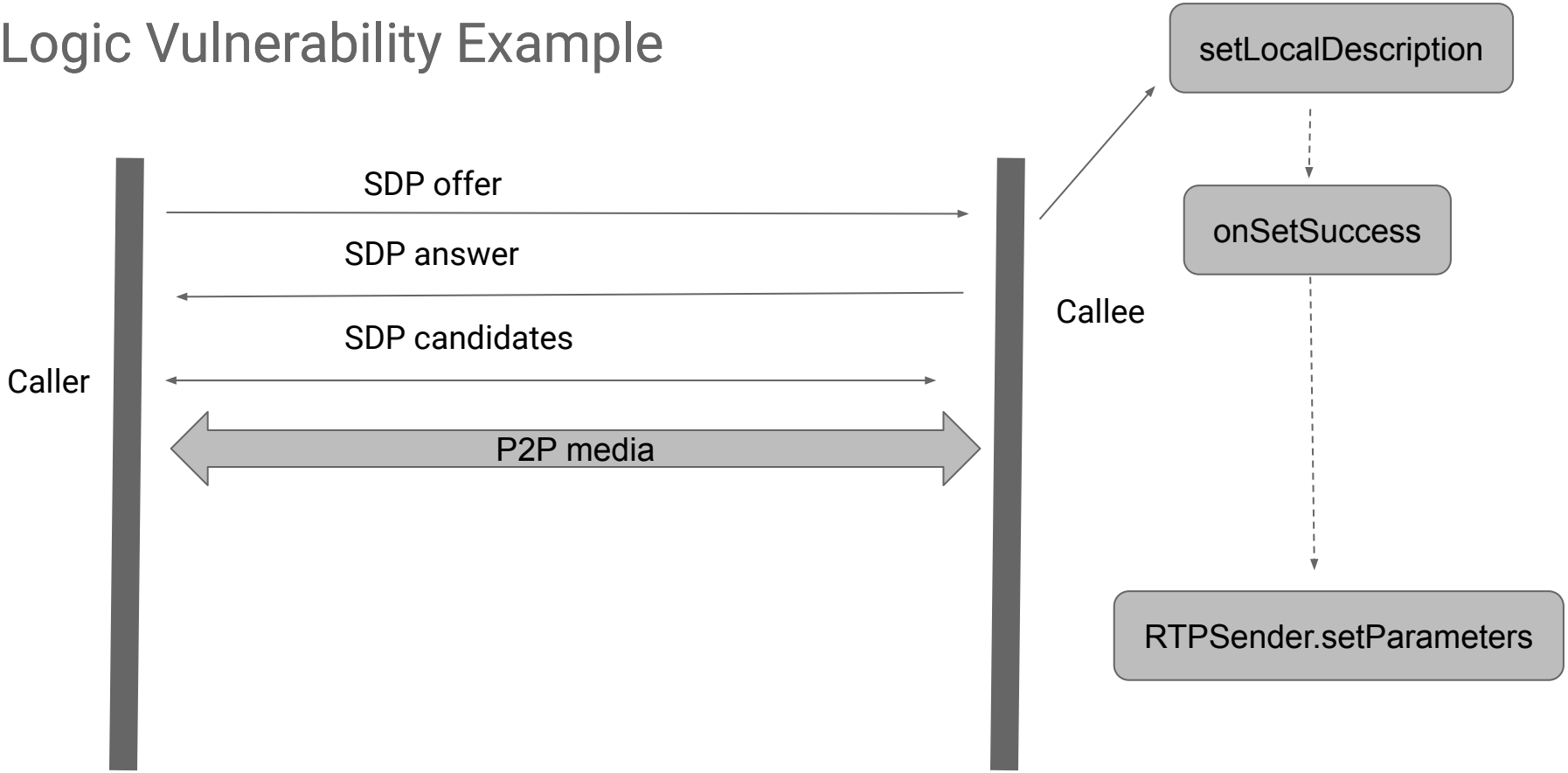
Google Duo Vulnerability

- Found and fixed in 2020
- Root cause is incorrect asynchronous logic (race condition)
- Allowed a few frames of video to be transmitted without consent

Logic Vulnerability Example



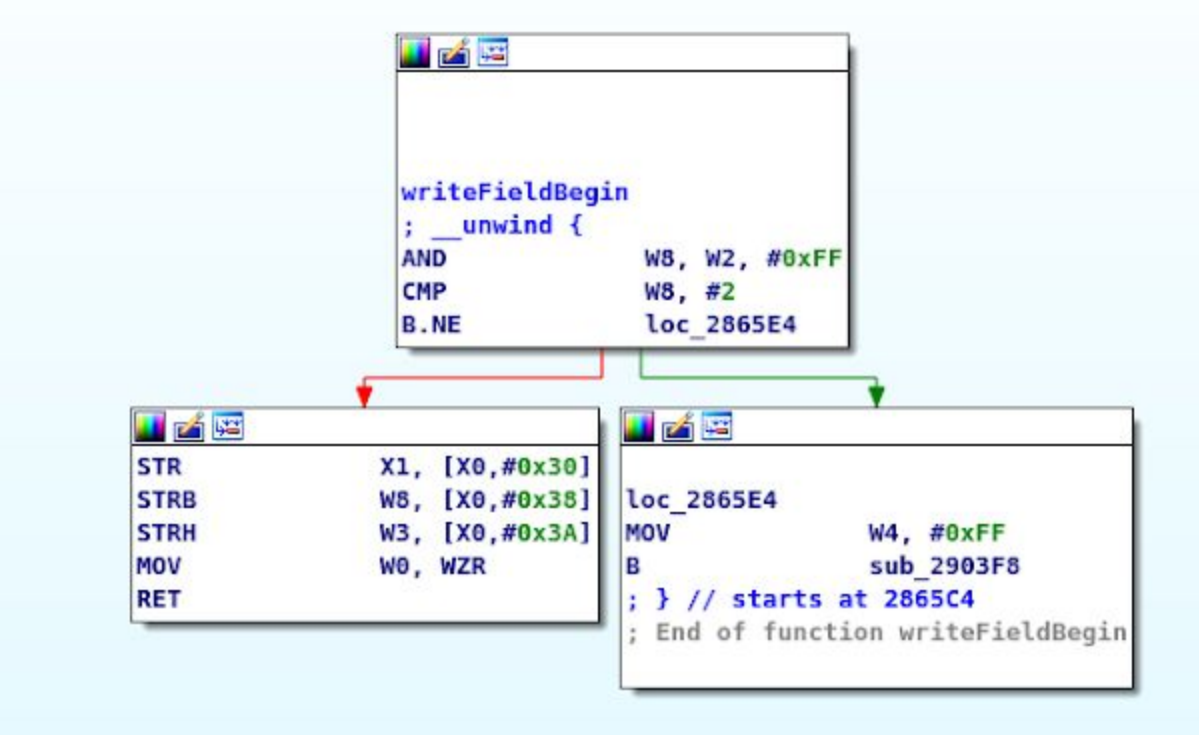
Logic Vulnerability Example



Testing Vulnerabilities

- Recompiled open source apps
- Otherwise, used Frida to change the state machine call flow
- This was painful
- Required extra step of including fbthrift-py for Facebook Messenger

Decoding fb-thrift



Decoding fb-thrift

```
add_uri
; __unwind {
ADRL            X1, aUri ; "uri"
MOV             W2, #8
MOV             W3, #2
MOV             X0, X19
ADD             X20, X20, #4
RET
; } // starts at 2A4DA8
; End of function add_uri
```

```
write_extmap

var_20= -0x20
var_10= -0x10

; FUNCTION CHUNK AT .text:00000000001E3ACC SIZE 00000008 BYTES

; __unwind {
STP             X21, X20, [SP,#var_20]!
STP             X19, X30, [SP,#0x20+var_10]
MOV             X19, X1
ADRP            X1, #aExtmap@PAGE ; "Extmap"
MOV             X20, X0
ADD             X1, X1, #aExtmap@PAGEOFF ; "Extmap"
BL              sub_2936DC
ADRP            X1, #(aUnexpectedMess_0+0x13)@PAGE ; "id"
MOV             W21, W0
ADD             X1, X1, #(aUnexpectedMess_0+0x13)@PAGEOFF ; "id"
BL              write_protocol
LDRB            W8, [X20,#9]
ADD             W21, W21, W0
CBZ             W8, loc_29C690
```

```
BL              add_uri
BL              writeFieldBegin
ADD             W21, W0, W21
BL              sub_188664
BL              sub_286648
ADD             W21, W21, W0
```

Decoding fb-thrift

```
struct Extmap{
        1: i32 id
        2: optional i32 uri
}
```

Decoding fb-thrift

```
struct P2PMessageRequest{

    1: WebrtcMessageHeader header
    2: WebrtcMessagePayload payload
}

struct WebrtcMessageHeader{

    1: optional i32 protocolVersion
    2: optional i64 messageId
    3: optional i64 callId
    4: optional i64 sender
    5: optional i64 receiver
    6: optional i64 capabilities
    7: optional i32  payloadType
    8: byte retryCount
    9: bool pranswerSupported
    10: optional i32 ackMessageType
```

Root Causes

- Lack of knowledge of vulnerability type
  - Poor state machine testing
- Misunderstanding WebRTC features
- Setting up P2P connection before call is answered

Conclusions

- Video conferencing signaling state bugs are common
- Some problems can be attributed to WebRTC design and documentation, but many can’t
- Developers should be careful when designing calling state machines
- This is an area that needs more research

Questions

http://googleprojectzero.blogspot.com/

@natashenka
natashenka@google.com