Fuzzing Android OMX

Mingjian Zhou and Chiachih Wu
CORE Team
About Us

• Mingjian Zhou, 周明建
  – Security researcher @ 360 CORE team
  – Focused on Android vulnerability research and exploit development

• Chiachih Wu, 吳家志 (@chiachih_wu)
  – Security researcher @ 360 CORE team
  – Android/Linux system security research
  – CORE team (c0reteam.org) founding member

• CORE Team
  – A security-focused group started in mid-2015
  – With a recent focus on the Android/Linux platform, the team aims to discover zero-day vulnerabilities, develop proof-of-concept exploits, and explore possible defenses
Stagefright: Scary Code in the Heart of Android

Researching Android Multimedia Framework Security

Joshua "jduck" Drake
August 5th 2015
Black Hat USA
Acknowledgements

We would like to thank these researchers for their contributions:

- Brennan Lautner: CVE-2015-3863
- Yajin Zhou, Lei Wu, and Xuxian Jiang of CORE Team from Qihoo 360: CVE-2015-3865

Acknowledgements

We would like to thank these researchers for their contributions:

- Android and Chrome Security Team: CVE-2016-0809, CVE-2016-0810
- Broadgate Team: CVE-2016-0801, CVE-2015-0802
- Chiachih Wu (@chiachih_wu), Mingjian Zhou (@Mingjian_Zhou), and Xuxian Jiang of CORE Team, Qihoo 360: CVE-2016-0804
Fuzzing Android System Services by Binder Call to Escalate Privilege

Guang Gong
Security Reacher
Qihoo 360
Twitter & Weibo: @oldfresher

Black Hat USA 2015
Agenda

• Introduction
• Fuzzing Android OMX
• Confirmed Vulnerabilities
• Patterns of OMX Vulnerabilities
INTRODUCTION
What is OMX (1/2)

- Open Media Acceleration, aka Open MAX, often shortened as “OMX”
- *WIKI*: a non-proprietary and royalty-free cross-platform set of C-language programming interfaces that provides abstractions for routines especially useful for audio, video, and still images processing.
What is OMX (2/2)

Media applications can be written portably, independent of the underlying media platform.

Media components can be integrated into flexible media graphs for advanced streaming media processing.

Media components can be written using primitives for portability across diverse parallel and serial silicon architectures.

OpenMAX layers can be implemented together or independently from the other layers.

OpenMAX|AL
“Application Layer”
Defines high-level playback and recording API

OpenMAX|IL
“Integration Layer”
Defines media component interfaces

OpenMAX|DL
“Development Layer”
Defines low-level media primitives and concurrency constructs

Platform Media Framework

Audio Components
- e.g. MP3

Video Components
- e.g. H.264

Image Components
- e.g. JPEG

Audio Components

Video Components

Image Components

Media Engines - CPUs, DSP, Hardware Accelerators etc.
OMX in Android (1/2)

• OMX Integration Layer (IL)
  – provides a standardized way for **Stagefright** to recognize and use custom hardware-based multimedia codecs called components.

• Vendors provide the **OMX plugin** which links custom codec components to Stagefright.

• Custom codecs **must** be implemented according to the OMX IL component standard.
OMX in Android (2/2)

User APPs

Binder IPC

MediaServer

OMX IL

Audio OMX Component

Video OMX Component

Soft A/V Codecs

Audio Drivers

Video Drivers

IOCTL

Kernel

Media Player Service

Stagefright

Binder

Music

MMS

…
OMX Codecs

• Android provides built-in software codecs for common media formats
• Vendors’ codecs

Built-in Soft Codecs Example

Vendor Codecs Example

OMX.google.aac.decoder
OMX.google.aac.encoder
OMX.google.amrnb.decoder
OMX.google.amrnb.encoder
OMX.google.amrwb.decoder
OMX.google.amrwb.encoder
OMX.google.flac.encoder
OMX.google.g711.alaw.decoder
OMX.google.g711.mlaw.decoder
OMX.google.gsm.decoder

OMX.qcom.audio.encoder.aac
OMX.qcom.audio.encoder.amrnb
OMX.qcom.audio.encoder.evrc
OMX.qcom.audio.encoder.qcelp13
OMX.qcom.file.muxer
OMX.qcom.video.decoder.avc
OMX.qcom.video.decoder.avc.secure
OMX.qcom.video.decoder.divx
OMX.qcom.video.decoder.divx311
OMX.qcom.video.decoder.divx4
OMX.qcom.video.decoder.h263
OMX.qcom.video.decoder.hevc
Why OMX?

- Exposed via multiple attack vectors
- Media native codes are often vulnerable
FUZZING ANDROID OMX
The Attack Surface (1/2)
The Attack Surface (2/2)
OMX Interfaces

- Defined in IOMX

<table>
<thead>
<tr>
<th>API</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>listNodes</td>
<td>List names of all the codec component</td>
</tr>
<tr>
<td>allocateNode</td>
<td>Create a codec component</td>
</tr>
<tr>
<td>allocateBuffer</td>
<td>Allocate input/output buffers for codec</td>
</tr>
<tr>
<td>useBuffer</td>
<td>Provide a share buffer to the server</td>
</tr>
<tr>
<td>emptyBuffer</td>
<td>Request (or receive) an empty input buffer, fill it up with data and send it to the codec for processing</td>
</tr>
<tr>
<td>fillBuffer</td>
<td>Request (or receive) a filled output buffer, consume its contents and release it back to the codec</td>
</tr>
<tr>
<td>sendCommand</td>
<td>Send commands to codecs, such as changing state, port disable/enable</td>
</tr>
<tr>
<td>getParameter</td>
<td>Get codecs’ parameters</td>
</tr>
<tr>
<td>setParameter</td>
<td>Set codecs’ parameters</td>
</tr>
</tbody>
</table>
Fuzzing Flow

Start

Select a component from the node list

Get the default codec parameters

Generate new parameters and set

Prepare input port buffers

Prepare output port buffers

Change the codec state from loaded to idle

Change the codec state from idle to executing

Empty/Fill buffers

Free node

end
CONFIRMED VULNERABILITIES
Confirmed Vulnerabilities (1/3)

• By 2016/07/07, total **21** vulnerabilities are confirmed.
  – **16** vulnerabilities (15 high, 1 moderate) have been disclosed on Android Security Bulletins.
  – **Others** will be disclosed on later Android Security Bulletins.

• **Almost all** the codecs implemented by **Google** and **vendors** (QualComm, Nvidia, MediaTek) are vulnerable.
Confirmed Vulnerabilities (2/3)

<table>
<thead>
<tr>
<th>NO.</th>
<th>CVE</th>
<th>Android ID</th>
<th>Codec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CVE-2016-2450</td>
<td>ANDROID-27569635</td>
<td>Google SoftVPX encoder</td>
</tr>
<tr>
<td>2</td>
<td>CVE-2016-2451</td>
<td>ANDROID-27597103</td>
<td>Google SoftVPX decoder</td>
</tr>
<tr>
<td>3</td>
<td>CVE-2016-2452</td>
<td>ANDROID-27662364</td>
<td>Google SoftAMR decoder</td>
</tr>
<tr>
<td>4</td>
<td>CVE-2016-2477</td>
<td>ANDROID-27251096</td>
<td>Qcom libOmxVdec</td>
</tr>
<tr>
<td>5</td>
<td>CVE-2016-2478</td>
<td>ANDROID-27475409</td>
<td>Qcom libOmxVdec</td>
</tr>
<tr>
<td>6</td>
<td>CVE-2016-2479</td>
<td>ANDROID-27532282</td>
<td>Qcom libOmxVdec</td>
</tr>
<tr>
<td>7</td>
<td>CVE-2016-2480</td>
<td>ANDROID-27532721</td>
<td>Qcom libOmxVdec</td>
</tr>
<tr>
<td>8</td>
<td>CVE-2016-2481</td>
<td>ANDROID-27532497</td>
<td>Qcom libOmxVenc</td>
</tr>
<tr>
<td>9</td>
<td>CVE-2016-2482</td>
<td>ANDROID-27661749</td>
<td>Qcom libOmxVdec</td>
</tr>
<tr>
<td>10</td>
<td>CVE-2016-2483</td>
<td>ANDROID-27662502</td>
<td>Qcom libOmxVenc</td>
</tr>
</tbody>
</table>
## Confirmed Vulnerabilities (3/3)

<table>
<thead>
<tr>
<th>NO.</th>
<th>CVE</th>
<th>Android ID</th>
<th>Codec</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>CVE-2016-2484</td>
<td>ANDROID-27793163</td>
<td>Google SoftG711 decoder</td>
</tr>
<tr>
<td>12</td>
<td>CVE-2016-2485</td>
<td>ANDROID-27793367</td>
<td>Google SoftGSM decoder</td>
</tr>
<tr>
<td>13</td>
<td>CVE-2016-2486</td>
<td>ANDROID-27793371</td>
<td>Google SoftMP3 decoder</td>
</tr>
<tr>
<td>14</td>
<td>CVE-2016-3747</td>
<td>ANDROID-27903498</td>
<td>Qcom libOmxVenc</td>
</tr>
<tr>
<td>15</td>
<td>CVE-2016-3746</td>
<td>ANDROID-27890802</td>
<td>Qcom libOmxVdec</td>
</tr>
<tr>
<td>16</td>
<td>CVE-2016-3765</td>
<td>ANDROID-28168413</td>
<td>Google SoftMPEG2 decoder</td>
</tr>
<tr>
<td>17</td>
<td>CVE-2016-3844</td>
<td>AndroidID-28299517</td>
<td>Not disclosed yet</td>
</tr>
<tr>
<td>18</td>
<td>CVE-2016-3835</td>
<td>AndroidID-28920116</td>
<td>Not disclosed yet</td>
</tr>
<tr>
<td>19</td>
<td>CVE-2016-3825</td>
<td>AndroidID-28816964</td>
<td>Not disclosed yet</td>
</tr>
<tr>
<td>20</td>
<td>CVE-2016-3824</td>
<td>AndroidID-28816827</td>
<td>Not disclosed yet</td>
</tr>
<tr>
<td>21</td>
<td>CVE-2016-3823</td>
<td>AndroidID-28815329</td>
<td>Not disclosed yet</td>
</tr>
</tbody>
</table>
PATTERNS OF CONFIRMED VULNERABILITIES
Patterns of Confirmed Vulnerabilities

• Mismatch between Android OMX framework and vendor codecs’ implementation
• Time of check to time of use
• Race condition
• Invalid input/output buffer length
Mismatch between Android OMX and vendors’ codec (1/2)

- CVE-2016-2480
Mismatch between Android OMX and vendors’ codec (2/2)

• CVE-2016-2477
Time of Check to Time of Use (1/2)

<table>
<thead>
<tr>
<th>NO.</th>
<th>CVE</th>
<th>Android ID</th>
<th>Codec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CVE-2016-2479</td>
<td>ANDROID-27532282</td>
<td>Qcom libOmxVdec</td>
</tr>
<tr>
<td>2</td>
<td>CVE-2016-2481</td>
<td>ANDROID-27532497</td>
<td>Qcom libOmxVenc</td>
</tr>
<tr>
<td>3</td>
<td>CVE-2016-2482</td>
<td>ANDROID-27661749</td>
<td>Qcom libOmxVdec</td>
</tr>
<tr>
<td>4</td>
<td>CVE-2016-2483</td>
<td>ANDROID-27662502</td>
<td>Qcom libOmxVenc</td>
</tr>
</tbody>
</table>
Time of Check to Time of Use (2/2)

APP

SET_PARAMETER

USE_BUFFER

SET_PARAMETER

USE_BUFFER/FREE_NODE

Media Server

Set codec input buffer count to 8

Check the buffer count and allocate buffers

Set codec input buffer count to 0x1234

Access buffers with 0x1234

OOB write & Heap overflow
Race Condition

• CVE-2016-3747
Invalid Input/Output Buffer Length

- Codecs don’t check the buffer length

<table>
<thead>
<tr>
<th>NO.</th>
<th>CVE</th>
<th>Android ID</th>
<th>Codec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CVE-2016-2450</td>
<td>ANDROID-27569635</td>
<td>Google SoftVPX encoder</td>
</tr>
<tr>
<td>2</td>
<td>CVE-2016-2451</td>
<td>ANDROID-27597103</td>
<td>Google SoftVPX decoder</td>
</tr>
<tr>
<td>3</td>
<td>CVE-2016-2452</td>
<td>ANDROID-27662364</td>
<td>Google SoftAMR decoder</td>
</tr>
<tr>
<td>4</td>
<td>CVE-2016-2484</td>
<td>ANDROID-27793163</td>
<td>Google SoftG711 decoder</td>
</tr>
<tr>
<td>5</td>
<td>CVE-2016-2485</td>
<td>ANDROID-27793367</td>
<td>Google SoftGSM decoder</td>
</tr>
<tr>
<td>6</td>
<td>CVE-2016-2486</td>
<td>ANDROID-27793371</td>
<td>Google SoftMP3 decoder</td>
</tr>
</tbody>
</table>
Invalid Input/output Buffer Length

Binder IPC

APP

USE_BUFFER
Buffer size: 256

USE_BUFFER
Buffer size: 8

MediaServer

Memory shared with APP

Input buffers
Size: 256

Output Buffers
Size: 8

codec

Read 256 bytes

Write 300 bytes
Conclusion

• Android OMX is vulnerable
  – OMX interfaces and OMX codecs are implemented by Google and vendors separately.
  – Media processing is complex.

• Fuzzing combined with code auditing is helpful for such modules.
  – Many codecs & parameters
Any Questions?

• If you prefer to ask offline, contact us:
  – Mingjian Zhou
    • Twitter/Weibo: @Mingjian_Zhou
    • Mail: cn.zhou.mingjian@gmail.com
  – Chiachih Wu
    • Twitter: @chiachih_wu
References

• Android
  – https://source.android.com/devices/media/

• OMX
  – https://www.khronos.org/openmax/