out-of-bounds memory access in Android media could be exploited to get media_server permission

Through this vulnerability, normal Apps in Android can corrupt the heap in media_server. By manipulating the heap carefully, normal Apps can get media_server permission. The PoC attached just show the corruption of the heap.

the vulnerable code is in ICrypto.cpp

http://androidxref.com/5.0.0_r2/xref/frameworks/av/media/libmedia/ICrypto.cpp#254

```
224     case DECRYPT:
225     {
226         CHECK_INTERFACE(ICrypto, data, reply);
227
228         bool secure = data.readInt32() != 0;
229         CryptoPlugin::Mode mode = (CryptoPlugin::Mode)data.readInt32();
230
231         uint8_t key[16];
232         data.read(key, sizeof(key));
233
234         uint8_t iv[16];
235         data.read(iv, sizeof(iv));
236
237         size_t totalSize = data.readInt32();      ----------------------> totalSize is read from Parcel, can be controlled by normal Apps, totalSize will be used as the size of decryption destination buffer
238         void *srcData = malloc(totalSize);
239         data.read(srcData, totalSize);
240
241         int32_t numSubSamples = data.readInt32();
242
243         CryptoPlugin::SubSample *subSamples =
244             new CryptoPlugin::SubSample[numSubSamples];
245
246         data.read(
247                 subSamples,                  ----------------> the length of data copied to  dstPtr(decryption destination buffer) is decide by the content in subSamples, which is also controlled by normall Apps.
248                 sizeof(CryptoPlugin::SubSample) * numSubSamples);
249
250         void *dstPtr;
251         if (secure) {
252             dstPtr = reinterpret_cast<void *>(static_cast<uintptr_t>(data.readInt64())); ----> can't  believe this ,pass a point from  other processes,but  this is not for this report.
253         } else {
254             dstPtr = malloc(totalSize);         --------------------> the allocated size of decryption destination buffer is up to totalSize
255         }
256
257         AString errorDetailMsg;
258         ssize_t result = decrypt(
259                 secure,
260                 key,
261                 iv,
262                 mode,
263                 srcData,
```

```
264                 subSamples, numSubSamples,
265                 dstPtr,
266                 &errorDetailMsg);
```

the decrypt function in line 258 above will call the decrypt function of relative crypto plugin, let take libdrmclearkeyplugin.so for example, which is supported by nexus 5 by default.
the decrypt function in drmclearkeyplugin is as follows:

```
ssize_t CryptoPlugin::decrypt(bool secure, const KeyId keyId, const Iv iv,
36                      Mode mode, const void* srcPtr,
37                      const SubSample* subSamples, size_t numSubSamples,
38                      void* dstPtr, AString* errorDetailMsg) {
39    if (secure) {
40        errorDetailMsg->setTo("Secure decryption is not supported with "
41                      "ClearKey.");
42        return android::ERROR_DRM_CANNOT_HANDLE;
43    }
44
45    if (mode == kMode_Unencrypted) {
46        size_t offset = 0;
47        for (size_t i = 0; i < numSubSamples; ++i) {
48            const SubSample& subSample = subSamples[i];
49
50            if (subSample.mNumBytesOfEncryptedData != 0) {
51                errorDetailMsg->setTo(
52                      "Encrypted subsamples found in allegedly unencrypted "
53                      "data.");
54                return android::ERROR_DRM_DECRYPT;
55            }
56
57            if (subSample.mNumBytesOfClearData != 0) {
58                memcpy(reinterpret_cast<uint8_t*>(dstPtr) + offset,
59                      reinterpret_cast<const uint8_t*>(srcPtr) + offset,
60                      subSample.mNumBytesOfClearData);      -------->here is the heap corruption position.
61                offset += subSample.mNumBytesOfClearData;
62            }
63        }
64        return static_cast<ssize_t>(offset);
65    } else if (mode == kMode_AES_CTR) {
```

Because the allocated buffer size and the copied data size are both controllable by other processes, by carefully manipulating the heap content, we can control the content copied to the destination buffer,(such as by call attachBuffer of IGraphicBufferProducer to manipulate the content of heap) so we can corrupt the heap with controllable data, it's exploitable.

the crash backtrace is as follows:

```
D/ClearKeyCryptoPlugin( 5332): Instantiating Session Library Singleton.
F/libc    ( 5332): Fatal signal 11 (SIGSEGV), code 1, fault addr 0xb4500000 in tid 5369 (Binder_3)
I/DEBUG   (  184): *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
I/DEBUG   (  184): Build fingerprint:
'Android/aosp_hammerhead/hammerhead:5.0/LRX21Q/ggong11131350:userdebug/test-keys'
I/DEBUG   (  184): Revision: '11'
I/DEBUG   (  184): ABI: 'arm'
I/DEBUG   (  184): pid: 5332, tid: 5369, name: Binder_3  >>> /system/bin/mediaserver <<<
I/DEBUG   (  184): signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0xb4500000
W/NativeCrashListener(  676): Couldn't find ProcessRecord for pid 5332
I/DEBUG   (  184):     r0 b44fffe0  r1 b44ffff8  r2 0ff0d040  r3 00000000
E/DEBUG   (  184): AM write failure (32 / Broken pipe)
```

I/DEBUG (184): r4 00000000 r5 b4413050 r6 b48fdbc8 r7 00000000
I/DEBUG (184): r8 b4414058 r9 00000002 sl b4414060 fp 00000002
I/DEBUG (184): ip b45fbdd4 sp b48fdb20 lr b45f852d pc b6ea0f0e cpsr 200b0030
I/DEBUG (184):
I/DEBUG (184): backtrace:
I/DEBUG (184): #00 pc 00012f0e /system/lib/libc.so (__memcpy_base+81)
I/DEBUG (184): #01 pc 00004529 /system/vendor/lib/mediadrm/libdrmclearkeyplugin.so (clearkeydrm::CryptoPlugin::decrypt(bool, unsigned char const*, unsigned char const*, android::CryptoPlugin::Mode, void const*, android::CryptoPlugin::SubSample const*, unsigned int, void*, android::AString*)+68)
I/DEBUG (184): #02 pc 00038e15 /system/lib/libmediaplayerservice.so (android::Crypto::decrypt(bool, unsigned char const*, unsigned char const*, android::CryptoPlugin::Mode, void const*, android::CryptoPlugin::SubSample const*, unsigned int, void*, android::AString*)+62)
I/DEBUG (184): #03 pc 00052213 /system/lib/libmedia.so (android::BnCrypto::onTransact(unsigned int, android::Parcel const&, android::Parcel*, unsigned int)+442)
I/DEBUG (184): #04 pc 0001a6d9 /system/lib/libbinder.so (android::BBinder::transact(unsigned int, android::Parcel const&, android::Parcel*, unsigned int)+60)
I/DEBUG (184): #05 pc 0001f787 /system/lib/libbinder.so (android::IPCThreadState::executeCommand(int)+582)
I/DEBUG (184): #06 pc 0001f8ab /system/lib/libbinder.so (android::IPCThreadState::getAndExecuteCommand()+38)
I/DEBUG (184): #07 pc 0001f8ed /system/lib/libbinder.so (android::IPCThreadState::joinThreadPool(bool)+48)
I/DEBUG (184): #08 pc 00023a5b /system/lib/libbinder.so
I/DEBUG (184): #09 pc 000104d5 /system/lib/libutils.so (android::Thread::_threadLoop(void*)+112)
I/DEBUG (184): #10 pc 00010045 /system/lib/libutils.so
I/DEBUG (184): #11 pc 000162e3 /system/lib/libc.so (__pthread_start(void*)+30)
I/DEBUG (184): #12 pc 000142d3 /system/lib/libc.so (__start_thread+6)
I/DEBUG (184):
I/DEBUG (184): Tombstone written to: /data/tombstones/tombstone_01
E/SharedPreferencesImpl(676): Couldn't create directory for SharedPreferences file shared_prefs/log_files.xml
I/BootReceiver(676): Copying /data/tombstones/tombstone_01 to DropBox (SYSTEM_TOMBSTONE)
I/ServiceManager(169): service 'media.sound_trigger_hw' died
I/ServiceManager(169): service 'media.audio_flinger' died
I/ServiceManager(169): service 'media.player' died
I/ServiceManager(169): service 'media.camera' died
I/ServiceManager(169): service 'media.audio_policy' died