



ANDROID'S NEW NIGHTMARE:

Media Files

08.09.15

Oğuzhan Topgül
@oguzhantopgul
www.oguzhantopgul.com



Only Takes One Text
llion Android

ES STAFF
culture.
ULL BIO ✓

per cent of Google GOOGL -1.70% Android
a simple multimedia text, a mobile
cases, where phones parse the attack
l, the exploits are silent and the user
eir data. The vulnerabilities are said to
red.

who reported the bugs in April this year,
its partners, he believes most
able to protect their customers. "All
le," Drake, vice president of platform
old FORBES. He believes as many as
ted, going on figures
e. Only Android phones below



A collage of news articles and websites from various sources discussing the Stagefright vulnerability in Android. The sources include NTV, threatpost, The Hacker News, and ShiftDelete.Net. The headlines highlight the ability to hack millions of Android devices using a single multimedia text message. The images in the collage feature a hand holding an Android smartphone displaying the Android logo, and a large green Android robot icon.

NTV
TÜRKİYE DÜNYA EKONOMİ SPOR SAĞLIK
MELO VE ALERİ
Anasayfa > Teknoloji > SİLES INTER'E TRANSFER OLDU
threathpost
08/31/15 8:50 | @CEP_Rou
Welcome > Blog Home > Black Hat > Android Stagefright Flaw:
Android STAGEFRIGTH RISK
by Michael Mimoso Follow
Vulnerabilities discovered in the Android operating system could allow unpatched devices to be hacked, putting their privacy, data, and tablets.

ShiftDelete.Net
Forum
Fotogaleri Nasıl Yapılır? UKT Android iOS Windows
SDNtv Incele
iPhone 6s ve iPhone 6s Plus ne zaman tanıtıldı?
İşte o tarih!
- @shiftdeletenet
Forbes / Tech
JUL 27, 2015 @ 1:00 PM 127,030 VIEWS
Stagefright: It Only Takes One Text Message to Hack Millions of Android Devices
GET FREE HACKING TRAINING NOW
How to Hack Millions of Android Phones Using Stagefright Bug, Without Sending MMS
Friday, July 31, 2015 by Swati Khandelwal
8+1 174 2.9k 2933 465 12 ShareThis 8019
New Ways to Hack Millions of Devices Remotely
Earlier this week, security researchers at Zimperium revealed a high-severity vulnerability in Android platforms that allowed a single multimedia text message to hack 950 Million Android smartphones

STAGEFRIGHT VULNERABILITY



- Result of a research conducted by **Joshua Drake** a.k.a **@jduck**
- Presented at 2015 BLACKHAT USA security conference with the title

Stagefright: Scary Code in the Heart of Android*

- Most of the Android devices are still vulnerable

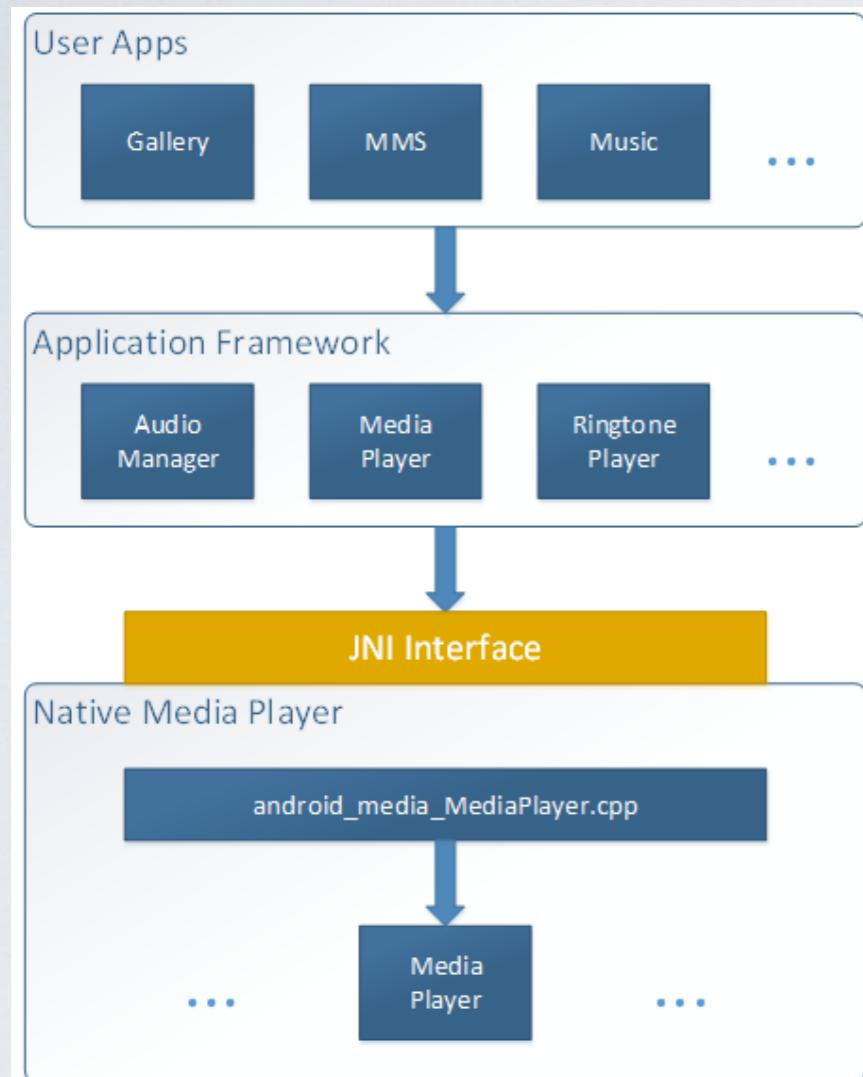
* <https://www.blackhat.com/us-15/briefings.html#stagefright-scary-code-in-the-heart-of-android>

STAGEFRIGHT

- Android's Multimedia library
- Processes video and audio files
- Extracts meta-data of media files
- Added to AOSP with Android 2.0
- Runs inside MediaServer



STAGEFRIGHT

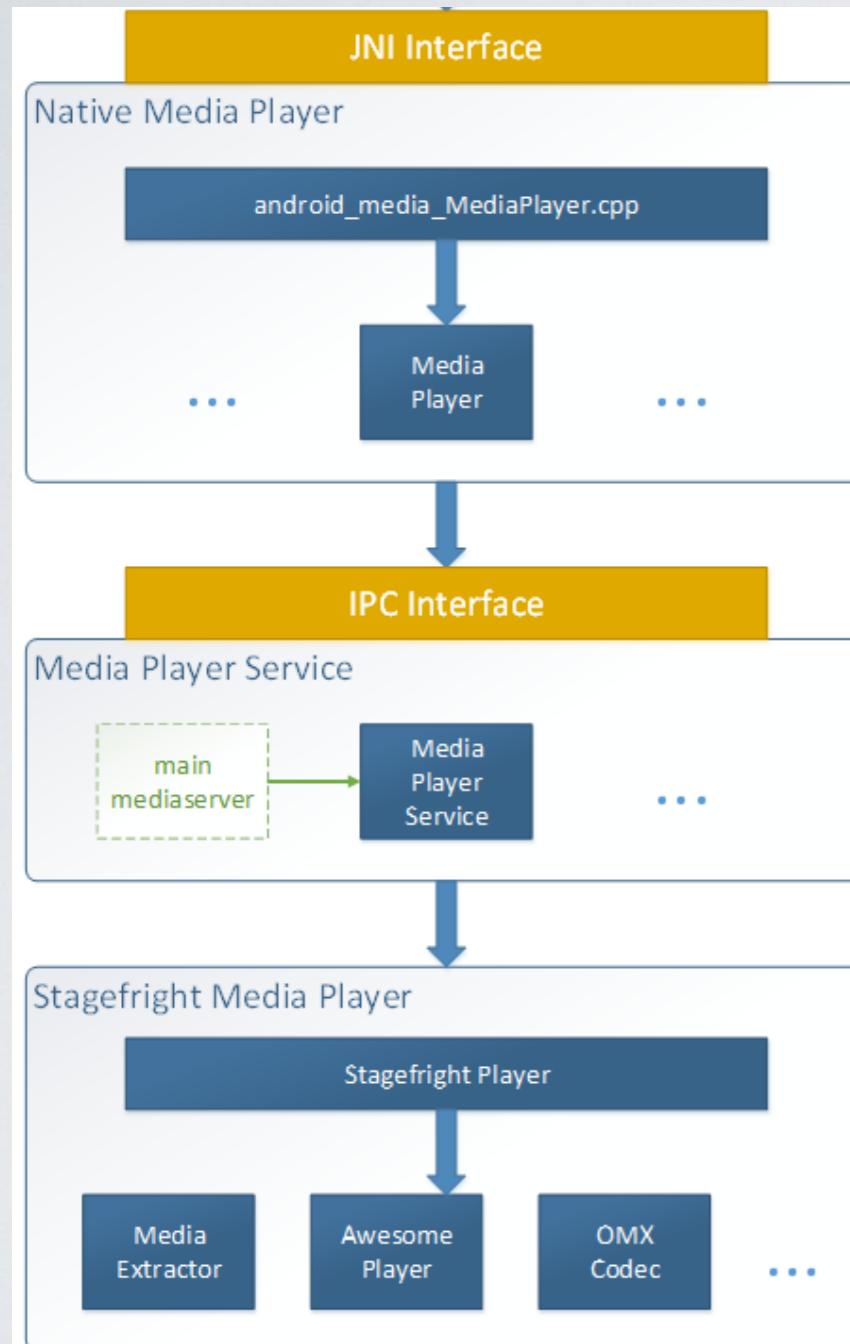


frameworks/base/media/java/android/media/**MediaPlayer.java**

frameworks/base/media/jni/**android_media_MediaPlayer.cpp**

* <https://android.googlesource.com/platform/frameworks/base/+/master/media/>

STAGEFRIGHT



frameworks/base/media/jni/**android_media_MediaPlayer.cpp**

frameworks/av/media/mediaserver/**main_mediaserver.cpp**

frameworks/av/media/libmediaplayerservice/**MediaPlayerService.cpp**

frameworks/av/media/libstagefright/**MediaSource.cpp**

frameworks/av/media/libstagefright/**MediaExtractor.cpp**

frameworks/av/media/libstagefright/**AwesomePlayer.cpp**

* <https://android.googlesource.com/platform/frameworks/av/+/master/media>

MEDIASERVER

mediaserver is run by **init** process

```
root@android:/ # ps | grep media
USER      PID    PPID   VSIZE   RSS      WCHAN      PC          NAME
media      310      1    49076   9112  ffffffff b7566dbe S /system/bin/mediaserver
```

It is a native service which is defined in **/init.rc** file

```
service media /system/bin/mediaserver
    class main
    user media
    group audio camera inet net_bt net_bt_admin net_bt_acct drmrpc
    ioprio rt 4
```

MEDIASERVER

mediaserver runs with high privileges

```
service media /system/bin/mediaserver
    class main
    user media
    group audio camera inet net_bt net_bt_admin net_bt_acct drmrpc
    ioprio rt 4
```

mediaserver restarts when crashes *

Services

Services are programs which init launches and (optionally) restarts when they exit.
Services take the form of:

```
service <name> <pathname> [ <argument> ]*
    <option>
    <option>
    ...
```

* <https://android.googlesource.com/platform/system/core/+/master/init/readme.txt>

STAGEFRIGHT

- **libstagefright** processes media files inside **mediaserver** service
- **mediaserver** is a privileged native service
 - In some devices it also runs even with **system** privileges

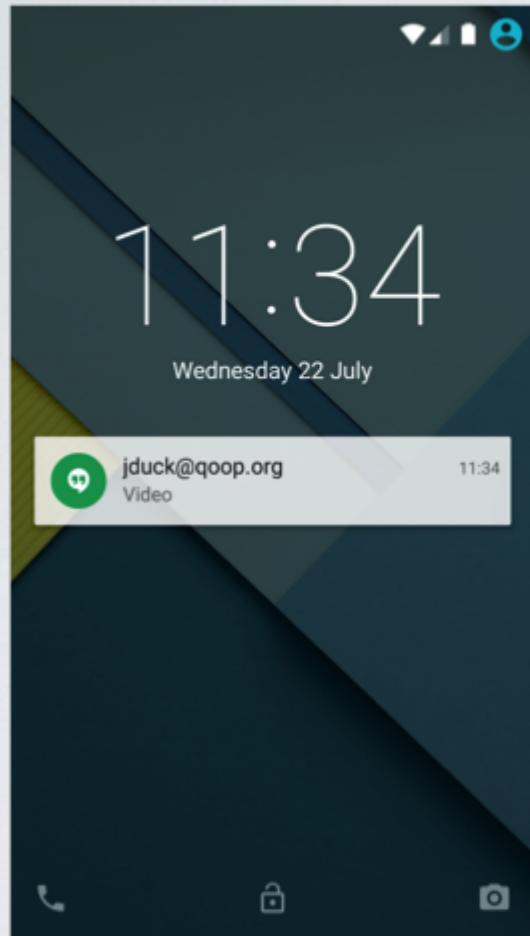
```
service media /system/bin/mediaserver
    class main
    user media
    group system audio camera inet net_bt net_bt_admin net_bw_acct drmrpc mediadrm sdcard_rw sdcard_r media_rw shell lgt_gid
    ioprio rt 4
```

* <https://android.googlesource.com/platform/system/core/+/master/init/readme.txt>

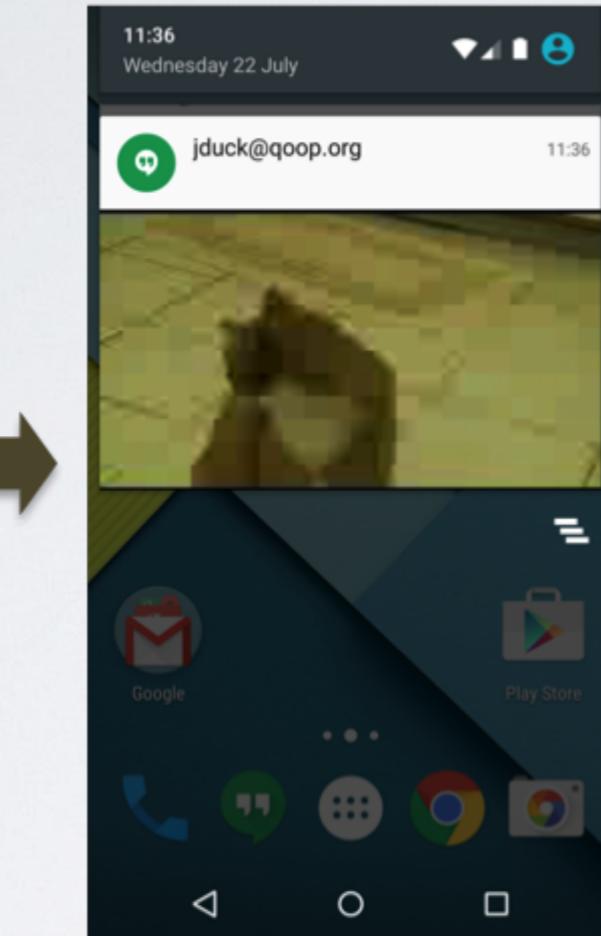
STAGEFRIGHT VULNERABILITY

- Lots of attack vectors present (||+)
 - MMS, <video> tag in a web page, downloaded the media file, e-mail, chat and messaging apps, NFC, Bluetooth, SDCard ...
- Exploited via playback or meta-data parse
- Binary parsers are mostly vulnerable
 - It is possible to exploit with different media formats like MP4, 3GPP, etc...
- Can be triggered with different ways
 - Rotating the screen, opening up the chat app, browsing the Gallery etc...

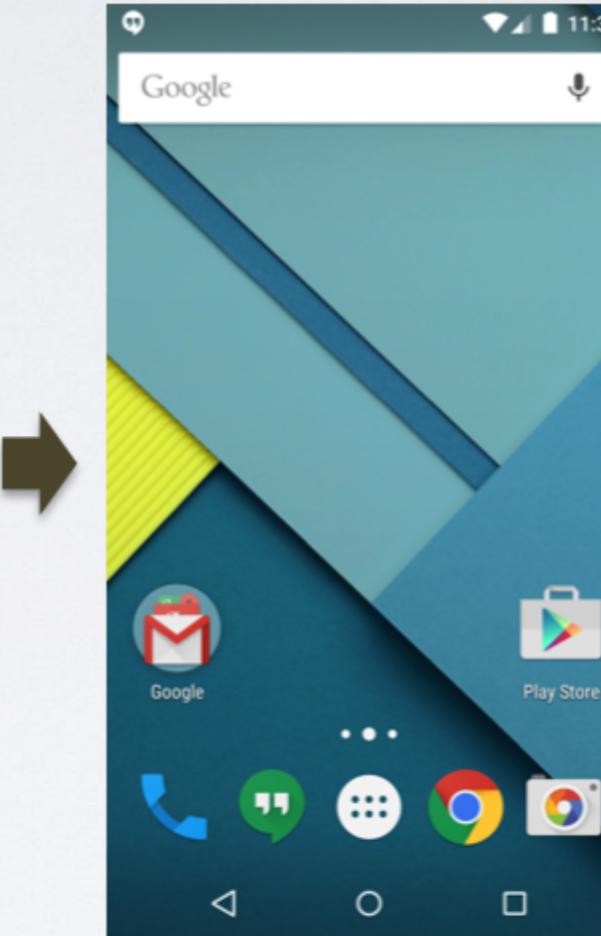
STAGEFRIGHT VULNERABILITY



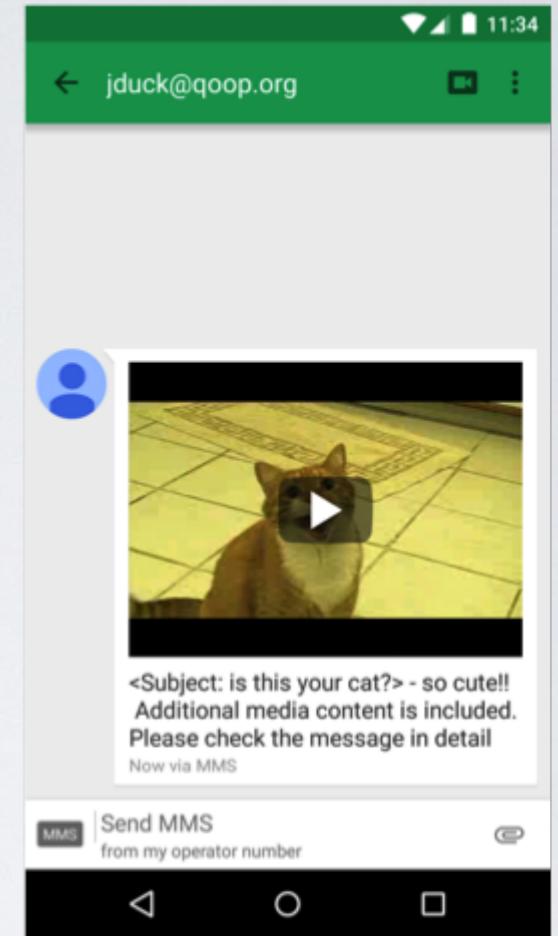
An MMS was received by Hangouts. At this point, an attacker may have already executed arbitrary code.



MMS notifications show a preview, which triggers the vulnerable code.



After unlocking the screen, no effects are apparent.



Viewing the MMS message triggers the vulnerable code again. Touching the video or rotating the screen triggers the vulnerable code again and again.



- **CVE-2015-1538**, P0006, Google Stagefright 'stsc' **MP4** Atom Integer Overflow Remote Code Execution
- **CVE-2015-1538**, P0004, Google Stagefright 'ctts' **MP4** Atom Integer Overflow Remote Code Execution
- **CVE-2015-1538**, P0004, Google Stagefright 'stts' **MP4** Atom Integer Overflow Remote Code Execution
- **CVE-2015-1538**, P0004, Google Stagefright 'stss' **MP4** Atom Integer Overflow Remote Code Execution
- **CVE-2015-1539**, P0007, Google Stagefright 'esds' **MP4** Atom Integer Underflow Remote Code Execution
- **CVE-2015-3827**, P0008, Google Stagefright 'covr' **MP4** Atom Integer Underflow Remote Code Execution
- **CVE-2015-3826**, P0009, Google Stagefright **3GPP** Metadata Buffer Overread
- **CVE-2015-3828**, P0010, Google Stagefright **3GPP** Integer Underflow Remote Code Execution
- **CVE-2015-3824**, P0011, Google Stagefright 'tx3g' **MP4** Atom Integer Overflow Remote Code Execution
- **CVE-2015-3829**, P0012, Google Stagefright 'covr' **MP4** Atom Integer Overflow Remote Code Execution

MPEG-4

- MPEG-4 files are made of units which are called **Atom** or **Box**

box		
Field	Type	Comment
Header	BOXHEADER	A consistent header that all boxes have
Payload	UI8[]	A number of bytes, the length of which is defined by the BOXHEADER
BOXHEADER		
Field	Type	Comment
TotalSize	UI32	The total size of the box in bytes, including this header
BoxType	UI32	The type of atom
ExtendedSize	If TotalSize equals 1 UI64	The total 64-bit length of the box in bytes, including this header

* http://www.adobe.com/content/dam/Adobe/en/devnet/flv/pdfs/video_file_format_spec_v10.pdf

MPEG-4

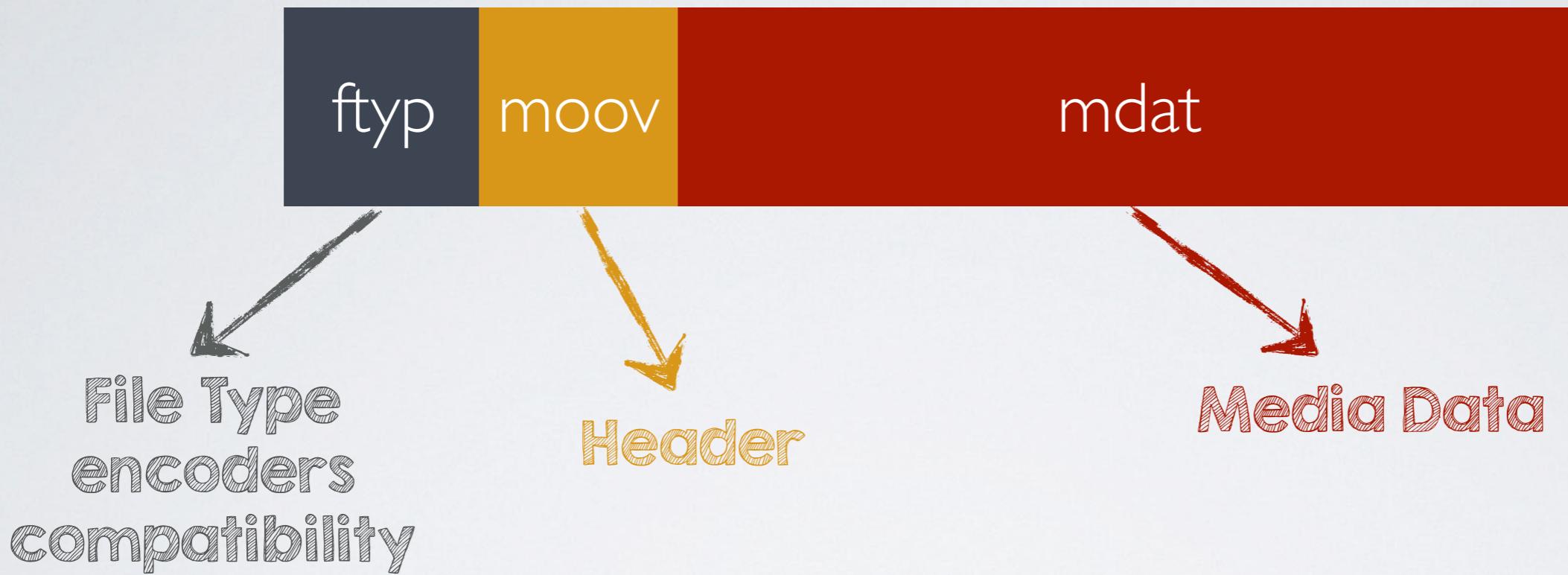
BOXHEADER

Field	Type	Comment
TotalSize	UI32	The total size of the box in bytes, including this header
BoxType	UI32	The type of atom
ExtendedSize	If TotalSize equals 1 UI64	The total 64-bit length of the box in bytes, including this header

- Atom type (BoxType) is a value which contains 4 characters (4 Byte):
 - E.g.: ftyp, moov, trak, covr, esds, mvhd...
- This 4 byte value is called FourCC (**Four Character Code**)

* http://www.adobe.com/content/dam/Adobe/en/devnet/flv/pdfs/video_file_format_spec_v10.pdf

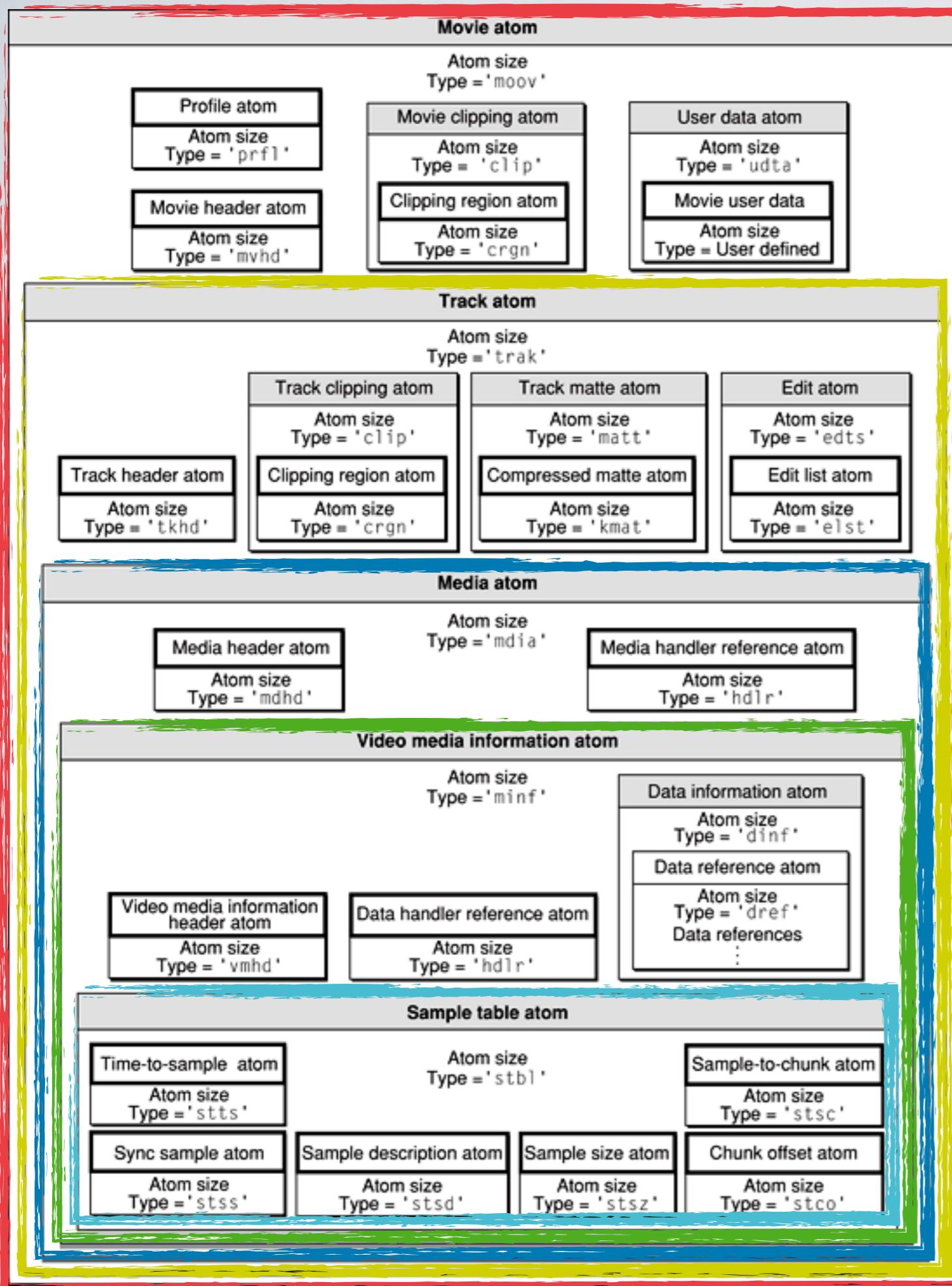
MPEG-4



- moov atom contains some nested atoms

* http://www.adobe.com/content/dam/Adobe/en/devnet/flv/pdfs/video_file_format_spec_v10.pdf

MOOV ATOM



```

▼ struct File
  ► struct FileTypeBox ftyp
  ► struct FreeBox free
  ► struct MediaDataBox mdat[0]
  ► struct MediaDataBox mdat[1]
  ▼ struct MovieBox moov
    uint32 size
    BoxType type
  ► struct MovieHeaderBox mvhd
  ▼ struct TrackBox trak[0]
    uint32 size
    BoxType type
  ► struct TrackHeaderBox tkhd
  ▼ struct MediaBox mdia
    uint32 size
    BoxType type
  ► struct MediaHeaderBox mdhd
  ► struct HandlerBox hdlr
  ▼ struct MediaInformationBox minf
    uint32 size
    BoxType type
  ► struct VideoMediaHeaderBox vmhd
  ► struct DataInformationBox dinf
  ▼ struct SampleTableBox stbl
    uint32 size
    BoxType type
  ► struct SampleDescriptionBox stsd
  ► struct TimeToSampleBox stts
  ► struct SyncSampleBox stss
  ► struct SampleToChunkBox stsc
  
```

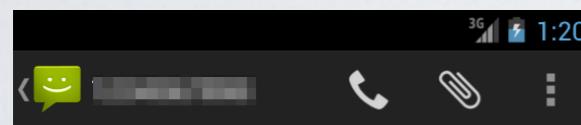
The code snippet defines the structure of the MOOV atom using C-like syntax. It includes nested structures for different types of boxes (FileTypeBox, FreeBox, MediaDataBox, MovieBox, etc.) and specific boxes like MovieHeaderBox, TrackHeaderBox, MediaHeaderBox, and HandlerBox. The `moov` box contains `mvhd`, `trak`, `mdia`, `mdhd`, `hdlr`, and `minf` boxes. The `minf` box contains `vmhd`, `dinf`, and `stbl`. The `stbl` box contains `stts`, `stss`, `stsd`, `sts`, `stsc`, and `stco`.



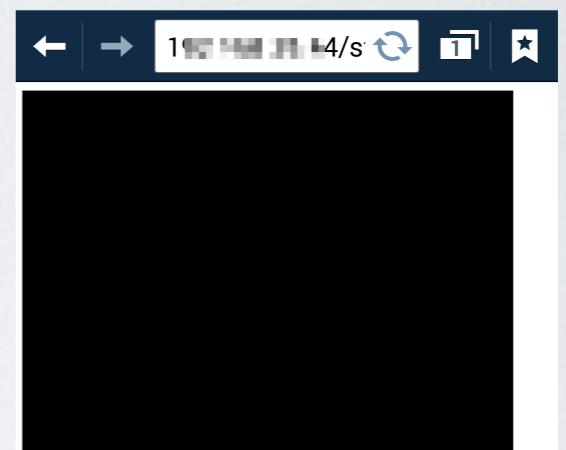
- **CVE-2015-1538**, P0006, Google Stagefright 'stsc' MP4 Atom Integer Overflow Remote Code Execution
- **CVE-2015-1538**, P0004, Google Stagefright 'ctts' MP4 Atom Integer Overflow Remote Code Execution
- **CVE-2015-1538**, P0004, Google Stagefright 'stts' MP4 Atom Integer Overflow Remote Code Execution
- **CVE-2015-1538**, P0004, Google Stagefright 'stss' MP4 Atom Integer Overflow Remote Code Execution
- **CVE-2015-1539**, P0007, Google Stagefright 'esds' MP4 Atom Integer Underflow Remote Code Execution
- **CVE-2015-3827**, P0008, Google Stagefright 'covr' MP4 Atom Integer Underflow Remote Code Execution
- **CVE-2015-3826**, P0009, Google Stagefright 3GPP Metadata Buffer Overread
- **CVE-2015-3828**, P0010, Google Stagefright 3GPP Integer Underflow Remote Code Execution
- **CVE-2015-3824**, P0011, Google Stagefright 'tx3g' MP4 Atom Integer Overflow Remote Code Execution
- **CVE-2015-3829**, P0012, Google Stagefright 'covr' MP4 Atom Integer Overflow Remote Code Execution

'STTS' ATOM INTEGER OVERFLOW

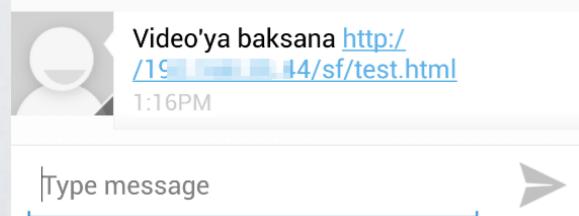
- PoC mp4 files: <https://s3.amazonaws.com/zhafiles/Zimperium-Handset-Alliance/ZHA-Crash-PoC.zip>
- sf-003.mp4 PoC mp4 file triggers 'stts' integer overflow.



<video> tag is introduced
with HTML5



```
<html>
<video width="320" height="240" autoplay>
  <source src="sf-003.mp4" type="video/mp4">
</video>
</html>
```



'STTS' ATOM INTEGER OVERFLOW

- Crash Logs are sent to **logcat** and recorded under **/data/tombstones/** directory with the naming **tombstone_xx**
- Attack vector in sf-003.mp4 file is Meta-data parsing

```
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***  
Build fingerprint: 'samsung/ja3gxx/ja3g:4.4.2/KOT49H/I9500XXUGNH2:user/release-keys'  
Revision: '10'  
pid: 7300, tid: 7300, name: mediaserver >>> /system/bin/mediaserver <<<  
signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 30da7f20  
r0 0000149a r1 30da2cb8 r2 b82cda30 r3 47102c91  
r4 00000b64 r5 00000000 r6 b82d2c50 r7 b6eff384  
...  
  
▼ backtrace:  
#00 pc 000a4b92 /system/lib/libstagefright.so (android::SampleTable::setTimeToSampleParams(long long, unsigned int)+169)  
#01 pc 00084b91 /system/lib/libstagefright.so (android::MPEG4Extractor::parseChunk(long long*, int)+3024)  
#02 pc 00084861 /system/lib/libstagefright.so (android::MPEG4Extractor::parseChunk(long long*, int)+3008)  
#03 pc 00084737 /system/lib/libstagefright.so (android::MPEG4Extractor::parseChunk(long long*, int)+2710)  
#04 pc 00083f6b /system/lib/libstagefright.so (android::MPEG4Extractor::parseChunk(long long*, int)+714)  
#05 pc 00083f6b /system/lib/libstagefright.so (android::MPEG4Extractor::parseChunk(long long*, int)+714)  
#06 pc 00083f6b /system/lib/libstagefright.so (android::MPEG4Extractor::parseChunk(long long*, int)+714)  
#07 pc 00083f6b /system/lib/libstagefright.so (android::MPEG4Extractor::parseChunk(long long*, int)+714)  
#08 pc 00083f6b /system/lib/libstagefright.so (android::MPEG4Extractor::parseChunk(long long*, int)+714)  
#09 pc 00085c99 /system/lib/libstagefright.so (android::MPEG4Extractor::readMetaData()+48)  
#10 pc 0008600d /system/lib/libstagefright.so (android::MPEG4Extractor::getMetaData()+8)  
#11 pc 000a6655 /system/lib/libstagefright.so (android::StagefrightMetadataRetriever::parseMetaData()+36)  
#12 pc 000a6c43 /system/lib/libstagefright.so (android::StagefrightMetadataRetriever::extractMetadata(int)+18)  
...
```

‘STTS’ ATOM INTEGER OVERFLOW

StageFrightMetadataRetriever.cpp

```
const char *StagefrightMetadataRetriever::extractMetadata(int keyCode) {
    if (mExtractor == NULL) {
        return NULL;
    }
    if (!mParsedMetaData) {
        parseMetaData();
        mParsedMetaData = true;
    }
    ...
}
```

```
void StagefrightMetadataRetriever::parseMetaData() {
    sp<MetaData> meta = mExtractor->getMetaData();
    if (meta == NULL) {
        ALOGV("extractor doesn't publish metadata, failed to initialize?");
        return;
    }
    ...
}
```

'STTS' ATOM INTEGER OVERFLOW

MPEG4Extractor.cpp

```
sp<MetaData> MPEG4Extractor::getMetaData() {
    status_t err;

    if ((err = readMetaData()) != OK) {
        return new Metadata;
    }
    return mFileMetaData;
}
```

```
status_t MPEG4Extractor::readMetaData() {
    if (mInitCheck != NO_INIT) {
        return mInitCheck;
    }
    off64_t offset = 0;
    status_t err;
    while (true) {
        err = parseChunk(&offset, 0);
        if (err == OK) {
            continue;
        }
    }
}
```

‘STTS’ ATOM INTEGER OVERFLOW

MPEG4Extractor.cpp

```
status_t MPEG4Extractor::parseChunk(off64_t *offset, int depth) {
    ALOGV("entering parseChunk %lld/%d", *offset, depth);
    uint32_t hdr[2];
    if (mDataSource->readAt(*offset, hdr, 8) < 8) {
        return ERROR_IO;
    }

    ...

    switch(chunk_type) {
        case FOURCC('m', 'o', 'o', 'v'):
        case FOURCC('t', 'r', 'a', 'k'):
        case FOURCC('m', 'd', 'i', 'a'):

        ...
        ...

        case FOURCC('s', 't', 't', 's'):
        {
            status_t err = mLastTrack->sampleTable->setTimeToSampleParams(data_offset, chunk_data_size);
            if (err != OK) {
                return err;
            }
            *offset += chunk_size;
            break;
        }

        ...
    }
}
```

‘STTS’ ATOM INTEGER OVERFLOW

SampleTable.cpp

```
status_t SampleTable::setTimeToSampleParams(off64_t data_offset, size_t data_size) {
    if (mTimeToSample != NULL || data_size < 8) {
        return ERROR_MALFORMED;
    }
    uint8_t header[8];
    if (mDataSource->readAt(data_offset, header, sizeof(header)) < (ssize_t)sizeof(header)) {
        return ERROR_IO;
    }
    if (U32_AT(header) != 0) {
        // Expected version = 0, flags = 0.
        return ERROR_MALFORMED;
    }

    mTimeToSampleCount = U32_AT(&header[4]);

    mTimeToSample = new uint32_t[mTimeToSampleCount * 2];

    size_t size = sizeof(uint32_t) * mTimeToSampleCount * 2;

    if (mDataSource->readAt(data_offset + 8, mTimeToSample, size) < (ssize_t)size) {
        return ERROR_IO;
    }
    for (uint32_t i = 0; i < mTimeToSampleCount * 2; ++i) {mTimeToSample[i] = ntohs(mTimeToSample[i]);}
    return OK;
}
```

'STTS' ATOM INTEGER OVERFLOW

SampleTable.cpp

< Android 4.4

```
mTimeToSampleCount = U32_AT(&header[4]); → Attacker  
Controlled  
mTimeToSample = new uint32_t[mTimeToSampleCount * 2];  
  
size_t size = sizeof(uint32_t) * mTimeToSampleCount * 2;  
  
if (mDataSource->readAt(  
    data_offset + 8, mTimeToSample, size) < (ssize_t)size) {  
    return ERROR_IO;  
}  
  
for (uint32_t i = 0; i < mTimeToSampleCount * 2; ++i) {  
    mTimeToSample[i] = ntohl(mTimeToSample[i]);  
}
```

'STTS' ATOM INTEGER OVERFLOW

SampleTable.cpp

< Android 4.4

```
mTimeToSampleCount = U32_AT(&header[4]);  
  
mTimeToSample = new uint32_t[mTimeToSampleCount * 2];  
  
size_t size = sizeof(uint32_t) * mTimeToSampleCount * 2;
```

Memory
allocation
for the
array



Allocated memory for the array is

$mTimeToSampleCount * 2 * sizeof(uint32_t)$

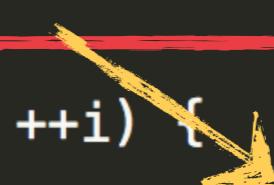
$mTimeToSampleCount * 2 * 4$

'STTS' ATOM INTEGER OVERFLOW

SampleTable.cpp

< Android 4.4

```
mTimeToSampleCount = U32_AT(&header[4]);  
  
mTimeToSample = new uint32_t[mTimeToSampleCount * 2];  
  
size_t size = sizeof(uint32_t) * mTimeToSampleCount * 2;  
  
if (mDataSource->readAt(  
    data_offset + 8, mTimeToSample, size) < (ssize_t)size) {  
    return ERROR_IO;  
}  
  
for (uint32_t i = 0; i < mTimeToSampleCount * 2; ++i) {  
    mTimeToSample[i] = ntohs(mTimeToSample[i]);  
}
```



Fill the array

'STTS' ATOM INTEGER OVERFLOW

SampleTable.cpp

< Android 4.4

```
mTimeToSampleCount = U32_AT(&header[4]);  
  
mTimeToSample = new uint32_t[mTimeToSampleCount * 2];  
  
size_t size = sizeof(uint32_t) * mTimeToSampleCount * 2;  
  
if (mDataSource->readAt(  
    data_offset + 8, mTimeToSample, size) < (ssize_t)size) {  
    return ERROR_IO;  
}  
  
for (uint32_t i = 0; i < mTimeToSampleCount * 2; ++i) {  
    mTimeToSample[i] = ntohs(mTimeToSample[i]);  
}
```

Endian
Swapping

'STTS' ATOM

stts box		
Field	Type	Comment
Header	BOXHEADER	BoxType = 'stts' (0x73747473)
Version	UI8	Expected to be 0
Flags	UI24	None defined, set to 0
Count	UI32	The number of STTSRECORD entries
Entries	STTSRECORD[Count]	An array of STTSRECORD structures

A regular mp4 file

Header Version Flags mTimeToSampleCount

	TILE POSITION												
10:0D90h:	40	10	E	04	40	00	00	00	00	40	00	00	00
10:0DA0h:	A8	01	00	04	68	EF	3C	80	00	00	00	18	73
10:0DB0h:	00	00	00	00	00	00	00	01	00	00	00	84	00
10:0DC0h:	00	00	00	14	73	74	73	73	00	00	00	00	01
10:0DD0h:	00	00	00	01	00	00	00	10	73	74	73	63	00
10:0DE0h:	00	00	00	01	00	00	00	01	00	00	00	01	01
10:0DF0h:	00	00	02	24	73	74	73	7A	00	00	00	00	00

'STTS' ATOM INTEGER OVERFLOW

SampleTable.cpp

< Android 4.4

```
mTimeToSampleCount = U32_AT(&header[4]); → mTimeToSampleCount = 1
Allocated memory
mTimeToSample = new uint32_t[mTimeToSampleCount * 2]; → for
mTimeToSample
size_t size = sizeof(uint32_t) * mTimeToSampleCount * 2;   | * 2 * 4 = 8

if (mDataSource->readAt(
    data_offset + 8, mTimeToSample, size) < (ssize_t)size) {
    return ERROR_IO;
}
Loop bound
mTimeToSampleCount * 2
for (uint32_t i = 0; i < mTimeToSampleCount * 2; ++i) {      = 2
    mTimeToSample[i] = ntohs(mTimeToSample[i]);
}
```

So, mTimeToSampleCount < Loop Bound < Array size

'STTS' ATOM

stts box		
Field	Type	Comment
Header	BOXHEADER	BoxType = 'stts' (0x73747473)
Version	UI8	Expected to be 0
Flags	UI24	None defined, set to 0
Count	UI32	The number of STTSRECORD entries
Entries	STTSRECORD[Count]	An array of STTSRECORD structures

sf-003.mp4 PoC file

	Header	Version	Flags	mTimeToSampleCount	
0B30h:	73 00 00 D9	00 03 80 80 80 22 00 00 00 04 80 80	S...U...€€€"....€€		
0B40h:	80 14 40 15	00 00 DE 00 00 20 D8 00 00 22 DE 05	€.@...þ.. 0.."þ.		
0B50h:	80 80 80 02	15 88 06 80 80 01 02 00 00 00 20	€€€..^..€€€.....		
0B60h:	73 74 74 73	00 00 00 00 40 00 00 02 00 00 00 10	stts....@.....		
0B70h:	00 00 04 00	00 00 00 01 00 00 03 00 00 00 00 58X		
0B80h:	73 74 73 7A	00 00 00 00 00 00 00 00 00 00 00 11	stsز.....		
0B90h:	00 00 00 DF	00 00 00 85 00 00 00 87 00 00 00 83	b + f		

'STTS' ATOM INTEGER OVERFLOW

SampleTable.cpp

< Android 4.4

```
mTimeToSampleCount = U32_AT(&header[4]); → mTimeToSampleCount  
= 0x40000002  
  
mTimeToSample = new uint32_t[mTimeToSampleCount * 2]; → Allocated memory  
for  
size_t size = sizeof(uint32_t) * mTimeToSampleCount * 2; mTimeToSample  
= 0x40000002 * 4 * 2  
if (mDataSource->readAt(  
    data_offset + 8, mTimeToSample, size) < (ssize_t)size) {  
    return ERROR_IO;  
}  
  
for (uint32_t i = 0; i < mTimeToSampleCount * 2; ++i) → Loop bound  
    mTimeToSample[i] = ntohs(mTimeToSample[i]);  
}
```

Lets check: **mTimeToSampleCount < Loop Bound < Array size**



'STTS' ATOM INTEGER OVERFLOW



'STTS' ATOM INTEGER OVERFLOW

... if you multiply two 32-bit integers, you get a 32-bit integer again, losing the upper 32 bits of the result

mTimeToSampleCount = 0x40000002

Allocated memory for mTimeToSample

$$\begin{aligned}0x40000002 * 4 * 2 \\= 0x100000010 \\= 0x00000010\end{aligned}$$

* <http://www.fefe.de/intof.html>

'STTS' ATOM INTEGER OVERFLOW

... if you multiply two 32-bit integers, you get a 32-bit integer again, losing the upper 32 bits of the result

mTimeToSampleCount = 0x40000002

**Allocated memory for
mTimeToSample = 0x00000010**

**Loop bound
0x40000002 * 2 = 0x80000004**

Check again: **mTimeToSampleCount < Loop Bound < Array size**



'STTS' ATOM INTEGER OVERFLOW

SampleTable.cpp

< Android 4.4

```
mTimeToSampleCount = U32_AT(&header[4]);  
  
mTimeToSample = new uint32_t[mTimeToSampleCount * 2];  
  
size_t size = sizeof(uint32_t) * mTimeToSampleCount * 2;  
  
if (mDataSource->readAt(  
    data_offset + 8, mTimeToSample, size) < (ssize_t)size) {  
    return ERROR_IO;  
}  
  
for (uint32_t i = 0; i < mTimeToSampleCount * 2; ++i) {  
    mTimeToSample[i] = ntohs(mTimeToSample[i]);  
}
```

Loop bound is too high, but size of the array is too small

'STTS' ATOM INTEGER OVERFLOW

Maximum value of a 32bit
Unsigned Integer:
0xFFFFFFFF

If

`mTimeToSampleCount >= 0x20000000`
integer overflow occurs

$$0x20000000 * 4 * 2 = 100000000$$

‘STTS’ ATOM INTEGER OVERFLOW

SampleTable.cpp

Android 5.0

```
mTimeToSampleCount = U32_AT(&header[4]);  
+ uint64_t allocSize = mTimeToSampleCount * 2 * sizeof(uint32_t);  
+  
+ if (allocSize > SIZE_MAX) {  
+     return ERROR_OUT_OF_RANGE;  
+ }  
  
mTimeToSample = new uint32_t[mTimeToSampleCount * 2];
```



This check fails to catch Integer Overflow

```
}  
  
for (uint32_t i = 0; i < mTimeToSampleCount * 2; ++i) {  
    mTimeToSample[i] = ntohs(mTimeToSample[i]);  
}
```

‘STTS’ ATOM INTEGER OVERFLOW

SampleTable.cpp

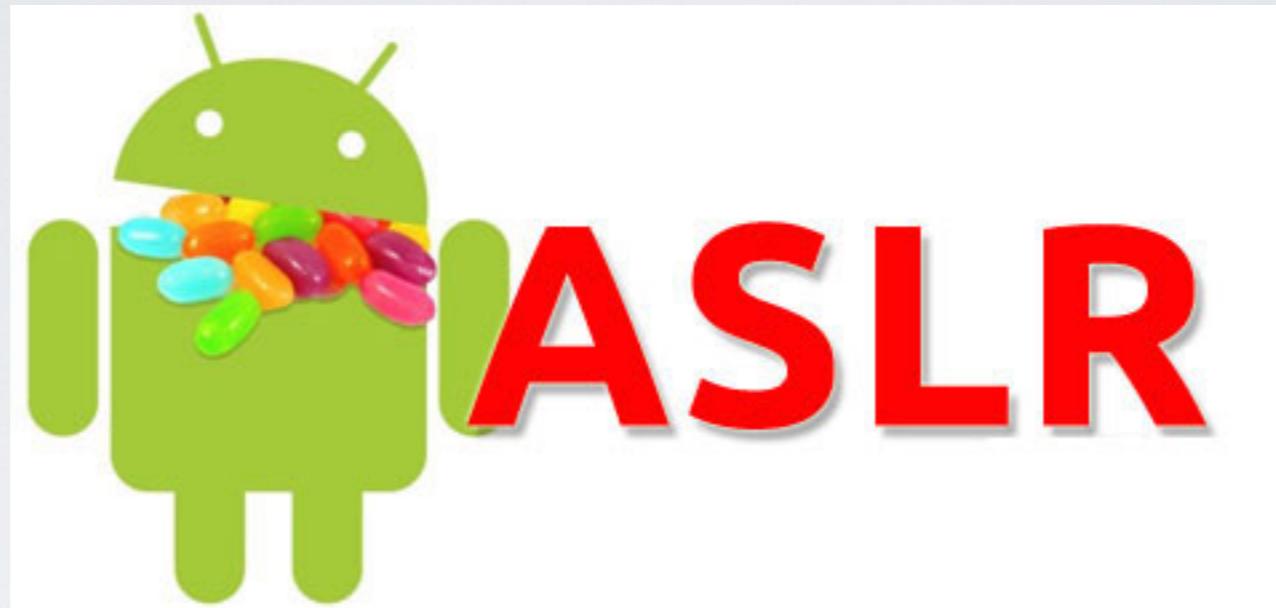
Android 5.1.1 r5

```
mTimeToSampleCount = U32_AT(&header[4]);  
- uint64_t allocSize = mTimeToSampleCount * 2 * sizeof(uint32_t);  
+ uint64_t allocSize = (uint64_t)mTimeToSampleCount * 2 * sizeof(uint32_t);  
  
if (allocSize > SIZE_MAX) {  
    return ERROR_OUT_OF_RANGE;  
}  
  
mTimeToSample = new uint32_t[mTimeToSampleCount * 2];  
  
size_t size = sizeof(uint32_t) * mTimeToSampleCount * 2;  
  
if (mDataSource->readAt(  
    data_offset + 8, mTimeToSample, size) < (ssize_t)size) {  
    return ERROR_IO;  
}  
  
for (uint32_t i = 0; i < mTimeToSampleCount * 2; ++i) {  
    mTimeToSample[i] = ntohs(mTimeToSample[i]);  
}
```

Android 5.0

Android 5.1.1 r5

STAGEFRIGHT VS ASLR



- ASLR is introduced with Android 4.0 Ice Cream
 - Stagefright can be successfully exploited for Ice Cream Sandwich
 - ASLR Bypass is required for the versions \geq Android 4.0

STAGEFRIGHT VULNERABILITY





ARE YOU
VULNERABLE?



stagefright detector - Google

<https://play.google.com/store/search?q=stagefright%20detector>

Google play

stagefright detector

Store Apps Movies & TV Music Books Newsstand Devices

Search All results

Apps

Stagefright Detector Lookout Mobile Security Zimperium INC. ESET Stagefright Detector ESET

★★★★★ FREE ★★★★★ FREE ★★★★★ FREE

Stagefright Detector

Your device is **vulnerable** to Stagefright

When the app is available immediately

If you recommend auto-restarting SMS a

HOW CAN I PROTECT?

Learn more Share your feedback

Download Link

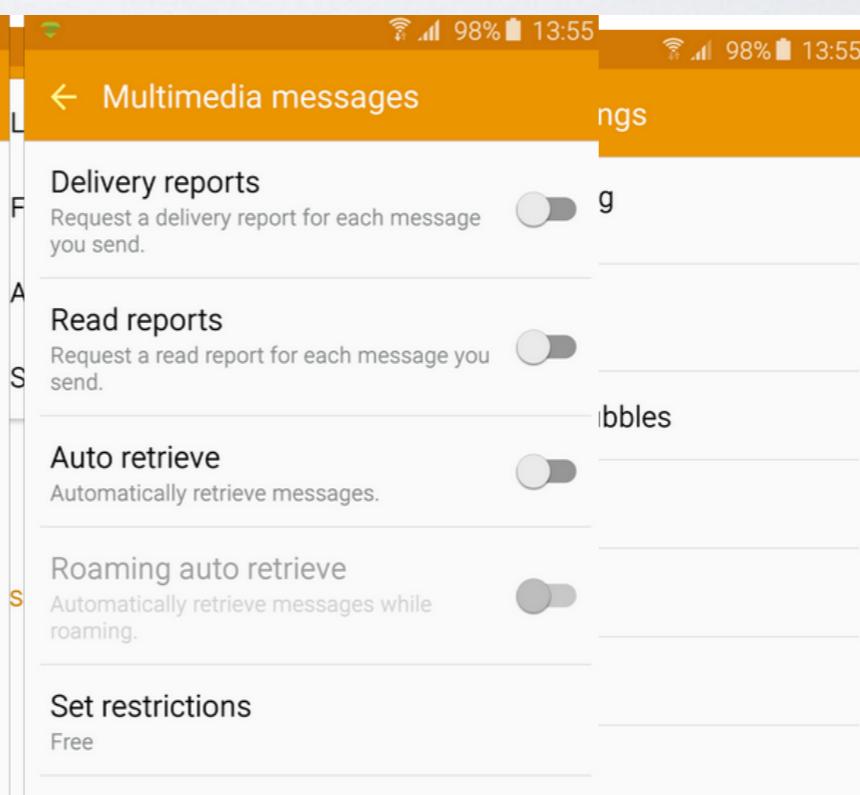
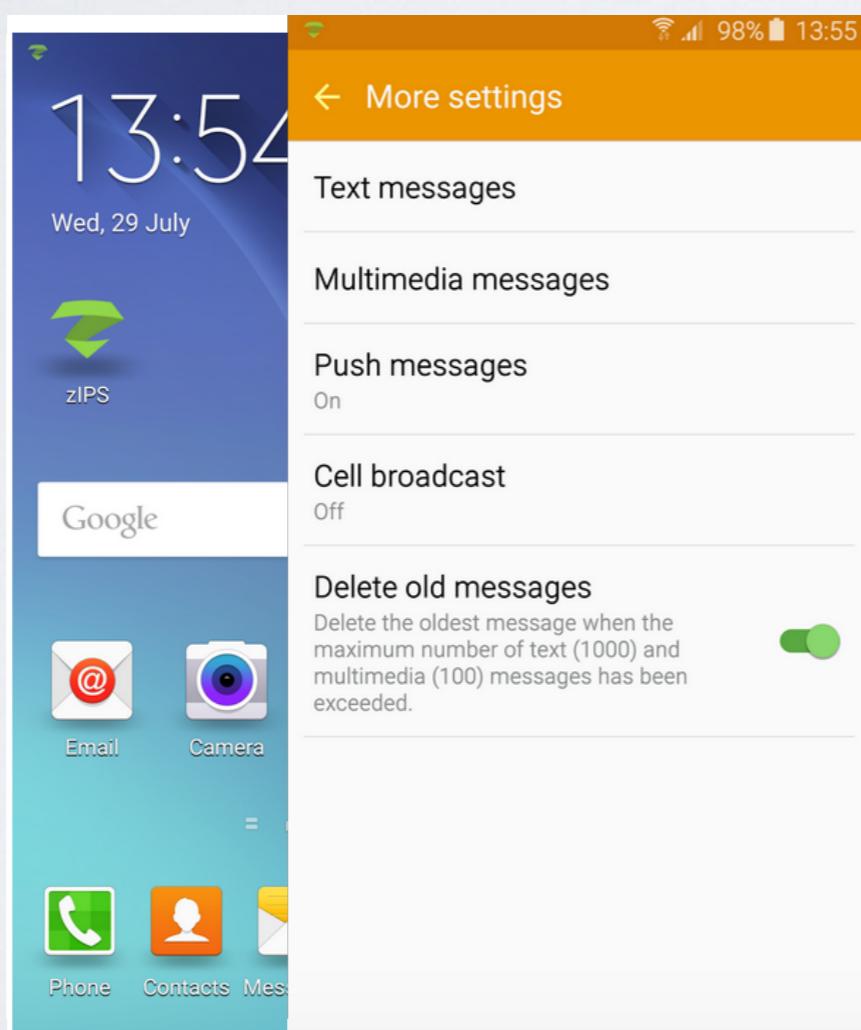
Increase your protection against other threats on Android

DOWNLOAD ESET MOBILE SECURITY

HOW TO PROTECT

- Update your device for the latest Android version which must be >= 5.1.1_r5
- Disable MMS auto-fetch feature (For Hangouts and Messages)

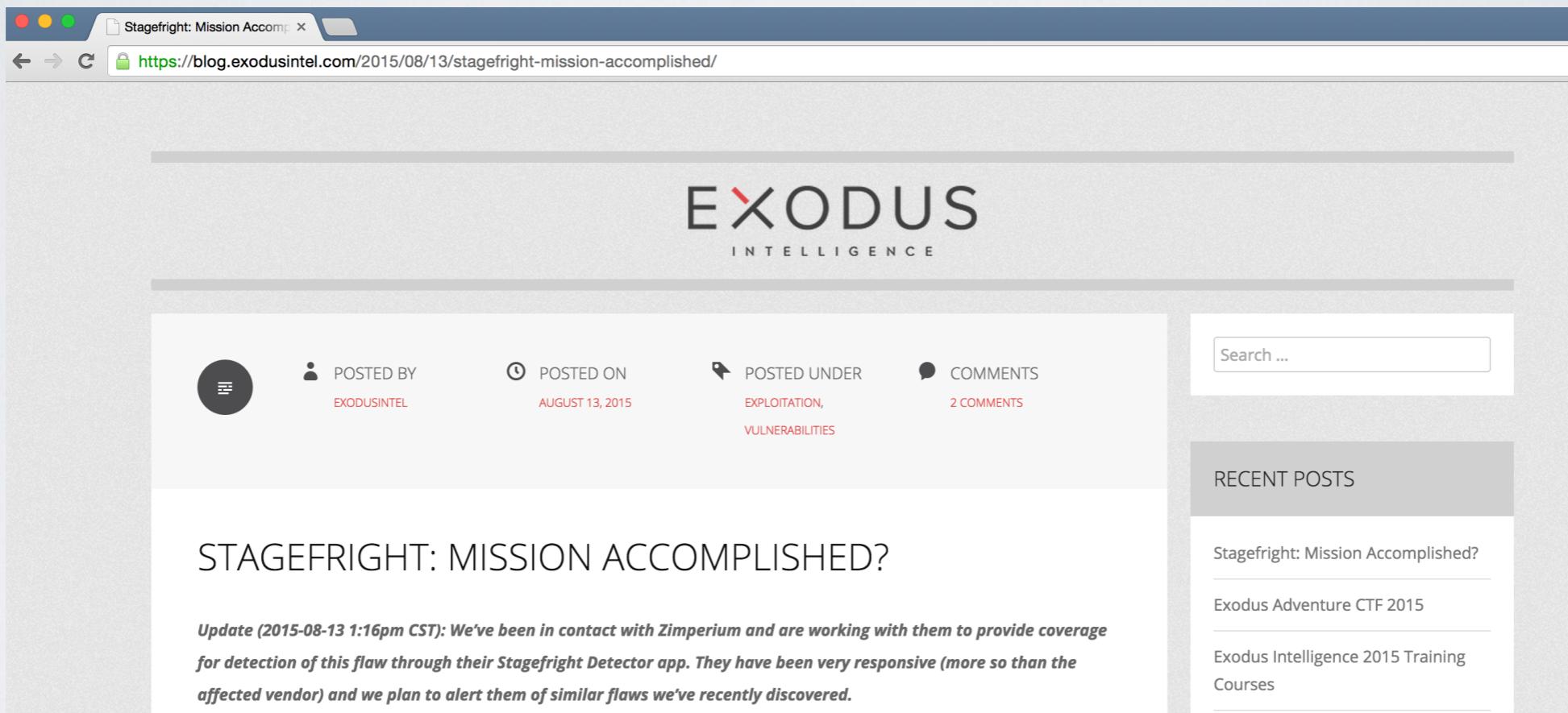
HANGOUT
↓
OPTIONS
↓
SETTINGS
↓
SMS



MESSAGES
↓
MORE
↓
SETTINGS
↓
MORE
SETTINGS
↓
MMS

UPDATES REALLY SOLVE THE PROBLEM?

- EXODUS Intelligence does not agree with that



- Problem related to “**CVE-2015-3824**, P0011, Google Stagefright ‘tx3g’ MP4 Atom Integer Overflow Remote Code Execution” remained unfixed

* <https://blog.exodusintel.com/2015/08/13/stagefright-mission-accomplished/>

STAGEFRIGHT PUBLIC EXPLOIT?

- **@jduck** put the PoC Exploit code in his private repo and...



Joshua J. Drake
@jduck

The #Stagefright public exploit release is held up by marketing. I'm doing a FrienNDA beta [github.com/jduck/cve-2015...](https://github.com/jduck/cve-2015-1538-1), tell me your github!

02/09/15 13:33

- **Update 09.09.15:** @jduck's private repo is publicly available

<https://github.com/jduck/cve-2015-1538-1>

MORE READING ON STAGEFRIGHT

Mobile Security by Oguzhan

<https://mobile-security.zeef.com/oguzhan.topgul>

Mobile Security

by Oguzhan Topgul

Login Make list

Books



- 1 [iOS Hacker's Handbook](#)
- 2 [Android Hacker's Handbook](#)
- 3 [Hacking and Securing iOS Applications](#)
- 4 [Learning Pentesting for Android Devices](#)
- 5 [Android Security Cookbook](#)
- 6 [Android Security Internals](#)
- 7 [iOS Application Security](#)
- 8 [Android Malware and Analysis](#)

▼ 10 more

+ ↻ ↺

StageFright & Android Media Vulns



- 1 [@jduck's BLACKHAT USA 2015 Presentation](#)
- 2 [Stagefright Vulnerability Details](#)
- 3 [Stagefright Vulnerability Disclosure](#)
- 4 [Stagefright vulnerability bulletin by Wooyun](#)
- 5 [StageFrighth Report by Baidu Security Lab](#)
- 6 [Revisited Stagefright vulnerability POC and EXP by Wooyun](#)
- 7 [CryptoGirl on StageFright](#)
- 8 [An overview of Stagefright player](#)

Mobile App Debugging



- 1 [Android Hacker's Handbook Chapter 7](#)
- 2 [Hacking and Securing iOS Applications Chapter 8](#)
- 3 [Debug Android APKs with Eclipse and DDMS](#)
- 4 [Debugging iOS applications using LLDB](#)
- 5 [Debugging iOS applications using LLDB Continued](#)
- 6 [Debugging Dalvik programs with IDA](#)
- 7 [Introduction to gikdbg.art \(aka Android Ollydbg\)](#)
- 8 [GikDbg Debugger](#)

<https://mobile-security.zeef.com/oguzhan.topgul>

One more thing...



OWASP TURKEY MOBILE SECURITY WORKSHOP

- 14 October 2015

The screenshot shows a web browser window with the following details:

- Title Bar:** Web Güvenlik Topluluğu » M X
- Address Bar:** www.webguvenligi.org/haberler/mobile-security-workshop-2015.html
- Page Content:**
 - Header:** Web Güvenlik Topluluğu (<http://www.webguvenligi.org>)
 - Navigation Bar:** Ana Sayfa, OWASP-TR ve WGT, Liste, Projeler, Belgeler, Çeviriler, Etkinlikler, Sözlük
 - Section:** Mobile Security Workshop 2015 (14th Aug, 2015)
 - Text:** OWASP Turkey Chapter is organizing a Mobile Security Workshop that will bring mobile software developers, mobile security professionals and academics together. The workshop will provide a common platform to share ideas and discuss latest developments in the security field of mobile devices and mobile applications.
 - Text:** The workshop sponsored by AVEA R&D Labs and Türk Telekom Group will take place on 14th October. You can find the Call for Presentation in this [link](#). The attendance registration will be open from 6th October.
 - Text:** Further details can be followed via our webpage and Twitter [account](#). For your questions, please contact with us via etkinlik@webguvenligi.org

* <http://www.webguvenligi.org/haberler/mobile-security-workshop-2015.html>

RERERENCES

- **StageFright - Zimperium Blog:** <https://blog.zimperium.com/experts-found-a-unicorn-in-the-heart-of-android/>
- **StageFright Vulnerability Details:** <https://blog.zimperium.com/stagefright-vulnerability-details-stagefright-detector-tool-released/>
- **Stagefriht Detailed Explanation:** <http://blog.fortinet.com/post/cryptogirl-on-stagefright-a-detailed-explanation>
- **FTYP Atom:** <http://www.ftyps.com/what.html>
- **F4V/MP4 File Format:** http://www.adobe.com/content/dam/Adobe/en/devnet/flv/pdfs/video_file_format_spec_v10.pdf
- **StageFrightMetadataReceiver.cpp:** https://android.googlesource.com/platform/frameworks/av/+/android-4.4.2_r2.0.1/media/libstagefright/StagefrightMetadataRetriever.cpp
- **MPEG4Extractor.cpp:** https://android.googlesource.com/platform/frameworks/av/+/android-4.4.2_r2.0.1/media/libstagefright/MPEG4Extractor.cpp
- **SampleTable.cpp:** https://android.googlesource.com/platform/frameworks/av/+/android-4.4.2_r2.0.1/media/libstagefright/SampleTable.cpp
- **Android 5.0 Integer Overflow Fix :** <https://android.googlesource.com/platform/frameworks/av/+/f106b19%5E/>
- **Android's StageFright Media Player Architecture:** <https://quandarypeak.com/2013/08/androids-stagefright-media-player-architecture/>

THANKS



Oğuzhan Topgül

@oguzhantopgul

www.oguzhantopgul.com