Fuzzing Android: A Recipe for Uncovering Vulnerabilities Inside System Components in Android

black hat EUROPE 2015

Alexandru Blanda
Intel OTC Security SQE

intel Software

Intel OpenSource TECHNOLOGY CENTER

Agenda

A fuzzing approach

- Set of basic methods and concepts for fuzzing in Android

Real-life fuzzing campaigns

- Fuzzing Stagefright
- Fuzzing the Android installer

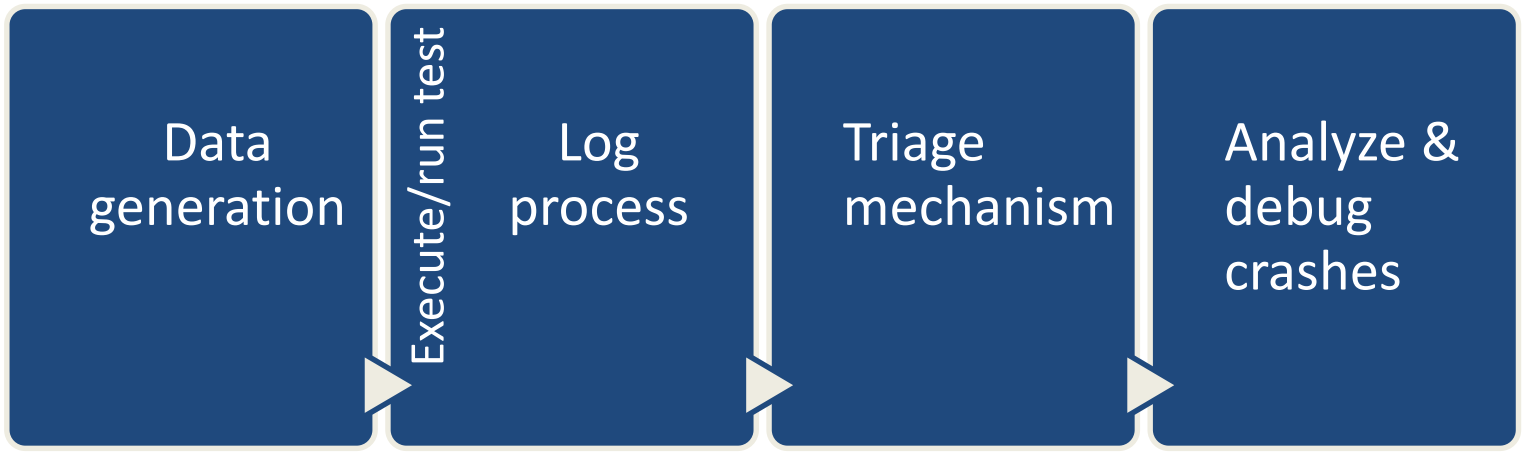
Alternatives

- Fuzzing with AFL in Android

A fuzzing approach in Android

black hat®
EUROPE 2015

A fuzzing approach in Android



Data generation

- Mutational vs. generational fuzzing

- Tools
  - Basic Fuzzing Framework (BFF)
  - FuzzBox
  - Radamsa
  - American Fuzzy Lop (AFL)*

- Seed gathering
  - Python mass downloader using Google and Bing search engines
  - -inurl:htm -inurl:html intitle:"index of" .mp3 + wget

Log process

- Log every test case with fatal priority

```
$ adb shell log  -p F -t <Component> <test_case_index> *** <reproducibility_info>
```

- Log template

```
$ adb shell logcat -v time *:F

01-16 17:46:12.240 F/<Component> (PID):  <test_case_index> ***
<reproducibility_info>
01-16 17:46:19.676 F/<Component> (PID):  <test_case_index> ***
<reproducibility_info>
01-16 17:46:24.328 F/<Component> (PID):  <test_case_index> ***
<reproducibility_info>
17:46:24.405 F/libc (8321): Fatal signal 11 (SIGSEGV) at 0x18 (code=1), thread
831 (process_name)
01-16 17:46:25.128 F/<Component> (PID):  <test_case_index> ***
<reproducibility_info>
01-16 17:46:55.933 F/<Component> (PID):  <test_case_index> ***
<reproducibility_info>
```

Triage mechanism

- Input that produces a crash generates an entry in /data/tombstones and /data/system/dropbox

```
pid: 3438, tid: 3438, name: stagefright  >>> stagefright <<<
signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr deadbaad
Abort message: 'invalid address or address of corrupt block 0x8004d748 passed to
dlfree'
    eax b3ee0ff8  ebx b7b18f38  ecx b7b1d900  edx b3ee0ff8
    esi 8004d748  edi af6d4dee
    xcs 00000073  xds 0000007b  xes 0000007b  xfs 00000000  xss 0000007b
    eip b7a7202c  ebp bffff418  esp bffff3d0  flags 00010286

backtrace:
    #00  pc 0001402c  /system/lib/libc.so (dlfree+1948)
    #01  pc 0000d630  /system/lib/libc.so (free+32)
    #02  pc 000dcf1c  /system/lib/libstagefright.so
(android::MediaBuffer::~MediaBuffer()+108)
    #03  pc 000dd6eb  /system/lib/libstagefright.so
(android::MediaBuffer::release()+267)
    #04  pc 000ddf7b  /system/lib/libstagefright.so
(android::MediaBufferGroup::~MediaBufferGroup()+187)
```

Triage mechanism

1. Parse generated logs
   - Identify input that causes crashes
2. Retest crashing input
3. For each identified test case
   - Grab generated tombstone
   - Parse tombstone – get the PC value
   - Check if PC value has been previously encountered
   - Save tombstone and input if issue is unique

Analyze & debug crashes

- /data/tombstones

```
pid: 3438, tid: 3438, name: stagefright  >>> stagefright <<<
signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr deadbaad
Abort message: 'invalid address or address of corrupt block 0x8004d748 passed to
dlfree'
    eax b3ee0ff8  ebx b7b18f38  ecx b7b1d900  edx b3ee0ff8
    esi 8004d748  edi af6d4dee
    xcs 00000073  xds 0000007b  xes 0000007b  xfs 00000000  xss 0000007b
    eip b7a7202c  ebp bffff418  esp bffff3d0  flags 00010286

backtrace:
    #00  pc 0001402c  /system/lib/libc.so (dlfree+1948)
    #01  pc 0000d630  /system/lib/libc.so (free+32)
    #02  pc 000dcf1c  /system/lib/libstagefright.so
(android::MediaBuffer::~MediaBuffer()+108)
    #03  pc 000dd6eb  /system/lib/libstagefright.so
(android::MediaBuffer::release()+267)
    #04  pc 000ddf7b  /system/lib/libstagefright.so
(android::MediaBufferGroup::~MediaBufferGroup()+187)
```

Analyze & debug crashes

- dmesg

```
<6>[73801.130320] stagefright[12469]: segfault at 14 ip 00000000f72a5fff sp
00000000fff98710 error 4 in libstagefright.so[f71c6000+1b5000]

<6>[73794.579462] stagefright[12455]: segfault at c ip 00000000f728bcfe sp
00000000ff9d6f90 error 6 in libstagefright.so[f71e8000+1b5000]
```

```
/*
 * Page fault error code bits:
 *
 *   bit 0 ==    0: no page found          1: protection fault
 *   bit 1 ==    0: read access            1: write access
 *   bit 2 ==    0: kernel-mode access     1: user-mode access
 *   bit 3 ==                              1: use of reserved bit detected
 *   bit 4 ==                              1: fault was an instruction fetch
 */
```

Analyze & debug crashes

- gdbserver (on device)

```
$ gdbserver :5039 --attach <process_pid>
    OR
$ gdbserver :5039 /path/to/executable <options> (ex: gdbserver :5039
/system/bin/stagefright -a file.mp3)
```

- gdb (on local machine)

```
$ adb forward tcp:5039 tcp:5039
$ gdb
    (gdb) target remote :5039 (from the gdb shell)
    (gdb) continue (to resume process execution)
```

- Load symbols for shared libraries

```
(gdb) set solib-absolute-prefixdb
/path/to/tree/out/target/product/<product_id>/symbols/

(gdb) set solib-search-path
/path/to/tree/out/target/product/<product_id>/symbols/system/lib/
```

Analyze & debug crashes

- addr2line

```
backtrace:
    #00  pc 0001402c  /system/lib/libc.so (dlfree+1948)
    #01  pc 0000d630  /system/lib/libc.so (free+32)
    #02  pc 000dcf1c  /system/lib/libstagefright.so
 (android::MediaBuffer::~MediaBuffer()+108)
```

```
$ addr2line -f -e
/path/to/tree/out/target/product/<product_id>/symbols/system/lib/libstagefright.s
o 000dcf1c
```

Fuzzing the Stagefright media framework

black hat®
EUROPE 2015

Audio and video as attack vectors

Binary streams containing complex data

Large variety of audio and video players and associated media codecs

User perception that media files are harmless

Media playback doesn’t require special permissions

Overview of testing process

- Create corrupt but structurally valid media files
- Direct them to the appropriate decoders
- Monitor the system for potential issues
- Pass the issues through a triage mechanism

The Stagefright CLI

- frameworks/av/cmds/stagefright

```
root@android:/ # stagefright -h
usage: stagefright
-h(elp)
-a(udio)
-n repetitions
-l(ist) components
-m max-number-of-frames-to-decode in each pass
-p(rofiles) dump decoder profiles supported
-t(humbnail) extract video thumbnail or album art
-s(oftware) prefer software codec
-r(hardware) force to use hardware codec
-o playback audio
-w(rite) filename (write to .mp4 file)
-x display a histogram of decoding times/fps (video only)
-S allocate buffers from a surface
-T allocate buffers from a surface texture
-d(ump) filename (raw stream data to a file)
-D(ump) filename (decoded PCM data to a file)
```

Stagefright log

- Media files are corrupted on the local machine using the Basic Fuzzing Framework (BFF) tool

```
04-14 05:02:07.698 F/Stagefright(20222): - sp_stagefright *** 958 -
Filename:zzuf.32732.c8jZzT.mp4
04-14 05:02:13.382 F/Stagefright(20255): - sp_stagefright *** 959 -
Filename:zzuf.26772.zh7c8g.mkv
04-14 05:02:13.527 F/libc    (20256): Fatal signal 11 (SIGSEGV), code 1, fault
addr 0x0 in tid 20256 (stagefright)
04-14 05:02:20.820 F/Stagefright(20270): - sp_stagefright *** 960 -
Filename:zzuf.12260.ayDuIA.mpg
04-14 05:02:21.259 F/Stagefright(20281): - sp_stagefright *** 961 -
Filename:zzuf.6488.F8drye.mp4
```

Results

- Initial fuzzing campaigns started in March 2014; first issues reported to Google
- Initial results were extremely surprising: thousands of crashes per week (triage mechanism)
- First severe issues in the September 2014 Android security bulletin:
  - Integer overflows in libstagefright (CVE-2014-7915, CVE-2014-7916, CVE-2014-7917)
- The tool was open-sourced in February 2015:
  - https://github.com/fuzzing/MFFA
- Currently used as a complementary solution along with AFL

Fuzzing the application install process

black hat®
EUROPE 2015

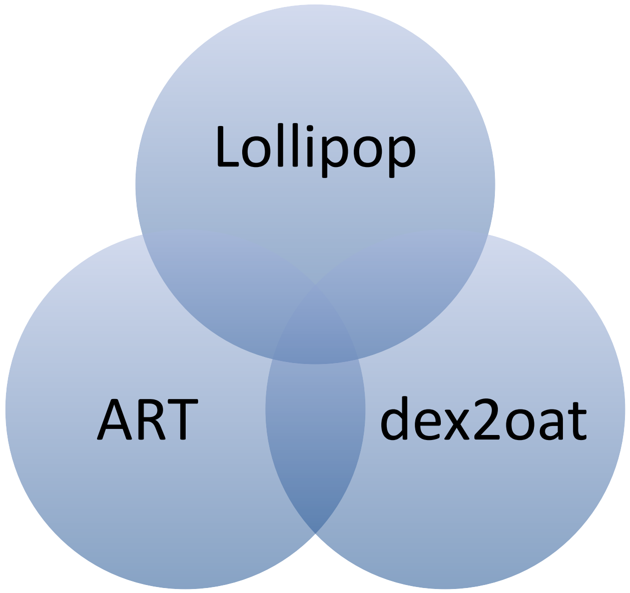
Motivation

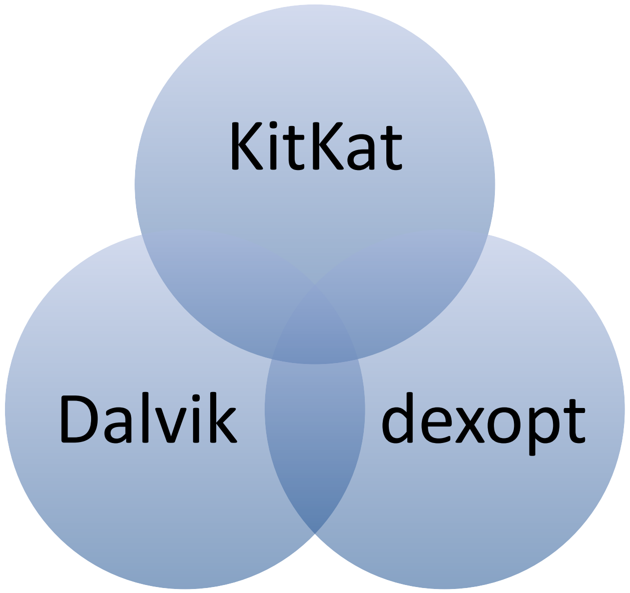
Attractive target – process runs with high system privileges

Method for unprivileged users to send input to system components

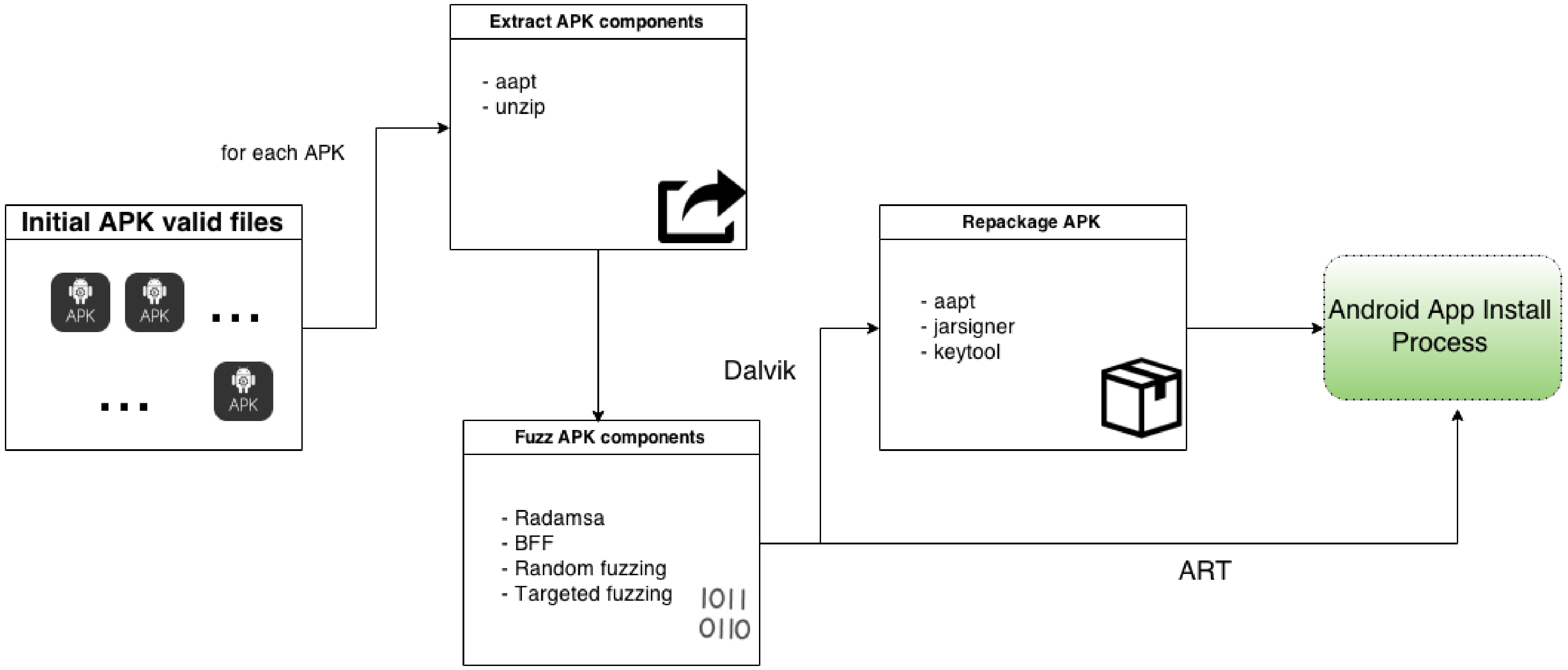
Check for issues that are not discovered during regular validation

Install process overview

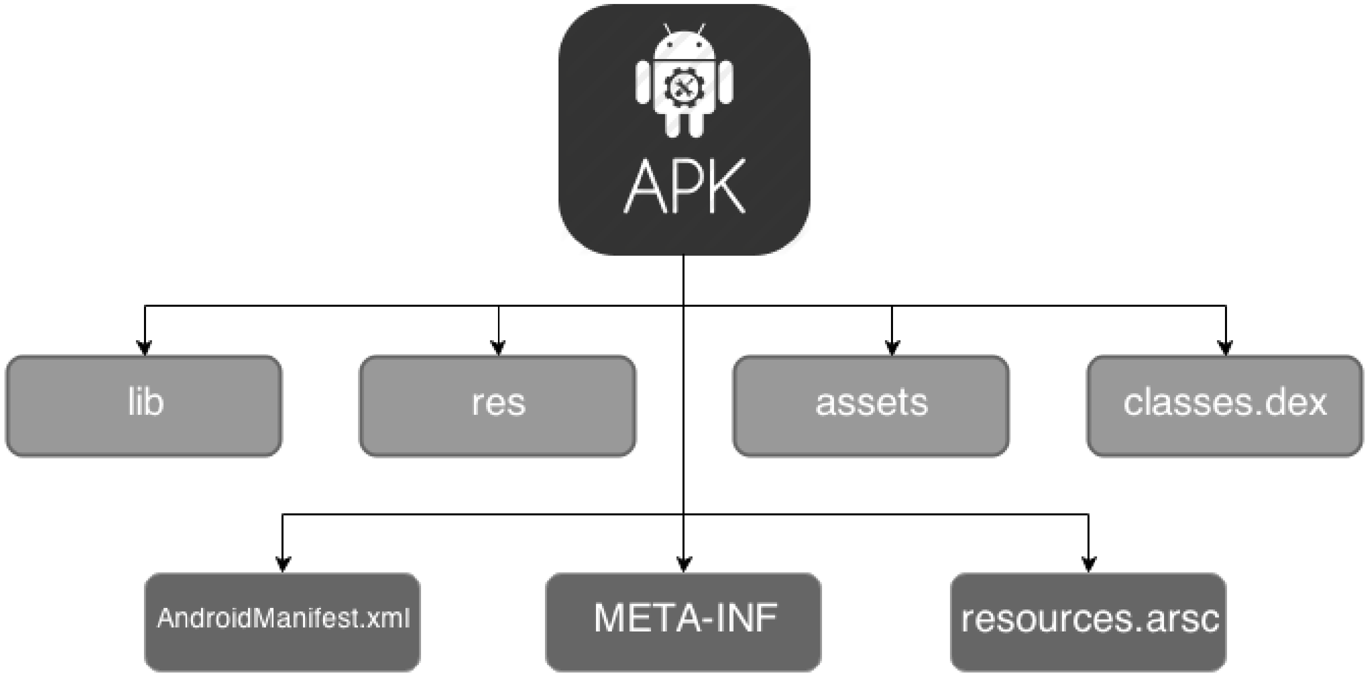




Testing process overview



APK structure



ART vs. Dalvik (dexopt)

- Approach on Dalvik: dexopt cannot be called as a standalone binary from the device shell

- Simulate the regular APK install process:

  - Extract classes.dex file from seed APK
    - `unzip -d  </local/path/> </apk/path/>`
  - Fuzz extracted dex file*
    - `<fuzz>-s <seed> classes.dex > fuzzed.dex`
  - Remove original .dex file from initial APK
    - `aapt r <original_apk> classes.dex`
  - Repackage APK with fuzzed APK
    - `aapt a <original_apk> classes.dex`

ART vs. Dalvik (dexopt)

- Simulate the regular APK install process:
  - Create local keystore
    - `keytool -genkey -v -keystore keystore.keystore -alias keystore -keyalg RSA -keysize 2048 -validity 10000`
  - Remove META-INF directory from APK
    - `zip --delete </apk/path/> META-INF/*`
  - Resign the APK using local keystore
    - `jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore </keystore/path> </apk/path> <keystore_alias>`

ART vs. Dalvik (dexopt)

- Log example:

```
06-26 17:43:05.568 F/dexopt (14769): - sp_lib.py - APK_id = imangi.templerun.apk
seed = radamsa -s 1927
06-26 17:43:29.732 F/dexopt (14881): - sp_lib.py - APK_id = imangi.templerun.apk
combination = radamsa -s 2086
06-26 17:43:54.620 F/dexopt (14988): - sp_lib.py - APK_id = imangi.templerun.apk
seed = radamsa -s 5011
06-26 17:44:19.763 F/dexopt (15105): - sp_lib.py - APK_id = imangi.templerun.apk
seed = radamsa -s 1543
06-26 17:44:43.524 F/dexopt (15215): - sp_lib.py - APK_id = imangi.templerun.apk
seed = radamsa -s 9090
06-26 17:44:44.079 F/libc    (15227): Fatal signal 11 (SIGSEGV) at 0xaa4c04f8
(code=1), thread 15227 (mangi.templerun)
06-26 17:45:09.950 F/dexopt (15338): - sp_lib.py - APK_id = imangi.templerun.apk
seed = radamsa -s 8098
06-26 17:45:33.771 F/dexopt (15451): - sp_lib.py - APK_id = imangi.templerun.apk
seed = radamsa -s 1069
06-26 17:45:59.802 F/dexopt (15570): - sp_lib.py - APK_id = imangi.templerun.apk
seed = radamsa -s 8925
```

ART vs. Dalvik (dex2oat)

- Approach on ART: dex2oat can be used directly from the command line for our fuzzing purposes

```
Usage: dex2oat [options]...
-j<number>: specifies the number of threads used for compilation.
--dex-file=<dex-file>: specifies a .dex file to compile.
--zip-fd=<file-descriptor>: specifies a file descriptor of a zip file
      containing a classes.dex file to compile.
--zip-location=<zip-location>: specifies a symbolic name for the file
--oat-file=<file.oat>: specifies the oat output destination via a filename.
--oat-fd=<number>: specifies the oat output destination via a file descriptor.
--oat-location=<oat-name>: specifies a symbolic name for the file corresponding
    to the file descriptor specified by --oat-fd.
...
...
...
```

black hat EUROPE 2015

ART vs. Dalvik (dex2oat)

- Log example:

```
09-29 11:32:20.460 F/dex2oat ( 8041): - sp_libd.py - dex_id = com.evernote.apk
seed = radamsa -s 1012528
09-29 11:32:33.405 F/dex2oat ( 8054): - sp_libd.py - dex_id = com.evernote.apk
seed = radamsa -s 6186726
09-29 11:32:46.277 F/dex2oat ( 8066): - sp_libd.py - dex_id = com.evernote.apk
seed = radamsa -s 7338683
09-29 11:32:49.121 F/libc    (15227): Fatal signal 11 (SIGSEGV) at 0xaa4c0302
(code=1), thread 15227 (evernote)
09-29 11:32:57.249 F/dex2oat ( 8079): - sp_libd.py - dex_id = com.evernote.apk
seed = radamsa -s 231131
09-29 11:33:08.528 F/dex2oat ( 8093): - sp_libd.py - dex_id = com.evernote.apk
seed = radamsa -s 4456070
```

Actual fuzzing methods

- 3 alternatives:

1. Completely random fuzzing (dex2oat)
2. Random fuzzing and partial header reconstruction (applies to dex2oat & dexopt)
3. Targeted fuzzing and complete header reconstruction (dex2oat & dexopt)

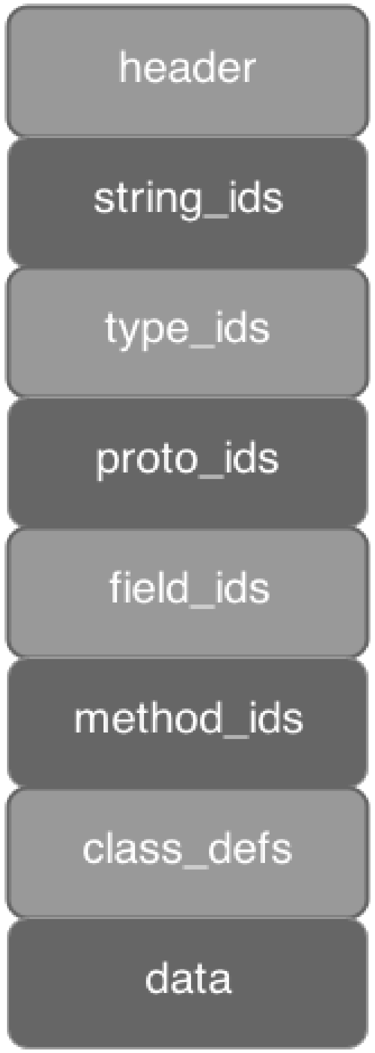
Install verification process

```
01-03 13:24:13.511 I/dex2oat ( 5671): dex2oat --dex-file=test7.dex --oat-
file=output.oat
01-03 13:24:13.125 W/dex2oat ( 5671): Failed to open .dex from file 'test7.dex':
verify
                   dex file 'test7.dex': Bad checksum (790931db, expected
745631bc)
01-03 13:24:13.115 E/dex2oat ( 5671): Failed to open some dex files: 1
01-03 13:24:13.447 I/dex2oat ( 5671): dex2oat took 255.693ms (threads: 4)

01-03 03:22:23.581 I/dex2oat ( 5671): dex2oat --dex-file=test7.dex --oat-
file=output.oat
01-03 03:22:23.635 W/dex2oat ( 5671): Failed to open .dex from file 'test7.dex':
verify
                   dex file 'test7.dex': Bad file size (143221ab, expected
435611cd)
01-03 03:22:23.635 E/dex2oat ( 5671): Failed to open some dex files: 1
01-03 03:22:23.837 I/dex2oat ( 5671): dex2oat took 255.693ms (threads: 4)

01-03 04:21:13.181 I/dex2oat ( 5671): dex2oat --dex-file=test7.dex --oat-
file=output.oat
01-03 04:21:13.235 W/dex2oat ( 5671): Failed to open .dex from file 'test7.dex':
verify
                   dex file 'test7.dex': Invalid header size (7f, expected 70)
01-03 04:21:13.641 E/dex2oat ( 5671): Failed to open some dex files: 1
01-03 04:21:13.857 I/dex2oat ( 5671): dex2oat took 255.693ms (threads: 4)
```

DEX file format



black hat EUROPE 2015

DEX file header

struct dex_magic magic	dex 035	0h	8h	Magic value
uint checksum	B3D20217h	8h	4h	Alder32 checksum of rest of file
SHA1 signature[20]	6DB8EDA774	Ch	14h	SHA-1 signature of rest of file
uint file_size	1430508	20h	4h	File size in bytes
uint header_size	112	24h	4h	Header size in bytes
uint endian_tag	12345678h	28h	4h	Endianness tag
uint link_size	0	2Ch	4h	Size of link section
uint link_off	0	30h	4h	File offset of link section
uint map_off	1430336	34h	4h	File offset of map list
uint string_ids_size	11029	38h	4h	Count of strings in the string ID list
uint string_ids_off	112	3Ch	4h	File offset of string ID list
uint type_ids_size	2068	40h	4h	Count of types in the type ID list
uint type_ids_off	44228	44h	4h	File offset of type ID list
uint proto_ids_size	2592	48h	4h	Count of items in the method prototype ID list
uint proto_ids_off	52500	4Ch	4h	File offset of method prototype ID list
uint field_ids_size	5335	50h	4h	Count of items in the field ID list
uint field_ids_off	83604	54h	4h	File offset of field ID list
uint method_ids_size	12925	58h	4h	Count of items in the method ID list
uint method_ids_off	126284	5Ch	4h	File offset of method ID list
uint class_defs_size	1427	60h	4h	Count of items in the class definitions list
uint class_defs_off	229684	64h	4h	File offset of class definitions list
uint data_size	1155160	68h	4h	Size of data section in bytes
uint data_off	275348	6Ch	4h	File offset of data section

black hat EUROPE 2015

Random fuzzing and partial header reconstruction

- Alter randomly the contents of all sections of the dex file
- Recompute and/or rewrite the header fields we have information about

Magic number (constant value)

Checksum (needs to be computed)

SHA1 signature (needs to be computed)

File size (needs to be computed)

Header size (constant value)

Endian tag (constant value)

Targeted fuzzing and complete header reconstruction

- Split the initial dex in 3 parts: data section, map section and the unmodified rest of the file section
- Fuzz only the data chunk as a separate file using Radamsa
- Glue all the chunks back together and rewrite the header

Magic number (constant value)

Checksum (needs to be computed)

SHA1 signature (needs to be computed)

File size (needs to be computed)

Header size (constant value)

Endian tag (constant value)

Map offset (needs to be computed)

Data size (needs to be computed)

Completely random fuzzing

- Generate random fuzzed dex files using BFF, starting from an initial set of seed dex files
- No changes made to the dex file header

```
09-19 11:57:00.346 F/dex2oat_bff(16102): - sp_libd.py - dex_id =
zzuf.16185.sOaX7i.dex
09-19 11:57:01.193 F/dex2oat_bff(16113): - sp_libd.py - dex_id =
zzuf.2554.pfKpqy.dex
09-19 11:57:03.488 F/dex2oat_bff(16125): - sp_libd.py - dex_id =
zzuf.4460.JGEqFa.dex
09-19 11:57:04.218 F/libc    (16127): Fatal signal 11 (SIGSEGV) at 0xaa2c14f4
(code=1), thread 16127 (evernote)
09-19 11:57:05.767 F/dex2oat_bff(16136): - sp_libd.py - dex_id =
zzuf.17117.vuTEiB.dex
09-19 11:57:08.651 F/dex2oat_bff(16146): - sp_libd.py - dex_id =
zzuf.5671.gHcnXq.dex
09-19 11:57:12.293 F/dex2oat_bff(16157): - sp_libd.py - dex_id =
zzuf.28549.ArCcd7.dex
09-19 11:57:14.143 F/dex2oat_bff(16167): - sp_libd.py - dex_id =
zzuf.1524.kHO8eC.dex
```

black hat EUROPE 2015

Results

- Number of crashes not as spectacular as in the case of Stagefright
- 1 critical issue affecting dex2oat – CVE-2014-7918
- A number of low priority issues reported and fixed both in KitKat and Lollipop
- Several issues under investigation

Fuzzing with AFL in Android

black hat

EUROPE 2015

American Fuzzy Lop (AFL)

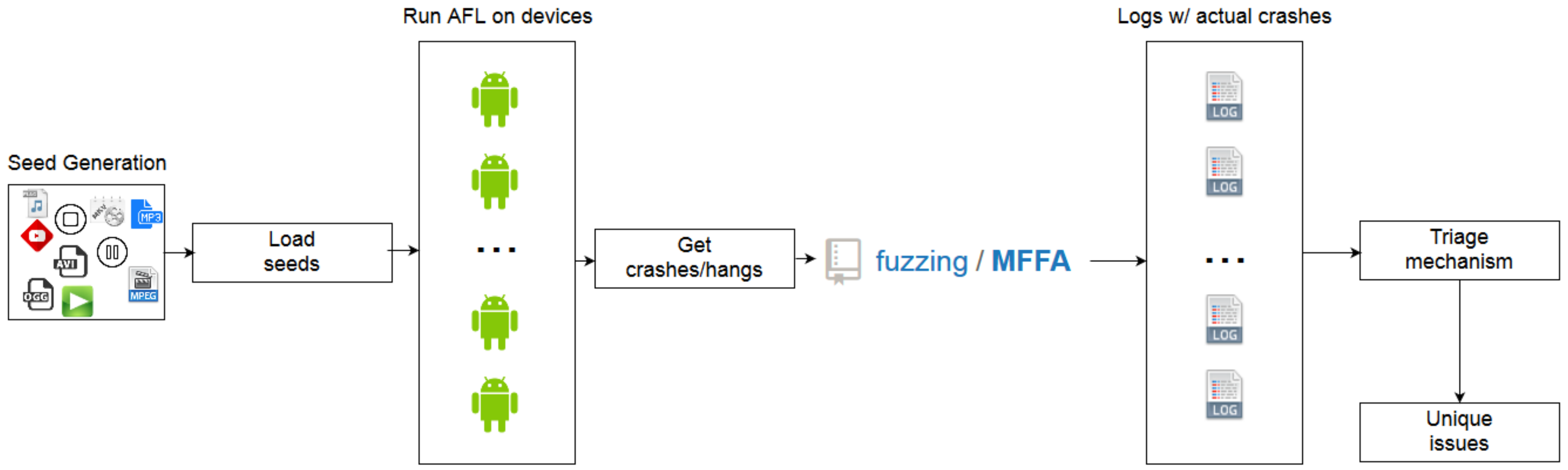
- Instrumentation based fuzzing tool developed by Michal Zalewski
- Two fuzzing modes: dumb-mode, instrumented-mode (peruvian rabbit mode)
- Instrumented mode detects changes to program control flow to find new code paths
- Detects both crashes and hangs and sorts out the unique issues
- Android port of the tool developed by Adrian Denkiewicz of Intel

black hat EUROPE 2015

Fuzzing Stagefright with AFL

```
1. Check device prerequisites
    1) Root
    2) Remount
    3) Push afl target binary
    4) Load initial seeds
    5) Set scalling governor
2. Eliminate crashing test cases from initial seeds on each
device
    1) Run AFL in a loop with timeout
    2) Identify crashing test case and delete it from input
    folder
    3) Restart AFL with timeout -> if crash occurs goto 2) else
    goto 4)
    4) No crash occurred after the timeout -> AFL successfully
    started -> kill the process
3. Restart the AFL process with clean input directory and
redirect output to /dev/null
```

Fuzzing Stagefright with AFL



Results

- 1 critical issue discovered using this approach:
  - heap corruption that can lead to arbitrary code execution in the mediaserver process (CVE-2015-3832)

- Multiple low priority issues reported to and fixed by Google (null-pointer dereferences, integer division by zero issues)

Acknowledgements

Adrian Denkiewicz – adrian.o.denkiewicz@intel.com

Costel Maxim – costel.maxim@intel.com

Ionut Gavaz – ionut.a.gavaz@intel.com

Intel OTC Security Romania Team

black hat®

EUROPE 2015

Q&A

black hat®
EUROPE 2015