Make iOS App more Robust and Security through Fuzzing

Wei Wang & Zhaowei Wang 2016-10-14

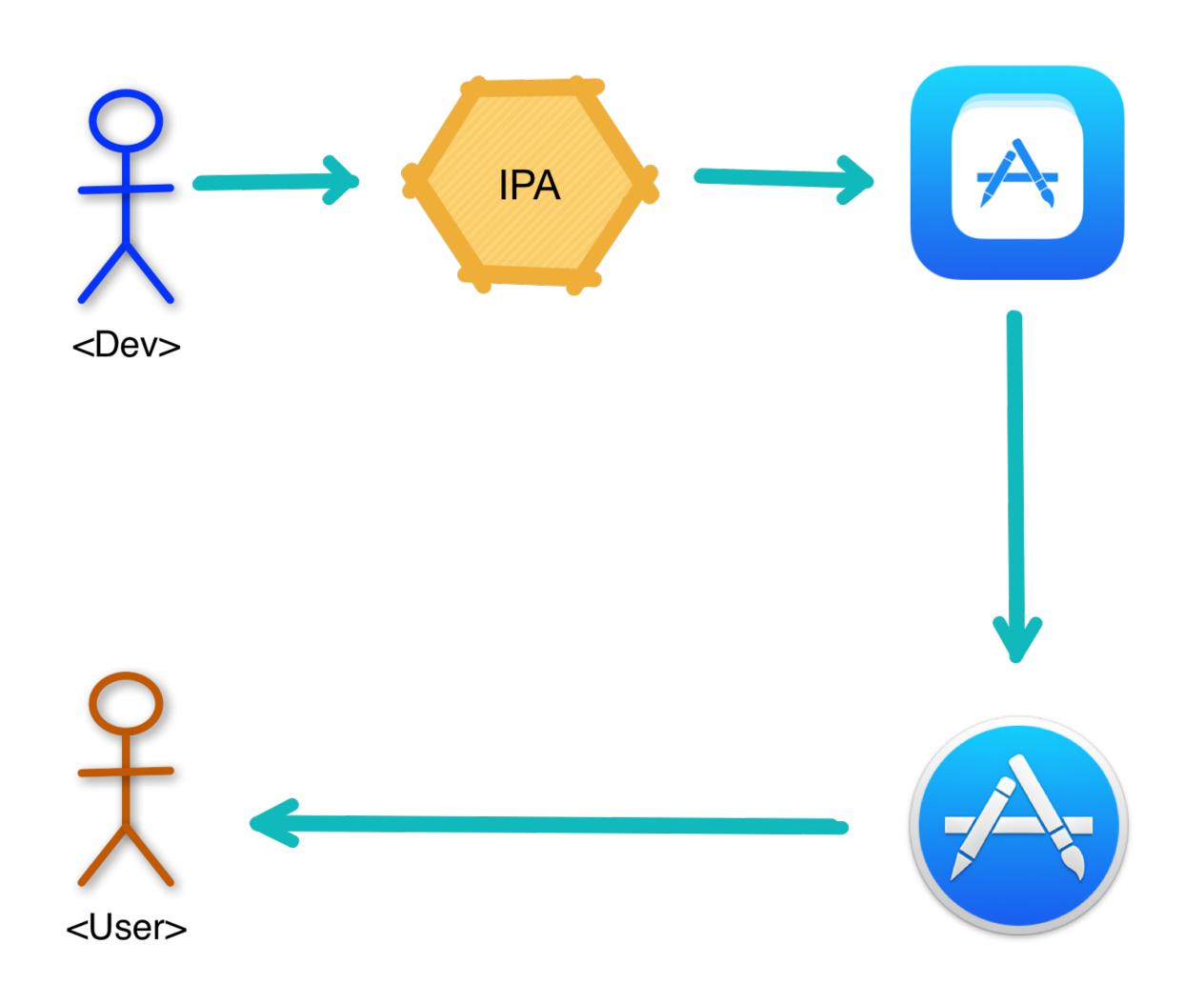
About us

- ID: Proteas, Shrek_wzw
- Working at: Qihoo 360 Nirvan Team
- Focused on: iOS and OS X Security Research
- Twitter: @ProteasWang, @Shrek_wzw

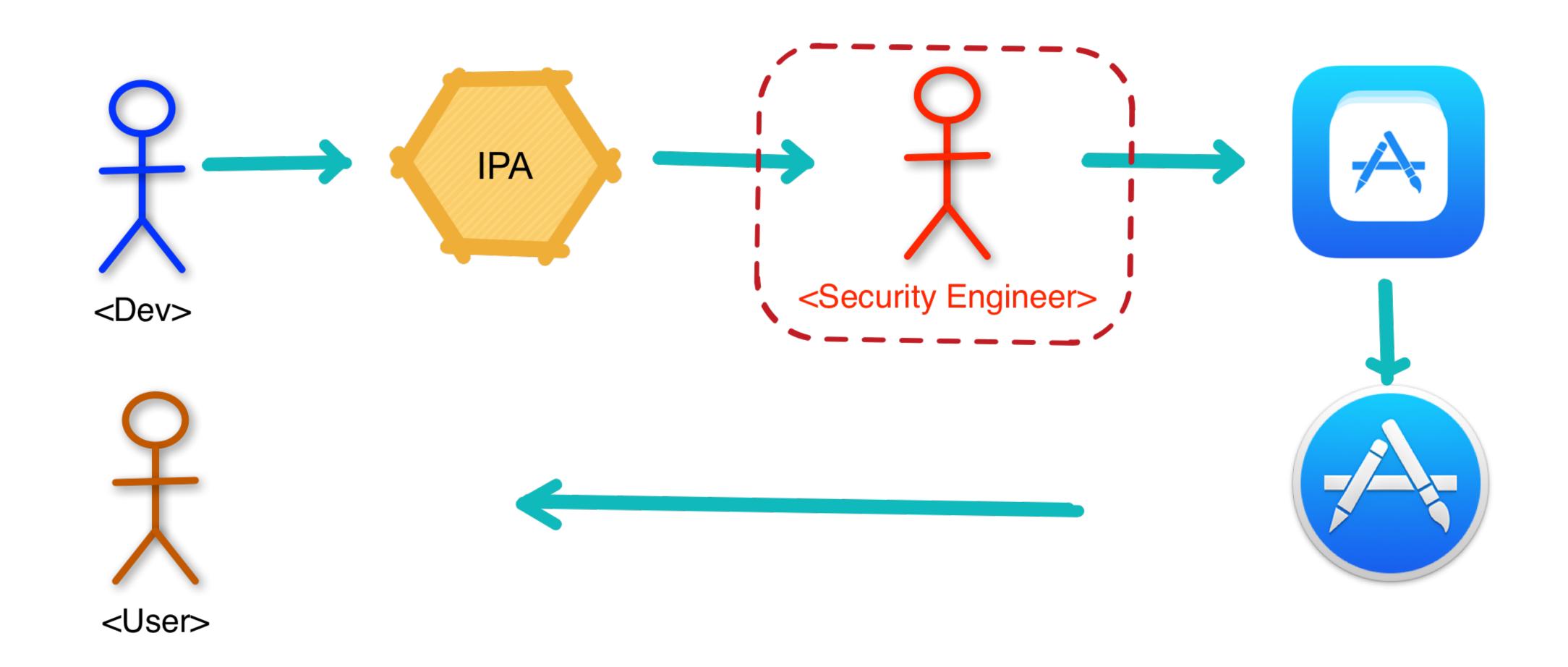
Agenda

- Status of iOS App Security Development Lifecycle
- Why Using AFL to Fuzz App during Development
- Port AFL to iOS
- Characteristics and Attacking Surfaces of iOS App
- Fuzz iOS App
- Fuzz 3rd Party Libraries

- There are about 2 million Apps on Apple AppStore as of June 2016
- Most developed by individual developers or small companies
- For most of those developers or companies, there is no security engineer to protect the Apps
- So the SDL may be like this:



- For companies with iOS security engineers
- Developers submit the App to security engineers first
- Security engineers assess the App using the blackbox way
- After security assessment, the App is submitted to iTunes Connect

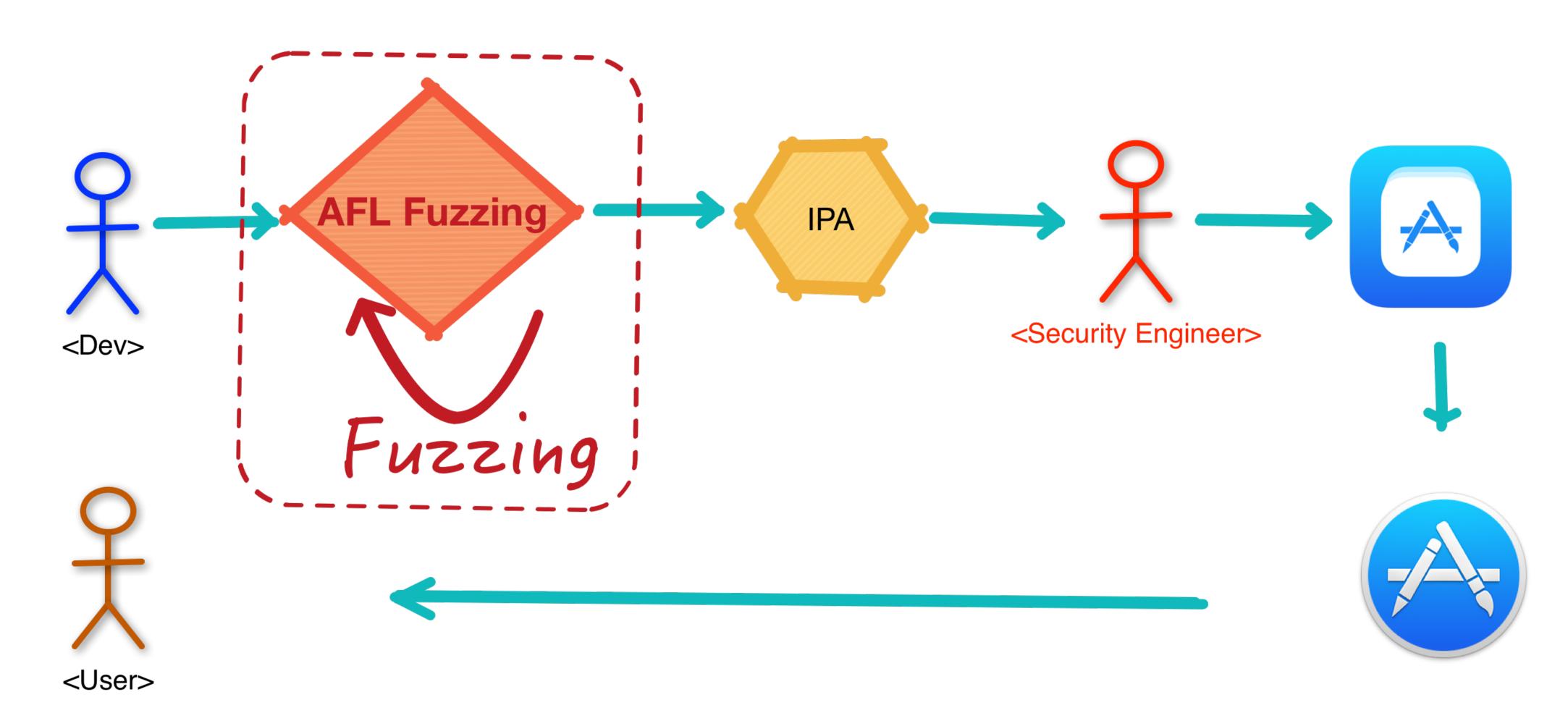


Why Using AFL to Fuzz App during Development

- Bugs should be found as earlier as possible
- We have the source code of our App, this is import for using AFL
- AFL is easy to config and easy to use
- Can be integrated with CI(Continuous Integration)
- When run unit tests with CI, should also run AFL fuzzing

Why Using AFL to Fuzz App during Development

SDL with AFL



Port AFL to iOS - Port Codes

- Change the API used to create shared memory: shmget() —> shm_open()
- All other changes are for this
- Get the code from my repo: https://github.com/Proteas/afl/tree/ios-afl-clang-fast
- This method is also compatible with AFL 2.35b(currently latest version)

Port AFL to iOS - Build Clang

- Before building AFL, should first build clang
- Get code from: http://opensource.apple.com/
- Using Apple's clang is for compatibility when building Xcode projects
- After building clang, add the result bin dir to PATH
- export PATH="\${CLANG_DIST_DIR}/bin:\${PATH}"

Port AFL to iOS - Build AFL

- Set Env param: export AFL_NO_X86=1
- Cross-compile targets:
 - afl-fuzz, afl-showmap, afl-tmin, afl-gotcpu, afl-analyze
 - ./llvm mode/afl-llvm-rt.o
- Native compile: afl-clang-fast
- Use lipo to merge the build results, then can fuzz macOS and iOS App using the same toolchain

Port AFL to iOS - Tips and Tricks

- Currently AFL-iOS can only fuzz arm64 binary
- Because AFL using C++11's thread local storage, the App deployment target should be >= 9.0
- Because of Jetsam, should limit the memory usage
- ./afl-fuzz -i \${TEST_CASES} -o \${RESULT_DIR} -m 80M \${TARGET_APP} @@

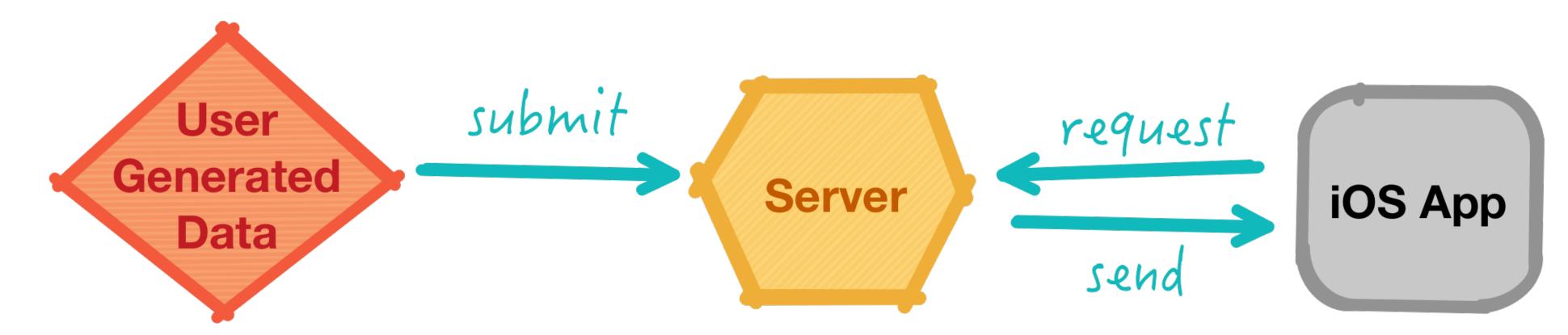
Port AFL to iOS



- Most of the Apps only communicate with their own server
- Requires HTTPS connections for iOS Apps by the end of this year
- The remote attacking surface is narrow relatively after using HTTPS
- If there are certificate validation vulnerabilities or config mistakes in iOS App
- Traditional remote attacking surfaces will be back

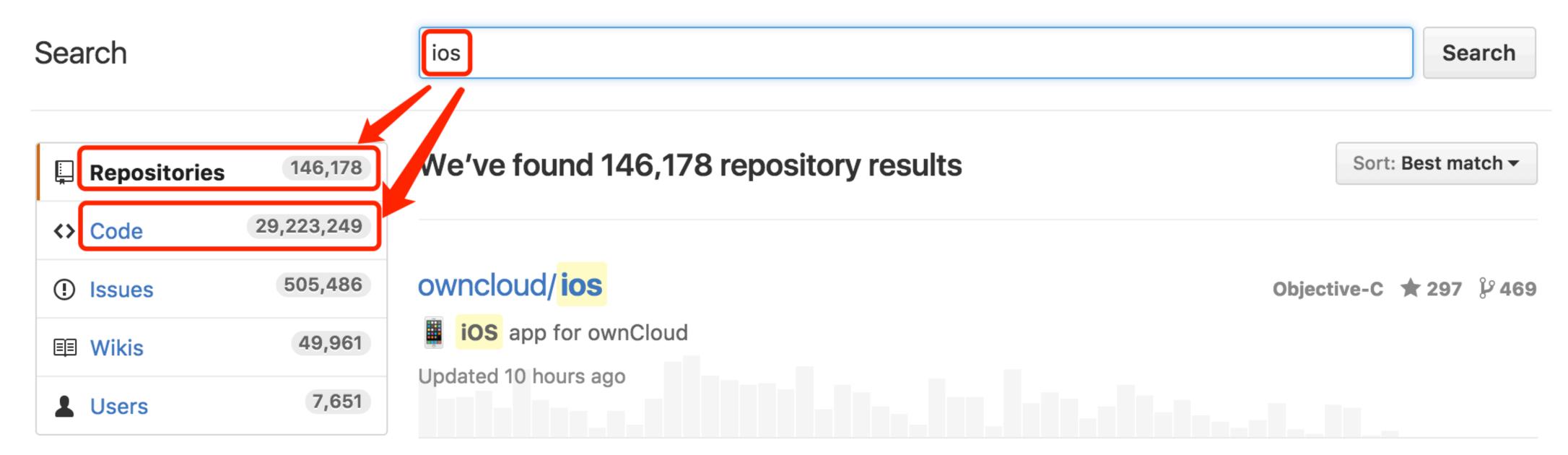
- Most of the communication protocol of iOS App based on:
 - JSON
 - XML
 - Protocol Buffers
- If can be hijacked, the type-confusion is a kind of issue
- We should validate the input data immediately after receiving it:
 - JSON Schema
 - XML Schema
- Not allow any malformed data come into our App

- If there are no certificate validation issues
- We should pay more attention to this kind of Apps:



- Apps like: iMessage, Twitter, Facebook, Dropbox, etc
- Different Apps have different attack surfaces depends on how it processing the user generated data

- There are lots of iOS libraries on Github
- Writing iOS App is more and more like "stacking wood"
- Search "ios" on Github(1476435790):



- Sharing is great
- There are so many codes on Github
- Some are shared by companies with fully testing or security assessment
- Some are written by individual developers
- Some are just demos
- We should do something to make the code more security
- Using AFL is a practical choice

- What libraries are more suitable for fuzzing with AFL?
 - Parsers: JSON Parser, XML Parser, DSLs Parser
 - Video & Audio Encoder and Decoder
 - Image Encoder and Decoder
 - Archive related libraries

•

- Introduce practical steps about how to fuzz our own codes
- We will use an open source app to demonstrate all the process
- The key point here is: the target function to be fuzzed is coupled seriously
- So the target function can't be fuzzed on macOS
- We need to do fuzzing on iDevice

- The demo App: https://github.com/songfei/ArchiveALL
- Function of ArchiveALL is unarchiving rar, Izma, zip on iOS
- Function code is seriously coupled with the demo app
- It is not easy to extract the specific function(for example: unrar)

- clone the repository, and create a new branch: AFL-Fuzz
- check out the newly created branch
- copy main.m to main-normal.m
- create file: main-afl.m
- add following contents to main-afl.m:

main-afl.m

```
#import "SFArchiveFileItem.h"
#import "SF7zArchive.h"
#import "SFRarArchive.h"
#import "SFZipArchive.h"
   DoFuzzing(int argc, char * argv[]);
int FuzzArchive(SFBaseArchive *archive);
int FuzzUnzip(NSString *fileName);
int FuzzUnrar(NSString *fileName);
   FuzzUn7z(NSString *fileName);
int main(int argc, char * argv[])
   @autoreleasepool {
        return DoFuzzing(argc, argv);
```

```
DoFuzzing(int argc, char * argv[])
if (argc != 3) {
   NSLog(@"Usage: ./ArchiveAll 0|1|2 ./test.zip");
    return -1;
NSFileManager *fileManager = [NSFileManager defaultManager];
NSString *inputFileName = [NSString stringWithUTF8String:argv[2]];
if (![fileManager fileExistsAtPath:inputFileName]) {
   NSLog(@"%s: file not exist", FUNCTION );
    return -1;
// Fuzz Type
int type = 0;
NSString *inputType = [NSString stringWithUTF8String:argv[1]];
type = (int)[inputType integerValue];
if (type == 0) {
    return FuzzUnzip(inputFileName);
else if (type == 1) {
    return FuzzUnrar(inputFileName);
else if (type == 2) {
    return FuzzUn7z(inputFileName);
   NSLog(@"error fuzz type");
    return -1;
```

• Edit main.m:

```
#ifdef AFL_FUZZ
     #include "./main-afl.m"
#else
     #include "./main-normal.m"
#endif
```

Key point of above code is using macro to control the entry of the App

• Create afl-ios.xcconfig to config build params for AFL building

```
ONLY ACTIVE ARCH = NO
ARCHS = arm64
VALID ARCHS = arm64
ENABLE BITCODE = NO
OTHER CFLAGS = "-DAFL FUZZ=1"
OTHER CPLUSPLUSFLAGS = "-DAFL FUZZ=1"
OTHER LDFLAGS = $(PATH TO AFL DIST)/afl/afl-llvm-rt.o
```

Build

```
AFL ROOT DIR="TODO"
export AFL PATH="${AFL ROOT DIR}"
export PATH="${AFL ROOT DIR}:${PATH}"
rm -rf "./Build"
xcodebuild \
    CC="${AFL ROOT DIR}/afl-clang-fast" \
    CXX="${AFL ROOT DIR}/afl-clang-fast++" \
    -project "ArchiveALL.xcodeproj" \
    -target "ArchiveALL" \
    -xcconfig "./afl-ios.xcconfig" \
    -configuration "Debug"
```

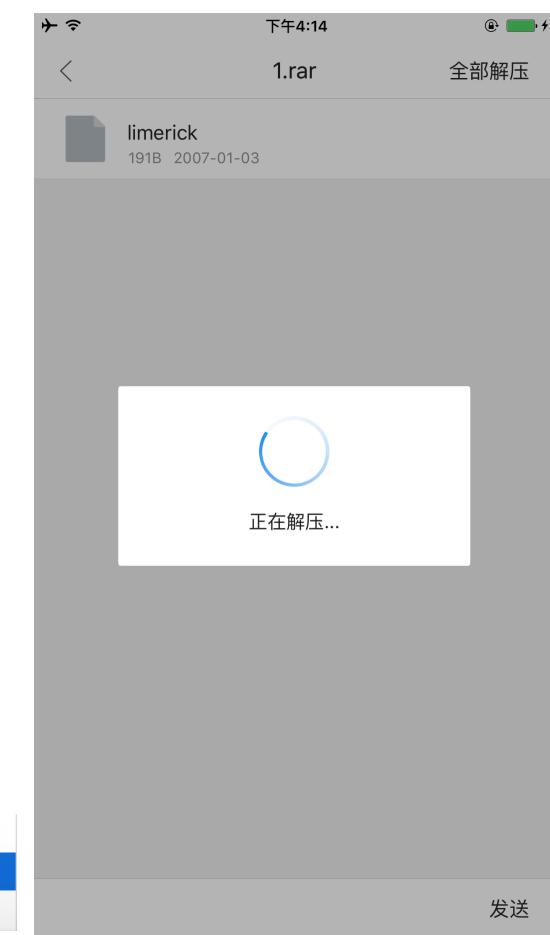
- Run it on iDevice
- Fuzzing Unrar

```
american fuzzy lop 2.13b (ArchiveALL)
                                                       overall results
process timing
                                                       cycles done: 0
      run time : 0 days, 0 hrs, 0 min, 39 sec
                                                       total paths: 41
 last new path: 0 days, 0 hrs, 0 min, 0 sec
                                                      uniq crashes: 0
last uniq crash : none seen yet
 last uniq hang: 0 days, 0 hrs, 0 min, 5 sec
                                                        uniq hangs: 1
cycle progress -
                                      map coverage
                                        map density : 1276 (1.95%)
now processing: 0 (0.00%)
                                     count coverage : 1.82 bits/tuple
paths timed out : 0 (0.00%)
stage progress -
                                      findings in depth -
now trying : calibration
                                     favored paths: 1 (2.44%)
stage execs : 7/10 (70.00%)
                                      new edges on: 34 (82.93%)
total execs : 714
                                     total crashes: 0 (0 unique)
                                       total hangs : 1 (1 unique)
execrspeed = 27.08/sec (slow!)
fuzzing strategy yields
                                                      path geometry
 bit flips: 0/0, 0/0, 0/0
                                                        levels : 2
byte flips: 0/0, 0/0, 0/0
                                                       pending: 41
arithmetics : 0/0, 0/0, 0/0
                                                      pend fav: 1
 known ints: 0/0, 0/0, 0/0
                                                     own finds: 39
                                                      imported : n/a
dictionary: 0/0, 0/0, 0/0
     havoc: 0/0, 0/0
                                                      variable : 0
      trim: 0.00%/99, n/a
                                                                 [cpu: 76%]
```

- As the image shows: In less than 1 minute, we got a DoS
- It can also DoS the App used this library.
- QQ Browser v6.7.2.2345
- All the following fuzzers and fuzzing results can be downloaded from:
- https://github.com/Proteas/fuzzers_based_on_afl

- QQ Browser v6.7.2.2345
- unrar DoS
- CPU Usage: 99.4%
- The GUI is freezing
- Need to kill the app

34,148	com.apple.dt.ins	root	0	2	1.14 MiB	664.89 MiB arm64	00.56069
34,156	mttlite	mobile	99.4	11	65.02 MiB	899.41 MiB arm64	1:35.719
34,158	com.apple.dt.ins	root	4.1	5	2.32 MiB	666.88 MiB arm64	secur 10/4te7/5:214:0:18



- With the doc of AFL and the previous information
- You can build your own fuzzers based on AFL
- Although we can fuzz on iOS, we prefer to do fuzzing on OS X
- The following will show some fuzzers and analysis some of the fuzzing results

- · ZXingObjC v3.1.0
- An Objective-C Port of ZXing
- Out-of-Bounds Read
- 140+ hangs(infinite loop)

- · Unrar4iOS 1.0.0 6c90561
- heap overflow: -[Unrar4iOS extractStream:]
- heap overflow in C, but ObjC object may be overwritten
- Unrar4iOS.mm

```
// alloc buffer
NSLog(@"buffer size: %lu", length);
UInt8 *buffer = (UInt8 *)malloc(length * sizeof(UInt8));
.....
// copy data to buffer
NSLog(@"memcpy size: %ld", P2);
memcpy(*buffer, (UInt8 *)P1, P2);
```

unrar[12069:2318258] buffer size: 191 unrar[12069:2318258] memcpy size: 214

- · opus codec
- Audio Codecs
- Versions
 - flac-1.3.0
 - libogg-1.3.2
 - opus-1.1
 - opus-tools-0.1.9
- Analysis the fuzzing results, you will find: stack overflows, integer overflows, ...

- · opus codec encode wav
- Some are exploitable
- Floating point exception: 8
- AddressSanitizer failed to allocate 0xfffffffffe0004 bytes
- AddressSanitizer: **stack-overflow** on address 0x7fff5b3ceb88
- AddressSanitizer: heap-buffer-overflow on address 0x00014ad3c800

```
american fuzzy lop 2.13b (opusenc)
                                                  — overall results —
process timing -
      run time : 9 days, 4 hrs, 12 min, 54 sec
                                                     cycles done : 0
 last new path: 0 days, 3 hrs, 41 min, 53 sec
                                                     total paths: 1539
last uniq crash : 0 days, 6 hrs, 48 min, 41 sec
                                                    uniq crashes : 34
last uniq hang: 0 days, 3 hrs, 15 min, 37 sec
                                                      uniq hangs : 45
cycle progress -
                                    map coverage —
                                      map density : 2893 (4.41%)
now processing: 959* (62.31%)
paths timed out : 0 (0.00%)
                                  | count coverage : 7.03 bits/tuple
                                  findings in depth —
stage progress ----
now trying : havoc
                                  | favored paths : 47 (3.05%)
stage execs: 151k/160k (94.42%) | new edges on: 89 (5.78%)
                            | total crashes : 43.1k (34 unique)
total execs : 7.32M
exec speed : 12.26/sec (zzzz...) | total hangs : 4061 (45 unique)
fuzzing strategy yields ----
                                                    path geometry
 bit flips: 265/177k, 72/177k, 36/177k
                                                      levels : 4
byte flips: 8/22.2k, 3/21.7k, 1/21.6k
                                                     pending: 1496
arithmetics : 210/1.21M, 4/843k, 0/104k
                                                    pend fav : 25
known ints: 29/100k, 70/500k, 25/933k
                                                   own finds : 1538
dictionary: 0/0, 0/0, 13/264k
                                                    imported : n/a
                                                    variable : 228
     havoc: 822/2.57M, 0/0
      trim: 20.33%/10.8k, 2.25%
                                                              Cpu: 73%
```

•

- · opus codec encode aif
- Some are exploitable
- AddressSanitizer: stack-overflow on address 0x7ffed2b175d8
- AddressSanitizer: **heap-buffer-overflow** on address 0x62e000000000
- AddressSanitizer failed to allocate 0xfffffffffe0004 bytes
- AddressSanitizer: SEGV on unknown address 0x62de00001dac
- AddressSanitizer: unknown-crash on address 0xffffff504c0d420

```
american fuzzy lop 2.13b (opusenc)
                                                    - overall results -
process timing -
      run time : 9 days, 0 hrs, 33 min, 49 sec
                                                     cycles done : 0
 last new path : 1 days, 16 hrs, 45 min, 25 sec
                                                     total paths : 1256
last uniq crash : 6 days, 23 hrs, 6 min, 50 sec
                                                    uniq crashes : 29
last uniq hang: 6 days, 8 hrs, 43 min, 3 sec
                                                      uniq hangs : 123
cycle progress -
                                     map coverage -
                                       map density : 2904 (4.43%)
now processing : 593* (47.21%)
paths timed out : 0 (0.00%)
                                  | count coverage : 6.83 bits/tuple
                                  findings in depth —
stage progress ----
now trying : interest 32/8
                                  | favored paths : 58 (4.62%)
stage execs: 63.6k/123k (51.62%) | new edges on: 87 (6.93%)
total execs: 7.89M
                                  | total crashes : 40.9k (29 unique)
exec speed: 0.83/sec (zzzz...) | total hangs: 681 (123 unique)
fuzzing strategy yields -
                                                   path geometry
 bit flips: 279/273k, 71/273k, 40/273k
                                                      levels: 3
byte flips: 8/34.1k, 3/33.1k, 5/33.1k
                                                     pending: 1233
arithmetics: 168/1.85M, 9/1.29M, 1/141k
                                                    pend fav : 44
known ints: 28/155k, 77/762k, 16/1.32M
                                                    own finds: 1255
dictionary: 0/0, 0/0, 12/265k
                                                    imported : n/a
     havoc: 567/1.09M, 0/0
                                                    variable : 190
      trim: 47.00%/17.0k, 3.06%
                                                               [cpu: 52%]
```

•

- · opus codec encode flac
- AddressSanitizer: SEGV on unknown address
 0x0000000000000
- Floating point exception: 8
- AddressSanitizer: SEGV ??:0 oi_strncasecmp

```
•
```

```
american fuzzy lop 2.13b (opusenc)
                                                     – overall results –
process timing
      run time : 9 days, 0 hrs, 31 min, 48 sec
                                                      cycles done: 0
 last new path : 1 days, 7 hrs, 56 min, 4 sec
                                                      total paths : 972
last uniq crash : 8 days, 16 hrs, 13 min, 5 sec
                                                     uniq crashes : 19
                                                      uniq hangs : 0
 last uniq hang : none seen yet
cycle progress ----
                                   — map coverage →
now processing : 297 (30.56%)
                                       map density: 3158 (4.82%)
paths timed out : 0 (0.00%)
                                   | count coverage : 5.23 bits/tuple
stage progress ----
                                  findings in depth —
now trying : auto extras (over)
                                   | favored paths : 152 (15.64%)
stage execs : 366k/435k (84.23%)
                                     new edges on: 208 (21.40%)
total execs : 20.3M
                                   | total crashes : 8423 (19 unique)
 exec speed: 3.15/sec (zzzz...)
                                      total hangs : 0 (0 unique)
fuzzing strategy yields -
                                                   ─ path geometry
 bit flips: 148/3.67M, 76/3.67M, 58/3.67M
                                                       levels: 3
byte flips: 6/459k, 3/26.9k, 2/27.3k
                                                      pending: 919
arithmetics: 161/1.49M, 18/1.32M, 1/1.20M
                                                     pend fav : 123
 known ints: 20/87.6k, 68/354k, 73/610k
                                                    own finds: 970
                                                     imported : n/a
 dictionary: 0/0, 0/0, 60/496k
     havoc: 271/2.80M, 0/0
                                                     variable : 122
      trim: 1.07%/57.0k, 94.13%
                                                                [cpu: 86%]
```

- · lame mp3 encoder 3.99.5
- AddressSanitizer: SEGV on unknown address 0x60bffff05b38
- AddressSanitizer: SEGV ??:0 fill_buffer
- AddressSanitizer: SEGV on unknown address 0x000000000000
- AddressSanitizer: **heap-buffer-overflow** on address 0x60c0000bd3c
- AddressSanitizer: heap-buffer-overflow ??:0 fill_buffer

```
american fuzzy lop 2.13b (lame)
process timing
                                                      overall results
      run time: 0 days, 0 hrs, 55 min, 18 sec
                                                      cycles done : 0
 last new path: 0 days, 0 hrs, 0 min, 54 sec
                                                      total paths: 971
last uniq crash : 0 days, 0 hrs, 19 min, 13 sec
                                                     uniq crashes: 24
last uniq hang : none seen yet
                                                       uniq hangs : 0
cycle progress —
                                     map coverage
                                        map density: 2667 (4.07%)
now processing : 32 (3.30%)
paths timed out : 0 (0.00%)
                                     count coverage : 4.50 bits/tuple
                                      findings in depth —
stage progress ——
                                     favored paths : 65 (6.69%)
now trying: interest 16/8
stage execs: 8166/21.1k (38.77%)
                                     new edges on: 89 (9.17%)
                                     total crashes: 1115 (24 unique)
total execs : 462k
                                       total hangs : 0 (0 unique)
exec speed: 110.8/sec
fuzzing strategy yields
                                                     path geometry
 bit flips: 345/12.3k, 63/12.3k, 29/12.3k
                                                       levels: 3
byte flips: 4/1536, 3/1528, 7/1512
                                                      pending : 964
arithmetics: 108/85.8k, 17/59.3k, 3/7633
                                                     pend fav : 59
known ints: 17/7137, 47/18.9k, 31/34.7k
                                                    own finds: 970
dictionary: 0/0, 0/0, 7/3230
                                                     imported : n/a
     havoc: 308/182k, 0/0
                                                     variable : 109
      trim : 72.80%/812, 0.00%
                                                                [cpu: 31%]
```

•

- KxMovie(ffmpeg decoder) -2c5324b0
- iOS movie player based on ffmpeg
- Fuzz results: decode flv
- You could clone the fuzzer and continue to fuzz other formats

```
american fuzzy lop 2.13b (ffmpeg-ios-decoder)
                                                      overall results -
process timing
      run time: 0 days, 15 hrs, 47 min, 43 sec
                                                      cycles done : 0
 last new path: 0 days, 0 hrs, 4 min, 33 sec
                                                      total paths: 1003
last uniq crash : 0 days, 0 hrs, 13 min, 38 sec
                                                     uniq crashes: 35
last uniq hang: 0 days, 15 hrs, 40 min, 53 sec
                                                       uniq hangs : 1
cycle progress ————
                                     map coverage
                                       map density : 5259 (8.02%)
now processing: 0 (0.00%)
                                    count coverage : 3.30 bits/tuple
paths timed out : 0 (0.00%)
                                     findings in depth —
stage progress —
now trying : interest 16/8
                                     favored paths: 1 (0.10%)
stage execs: 12.0k/160k (7.46%)
                                     new edges on: 160 (15.95%)
total execs : 546k
                                     total crashes : 558 (35 unique)
                                      total hangs : 1 (1 unique)
exec speed: 9.68/sec (zzzz...)
fuzzing strategy yields
                                                     path geometry
 bit flips: 577/35.3k, 110/35.3k, 68/35.3k
                                                       levels : 2
byte flips: 4/4412, 8/4411, 21/4409
                                                      pending: 1003
arithmetics: 129/246k, 32/126k, 14/11.8k
                                                     pend fav: 1
known ints: 45/19.9k, 0/0, 0/0
                                                    own finds: 1002
dictionary: 0/0, 0/0, 0/0
                                                     imported : n/a
                                                     variable : 3
     havoc: 0/0, 0/0
      trim : 0.00%/1090, 0.00%
                                                                [cpu: 98%]
```

Thanks

- Thanks To Michal Zalewski <lcamtuf@google.com>
- For developing and sharing AFL

Reference

- Number of apps available in leading app stores as of June 2016
- American Fuzzy Lop: http://lcamtuf.coredump.cx/afl/
- ArchiveALL: https://github.com/songfei/ArchiveALL
- ZXingObjC: https://github.com/TheLevelUp/ZXingObjC
- Unrar4iOS: https://github.com/ararog/Unrar4iOS
- opus codec: https://www.opus-codec.org/
- KxMovie: https://github.com/kolyvan/kxmovie

Question?