

0x00 前言

跟着[1]调试CVE-2017-13253 ,
[1]使用的调试环境为nexus，HAL的实现没有分开，我的调试环境为pixel，HAL的实现分开了，我在这里整理成一篇文章，记录踩过的坑。
CVE-2017-13253 为Android Drm服务中的堆溢出漏洞。

0x01 理论

1 DRM

Android Drm 属于 Android Native 多媒体框架中的一部分。在播放受DRM保护的内容 如Google Play电影中的影片时，会使用DRM服务器。该服务器会采用安全方式对加密的数据进行解密，因此可以访问证书和密钥存储以及其他敏感组件。但由于供应商依赖关系，DRM 的架构如下图所示：

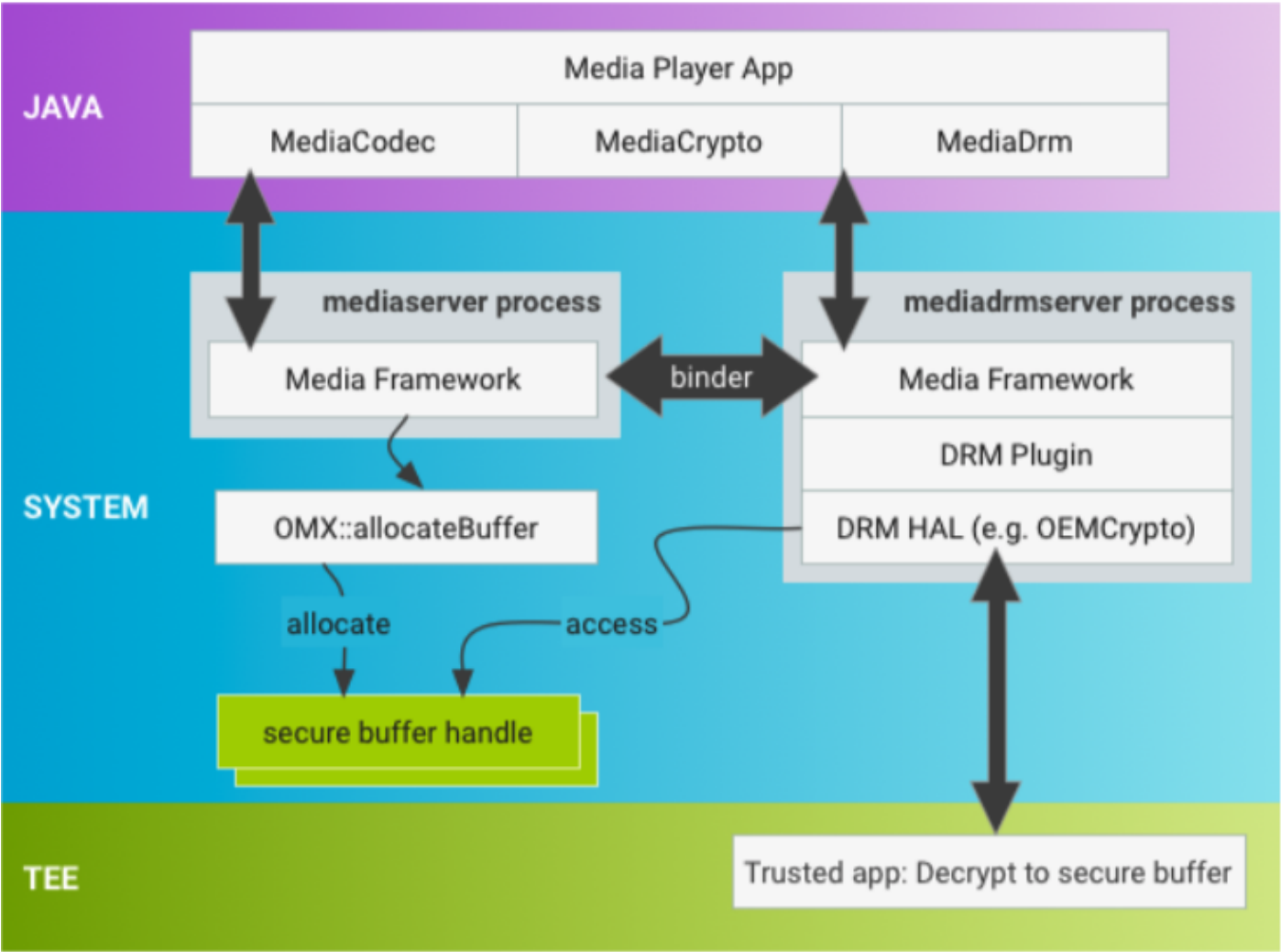


图 3. Android 7.0 及更高版本中的 mediaserver 中的缓冲区分配。

2 Crypto Plugins

Crypto Plugins
为DRM方案之一，在Android术语中，每个DRM方案的处理程序称为插件。供应商负责提供这些插件，但供应商可以使用AOSP中一些有用的代码。例如，AOSP包含ClearKey DRM方案插件的完整开源实现。
java层 所涉及到的参数变量

Fields	
public byte[]	iv A 16-byte initialization vector
public byte[]	key A 16-byte key id
public int	mode The type of encryption that has been applied, see MediaCodec.CRYPTO_MODE_UNENCRYPTED , MediaCodec.CRYPTO_MODE_AES_CTR and MediaCodec.CRYPTO_MODE_AES_CBC
public int[]	numBytesOfClearData The number of leading unencrypted bytes in each subSample.
public int[]	numBytesOfEncryptedData The number of trailing encrypted bytes in each subSample.
public int	numSubSamples The number of subSamples that make up the buffer's contents.

<https://developer.android.com/reference/android/media/MediaCodec.CryptoInfo.html#numSubSamples>

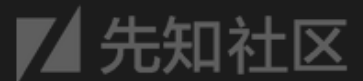
Binder层的解释 这里引用原文的图


```

struct SourceBuffer {
    sp<IMemory> mSharedMemory;
    int32_t mHeapSeqNum;
};

...
struct DestinationBuffer {
    DestinationType mType;
    native_handle_t *mHandle;
    sp<IMemory> mSharedMemory;
};

```



这里相关的结构成员是mHeapSeqNum和两个mSharedMemory成员(DestinationBuffer的其余部分是在目标未存储为共享内存的情况下，与此漏洞无关的情况)。这里使用mHeapSeqNum是这样的内存的标识符，之前使用ICrypto的方法（称为setHeap）共享。两个mSharedMemory成员仅表示堆内缓冲区的偏移和大小。这意味着尽管mHeapSeqNum值得注意的，参数结构的某些部分有点奇怪。

mSharedMemory是一个IMemory，它实际上连接到自己的堆，并且应该在其中表示一个缓冲区，但是这个堆被忽略，偏移量和大小用于mHeapSeqNum堆。源结构中也由Treble的主要重新架构师的一部分而制作

4 源码

source.mSharedMemory 是什么含义呢，我猜是图里面的这个 shared memory？仅表示堆内缓冲区的偏移和大小，实际上连接到自己的堆，应该用于表示一个缓冲区，但是这个堆被忽略，偏移量和大小用于mHeapSeqNum堆。在ICrypto interface 的“服务端”代码，用于验证共享内存缓冲区的subsample。这段代码检查 subsample Encrypt Clear data都加起来 sum<= SIZE_MAX subsampleSizes == totalSize totalSize <= source.mSharedMemory->size() Offset <= source.mSharedMemory->size() - totalSize

```

size_t sumSubsampleSizes = 0;
bool overflow = false;
for (int32_t i = 0; i < numSubSamples; ++i) {
    CryptoPlugin::SubSample &ss = subSamples[i];
    if (sumSubsampleSizes <= SIZE_MAX - ss.mNumBytesOfEncryptedData) {
        sumSubsampleSizes += ss.mNumBytesOfEncryptedData;
    } else {
        overflow = true;
    }
    if (sumSubsampleSizes <= SIZE_MAX - ss.mNumBytesOfClearData) {
        sumSubsampleSizes += ss.mNumBytesOfClearData;
    } else {
        overflow = true;
    }
}

if (overflow || sumSubsampleSizes != totalSize) {
    result = -EINVAL;
} else if (totalSize > source.mSharedMemory->size()) {
    result = -EINVAL;
} else if ((size_t)offset > source.mSharedMemory->size() - totalSize) {
    result = -EINVAL;
} else {
    result = decrypt(key, iv, mode, pattern, source, offset,
                    subSamples, numSubSamples, destination, &errorDetailMsg);
}

```

http://androidxref.com/8.0.0_r4/xref/frameworks/av/drm/libmediadrm/ICrypto.cpp#370

接着进入 decrypt，对数据流进行序列化并继续进行验证

```

if (source.offset + offset + source.size > sourceBase->getSize()) {
    _hidl_cb(Status::ERROR_DRM_CANNOT_HANDLE, 0, "invalid buffer size");
    return Void();
}

uint8_t *base = static_cast<uint8_t *>
    (static_cast<void *>(sourceBase->getPointer()));
void *srcPtr = static_cast<void *>(base + source.offset + offset);

...
if (destBuffer.offset + destBuffer.size > destBase->getSize()) {
    _hidl_cb(Status::ERROR_DRM_CANNOT_HANDLE, 0, "invalid buffer size");
    return Void();
}
destPtr = static_cast<void *>(base + destination.nonsecureMemory.offset);

...
ssize_t result = mLegacyPlugin->decrypt(secure, keyId.data(), iv.data(),
    legacyMode, legacyPattern, srcPtr, legacySubSamples,
    subSamples.size(), destPtr, &detailMessage);

```

http://androidxref.com/8.0.0_r4/xref/hardware/interfaces/drm/1.0/default/CryptoPlugin.cpp#111

sourceBase 指向 mSharedBuffer 上的 source

...

```

111 if (source.offset + offset + source.size > sourceBase->getSize()) {
112 _hidl_cb(Status::ERROR_DRM_CANNOT_HANDLE, 0, "invalid buffer size");
113 return Void();
114 }
...

```

此处的本意是拷贝的 size = offset + 所有的 subSamples。

```

...
uint8_t *base = static_cast<uint8_t *>(static_cast<void *>(sourceBase->getPointer()));
Const SHaredBuufer& destBuffer = destination.nonsecureMemory;
Sp<IMemory> destBase = mSharedBufferMap[destBuffer.bufferId]
■■■
source.offset + offset + source.size > sourceBase->getSize()
source.offset ■source ■heap■■■■■■■■■■
destBuffer.offset+destBuffer.size>destBase->getSize()
...

```

第一处检查是offsets和buffer size没有超出堆的size。SourceBase是堆，而source现在是source.mSharedMemory。此处的本意是检查偏移+需要拷贝的数据是否有超出这段sharedMemory的size??

另一处检查类似，但是在destBuffer上执行。destBuffer 是 destination.mSharedMemory、destBase “same heap as”sourceBase。也就是说 destBuffer 和 sourceBuffer 都在同一段SharedMemory上。

最终，每个buffer简化为一个指向内存的指针，偏移现在是指针的一部分。
最后一处代码

```

if (mode == kMode_Unencrypted) {
    size_t offset = 0;
    for (size_t i = 0; i < numSubSamples; ++i) {
        const SubSample& subSample = subSamples[i];

        if (subSample.mNumBytesOfEncryptedData != 0) {
            errorDetailMsg->setTo(
                "Encrypted subsamples found in allegedly unencrypted "
                "data.");
            return android::ERROR_DRM_DECRYPT;
        }

        if (subSample.mNumBytesOfClearData != 0) {
            memcpy(reinterpret_cast<uint8_t*>(dstPtr) + offset,
                reinterpret_cast<const uint8_t*>(srcPtr) + offset,
                subSample.mNumBytesOfClearData);
            offset += subSample.mNumBytesOfClearData;
        }
    }
    return static_cast<ssize_t>(offset);
}

```



http://androidxref.com/8.0.0_r4/xref/frameworks/av/drm/mediadrm/plugins/clearkey/CryptoPlugin.cpp#45

当 满足mode == kMode_Unencrypted时，
会执行到
...

```

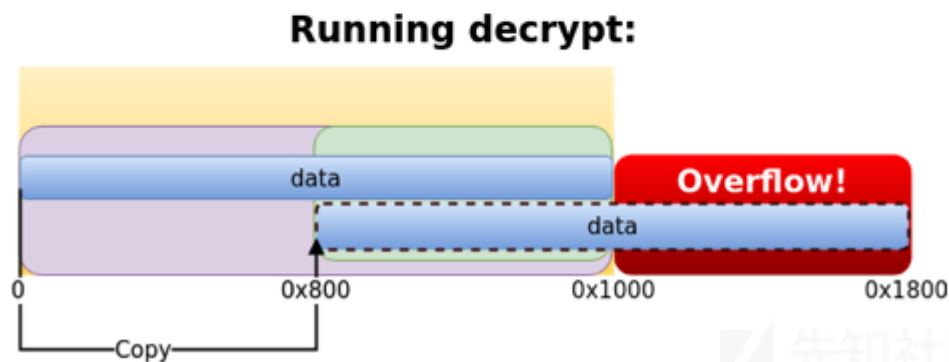
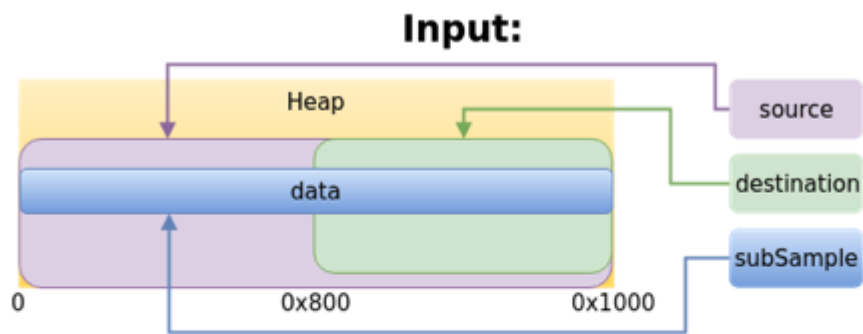
memcpy(reinterpret_cast<uint8_t*>(dstPtr) + offset,
reinterpret_cast<const uint8_t*>(srcPtr) + offset,
subSample.mNumBytesOfClearData);
...

```

4 漏洞处

漏洞原因：没有检查 要复制的数据+目标缓冲区的位置是否超过 堆的 size，属于检查不完整。

只有一个简单的检查 源缓冲区，BnCrypto的第三次检查了这一点，下一处的检查 考虑了源缓冲区+offset。唯一检查和目标缓冲区有关的是第二处的检查，但是太简单不足以阻止这种问题。第二处的检查，确认目标缓冲区在堆内，且没有超出缓冲区的边界。



0x02 动态调试

动态调试[6][7][8]

在poc作者的github[9]里说到运行的结果应该是

1. 如果在2018年3月之后的Android版本 decrypt 会返回bad_value (-22)。
2. 如果没有crash (overwritten data 是可写的) decrypt return 他copy的数据的量。
3. 如果供应商将HAL实现为单独的进程? (例如Pixel 2) , 则解密应该返回UNKNOWN_ERROR (-32)。
4. 如果供应商在同一过程中实施HAL? (例如Nexus 5X) , 则解密应返回0。

HAL的实现会影响是否造成Crash。此处没有返回UNKNOWN_ERROR

本漏洞涉及三处函数，分别是

android::BnCrypto::onTransact 序列化+验证

Android::CryptoHal::decrypt 序列化+验证

Clearkeydrm::CryptoPlugin::decrypt memcpy

前面两次进入BnCrypto::onTransact函数是调用createPlugin和setHeap，第三次调用decrypt对android::BnCrypto::onTransact、android::CryptoHal::decrypt的调试部分可参考[1]调试。


```

r0 : 0x0
r1 : 0x0
r2 : 0xc
r3 : 0xe86cd0b0 -> 0x00400000
r4 : 0x7708800 -> 0x00000000
r5 : 0x0
r6 : 0xe770883c -> 0x00000000
r7 : 0xe770883c -> 0x00000000
r8 : 0xe7708928 -> 0xe0797f90 -> 0xe912cf04 -> 0xe911c07d -> <android:ThreadPool->ThreadPool()>+ push {r4, r}
r9 : 0x1737
r10 : 0xe84dc600 -> 0xe9237e9c -> 0xe922e040 -> <android:CryptoTotal->CryptoTotal()>+ push {r6, r}
r11 : 0xe7708800 -> 0x00000000
r12 : 0xe92388ec -> 0xe740b790 -> <android:Parcel:>readInt32()+> ldr r1, [r0, #16]
sp : 0xe7708740 -> 0xe87d082c -> 0x00000000
lir : 0xe9226000 -> 0x0a20004a ("??")
qpc : 0xe9226000 -> <android:BnCrypto:>onTransact(unsigned=0) mov r9, r0
qpcr : TCM00 Fast Interrupt overflow carry zero NEGATIVE
----- stack -----
0xe7708740|+0x0000: 0xe87d082c -> 0x00000000 <- $ip
0xe7708744|+0x0004: 0x0000720a -> 0x00000000
0xe7708748|+0x0008: 0x00000000 -> 0x00000000
0xe770874c|+0x000c: 0x00000000 -> 0x00000000
0xe7708750|+0x0010: 0x00000000 -> 0x00000000
0xe7708754|+0x0014: 0x0004720a -> 0x00000000
0xe7708758|+0x0018: 0x00000000 -> 0x00000000
0xe770875c|+0x001c: 0x00000000 -> <android:IPCThreadState:>waitForResponse(android:Parcel*,0) cmp r0, #0
----- code:arm:ARM -----
0xe92260f0 <android:BnCrypto:>onTransact(unsigned=0) movs r2, #0
0xe92260fa <android:BnCrypto:>onTransact(unsigned=0) blx 0xe92230f4 <?ZmkAndroidParcel:>readPaj()blt
0xe92260fe <android:BnCrypto:>onTransact(unsigned=0) mov r0, r7
0xe92260ff <android:BnCrypto:>onTransact(unsigned=0) mov r0, r0
0xe9226100 <android:BnCrypto:>onTransact(unsigned=0) mov r0, #0
0xe9226108 <android:BnCrypto:>onTransact(unsigned=0) add r0, sp, #40 ; 0x28
0xe922610e <android:BnCrypto:>onTransact(unsigned=0) str r0, [sp, #68] ; 0x44
0xe9226114 <android:BnCrypto:>onTransact(unsigned=0) mov r0, r0
0xe922611e <android:BnCrypto:>onTransact(unsigned=0) mov r1, r7
----- source:frameworks/av/d... cpg-324 -----
319 data.read(key, sizeof(key));
320
321 uint8_t iv[16];
322 data.read(iv, sizeof(iv));
323
324 size_t totalSize = data.readInt32();
325
326 SourceBuffer source;
327
328 source.mSharedMemory =
329 interface_cast<Memory>(data.readStrongBinder());
----- threads -----
[#0] Id 1, Name: "mediadrmserver", stopped, reason: SINGLE STEP
[#1] Id 2, Name: "Binder:706_1", stopped, reason: SINGLE STEP
[#2] Id 3, Name: "Binder:706_2", stopped, reason: SINGLE STEP
[#3] Id 4, Name: "Binder:706_3", stopped, reason: SINGLE STEP
[#4] Id 5, Name: "Binder:706_4", stopped, reason: SINGLE STEP
[#5] Id 6, Name: "Binder:706_5", stopped, reason: SINGLE STEP
[#6] Id 7, Name: "Binder:706_6", stopped, reason: SINGLE STEP
[#7] Id 8, Name: "Binder:706_7", stopped, reason: SINGLE STEP
[#8] Id 9, Name: "Binder:706_8", stopped, reason: SINGLE STEP
[#9] Id 10, Name: "Binder:706_9", stopped, reason: SINGLE STEP
[#10] Id 11, Name: "Binder:706_A", stopped, reason: SINGLE STEP
[#11] Id 12, Name: "Binder:706_B", stopped, reason: SINGLE STEP
[#12] Id 13, Name: "Binder:706_C", stopped, reason: SINGLE STEP
[#13] Id 14, Name: "Binder:706_D", stopped, reason: SINGLE STEP
[#14] Id 15, Name: "Binder:706_E", stopped, reason: SINGLE STEP
[#15] Id 16, Name: "Binder:706_F", stopped, reason: SINGLE STEP
[#16] Id 17, Name: "Binder:706_10", stopped, reason: SINGLE STEP
----- trace -----
[#0] 0xe922609a-><android:BnCrypto:>onTransact(this=0xe84dc600, cmd=ciphertextOut*, data=@0xe770883c, reply=@0xe7708800, flags=ciphertextOut*)
[#1] 0xe9108070-><android:BnCrypto:>onTransact(this=0xe84dc604, cmd=drv, data=@0xe770883c, reply=@0xe7708800, flags=ciphertextOut*)
[#2] 0xe9108000-><android:IPCThreadState:>executeCommand(this=0xe87d0800, cmd=ciphertextOut*)
[#3] 0xe91060c4-><android:IPCThreadState:>getAndExecuteCommand(this=0xe87d0800)
[#4] Command 'context' failed to execute properly, reason: access outside bounds of object referenced via synthetic pointer

```

```

00000000 bt
00000001 0x0e8ef3e5a in android:hardware:drm:V1.0/BpHwCryptoPlugin:~hidl_decrypt(android:hardware:~Interfaces, android:hardware:details:~HidlInstrumentor, bool, android:hardware:hidl_1_array<unsigned char, 16u> const&, android:hardware:hidl_array<unsigned char, 16u> const&, android:hardware:drm:V1.0/Mode, android:hardware:drm:V1.0/Pattern const&, android:hw_
00000002 nd:hardware:hidl_vec<android:hardware:drm:V1.0/SubSample> const&, android:hardware:drm:V1.0/SharedBuffer const&, unsigned long long, android:hardware:drm:V1.0/DestinationBuffer co
00000003 nst&, std::1.1:~function<void (android:hardware:drm:V1.0/Status, unsigned int, android:hardware:hidl_string const&)> (&hidl_this=<optimized out>, &hidl_this_instrumentor=0xe848c918
00000004 secure=<optimized out>, keyId=..., iv=..., mode=<optimized out>, pattern=..., subSamples=..., source=..., offset=<optimized out>, destination=..., &hidl_cb=...) at out/soong/intermedia
00000005 tes/hardware/interfaces/drm/v1.0/android.hardware.drm.v1.0/gen/android.hardware.drm/v1.0/CryptoPluginAll.cpp:507
00000006 01 0x0e84f0f6 in android:hardware:drm:V1.0/BpHwCryptoPlugin:~decrypt(bool, android:hardware:hidl_array<unsigned char, 16u> const&, android:hardware:hidl_array<unsigned char, 16u>
00000007 const&, android:hardware:drm:V1.0/Mode, android:hardware:drm:V1.0/iPattern const&, android:hardware:hidl_vec<android:hardware:drm:V1.0/SubSample> const&, android:hw_
00000008 nd:hardware:hidl_vec<android:hardware:drm:V1.0/SharedBuffer const&, unsigned long long, and oid:hardware:drm:V1.0/DestinationBuffer const&, std::1.1:~function<void (android:hardwa
00000009 re:hardware:hidl_string const&)> (&this=0xe848c918, secure=<error reading variable: access outside bounds of object referenced via synthetic pointer>, keyId=..., iv=..., m=se<optimi
0000000a zed out>, pattern=..., subSamples=..., source=..., offset=<optimized out>, destination=..., &hidl_cb=...) at out/soong/intermediates/hardware/interfaces/drm/v1.0/android.hardware.drm.v1.0_
0000000b gen/android.hardware.drm/v1.0/CryptoPluginAll.cpp:576
0000000c 02 0x0e922f0ee in android:CryptHidl:~decrypt (this=<optimized out>, keyId=<optimized out>, iv=<optimized out>, mode=android:CryptPlugin::KMode Unencrypted, pattern=..., source=..., off
0000000d set=<optimized out>, subSamples=<optimized out>, numSubSamples=<optimized out>, destination=..., replyDetail=0xe760b768) at frameworks/av/drm/libmediadrm/CryptoHidl.cpp:349
0000000e 03 0x0e922681a in android:~BnCrypto:~onTransact (this=0xe87aeae0, code=<optimized out>, data=..., reply=0xe760b808, flags=<optimized out>) at frameworks/av/drm/libmediadrm/ICrypto.cpp:393
0000000f 04 0x0e9100878 in android:~BBinder:~transact (this=0xe87aeae4, code=0x6, data=..., reply=0xe760b808, flags=<optimized out>) at frameworks/native/libs/binder/Binder.cpp:129
00000010 05 0x0e9106a0e in android:IPCThreadState:~executeCommand (this=0xe8484500, cmd=<optimized out>) at frameworks/native/libs/binder/IPCThreadState.cpp:1080
00000011 06 0x0e91066c4 in android:IPCThreadState:~getAndExecuteCommand (this=0xe8484500) at frameworks/native/libs/binder/IPCThreadState.cpp:449
00000012 07 0x0e9106bb6 in android:IPCThreadState:~joinThreadPool (this=0xe8484500, isMain=<error reading variable: access outside bounds of object referenced via synthetic pointer>) at framew
00000013 rk/native/libs/binder/IPCThreadState.cpp:503
00000014 08 0x0e911c0d8 in android:~ThreadPool:~threadLoop (this=0xe879fa20) at frameworks/native/libs/binder/ProcessState.cpp:61
00000015 09 0x0e8e9719e in android:~Thread:~threadLoop (user=0xe879fa20) at system/core/libutils/Threads.cpp:744
00000016 10 0x0e88e6ee8 in ~pthread_start (arg=0xe760b970) at bionic/libc/bionic/pthread_create.cpp:226
00000017 11 0x0e88ba1da in ~_start_thread (fn=0xe88e6ed1 <_pthread_start(void*)>, arg=0xe760b970) at bionic/libc/bionic/clone.cpp:47
00000018 12 0x00000000 in ?? ()

```

1.通过Android.mk or Android.bp文件 找到local module

```
gongqi@hulk:~$ adb shell ps | grep 'drm'
```

Process Name	PPID	PID	UID	State	TTY	Session	Service
media	652	1	17428	3692	ptrace_stop	efcef032	t android.hardware.drm@1.0-service
drm	703	1	17772	6256	binder_wait_for_work	e986c644	S drmservice
media	706	1	28280	4292	binder_wait_for_work	e88e8644	S mediadrmservice


```

0xff9626b8|+0x0008: 0x00000000
0xff9626bc|+0x000c: 0x00000000
0xff9626c0|+0x0010: 0x60d0f931
0xff9626c4|+0x0014: 0x00000008
0xff9626c8|+0x0018: 0x60d0f931
0xff9626cc|+0x001c: 0xff96278c -> 0xfeaf56 -> 0x74534100

----- code:arm:ARM -----
0xefcef026 <clearkeydrm::CryptoPlugin::decrypt(bool,+0> lsls r7, r0, #3
0xefcef028 <clearkeydrm::CryptoPlugin::decrypt(bool,+0> ldreq r0, [r0, #4]
0xefcef02a <clearkeydrm::CryptoPlugin::decrypt(bool,+0> cbnz r0, 0xefcef0a0 <clearkeydrm::CryptoPlugin::decrypt(bool, unsigned char const*, unsigned char const*, android
::CryptoPlugin::Mode, android::CryptoPlugin::Pattern const&, void const*, android::CryptoPlugin::SubSample const&, unsigned int, void*, android::AString*)+180> ; unpredic
table <IT:req>
->0xefcef032 <clearkeydrm::CryptoPlugin::decrypt(bool,+0> add.w r0, r11, r5
0xefcef036 <clearkeydrm::CryptoPlugin::decrypt(bool,+0> add.w r1, r10, r5
0xefcef03a <clearkeydrm::CryptoPlugin::decrypt(bool,+0> mov.w r8, r7, lsl #1
0xefcef03e <clearkeydrm::CryptoPlugin::decrypt(bool,+0> blx 0xefcee0fc <__aeabi_memcpy@plt>
0xefcef042 <clearkeydrm::CryptoPlugin::decrypt(bool,+0> ldr.w r0, [r6, r8, lsl #2]
0xefcef046 <clearkeydrm::CryptoPlugin::decrypt(bool,+0> add r5, r0

----- source:frameworks/av/d[...]cpp:58 -----
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);
61         offset += subSample.mNumBytesOfClearData;
62     }
63 }

----- threads -----
[#0] Id 1, Name: "drm@1.0-service", stopped, reason: BREAKPOINT
[#1] Id 2, Name: "HwBinder:652_1", stopped, reason: BREAKPOINT
[#2] Id 3, Name: "HwBinder:652_2", stopped, reason: BREAKPOINT
[#3] Id 4, Name: "HwBinder:652_3", stopped, reason: BREAKPOINT
[#4] Id 5, Name: "HwBinder:652_4", stopped, reason: BREAKPOINT
[#5] Id 6, Name: "HwBinder:652_5", stopped, reason: BREAKPOINT
[#6] Id 7, Name: "HwBinder:652_6", stopped, reason: BREAKPOINT

----- trace -----
[#0] 0xefcef032-><clearkeydrm::CryptoPlugin::decrypt(this=<optimized out>, secure=<optimized out>, keyId=<optimized out>, iv=0xffff11a0 "", mode=android::CryptoPlugin::kMode_Unenc
rypted, srcPtr=0xf0add000, subSamples=<optimized out>, numSubSamples=0x1, dstPtr=<optimized out>, errorDetailMsg=<optimized out>)
[1] Command 'context' failed to execute properly, reason: access outside bounds of object referenced via synthetic pointer
gef> 1
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);
61         offset += subSample.mNumBytesOfClearData;
62     }

```



```

$R0 : 0xf0adefff -> 0x41414141 ("AAAA"?)
$R1 : 0xf0add000 -> 0x41414141 ("AAAA"?)
$R2 : 0x2000
$R3 : 0xffff11a0 -> 0x00000000
$R4 : 0x1
$R5 : 0x0
$R6 : 0xf0217198 -> 0x00002000
$R7 : 0x0
$R8 : 0xff962888 -> 0x00000000
$R9 : 0xfffff82a
$R10: 0xf0add000 -> 0x41414141 ("AAAA"?)
$R11: 0xf0adefff -> 0x41414141 ("AAAA"?)
$R12: 0xefe53ca8 -> 0xf0897c3d -> <android::RefBase::decStrong(void+0> push {r4, r5, r6, lr}
$SP : 0xff9626b0 -> 0x00000000
$LR : 0xefe4f3b9 -> <android::hardware::drm::V1_0::implementation::CryptoPlugin::decrypt(bool,+0> mov r5, r0
$PC : 0xefcef03a -> <clearkeydrm::CryptoPlugin::decrypt(bool,+0> mov.w r8, r7, lsl #1
$CPSR: [THUMB fast interrupt overflow carry ZERO negative]

```



0x03 结论与补丁

1

受影响的流程取决于供应商如何实现。如果供应商未将HAL分成不同的进程，则mediadrmserver受影响。如果供应商讲HAL分开，那么使用默认加密插件的Crypto插件的每个

2

对这个漏洞的补丁很简单，增加对要传递数据检查的完整性，dest->size()
目标缓冲区堆的size，检查目标缓冲区的偏移+要拷贝的数据是否超出目标缓冲区堆的总和。

0x04 参考

- [1] <https://bbs.pediy.com/thread-225398.htm>
- [2] <http://wangkuiwu.github.io/page2/> Android Binder机制
- [3] <https://www.anquanke.com/vul/id/1124887> 漏洞简介
- [4] <https://source.android.com/devices/drm> DRM，安卓版权管理框架
- [5] <https://developer.android.com/reference/android/media/MediaDrm.html> MediaDrm
- [6] <https://source.android.com/setup/build/building> 编译流程
- [7] <https://developers.google.com/android/drivers>

[8]<https://source.android.com/setup/start/build-numbers#source-code-tags-and-builds>
[9]<https://github.com/tamirzb/CVE-2017-13253>

点击收藏 | 0 关注 | 1

[上一篇：渗透测试：从XSLT注入到Gets...](#) [下一篇：渗透测试：从XSLT注入到Gets...](#)

1. 0 条回复
- 动动手指，沙发就是你的了！

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)