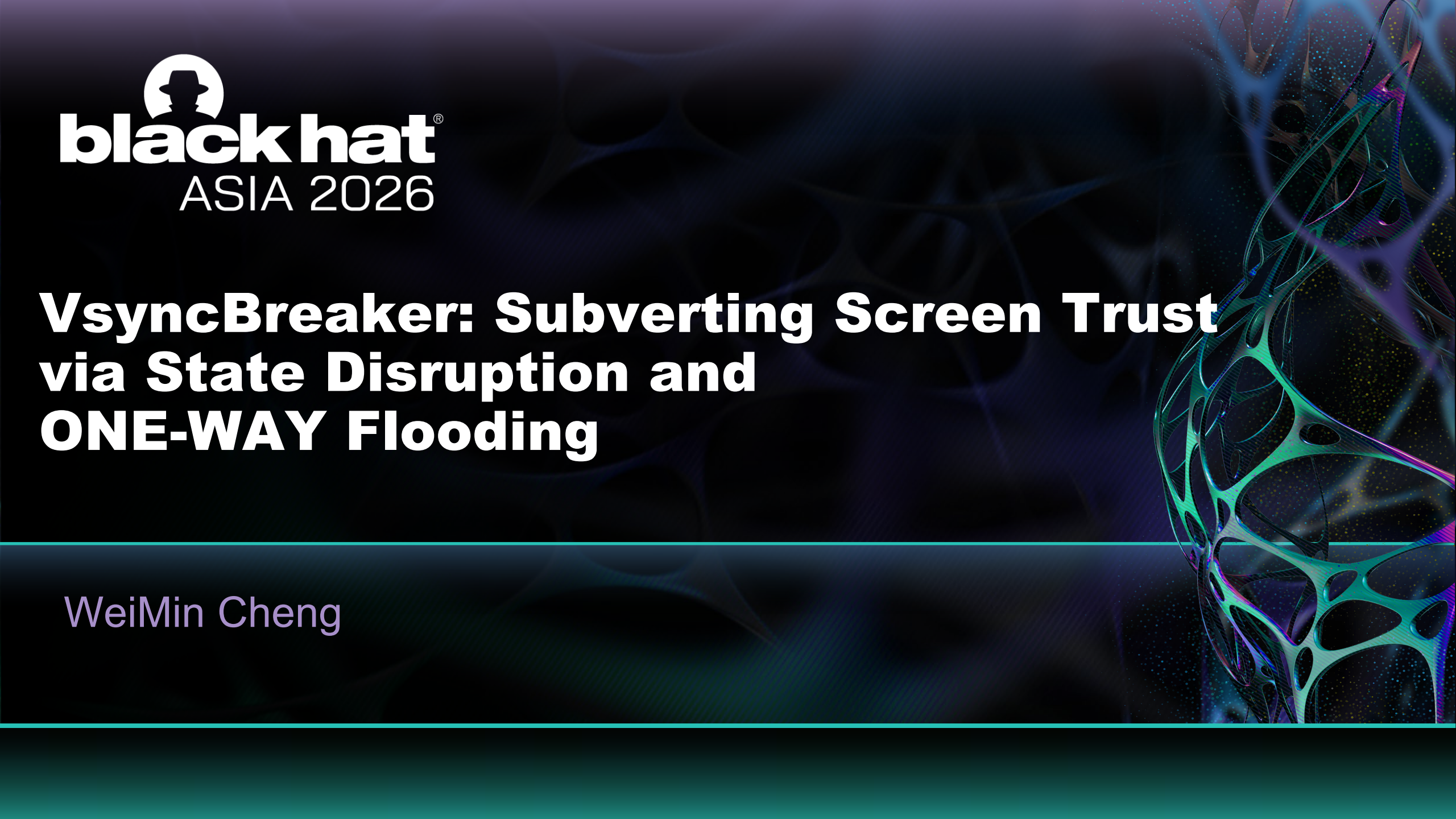


VsyncBreaker: Subverting Screen Trust via State Disruption and ONE-WAY Flooding

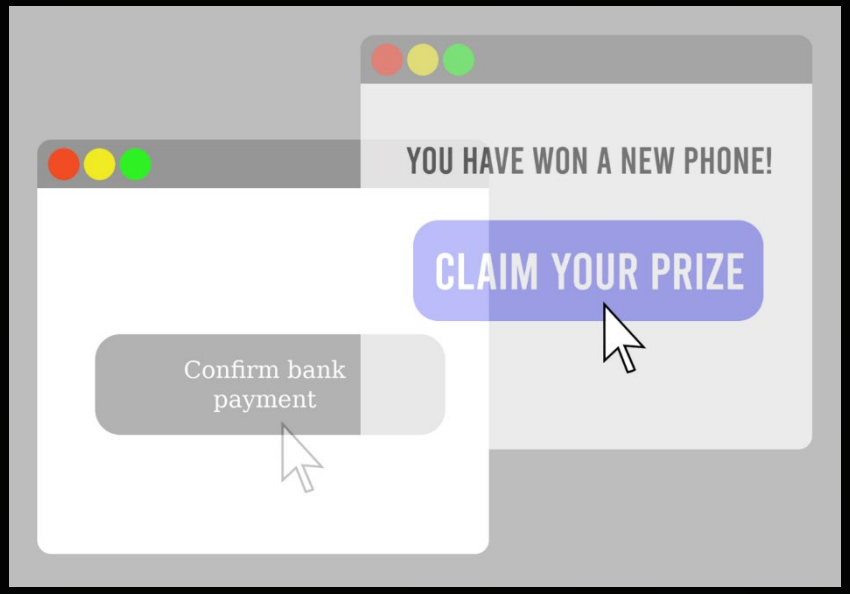
WeiMin Cheng

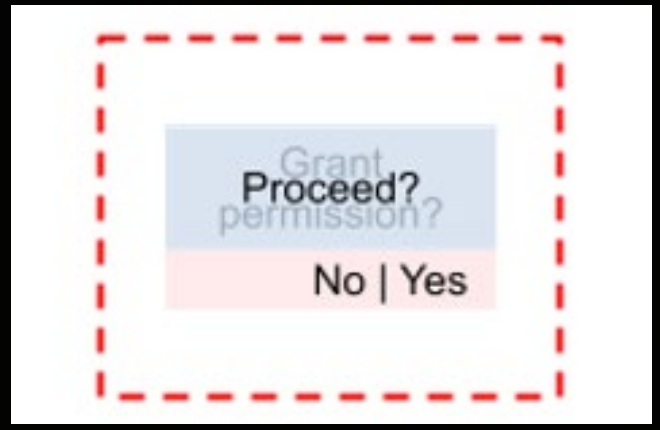
WHOAMI

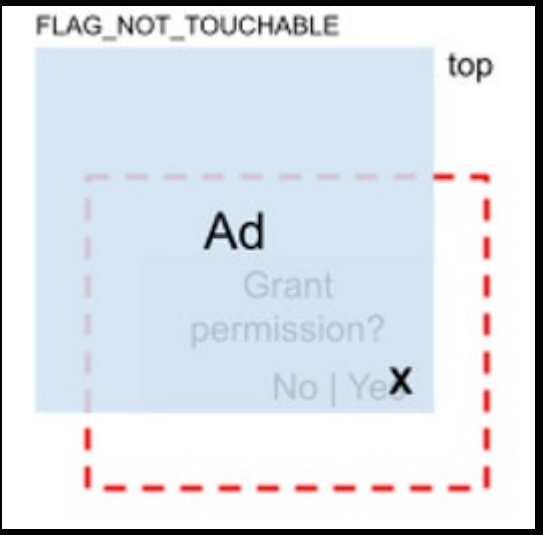
- Mobile Security
- AOSP Researcher
- https://github.com/MG1937/

What is Tapjacking

- Web Clickjacking := IFrame + CSS + CSP Bypass
- Android Tapjacking := ?

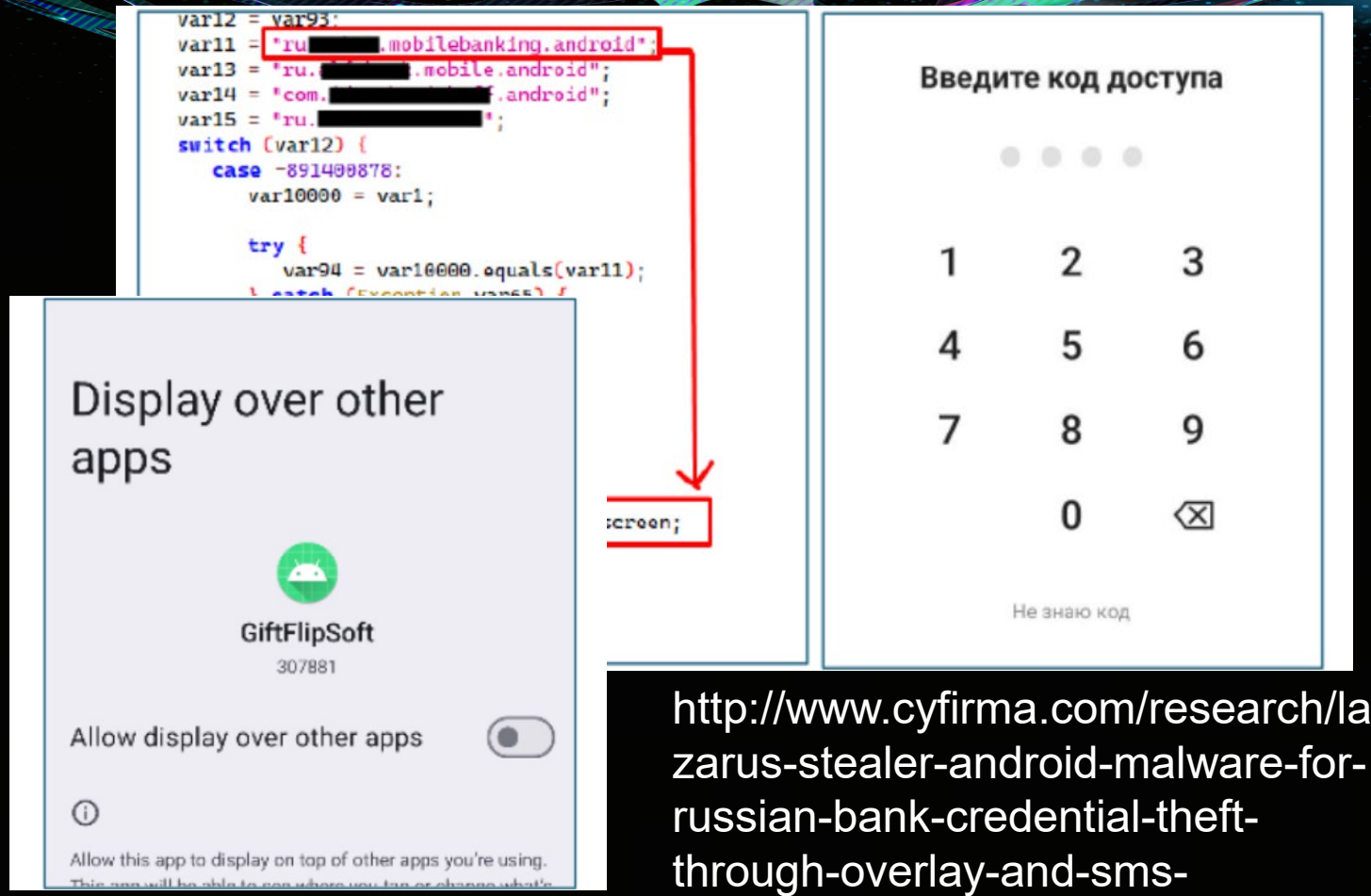






Why Tapjacking

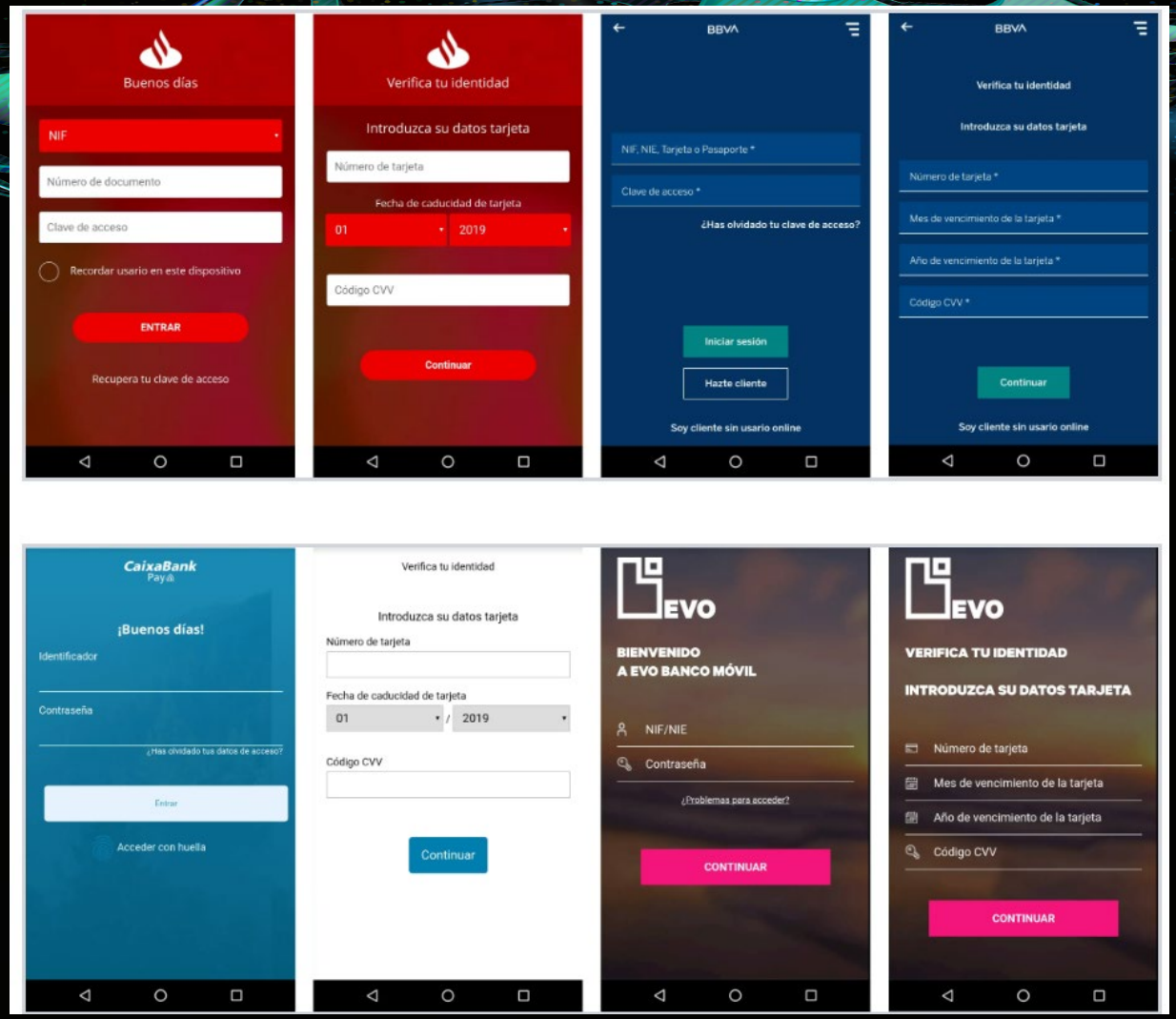
- **APT/BankBot**
  - **Lazarus**
  - Anubis
- Palo Alto CVE-2017-0752
- Toast Overlay Weaponized



http://www.cyfirma.com/research/lazarus-stealer-android-malware-for-russian-bank-credential-theft-through-overlay-and-sms-manipulation/

Why Tapjacking

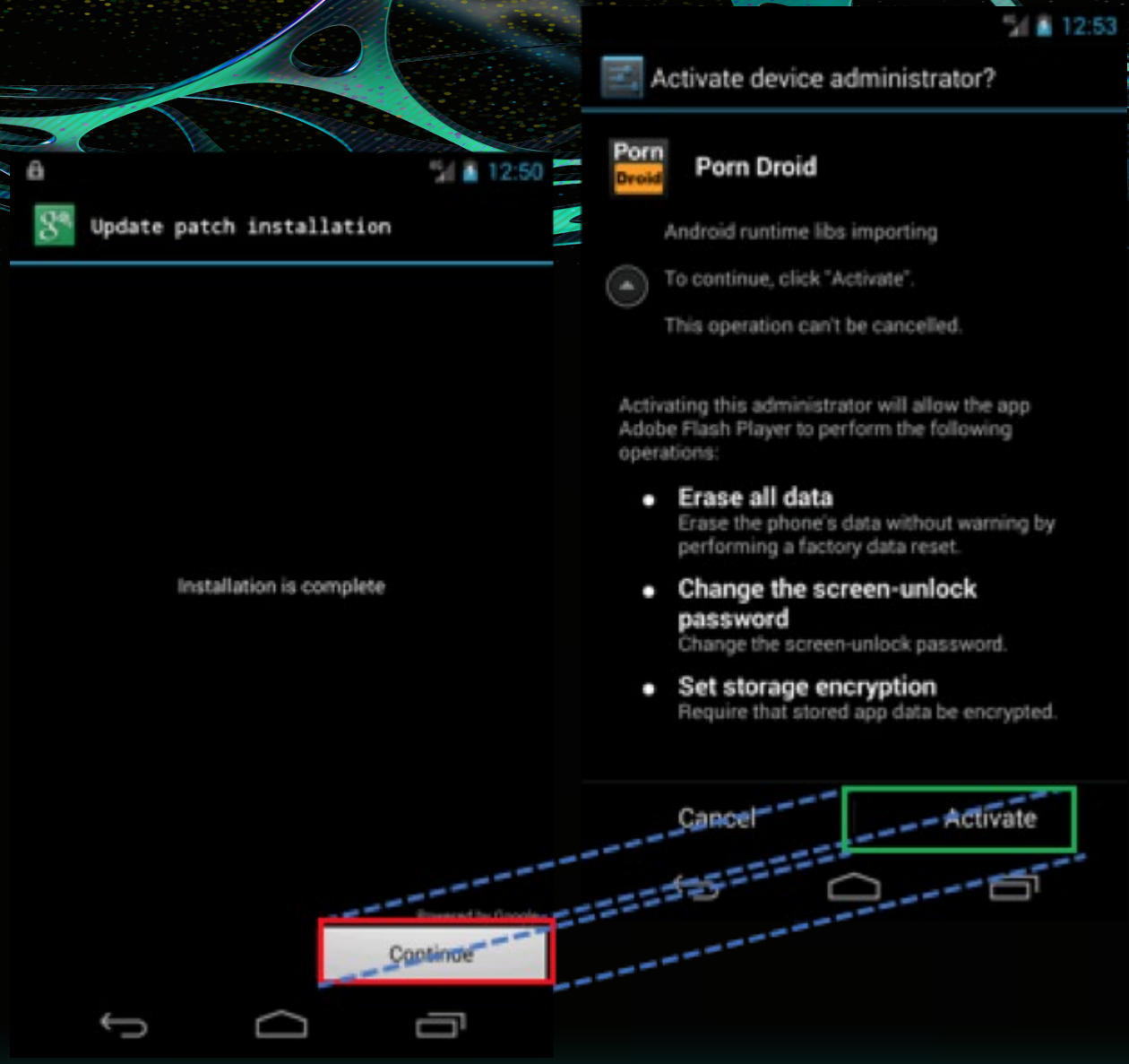
- **APT/BankBot**
  - Lazarus
  - **Anubis**
- Palo Alto CVE-2017-0752
- Toast Overlay Weaponized



http:://www.threatfabric.com/blogs/ginp_a_malware_patchwork_borrowing_from_anubis

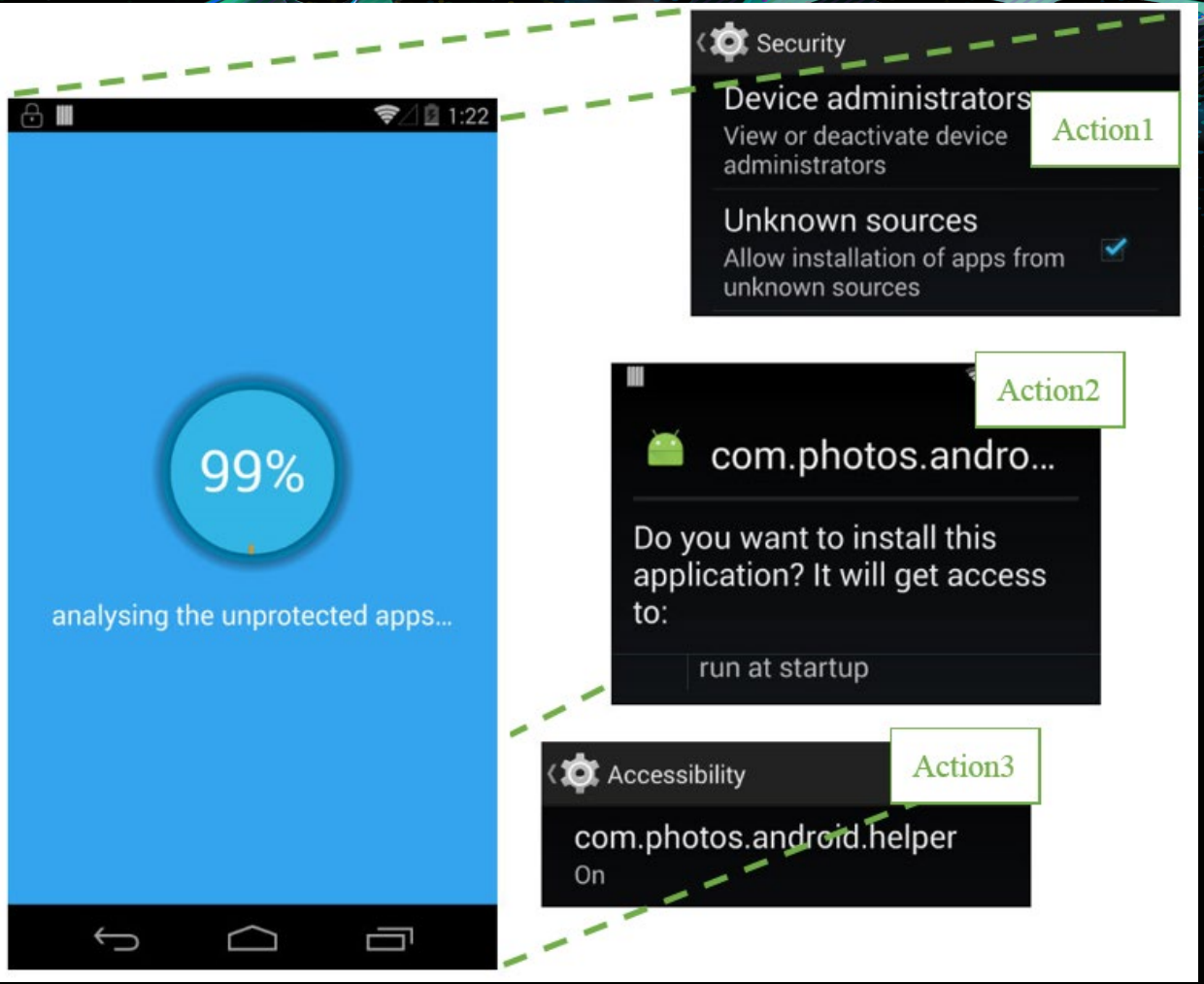
Why Tapjacking

- APT/BankBot
  - Lazarus
  - Anubis
- **Palo Alto CVE-2017-0752**
- Toast Overlay Weaponized



Why Tapjacking

- APT/BankBot
  - Lazarus
  - Anubis
- Palo Alto CVE-2017-0752
- **Toast Overlay Weaponized**



https://www.trendmicro.com/en_us/research/17/k/toast-overlay-weaponized-install-android-malware-single-attack-chain.html

Classic Malware

#1

- SYSTEM_ALERT_WINDOW
  - Display over victim
- FLAG_NOT_TOUCHABLE
  - Click pass-through

Only work for Android <= 10

#2 CVE-2021-39617

- Activity + FLAG_NOT_TOUCHABLE

Protect Mechanism

- SYSTEM_ALERT_WINDOW

```
/**
 * Returns true if any window added by an application process that if of type
 * {@link android.view.WindowManager.LayoutParams#TYPE_TOAST} or that requires that requires
 * {@link android.app.AppOpsManager#OP_SYSTEM_ALERT_WINDOW} permission should be hidden when
 * this window is visible.
 */
boolean hideNonSystemOverlayWindowsWhenVisible() {
    return (mAttrs.privateFlags & SYSTEM_FLAG_HIDE_NON_SYSTEM_OVERLAY_WINDOWS) != 0
            && mSession.mCanHideNonSystemOverlayWindows;
}
```

```
    mRoot.forAllWindows((w) -> {
        w.setForceHideNonSystemOverlayWindowIfNeeded(hideSystemAlertWindows);
    }, false /* traverseTopToBottom */);
}
```

SYSTEM_FLAG_HIDE_NON_SYSTEM_OVERLAY_WINDOWS

HIDE_SAW

CVE-2021-39669 — CNA: Android (associated with Google Inc. or Open Handset Alliance)

In onCreate of InstallCaCertificateWarning.java, there is a possible way to mislead an user about CA installation circumstances due to a tapjacking/overlay attack. This could lead to local escalation of privilege...

Show more

CVE-2021-1040 — CNA: Android (associated with Google Inc. or Open Handset Alliance)

In onCreate of BluetoothPairingSelectionFragment.java, there is a possible EoP due to a tapjacking/overlay attack. This could lead to local escalation of privilege with no additional execution privileges needed. User interaction...

Show more

CVE-2021-1039 — CNA: Android (associated with Google Inc. or Open Handset Alliance)

In NotificationAccessActivity of AndroidManifest.xml, there is a possible EoP due to a tapjacking/overlay attack. This could lead to local escalation of privilege with no additional execution privileges needed. User interaction...

Show more

CVE-2021-1038 — CNA: Android (associated with Google Inc. or Open Handset Alliance)

In UserDetailsActivity of AndroidManifest.xml, there is a possible DoS due to a tapjacking/overlay attack. This could lead to local denial of service with no additional execution privileges needed. User interaction...

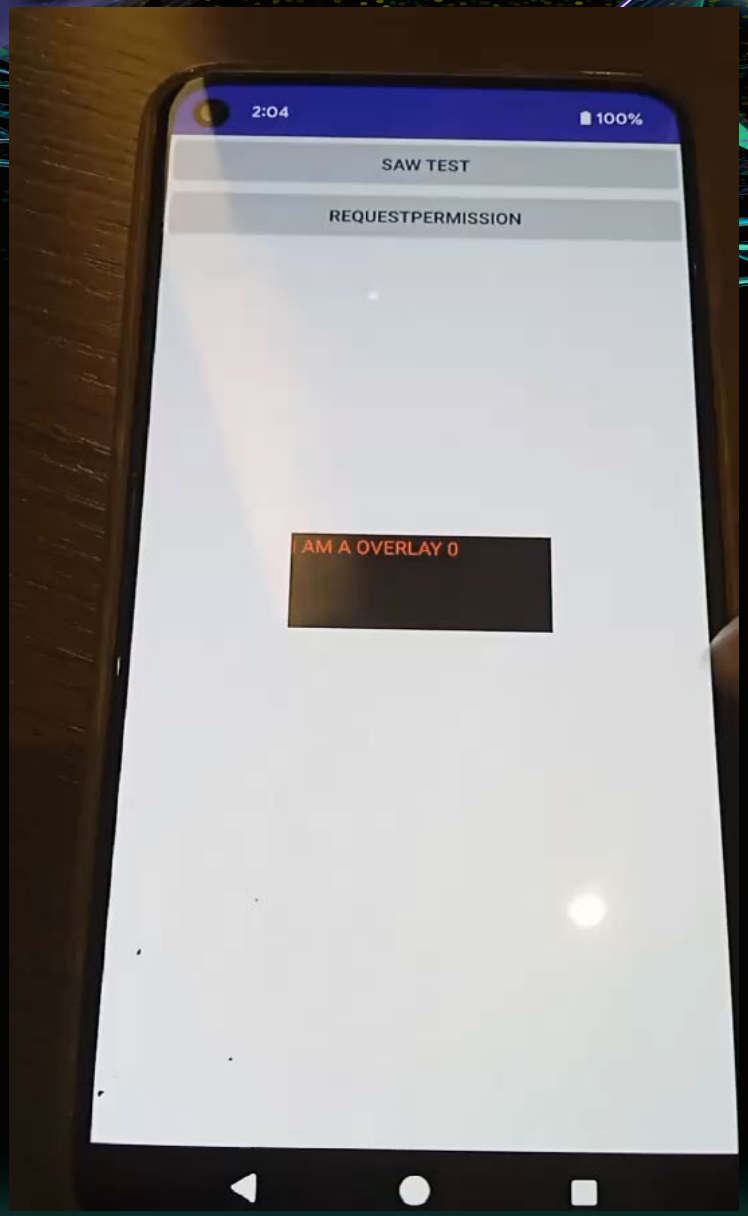
Show more

```
+         d.window?.addSystemFlags(SYSTEM_FLAG_HIDE_NON_SYSTEM_OVERLAY_WINDOWS)
+
          SystemUIDialog.registerDismissListener(d) { dialog = null }
```

```
+         getWindow().addPrivateFlags(
+                 WindowManager.LayoutParams.SYSTEM_FLAG_HIDE_NON_SYSTEM_OVERLAY_WINDOWS);
          Intent intent = getIntent();
          mDevice = (UsbDevice)intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);
```

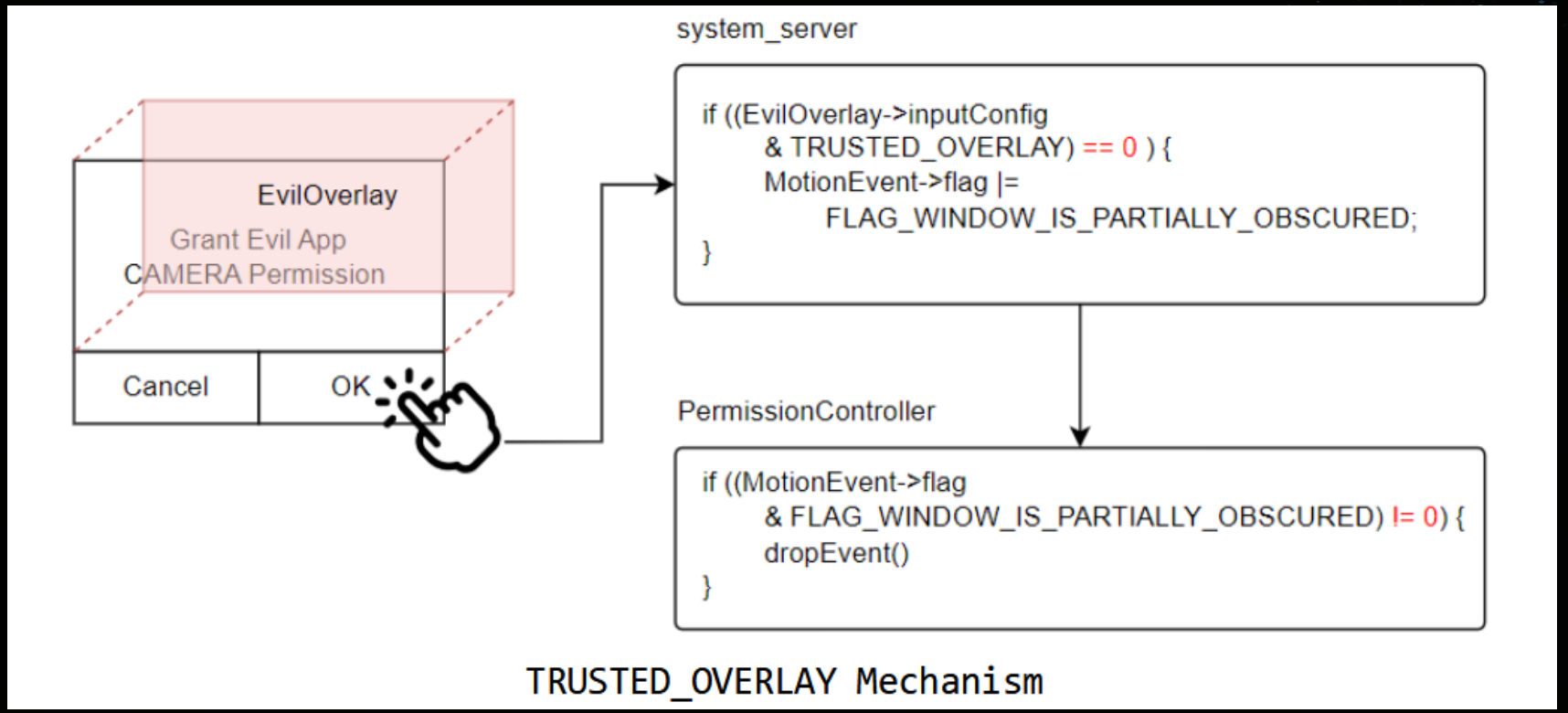
```
Add non-system overlay flag for usb perm dialog

Add private flag SYSTEM_FLAG_HIDE_NON_SYSTEM_OVERLAY_WINDOWS for
UsbPermissionActivity.

With this flag, UsbPermissionActivity will not been overlapped with
views which are displayed on system alert window
```

Demo



Protect Mechanism

- FLAG_NOT_TOUCHABLE

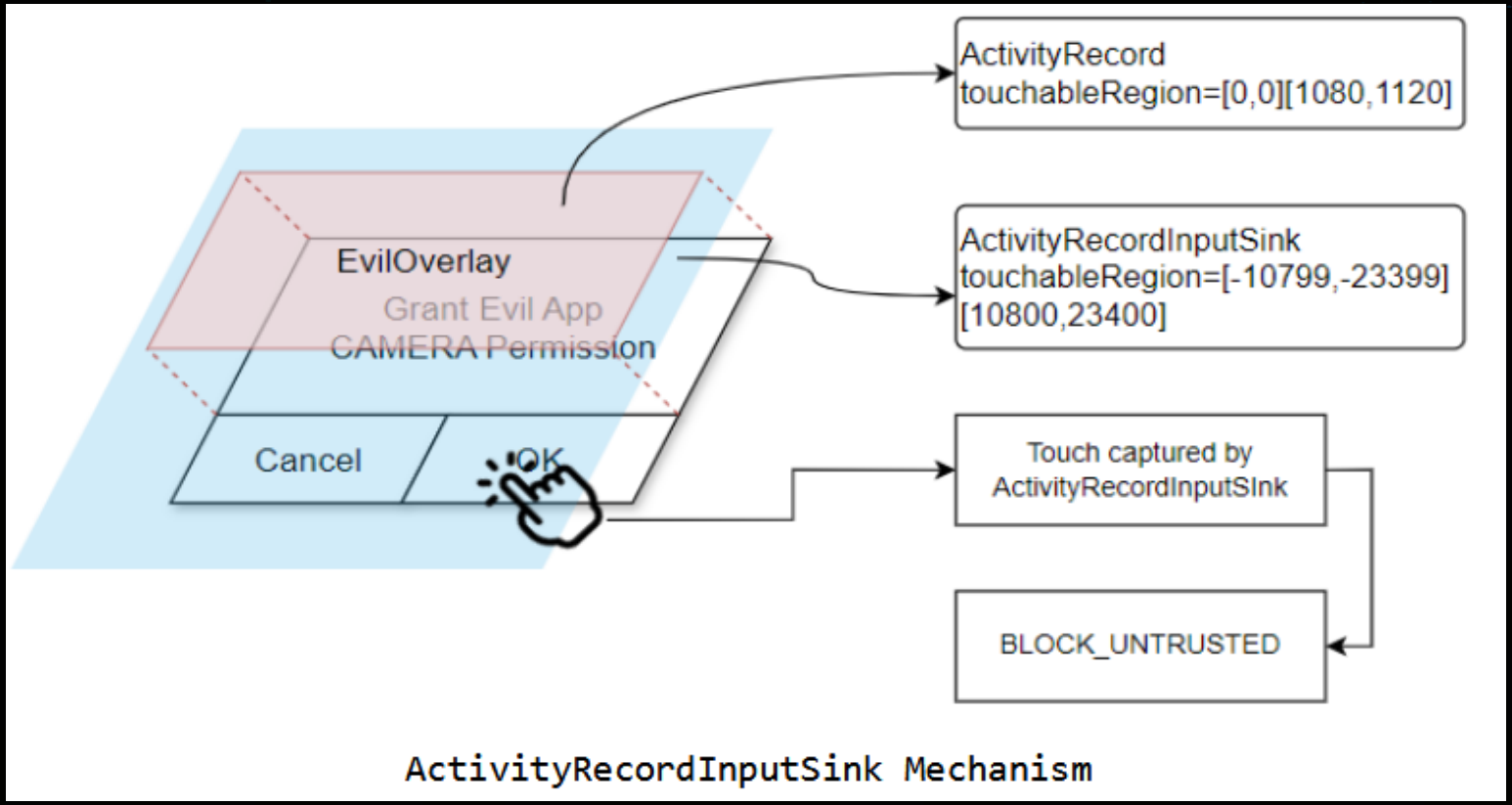


WINDOW_IS_PARTIALLY_OBSCURED

Demo

Protect Mechanism

- CVE-2021-39617



ActivityRecordInputSink Mechanism

ActivityRecordInputSink

CVE-2021-39617

- shrink Activity touchableRegion + FLAG_NOT_TOUCHABLE
- Bypass HIDE_SAW
  - Can’t hide TYPE_APPLICATION

```
      4: name='37bcb44 com.poc.poctester/com.poc.poctester.OverlayActivity', id=5437, displayId=0, inputConfi
g=NOT_FOCUSABLE | NOT_TOUCHABLE, alpha=1.00, frame=[490,1120][590,1220], globalScale=1.000000, applicationInf
o.name=ActivityRecord{5764f49 u0 com.poc.poctester/.OverlayActivity t286}, applicationInfo.token=0xb4000078b2
57e490, touchableRegion=[490,1120][590,1220], ownerPid=5055, ownerUid=10259, dispatchingTimeout=5000ms, hasTo
ken=0xb40000793256ab10, touchOcclusionMode=BLOCK_UNTRUSTED
```

CVE-2021-39617 Patch

- TRUSTED_OVERLAY
  - SecureButton

```
public boolean onFilterTouchEventForSecurity(MotionEvent event) {
    if (SdkLevel.isAtLeastS()) {
        return (event.getFlags() & FLAGS_WINDOW_IS_OBSCURED) == 0
                && super.onFilterTouchEventForSecurity(event);
    }

    return super.onFilterTouchEventForSecurity(event);
```

- ActivityRecordInputSink
  - touchableRegion=[-10799,-23399][10800,23400]
  - 10xScreen Size

```
    7: name='8adf150 ActivityRecordInputSink com.poc.poctester/.OverlayActivity', id=5397, displayId=0, inp
utConfig=NO_INPUT_CHANNEL | NOT_VISIBLE | NOT_FOCUSABLE | NOT_TOUCHABLE, alpha=1.00, frame=[0,0][0,0], global
Scale=0.000000, applicationInfo.name=, applicationInfo.token=<null>, touchableRegion=[-10799,-23399][10800,23
400], ownerPid=19715, ownerUid=1000, dispatchingTimeout=0ms, hasToken=<null>, touchOcclusionMode=BLOCK_UNTRUS
TED
```

Tapjacking :=

- Bypass HIDE_SAW
- Bypass TRUSTED_OVERLAY
- Bypass ARIS

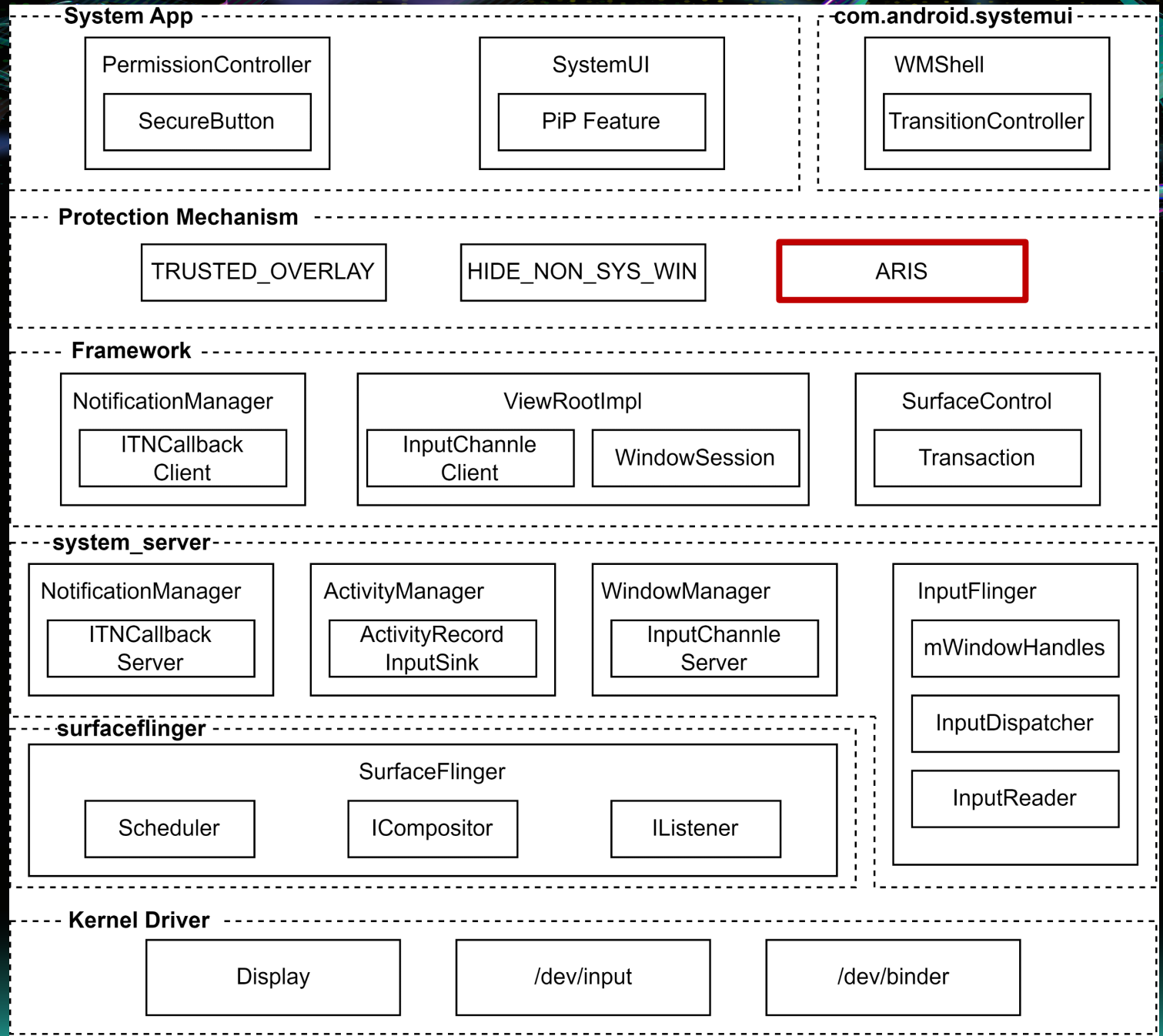
Tapjacking :=

- Bypass HIDE_SAW
- Bypass TRUSTED_OVERLAY
- Bypass ARIS

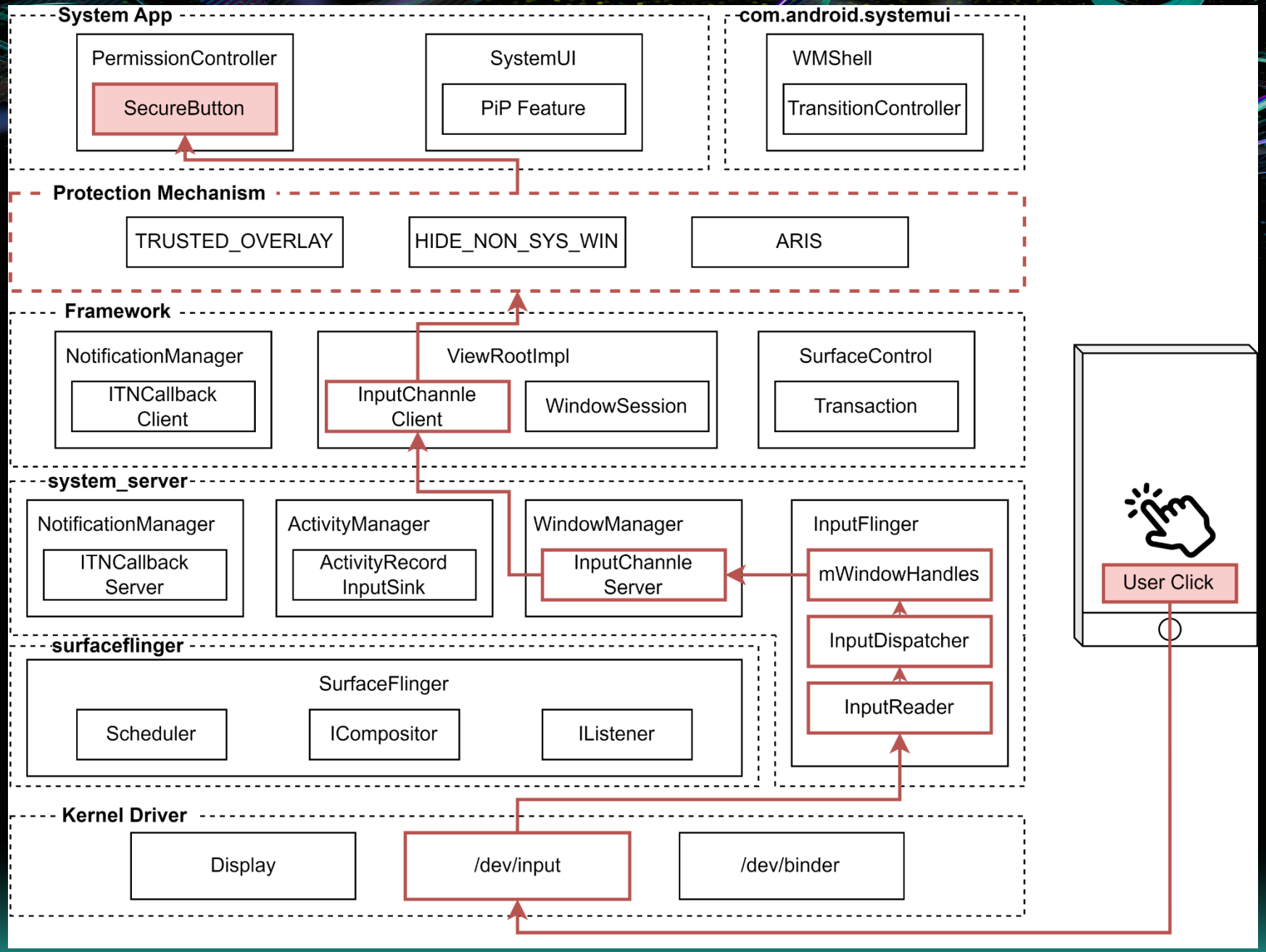
NO Runtime Permission requires

NO Privilege requires

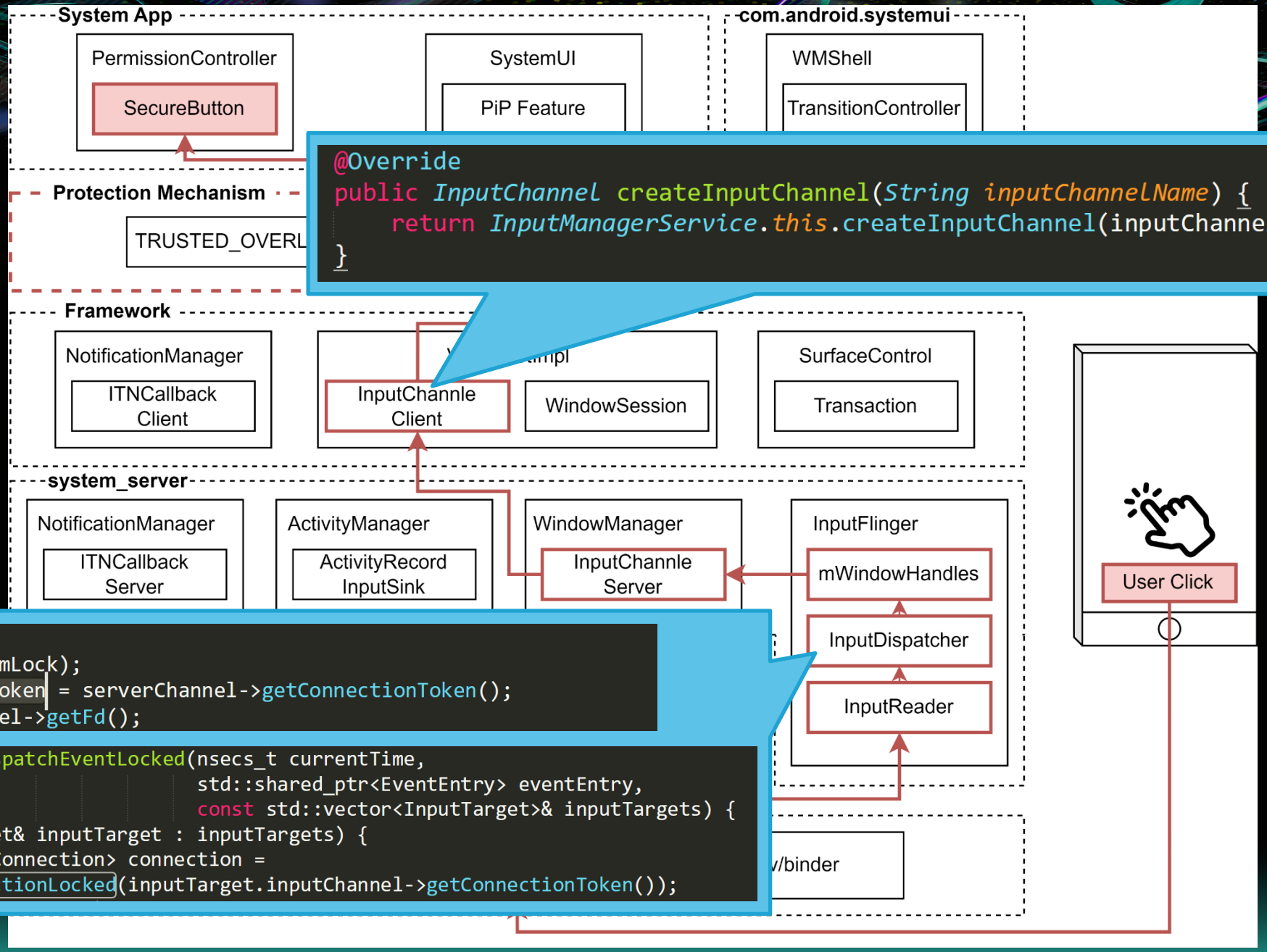
Attack Surface



How User Tap Work?



How User Tap Work?



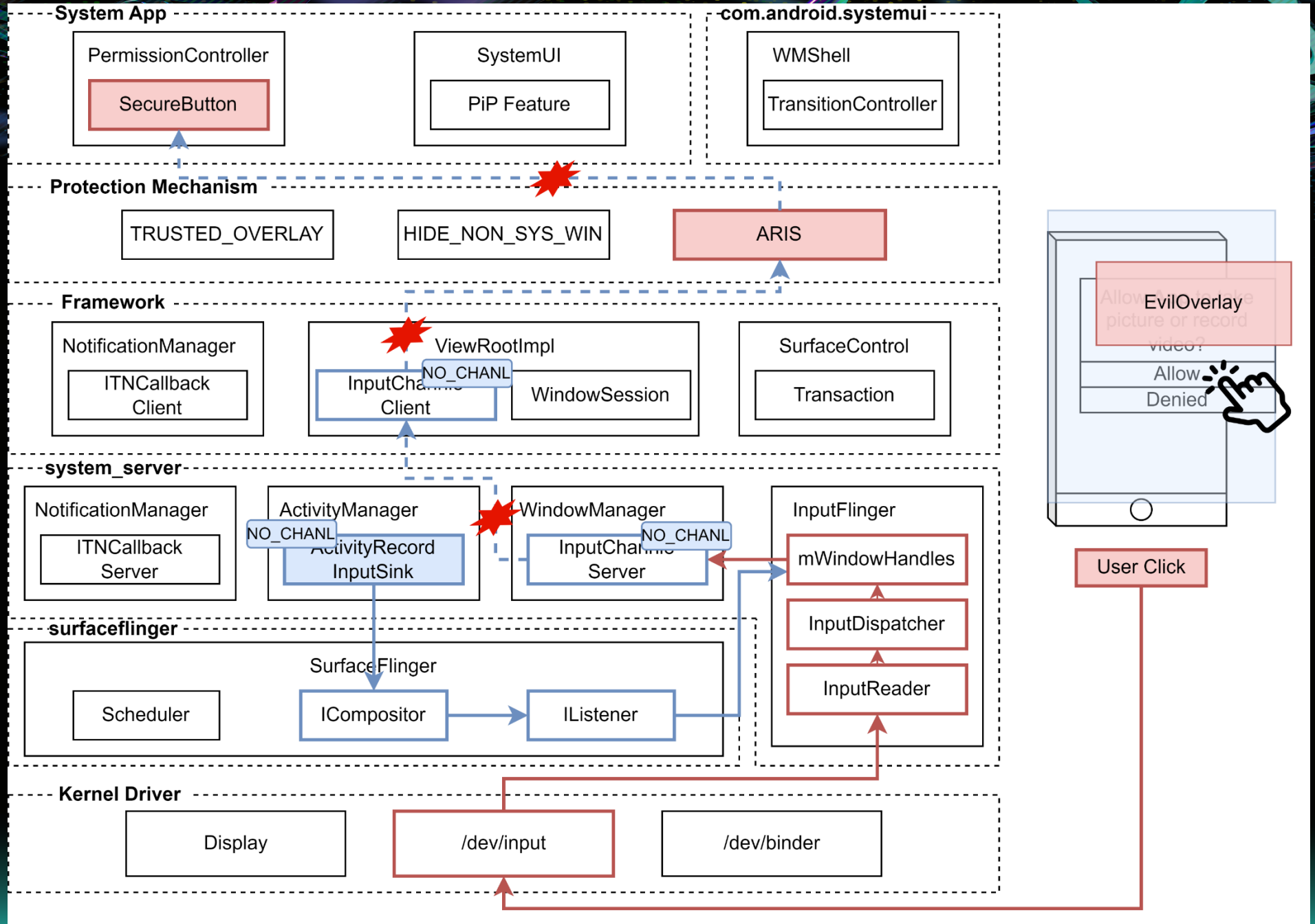
ActivityRecordInputSink

- Insert ARIS behind every Activity!
  - **ARIS has no InputChannel!**
- InputDispatcher Drop Touches!

```
private InputWindowHandle createInputWindowHandle() {
    InputWindowHandle inputWindowHandle = new InputWindowHandle(null,
            mActivityRecord.getDisplayId());
    inputWindowHandle.replaceTouchableRegionWithCrop = true;
    inputWindowHandle.name = mName;
    inputWindowHandle.layoutParamsType = WindowManager.LayoutParams.TYPE_INPUT_CONSUMER;
    inputWindowHandle.ownerPid = WindowManagerService.MY_PID;
    inputWindowHandle.ownerUid = WindowManagerService.MY_UID;
    inputWindowHandle.inputConfig = InputConfig.NOT_FOCUSABLE | InputConfig.NO_INPUT_CHANNEL;
    return inputWindowHandle;
}
```

```
bool InputDispatcher::canWindowReceiveMotionLocked(const sp<WindowInfoHandle>& window,
                                                   const MotionEntry& motionEntry) const

    if (info.inputConfig.test(WindowInfo::InputConfig::NO_INPUT_CHANNEL)) {
        ALOGW("Not sending touch gesture to %s because it has config NO_INPUT_CHANNEL",
              window->getName().c_str());
        return false;
    }
```

How ARIS Work?



The Price of Aggression

- **b9d9d7adeb**
- Resolve app compatibility issue
- Block Touch when:
  - Cross task boundry
  - Touch between different UID app

```
Allow some cross activity touches

Resolve app compatibility issues caused by ActivityRecordInputSink
agressivly blocking all touches. This change makes
ActivityRecordInputSink more conservable and it now only blocks touches
if the touches would otherwise:
1. Cross a task boundry
2. Go to an activity with a different uid AND that activity hasn't
   explicilty launched an activity with the same uid as this activity

Bug: 231701903
Bug: 194480991
Test: atest CtsWindowManagerDeviceTestCases:ActivityRecordInputSinkTests
Test: atest CtsWindowManagerDeviceTestCases:CrossAppDragAndDropTests
Test: atest CtsWindowManagerDeviceTestCases:PinnedStackTests
Test: atest CtsWindowManagerDeviceTestCases:WindowUntrustedTouchTest
Change-Id: Icd1668f465ba01b330c056d9ea95aed6104fc6a9
Merged-In: Icd1668f465ba01b330c056d9ea95aed6104fc6a9
```

The Price of Aggression

- **b9d9d7adeb**
- Resolve app compatibility issue
- Block Touch when:
  - Cross task boundry
  - Touch between different UID app
- UnBlock Touch when:
  - same UID app
  - **isTransition???**

```
Allow some cross activity touches

Resolve app compatibility issues caused by ActivityRecordInputSink
agressivly blocking all touches. This change makes
ActivityRecordInputSink more conservable and it now only blocks touches
if the touches would otherwise:
1. Cross a task boundry
2. Go to an activity with a different uid AND that activity hasn't
   explicilty launched an activity with the same uid as this activity

Bug: 231701903
```

```
+        final boolean allowPassthrough = activityBelowInTask != null && (
+                activityBelowInTask.mAllowedTouchUid == mActivityRecord.getUid()
+                        || activityBelowInTask.isUid(mActivityRecord.getUid()));
+        if (allowPassthrough || !mIsCompatEnabled || mActivityRecord.isInTransition()) {
             mInputWindowHandleWrapper.setInputConfigMasked(InputConfig.NOT_TOUCHABLE,
                     InputConfig.NOT_TOUCHABLE);
         } else {
```

???????

DEBUG Priority

- ARIS + NOT_TOUCHABLE + NO_INPUT_CHANNEL
  - Debug **InputDispatcher::windowAcceptsTouchAt**

```
const bool windowCanInterceptTouch = isStylus && windowInfo.interceptsStylus();
if (inputConfig.test(WindowInfo::InputConfig::NOT_TOUCHABLE) && !windowCanInterceptTouch) {
    return false;
}
```

```
(lldb) image list libinputflinger.so
[  0] 362E5F45-F4A2-0AA1-6998-714B135BFC4E 0x0000007af5b0a000 C:\Users\Admin\.lldb\module_cache\remote-android\.cache\
(lldb) disas -s 0x0000007af5b0a000+0x000000000009c220
libinputflinger.so`android::inputdispatcher::(anonymous namespace)::windowAcceptsTouchAt:
0x7af5ba6220 <-18446743545600908767>: tbnz   w0, #0x0, 0x7af5ba622c ; libinputflinger.so.PT_LOAD[1]..text + 131628
0x7af5ba6224 <-18446743545600908763>: mov    w21, wzr
0x7af5ba6228 <-18446743545600908759>: b      0x7af5ba627c   ; libinputflinger.so.PT_LOAD[1]..text + 131708
0x7af5ba622c <-18446743545600908755>: movi   v0.2d, #0000000000000000
0x7af5ba6230 <-18446743545600908751>: add    x1, x20, #0x88
0x7af5ba6234 <-18446743545600908747>: mov    x8, sp
0x7af5ba6238 <-18446743545600908743>: mov    x0, x19
0x7af5ba623c <-18446743545600908739>: str    xzr, [sp, #0x60]
(lldb)
All breakpoints enabled. (1 breakpoints)
(lldb)  p *(char**)($x20+48)
(char *) 0xb400007d314b3f50 "d7a7993 ScreenDecorOverlayBottom"
(lldb)  p *(char**)($x20+48)
(char *) 0xb400007c21630380 "e51f915 ScreenDecorOverlay"
(lldb)  p *(char**)($x20+48)
(char *) 0xb400007cf14bce70 "b144211 com.poc.abortpiptapjack/com.poc.abortpiptapjack.PocActivity"
(lldb)  p *(char**)($x20+48)
(char *) 0xb400007cf14bbc10 "f03ad4c ActivityRecordInputSink com.poc.abortpiptapjack/.PocActivity"
```

DEBUG Priority

- ARIS + NOT_TOUCHABLE + NO_INPUT_CHANNEL
  - Debug **InputDispatcher::windowAcceptsTouchAt**

```
const bool windowCanInterceptTouch = isStylus && windowInfo.interceptsStylus();
if (inputConfig.test(WindowInfo::InputConfig::NOT_TOUCHABLE) && !windowCanInterceptTouch) {
    return false;
}
```

```
(lldb) image list libinputflinger.so
[  0] 362E5F45-F4A2-0AA1-6998-714B135BFC4E 0x0000007af5b0a000 C:\Users\Admin\.lldb\module_cache\remote-android\.cache\
(lldb) disas -s 0x0000007af5b0a000+0x000000000009c220
libinputflinger.so`android::inputdispatcher::(anonymous namespace)::windowAcceptsTouchAt:
0x7af5ba6220 <-18446743545600908767>: tbnz   w0, #0x0, 0x7af5ba622c ; libinputflinger.so.PT_LOAD[1]..text + 131628
0x7af5ba6224 <-18446743545600908763>: mov    w21, wzr
0x7af5ba6228 <-18446743545600908759>: b      0x7af5ba627c   ; libinputflinger.so.PT_LOAD[1]..text + 131708
0x7af5ba622c <-18446743545600908755>: movi   v0.2d, #0000000000000000
0x7af5ba6230 <-18446743545600908751>: add    x1, x20, #0x88
0x7af5ba6234 <-18446743545600908747>: mov    x8, sp
0x7af5ba6238 <-18446743545600908743>: mov    x0, x19
0x7af5ba623c <-18446743545600908739>: str    xzr, [sp, #0x60]
(lldb)
All breakpoints enabled. (1 breakpoints)
(lldb)  p *(char**)($x20+48)
(char *) 0xb400007d314b3f50 "d7a7993 ScreenDecorOverlayBottom"
(lldb)  p *(char**)($x20+48)
(char *) 0xb400007c21630380 "e51f915 ScreenDecorOverlay"
(lldb)  p *(char**)($x20+48)
(char *) 0xb400007cf14bce70 "b144211 com.poc.abortpiptapjack/com.poc.abortpiptapjack.PocActivity"
(lldb)  p *(char**)($x20+48)
(char *) 0xb400007cf14bbc10 "f03ad4c ActivityRecordInputSink com.poc.abortpiptapjack/.PocActivity"
```

Bypassed!

TRUSTED_OVERLAY

- Fallback Logic
  - ARIS Bypass + HIDE_SAW Bypass != Tapjack System
- Only SysUI/Privilege have TRUSTED_OVERLAY

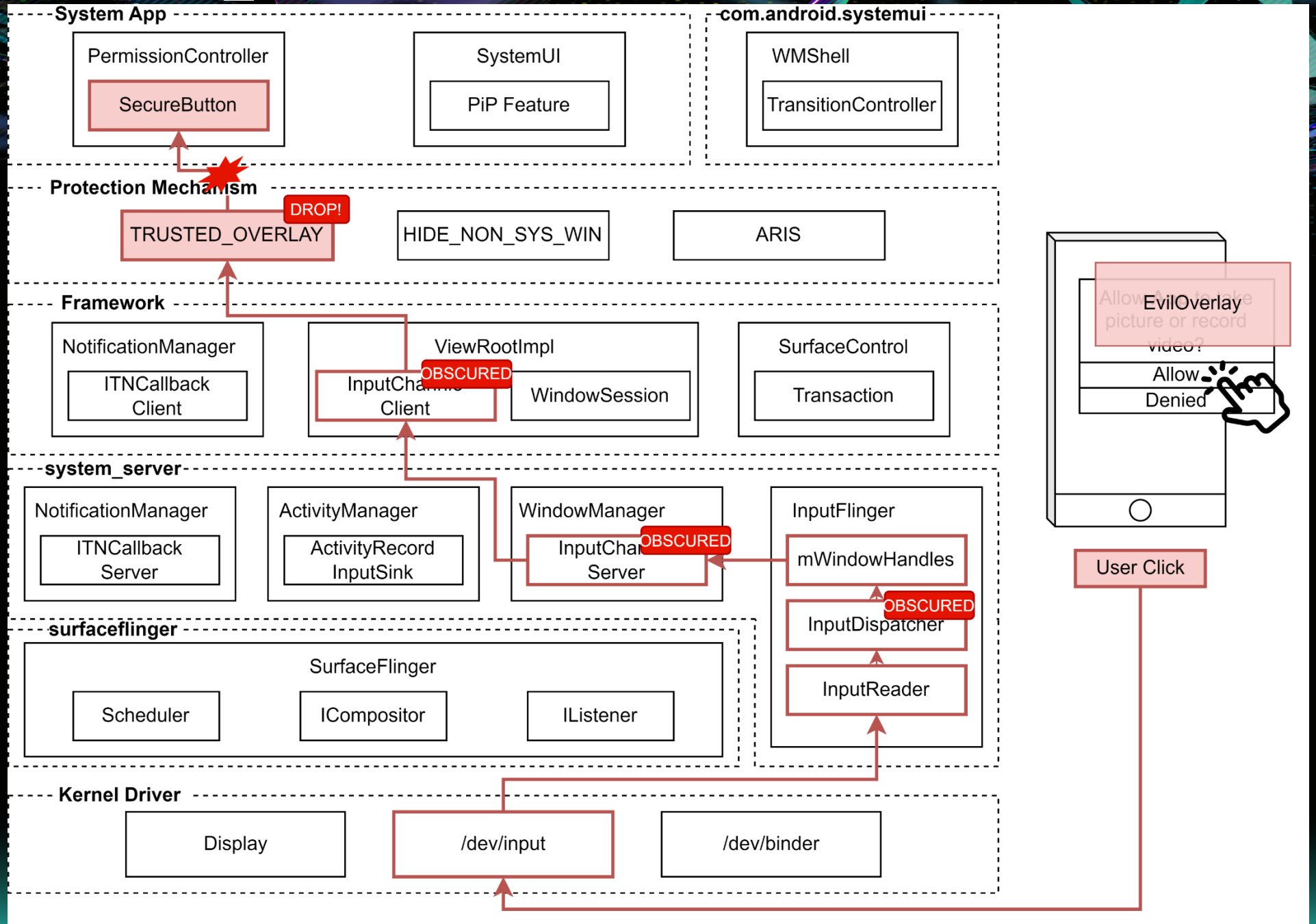
```
static bool canBeObscuredBy(const sp<WindowInfoHandle>& windowHandle,
                            const sp<WindowInfoHandle>& otherHandle) {
    ...
    } else if (otherInfo->inputConfig.test(gui::WindowInfo::InputConfig::TRUSTED_OVERLAY)) {
        return false;
    } else if (otherInfo->displayId != info->displayId) {
```

InputDispatcher::canBeObscuredBy

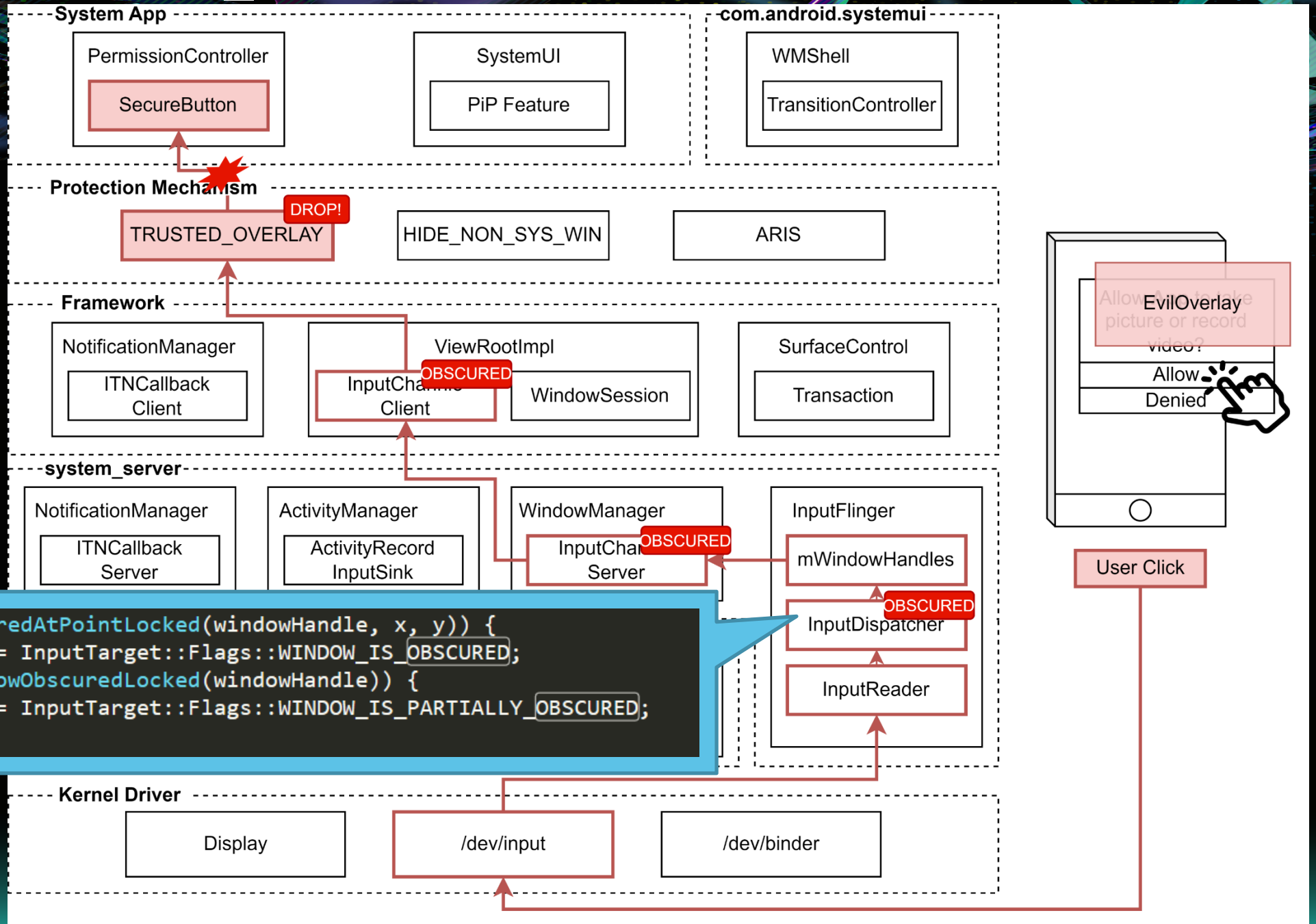
```
if (isWindowObscuredAtPointLocked(windowHandle, x, y)) {
    targetFlags |= InputTarget::Flags::WINDOW_IS_OBSCURED;
} else if (isWindowObscuredLocked(windowHandle)) {
    targetFlags |= InputTarget::Flags::WINDOW_IS_PARTIALLY_OBSCURED;
}
```

InputDispatcher::findTouchedWindowTargetsLocked

How TRUSTED_OVERLAY Work?



How TRUSTED_OVERLAY Work?




```
if (isWindowObscuredAtPointLocked(windowHandle, x, y)) {
    targetFlags |= InputTarget::Flags::WINDOW_IS_OBSCURED;
} else if (isWindowObscuredLocked(windowHandle)) {
    targetFlags |= InputTarget::Flags::WINDOW_IS_PARTIALLY_OBSCURED;
}
```

How to setTrustedOverlay

- SysUI/SystemServer
  - Obtain SurfaceControl$Transaction
  - Call setTrustedOverlay

```
SurfaceControl.Transaction t = new SurfaceControl.Transaction();
t.reparent(sc, parent).setTrustedOverlay(sc, true).apply();
t.close();
```

```
createTransaction()
        .setLayer(mSurfaceControl->get(), INT32_MAX - 2)
        .setTrustedOverlay(mSurfaceControl->get(), true)
        .apply();
```

Code snipe of SystemServer/SurfaceFlinger

- Non-privilege process?
  - Is SurfaceControl handled by System?
  - Can App get its SurfaceControl?

SurfaceControl

outSurfaceControl

ViewRootImpl → WindowSession → WMS → WSC → …

```
private int createSurfaceControl(SurfaceControl outSurfaceControl, int result,
        WindowState win, WindowStateAnimator winAnimator) {
    if (!win.mHasSurface) {
        result |= RELAYOUT_RES_SURFACE_CHANGED;
    }

    WindowSurfaceController surfaceController;
    try {
        Trace.traceBegin(TRACE_TAG_WINDOW_MANAGER, "createSurfaceControl");
        surfaceController = winAnimator.createSurfaceLocked();
    } finally {
        Trace.traceEnd(TRACE_TAG_WINDOW_MANAGER);
    }
    if (surfaceController != null) {
        surfaceController.getSurfaceControl(outSurfaceControl);
        ProtoLog.i(WM_SHOW_TRANSACTIONS, "OUT SURFACE %s: copied", outSurfaceControl);
```

WindowManagerService#createSurfaceControl

SurfaceControl

outSurfaceControl

ViewRootImpl → WindowSession → WMS → WSC → …

↓

SurfaceControl

```
final SurfaceControl.Builder b = win.makeSurface()
        .setParent(win.getSurfaceControl())
        .setName(name)
        ...
        .setCallsite("WindowSurfaceController");

final boolean useBLAST = mService.mUseBLAST && ((win.getAttrs().privateFlags
        & WindowManager.LayoutParams.PRIVATE_FLAG_USE_BLAST) != 0);

if (useBLAST) {
    b.setBLASTLayer();
}

mSurfaceControl = b.build();
```

WindowSurfaceController#<init>

SurfaceControl

outSurfaceControl


```
private int createSurfaceControl(SurfaceControl outSurfaceControl, int result,
        WindowState win, WindowStateAnimator winAnimator) {

    if (surfaceController != null) {
        surfaceController.getSurfaceControl(outSurfaceControl);
        ProtoLog.i(WM_SHOW_TRANSACTIONS, "OUT SURFACE %s: copied", outSurfaceControl);
```

```
/**
 * @hide
 */
public void copyFrom(@NonNull SurfaceControl other, String callsite) {
    mName = other.mName;
    mWidth = other.mWidth;
    mHeight = other.mHeight;
    mLocalOwnerView = other.mLocalOwnerView;
    assignNativeObject(nativeCopyFromSurfaceControl(other.mNativeObject), callsite);
}
```

SurfaceControl “Reference”

- System And App Both hold this “Reference”
  - From Binder pool…

```
/**
 * Note: do not rename, this field is used by native code.
 * @hide
 */
public long mNativeObject;
private long mNativeHandle;
```

```
private void assignNativeObject(long nativeObject, String callsite) {
    if (mNativeObject != 0) {
        release();
    }
    if (nativeObject != 0) {
        mFreeNativeResources =
                sRegistry.registerNativeAllocation(this, nativeObject);
    }
    mNativeObject = nativeObject;
    mNativeHandle = mNativeObject != 0 ? nativeGetHandle(nativeObject) : 0;
    if (sDebugUsageAfterRelease && mNativeObject == 0) {
        mReleaseStack = new Throwable("Assigned invalid nativeObject");
```

Obtain SurfaceControl

```
Class VRI = Class.forName("android.view.ViewRootImpl");
Field mSurfaceControlF = VRI.getDeclaredField("mSurfaceControl");
Field mSurfaceF = VRI.getDeclaredField("mSurface");
Field mTransactionF = VRI.getDeclaredField("mTransaction");
Field mChoreographerF_ = VRI.getDeclaredField("mChoreographer");
Field mTraversalScheduledF = VRI.getDeclaredField("mTraversalScheduled");
mTraversalScheduledF.setAccessible(true);
mChoreographerF_.setAccessible(true);
mTransactionF.setAccessible(true);
mSurfaceF.setAccessible(true);
mSurfaceControlF.setAccessible(true);
```

```
for (int i=0;i<mRoots.size();i++) {
    Object Root = mRoots.get(i);

    SurfaceControl.Transaction mTransaction = (SurfaceControl.Transaction) mTransactionF.get(Root);

    SurfaceControl mSurfaceControl = (SurfaceControl) mSurfaceControlF.get(Root);
    System.out.println("PAYLOAD mSurfaceControl >>> " + mSurfaceControl);
}
```

Sanitize

- eTrustedOverlayChanged
  - sanitize for Permission::ACCESS_SURFACE_**FLINGER**
  - Only grant to privilege process
  - **But who call sanitize?**

```
void layer_state_t::sanitize(int32_t permissions) {
    // TODO: b/109894387
    ...
    if (what & layer_state_t::eTrustedOverlayChanged) {
        if (!(permissions & Permission::ACCESS_SURFACE_FLINGER)) {
            what &= ~eTrustedOverlayChanged;
            ALOGE("Stripped attempt to set eTrustedOverlay in sanitize");
        }
    }
```

layer_state_t::sanitize

Pass-By-Value Misuse

- c78f53cd0e9ce
  - Make sanitize closer to client
  - But…

```
In SF, call sanitize as soon as setTransactionState is called since
that's the point where the Transaction has been passed over binder so we
can identify the calling uid. This allows us to remove the permission
values passed to applyTransactionState and unifies the places that were
calling sanitize.

Test: CredentialsTest
Bug: 267794530
Change-Id: I30c1800f0fee43df1cee82464139db7b56a7d911
```

```
 uint32_t SurfaceFlinger::setClientStateLocked(const FrameTimelineInfo& frameTimelineInfo,
                                               ResolvedComposerState& composerState,
                                               int64_t desiredPresentTime, bool isAutoTimestamp,
-                                              int64_t postTime, uint32_t permissions,
-                                              uint64_t transactionId) {
+                                              int64_t postTime, uint64_t transactionId) {
     layer_state_t& s = composerState.state;
-    s.sanitize(permissions);
```

Pass-By-Value Misuse

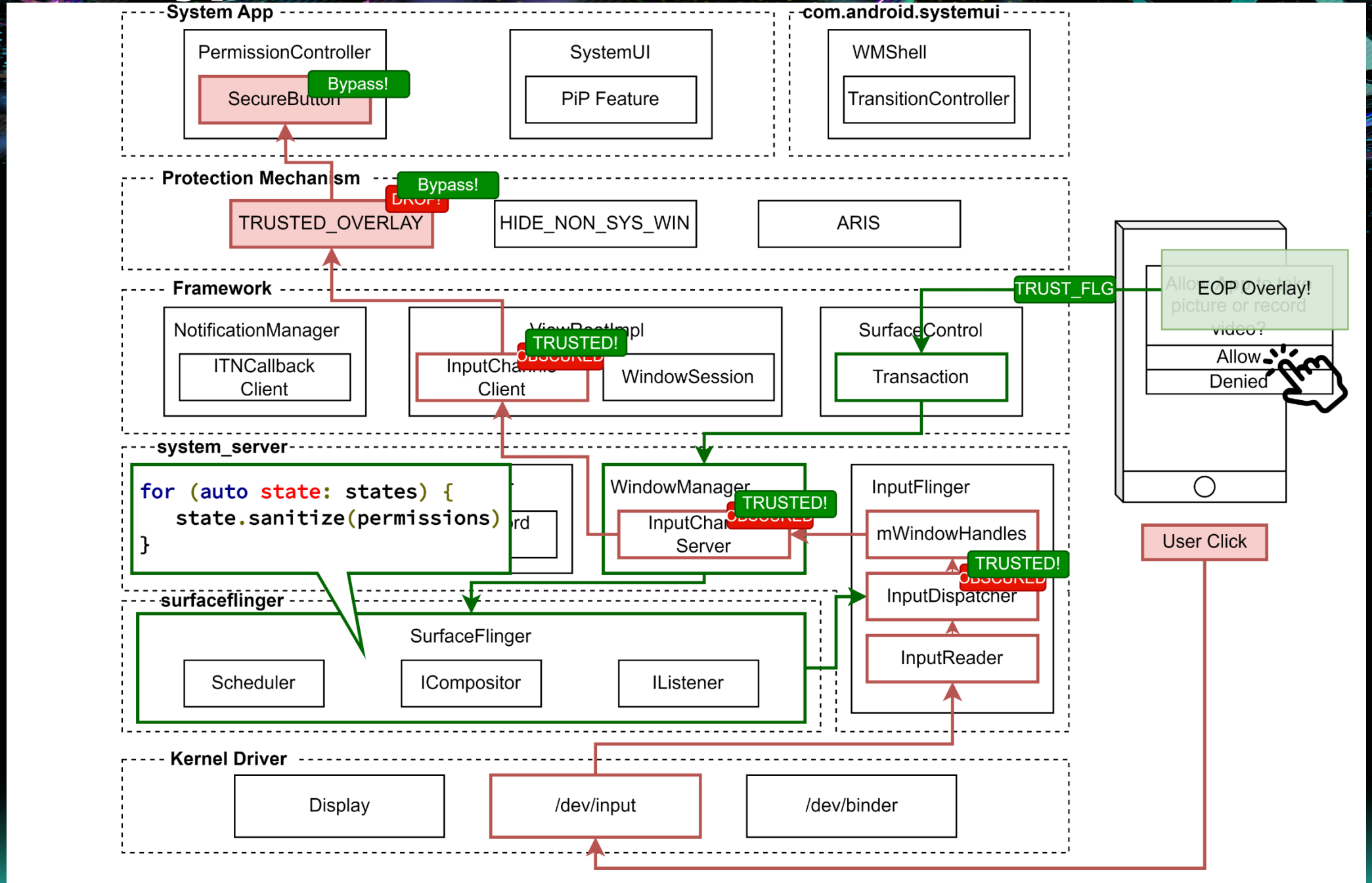
- c78f53cd0e9ce
  - Make sanitize closer to client
  - **BUT PASS BY VALUE!!!**

```
In SF, call sanitize as soon as setTransactionState is called since
that's the point where the Transaction has been passed over binder so we
can identify the calling uid. This allows us to remove the permission
values passed to applyTransactionState and unifies the places that were
calling sanitize.

Test: CredentialsTest
```

```
+    IPCThreadState* ipc = IPCThreadState::self();
+    const int originPid = ipc->getCallingPid();
+    const int originUid = ipc->getCallingUid();
+    uint32_t permissions = LayerStatePermissions::getTransactionPermissions(originPid, originUid);
+    for (auto composerState : states) {
+        composerState.state.sanitize(permissions);
     }
```

```
     layer_state_t& s = composerState.state;
-    s.sanitize(permissions);
```

How we Bypass it?



First Full Chain Bypass

- HIDE_SAW BYPASS
- ARIS BYPASS
- TRUSTED_OVERLAY BYPASS

You can’t be lucky forever

- Patch to Bug Issue
  - **Not Fun**

- **What If…**
  - **Protect machenism have no logic issue?**

PictureInPicture

- PiP feature
  - Require no permission
  - Render by SysUI
  - Display untrusted content
  - **Set Trusted Overlay!**

```
void notifyActivityPipModeChanged(@NonNull Task task, @Nullable ActivityRecord r) {
    final boolean inPip = r != null;
    if (inPip) {
        mService.getTaskChangeNotificationController().notifyActivityPinned(r);
    } else {
        mService.getTaskChangeNotificationController().notifyActivityUnpinned();
    }
    mWindowManager.mPolicy.setPipVisibilityLw(inPip);
    mWmService.mTransactionFactory.get()
            .setTrustedOverlay(task.getSurfaceControl(), inPip)
            .apply();
}
```


```
    5: name='b368841 com.poc.abortpiptapjack/com.poc.abortpiptapjack.PocActivity', id=2098, displayId=2,
 inputConfig=NOT_FOCUSABLE | NOT_TOUCHABLE | TRUSTED_OVERLAY | CLONE, alpha=1.00, frame=[441,247][1035,546
], globalScale=1.000000, applicationInfo.name=ActivityRecord{b3f64c5 u0 com.poc.abortpiptapjack/.PocActivi
ty t548}, applicationInfo.token=0xb4000071bb047350, touchableRegion=[441,247][1035,546], ownerPid=21410, o
wnerUid=10338, dispatchingTimeout=5000ms, hasToken=0xb40000728af780d0, touchOcclusionMode=BLOCK_UNTRUSTED
        transform (ROT_0) (SCALE TRANSLATE)
            1.0017  -0.0000  -442.0017
            -0.0000  1.0017  -247.0000
            0.0000  0.0000  1.0000
    6: name='6583f3c ActivityRecordInputSink com.poc.abortpiptapjack/.PocActivity', id=2096, displayId=2
, inputConfig=NO_INPUT_CHANNEL | NOT_FOCUSABLE | TRUSTED_OVERLAY | CLONE, alpha=1.00, frame=[441,230][441,
230], globalScale=0.000000, applicationInfo.name=, applicationInfo.token=<null>, touchableRegion=[441,230]
[1035,564], ownerPid=9484, ownerUid=1000, dispatchingTimeout=0ms, hasToken=<null>, touchOcclusionMode=BLOC
K_UNTRUSTED
```

PictureInPicture

- PiP feature
  - Require no permission
  - Render by SysUI
  - Display untrusted content
  - **Set Trusted Overlay!**

```
void notifyActivityPipModeChanged(@NonNull Task task, @Nullable ActivityRecord r) {
    final boolean inPip = r != null;
    if (inPip) {
        mService.getTaskChangeNotificationController().notifyActivityPinned(r);
    } else {
        mService.getTaskChangeNotificationController().notifyActivityUnpinned();
    }
    mWindowManager.mPolicy.setPipVisibilityLw(inPip);
    mWmService.mTransactionFactory.get()
            .setTrustedOverlay(task.getSurfaceControl(), inPip)
            .apply();
}
```

- **However**
  - **Can’t control SysUI**
  - **Still far away from exploit**


```
    5: name='b368841 com.poc.abortpiptapjack/com.poc.abortpiptapjack.PocActivity', id=2098, displayId=2,
 inputConfig=NOT_FOCUSABLE | NOT_TOUCHABLE | TRUSTED_OVERLAY | CLONE, alpha=1.00, frame=[441,247][1035,546
], globalScale=1.000000, applicationInfo.name=ActivityRecord{b3f64c5 u0 com.poc.abortpiptapjack/.PocActivi
ty t548}, applicationInfo.token=0xb4000071bb047350, touchableRegion=[441,247][1035,546], ownerPid=21410, o
wnerUid=10338, dispatchingTimeout=5000ms, hasToken=0xb40000728af780d0, touchOcclusionMode=BLOCK_UNTRUSTED
        transform (ROT_0) (SCALE TRANSLATE)
            1.0017  -0.0000  -442.0017
            -0.0000  1.0017  -247.0000
            0.0000  0.0000  1.0000
    6: name='6583f3c ActivityRecordInputSink com.poc.abortpiptapjack/.PocActivity', id=2096, displayId=2
, inputConfig=NO_INPUT_CHANNEL | NOT_FOCUSABLE | TRUSTED_OVERLAY | CLONE, alpha=1.00, frame=[441,230][441,
230], globalScale=0.000000, applicationInfo.name=, applicationInfo.token=<null>, touchableRegion=[441,230]
[1035,564], ownerPid=9484, ownerUid=1000, dispatchingTimeout=0ms, hasToken=<null>, touchOcclusionMode=BLOC
K_UNTRUSTED
```

BrainStorm

- How to control PiP-SysUI?
  - Can I control it by control SurfaceControl$Transaction?

BrainStorm

- How to control PiP-SysUI?
  - Can I control it by control SurfaceControl$Transaction?

  - getWindow().setWindowAnimations(R.style.AnimPoc);
  - App -> SurfaceControl$Transaction -> WindowState#startAnimation
  - **App render inside SysUI, make SysUI force run our Transaction!**

BrainStorm

- How to control PiP-SysUI?
  - Can I control it by control SurfaceControl$Transaction?
  - **Pip Also execute Transaction, merged by SysUI ❌**

```
@Override
public void mergeAnimation(@NonNull IBinder transition, @NonNull TransitionInfo info,
        @NonNull SurfaceControl.Transaction t, @NonNull IBinder mergeTarget,
        @NonNull Transitions.TransitionFinishCallback finishCallback) {
    ArrayList<Animator> anims = mAnimations.get(mergeTarget);
    if (anims == null) return;
    for (int i = anims.size() - 1; i >= 0; --i) {
        final Animator anim = anims.get(i);
        mAnimExecutor.execute(anim::end);
    }
}
```

DefaultTransitionHandler#mergeAnimation

```
080, 2340) d=0}]}
08-03 22:23:31.095  9718  9736 V WindowManagerShell: Transition (#16)android.os.BinderProxy@b0fd4c@0 ready
 while (#15)android.os.BinderProxy@2028c6@0 is still animating. Notify the animating transition in case th
ey can be merged
08-03 22:23:31.096  9718  9736 V WindowManagerShell: Transition animation finished (aborted=false), notify
```

How PiP talk to WMShell?

- ITransitionPlayer AIDL
  - ONEWAY Mode
  - Anyway to make IPC fail?

```
if (newTransition != null) {
    // Request at end since we want task-organizer events from ensureActivitiesVisible
    // to be recognized.
    transitionController.requestStartTransition(newTransition, rootTask,
            null /* remoteTransition */, null /* displayChange */);
    // A new transition was created just for this operations. Since the operation is
    // complete, mark it as ready.
    newTransition.setReady(rootTask, tr
}

resumeFocusedTasksTopActivities();

notifyActivityPipModeChanged(r.getTask(
```

```
oneway interface ITransitionPlayer {
    /**
     * Called when something in WMCore requires a transition to play -- for example when an Activity
     * is started in a new Task.
     *
     * @param transitionToken An identifying token for the transition that needs to be started.
     *                        Pass this to {@link IWindowOrganizerController#startTransition}.
     * @param request Information about this particular request.
     */
    void requestStartTransition(in IBinder transitionToken, in TransitionRequestInfo request);
```

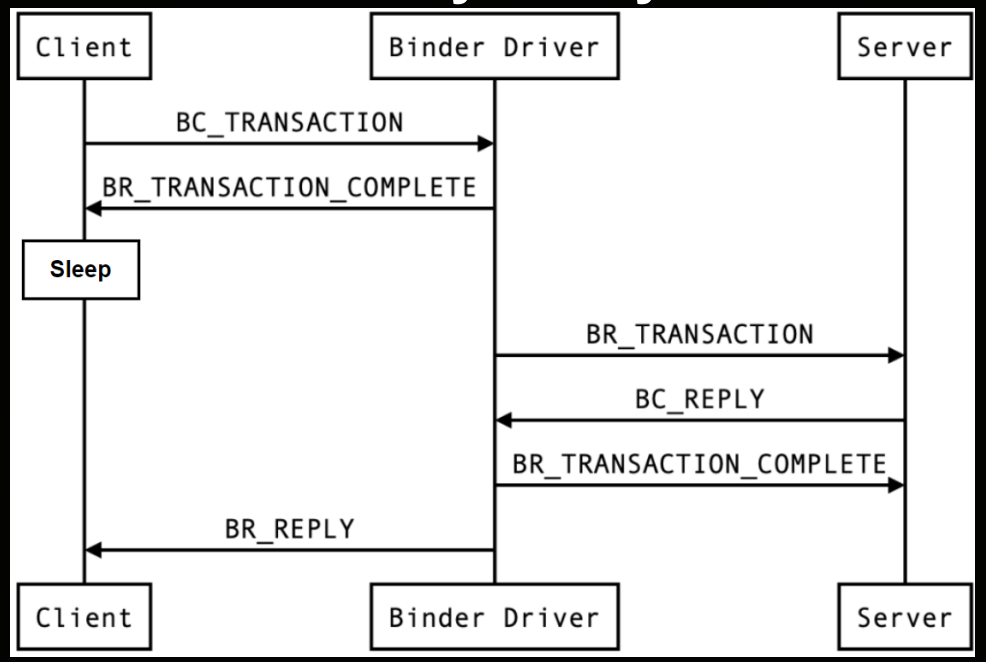
ONE-WAY Mode

- What is one-way mode?

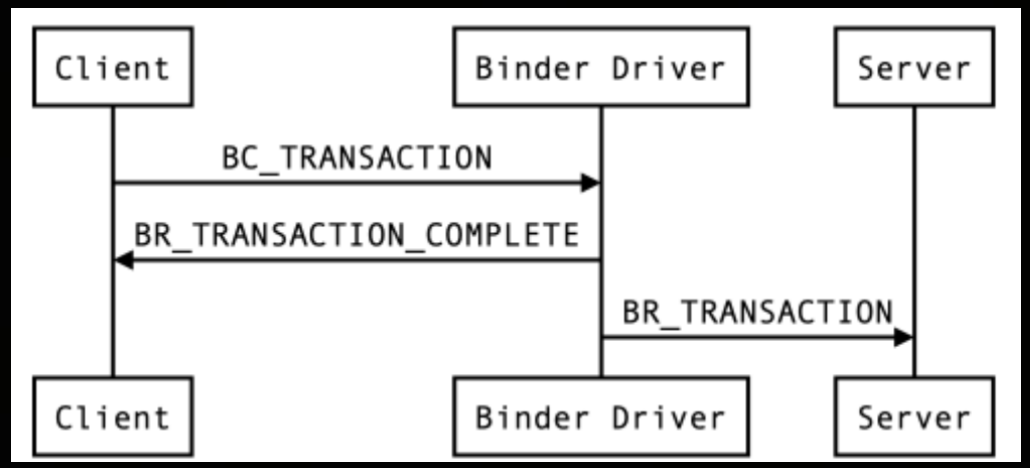
```
oneway interface ITransitionPlayer {
    /**
     * Called when something in WMCore requires a transition to play -- for example when an Activity
     * is started in a new Task.
     *
     * @param transitionToken An identifying token for the transition that needs to be started.
     *                        Pass this to {@link IWindowOrganizerController#startTransition}.
     * @param request Information about this particular request.
     */
    void requestStartTransition(in IBinder transitionToken, in TransitionRequestInfo request);
```

ONE-WAY Mode

- What is one-way mode?
  - Non-oneway := Sync IPC
  - Oneway := Async IPC



Non-oneway



Oneway

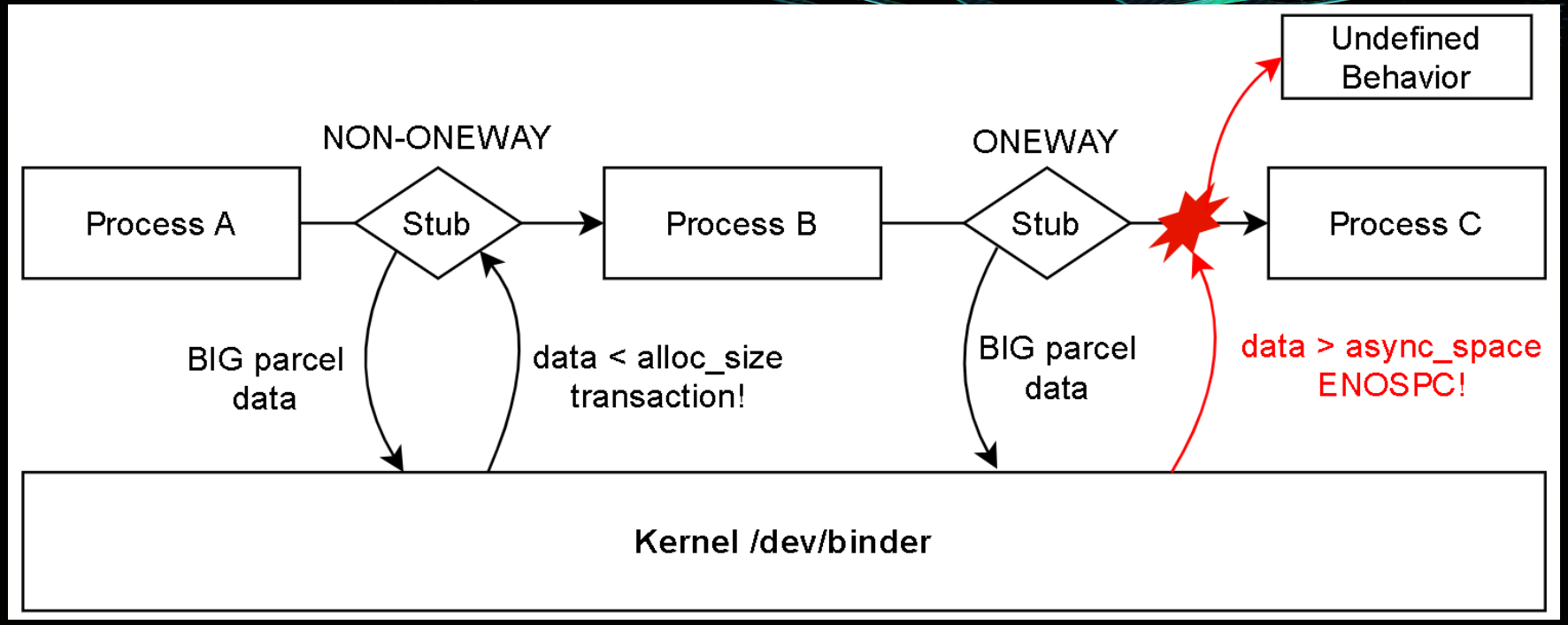
ONE-WAY Mode

- What is one-way mode?
  - Oneway IPC size := Non-oneway / 2
  - Oversize lead to IPC fail! Ret -ENOSPC

```
/* binder_alloc.c#binder_alloc_mmap_handler */
binder_insert_free_buffer(alloc, buffer);
alloc->free_async_space = alloc->buffer_size / 2;
binder_alloc_set_vma(alloc, vma);
mmgrab(alloc->vma_vm_mm);
```

```
/* binder_alloc.c#binder_alloc_new_buf_locked */
if (is_async &&
    alloc->free_async_space < size + sizeof(struct binder_buffer)) {
        binder_alloc_debug(BINDER_DEBUG_BUFFER_ALLOC,
                           "%d: binder_alloc_buf size %zd failed, no async space left\n",
                            alloc->pid, size);
        return ERR_PTR(-ENOSPC);
```

Attack Scheme



DEBUG Kernel

- Boot.img + kallsyms+ Kprobe/eBPF
  - Need to comfirm Size of ONEWAY~NON-ONEWAY

```
FFFFFFFF81F9CFF4 48 39 CB                                cmp     rbx, rcx
FFFFFFFF81F9CFF7 0F 82 FB 00 00 00                       jb      loc_FFFFFFFF81F9D0F8
FFFFFFFF81F9CFFD 4D 8D 5F 68                             lea     r11, [r15+68h]  ; r15 = a1 = alloc_ptr
FFFFFFFF81F9D001 0F 1F 44 00 00                          nop     dword ptr [rax+rax+00h]
FFFFFFFF81F9D006
FFFFFFFF81F9D006                         loc_FFFFFFFF81F9D006:                   ; CODE XREF: .text:FFFFFFFF81F9D6E8↓j
FFFFFFFF81F9D006                                                                 ; .text:FFFFFFFF81F9D746↓j ...
FFFFFFFF81F9D006 45 85 C0                                test    r8d, r8d
FFFFFFFF81F9D009 0F 84 36 01 00 00                       jz      loc_FFFFFFFF81F9D145
FFFFFFFF81F9D00F 48 8D 43 68                             lea     rax, [rbx+68h]  ; rbx = size
FFFFFFFF81F9D00F                                                                 ; 0x68 = sizeof(struct binder_buffer)
FFFFFFFF81F9D013 49 39 03                                cmp     [r11], rax
```

ONEWAY

```
FFFFFFFF81F9D1A0
FFFFFFFF81F9D1A0                         loc_FFFFFFFF81F9D1A0:                   ; CODE XREF: binder_alloc_new_buf_locked+208↑j
FFFFFFFF81F9D1A0 4D 8B 6F 40                                 mov     r13, [r15+40h]
FFFFFFFF81F9D1A4 4D 03 6F 78                                 add     r13, [r15+78h]
FFFFFFFF81F9D1A8
FFFFFFFF81F9D1A8                         loc_FFFFFFFF81F9D1A8:                   ; CODE XREF: binder_alloc_new_buf_locked+20E↑j
FFFFFFFF81F9D1A8 4D 2B 6E 58                                 sub     r13, [r14+58h]  ; r13 = buffer_size
FFFFFFFF81F9D1AC 4C 39 EF                                    cmp     rdi, r13
FFFFFFFF81F9D1AF 72 BF                                       jb      short loc_FFFFFFFF81F9D170
FFFFFFFF81F9D1B1 0F 86 C5 00 00 00                           jbe     loc_FFFFFFFF81F9D27C
```

NON-ONEWAY

DEBUG Kernel

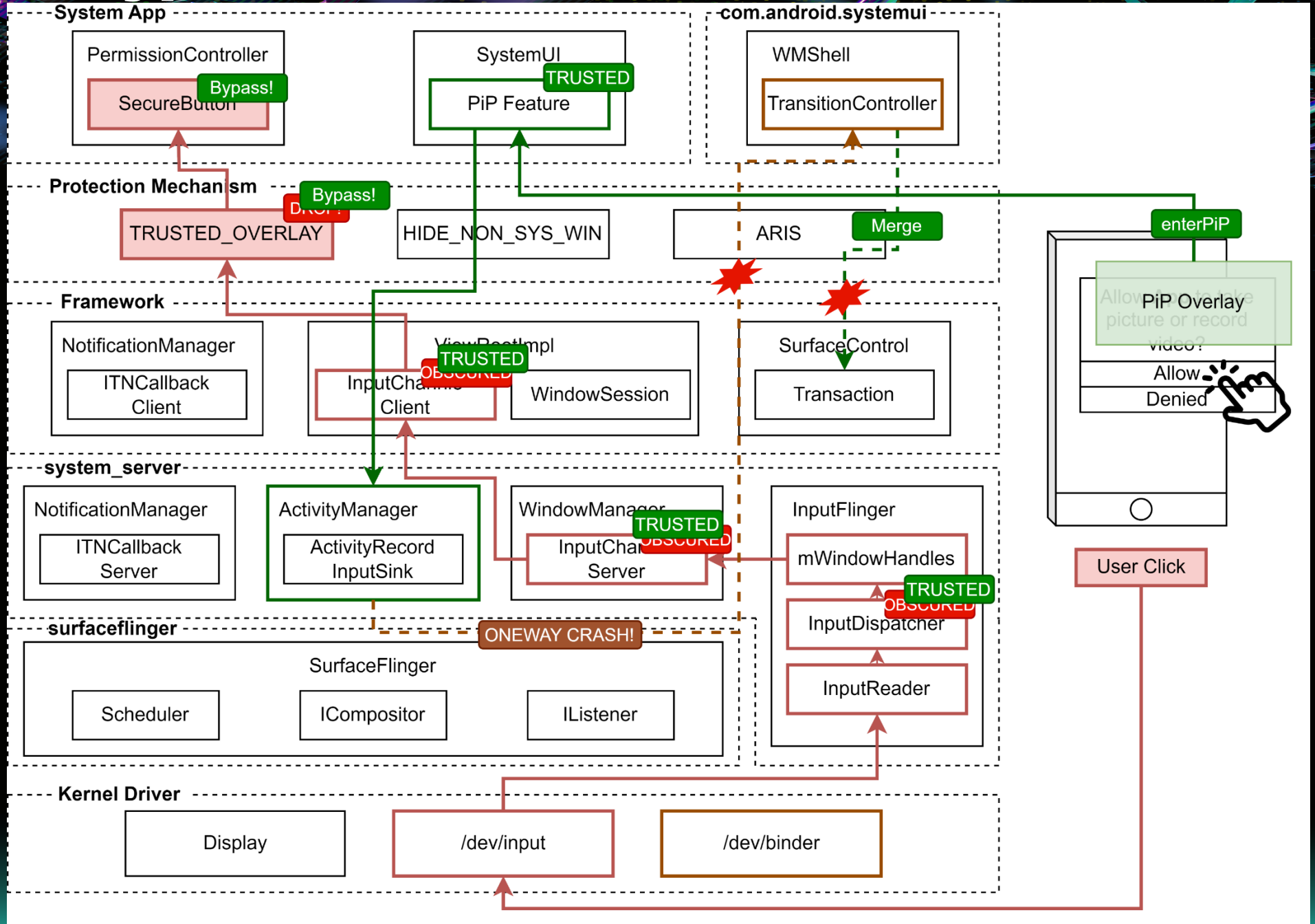
- 0x7f000~0xfe000
  - 508kb~1016kb

```
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| / _-=> migrate-disable
#                            |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
 m.android.phone-1011    [000] d.Z..   954.389850: oneway_alloc: (0xffffffff81f9d006/0x7d0) async_space=0x7f000 r11=0xffff88802227ea10
 m.android.phone-1011    [000] d.Z..   954.389857: non_oneway_alloc: (0xffffffff81f9d1ac/0x7d0) buffer_size=0xfe000
    binder:568_C-1081    [000] d.Z..   954.390069: oneway_alloc: (0xffffffff81f9d006/0x7d0) async_space=0x7f000 r11=0xffff8880222ab210
    binder:568_C-1081    [000] d.Z..   954.390072: non_oneway_alloc: (0xffffffff81f9d1ac/0x7d0) buffer_size=0xfe000
 m.android.phone-1011    [000] d.Z..   954.390161: oneway_alloc: (0xffffffff81f9d006/0x7d0) async_space=0x7f000 r11=0xffff88802227ea10
 m.android.phone-1011    [000] d.Z..   954.390163: non_oneway_alloc: (0xffffffff81f9d1ac/0x7d0) buffer_size=0xfe000
    binder:568_C-1081    [000] d.Z..   954.390396: oneway_alloc: (0xffffffff81f9d006/0x7d0) async_space=0x7f000 r11=0xffff8880222ab210
    binder:568_C-1081    [000] d.Z..   954.390398: non_oneway_alloc: (0xffffffff81f9d1ac/0x7d0) buffer_size=0xfe000
 m.android.phone-1011    [000] d.Z..   954.390474: oneway_alloc: (0xffffffff81f9d006/0x7d0) async_space=0x7f000 r11=0xffff88802227ea10
 m.android.phone-1011    [000] d.Z..   954.390476: non_oneway_alloc: (0xffffffff81f9d1ac/0x7d0) buffer_size=0xfe000
    binder:568_C-1081    [000] d.Z..   954.390543: oneway_alloc: (0xffffffff81f9d006/0x7d0) async_space=0x7f000 r11=0xffff8880222ab210
```

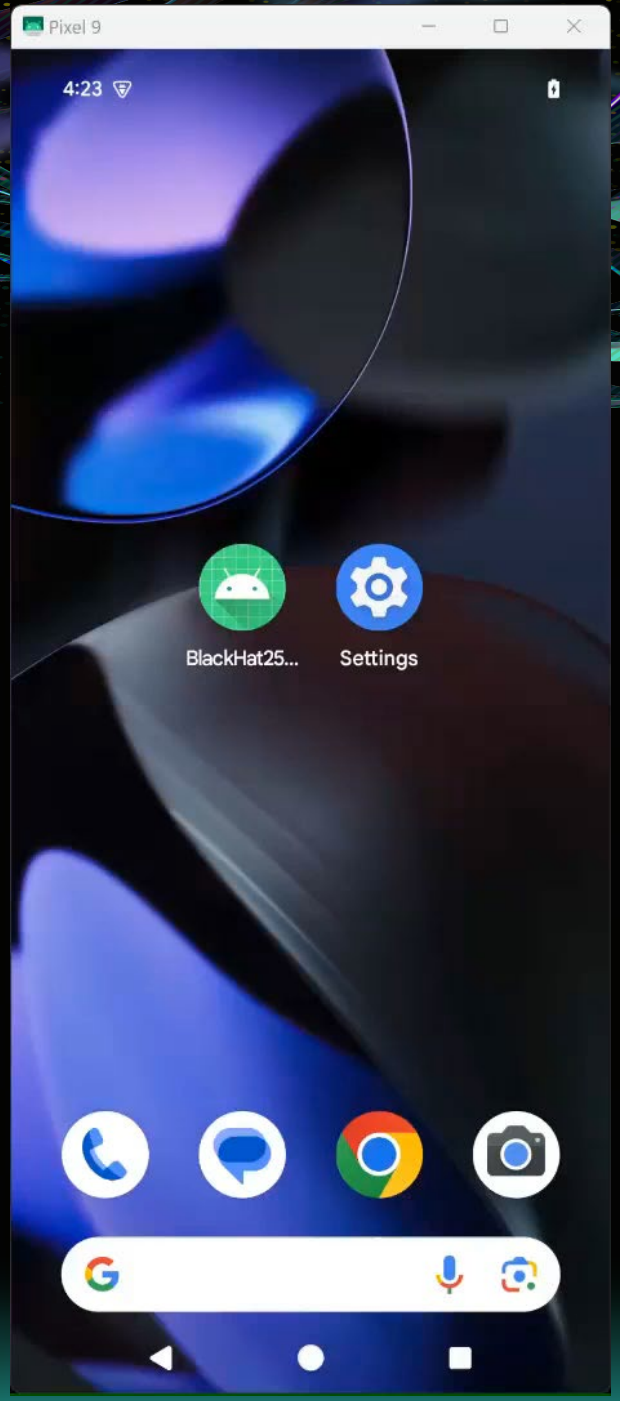
Attack WMShell IPC

```
JavaBinder: !!! FAILED BINDER TRANSACTION !!!  (parcel size = 617408)
TransitionController: Error requesting transition
TransitionController: android.os.TransactionTooLargeException: data parcel size 617408 bytes
TransitionController:  at android.os.BinderProxy.transactNative(Native Method)
TransitionController:  at android.os.BinderProxy.transact(BinderProxy.java:584)
TransitionController:  at android.window.ITransitionPlayer$Stub$Proxy.requestStartTransition(ITransitionPlayer.java:212)
TransitionController:  at com.android.server.wm.TransitionController.requestStartTransition(TransitionController.java:713)
TransitionController:  at com.android.server.wm.RootWindowContainer.moveActivityToPinnedRootTask(RootWindowContainer.java:217

TransitionController:  at com.android.server.wm.ActivityTaskManagerService.lambda$enterPictureInPictureMode$7(ActivityTaskMan

TransitionController:  at com.android.server.wm.ActivityTaskManagerService.$r8$lambda$szn0wXrSHXHk47wGyBaix3VcytI(ActivityTas
```

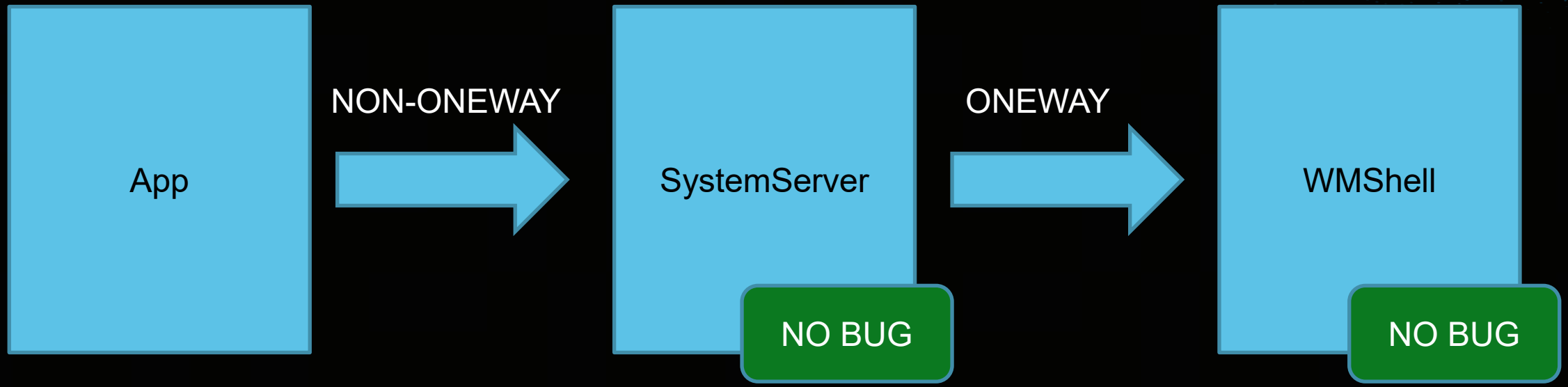
How we Bypass it?



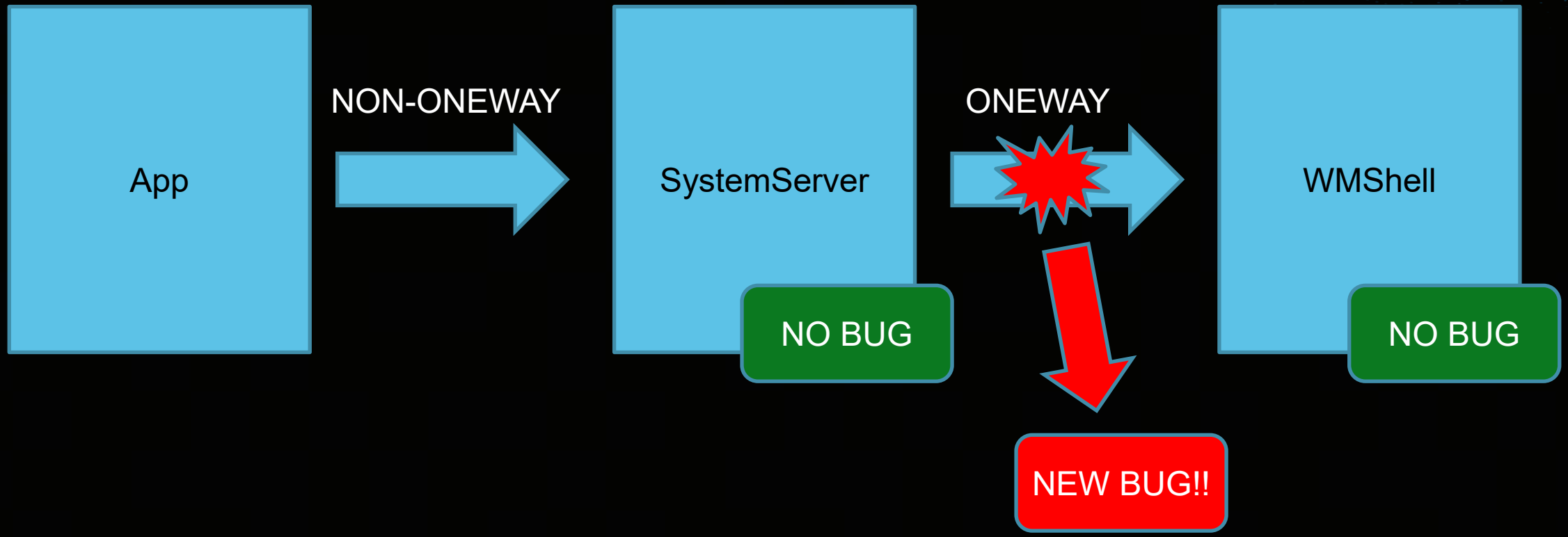
DEMO



Binder IPC Difference



Binder IPC Difference



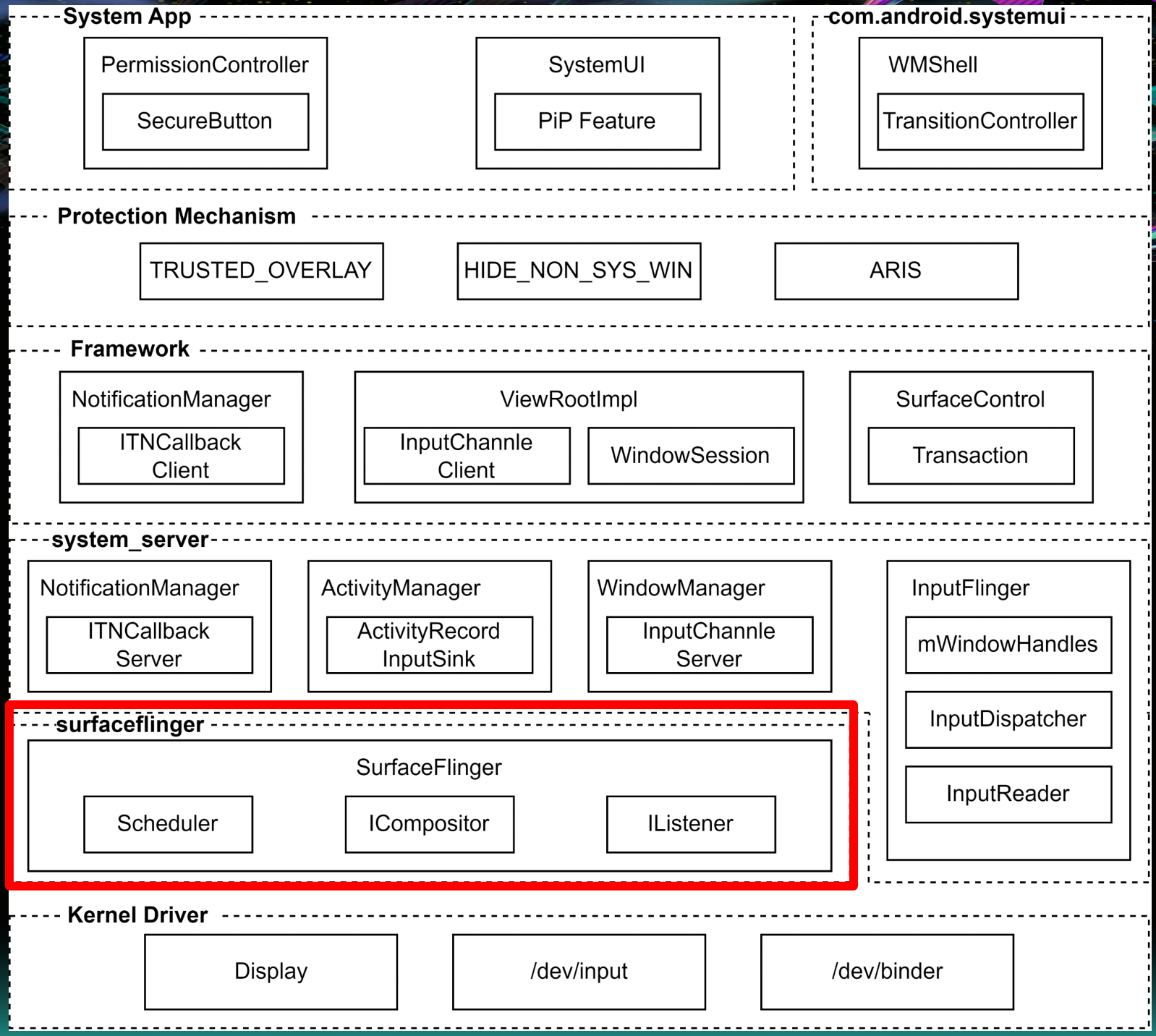
Binder IPC Difference

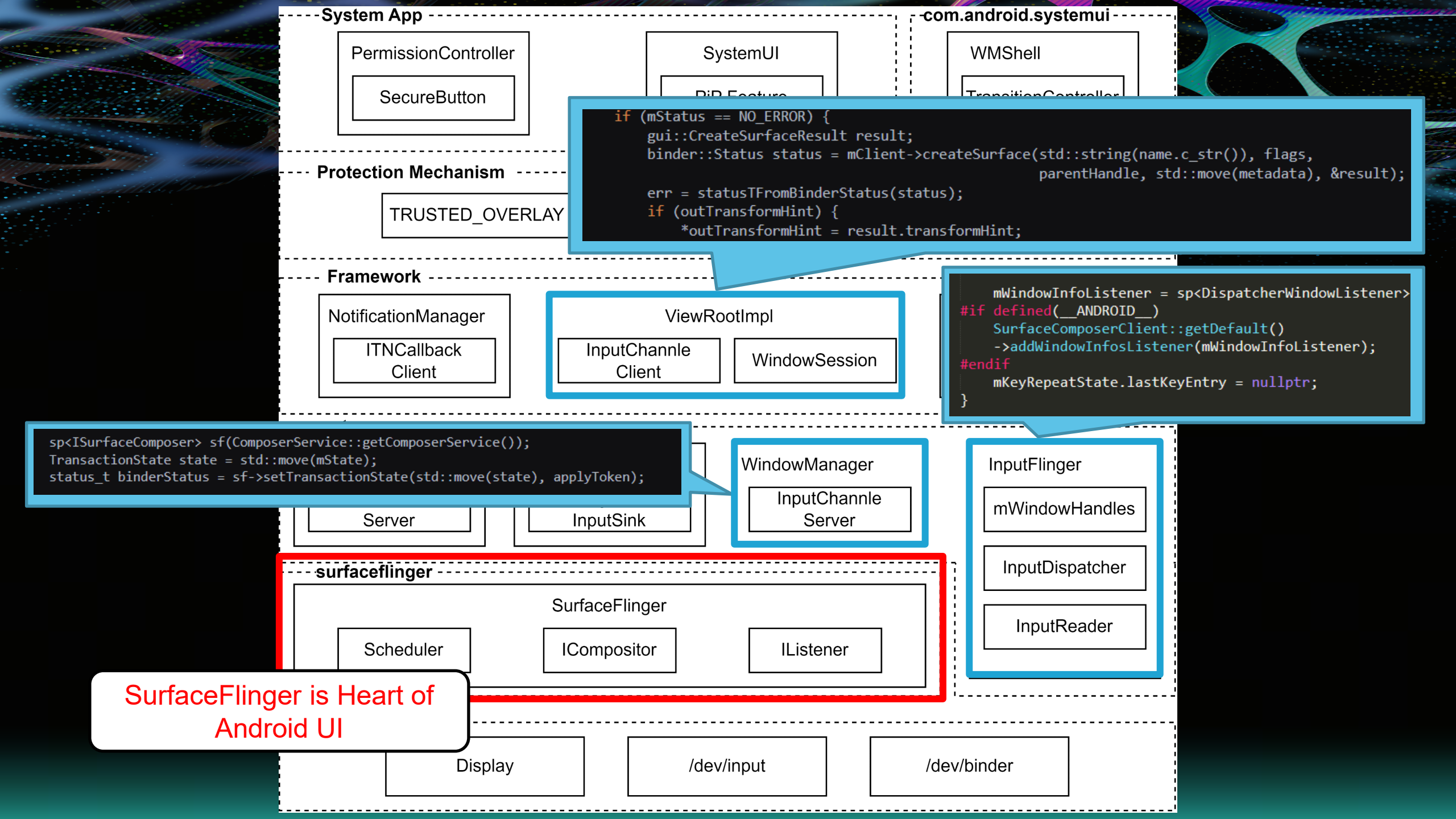
- Not limit to WMShell, Go find a bigger target!
- Not only one Component walk ONEWAY IPC

Binder IPC Difference

- Not limit to WMShell, Go find a bigger target!
- Not only one Component walk ONEWAY IPC

We know Binder IPC is heart of Android, but what is heart of Android UI?


System App
PermissionController
SecureButton
SystemUI
PiP Feature
com.android.systemui
WMShell
TransitionController
Protection Mechanism
TRUSTED_OVERLAY
HIDE_NON_SYS_WIN
ARIS
Framework
NotificationManager
ITNCallback Client
ViewRootImpl
InputChannle Client
WindowSession
SurfaceControl
Transaction
system_server
NotificationManager
ITNCallback Server
ActivityManager
ActivityRecord InputSink
WindowManager
InputChannle Server
InputFlinger
mWindowHandles
InputDispatcher
InputReader
surfaceflinger
SurfaceFlinger
Scheduler
ICompositor
IListener
Kernel Driver
Display
/dev/input
/dev/binder

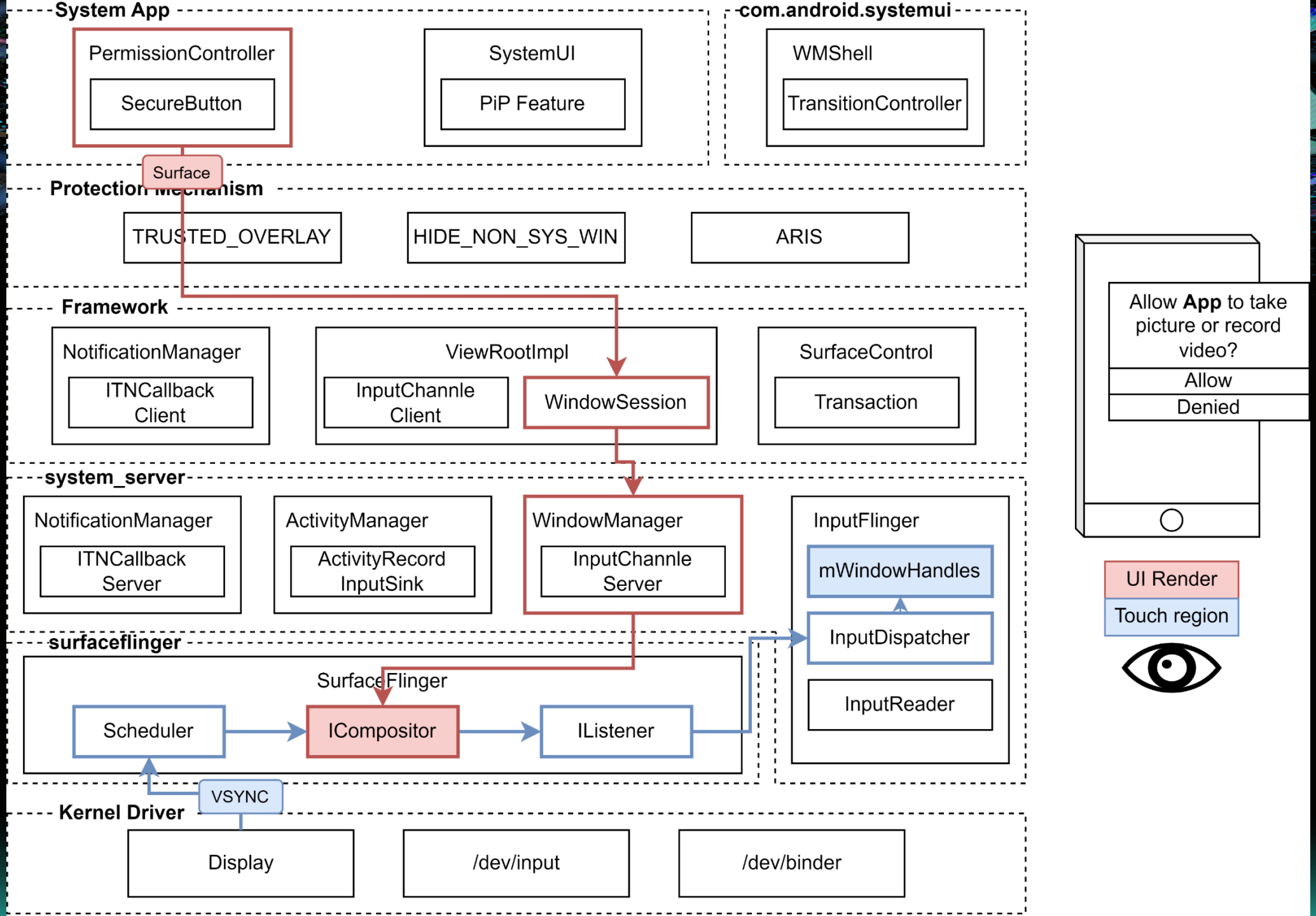

System App
PermissionController
SecureButton
SystemUI
com.android.systemui
WMShell
Protection Mechanism
TRUSTED_OVERLAY
if (mStatus == NO_ERROR) {
gui::CreateSurfaceResult result;
binder::Status status = mClient->createSurface(std::string(name.c_str()), flags,
parentHandle, std::move(metadata), &result);
err = statusTFromBinderStatus(status);
if (outTransformHint) {
*outTransformHint = result.transformHint;
Framework
NotificationManager
ITNCallback Client
ViewRootImpl
InputChannle Client
WindowSession
mWindowInfoListener = sp<DispatcherWindowListener>
#if defined(__ANDROID__)
SurfaceComposerClient::getDefault()
->addWindowInfosListener(mWindowInfoListener);
#endif
mKeyRepeatState.lastKeyEntry = nullptr;
}
sp<ISurfaceComposer> sf(ComposerService::getComposerService());
TransactionState state = std::move(mState);
status_t binderStatus = sf->setTransactionState(std::move(state), applyToken);
Server
InputSink
WindowManager
InputChannle Server
InputFlinger
mWindowHandles
InputDispatcher
InputReader
surfaceflinger
SurfaceFlinger
Scheduler
ICompositor
IListener
SurfaceFlinger is Heart of Android UI
Display
/dev/input
/dev/binder

Binder IPC Difference

- Not limit to WMShell, Go find a bigger target!
- Not only one Component walk ONEWAY IPC

We know Binder IPC is heart of Android, but what is heart of Android UI?

- SurfaceFlinger is Heart of Android UI
- There must have IPC Difference between SF and other components


System App
PermissionController
SecureButton
SystemUI
PiP Feature
com.android.systemui
WMShell
TransitionController
Surface
Protection Mechanism
TRUSTED_OVERLAY
HIDE_NON_SYS_WIN
ARIS
Framework
NotificationManager
ITNCallback Client
ViewRootImpl
InputChannle Client
WindowSession
SurfaceControl
Transaction
system_server
NotificationManager
ITNCallback Server
ActivityManager
ActivityRecord InputSink
WindowManager
InputChannle Server
InputFlinger
mWindowHandles
InputDispatcher
InputReader
surfaceflinger
SurfaceFlinger
Scheduler
ICompositor
IListener
VSYNC
Kernel Driver
Display
/dev/input
/dev/binder
Allow App to take picture or record video?
Allow
Denied
UI Render
Touch region

SurfaceFlinger / InputDispatcher

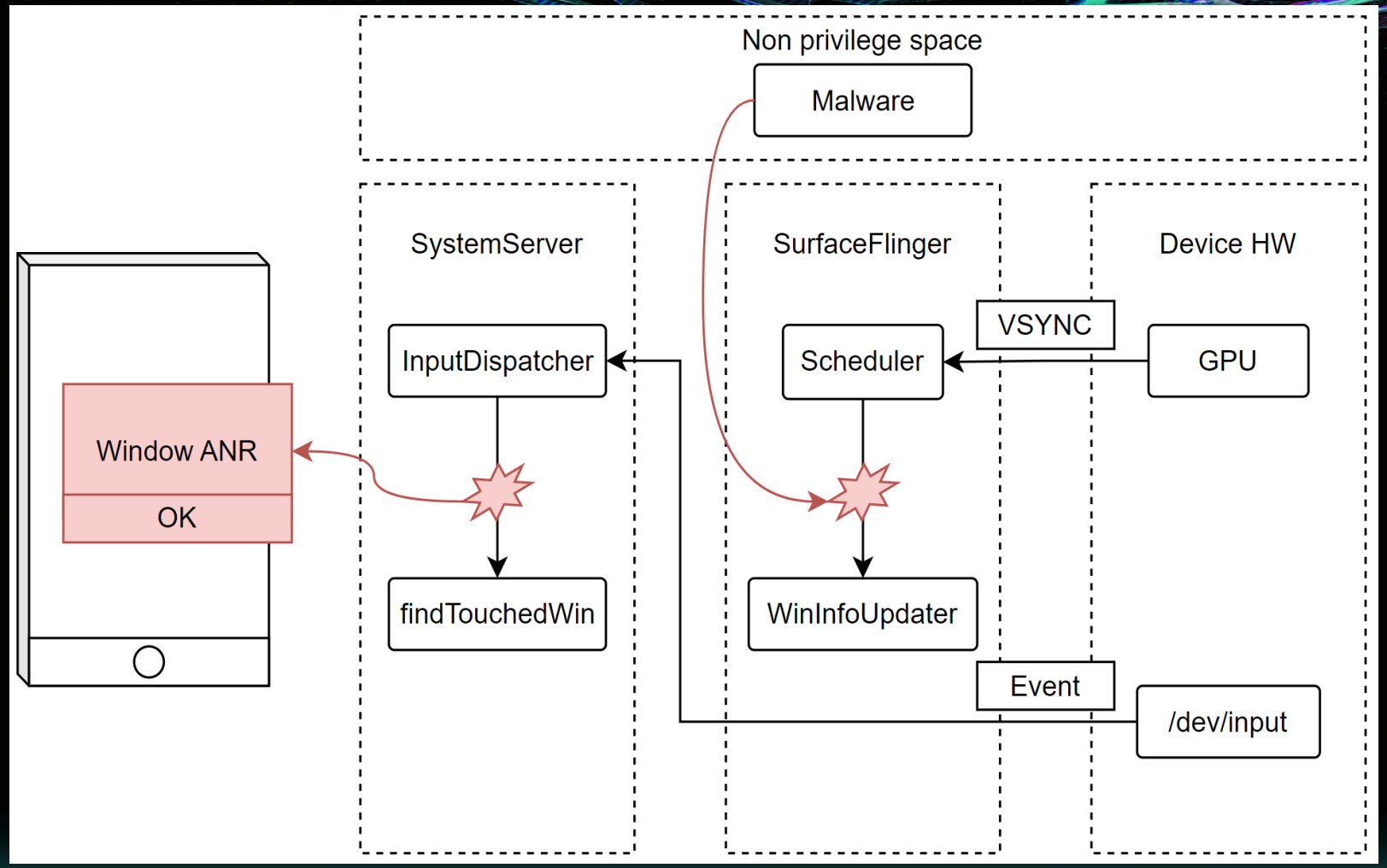
- SF reaction to VSYNC signal
- InputDispatcher rely on the data provided by SF
- Different but closely cooperating

SurfaceFlinger / InputDispatcher

- SF reaction to VSYNC signal
- InputDispatcher rely on the data provided by SF
- Different but closely cooperating

  Different can lead to bugs

SurfaceFlinger / InputDispatcher



InputDispatcher

- How InputDispatcher know which window you click?

```
std::pair<sp<WindowInfoHandle>, std::vector<InputTarget>>
InputDispatcher::findTouchedWindowAtLocked(int32_t displayId, float x, float y, bool isStylus,
                                           bool ignoreDragWindow) const {
    // Traverse windows from front to back to find touched window.
    std::vector<InputTarget> outsideTargets;
    const auto& windowHandles = getWindowHandlesLocked(displayId);
    for (const sp<WindowInfoHandle>& windowHandle : windowHandles) {
        if (ignoreDragWindow && haveSameToken(windowHandle, mDragState->dragWindow)) {
            continue;
        }

        const WindowInfo& info = *windowHandle->getInfo();
        if (!info.isSpy() &&
            windowAcceptsTouchAt(info, displayId, x, y, isStylus, getTransformLocked(displayId))) {
            return {windowHandle, outsideTargets};
        }
```

InputDispatchrt#findTouchedWindowAtLocked

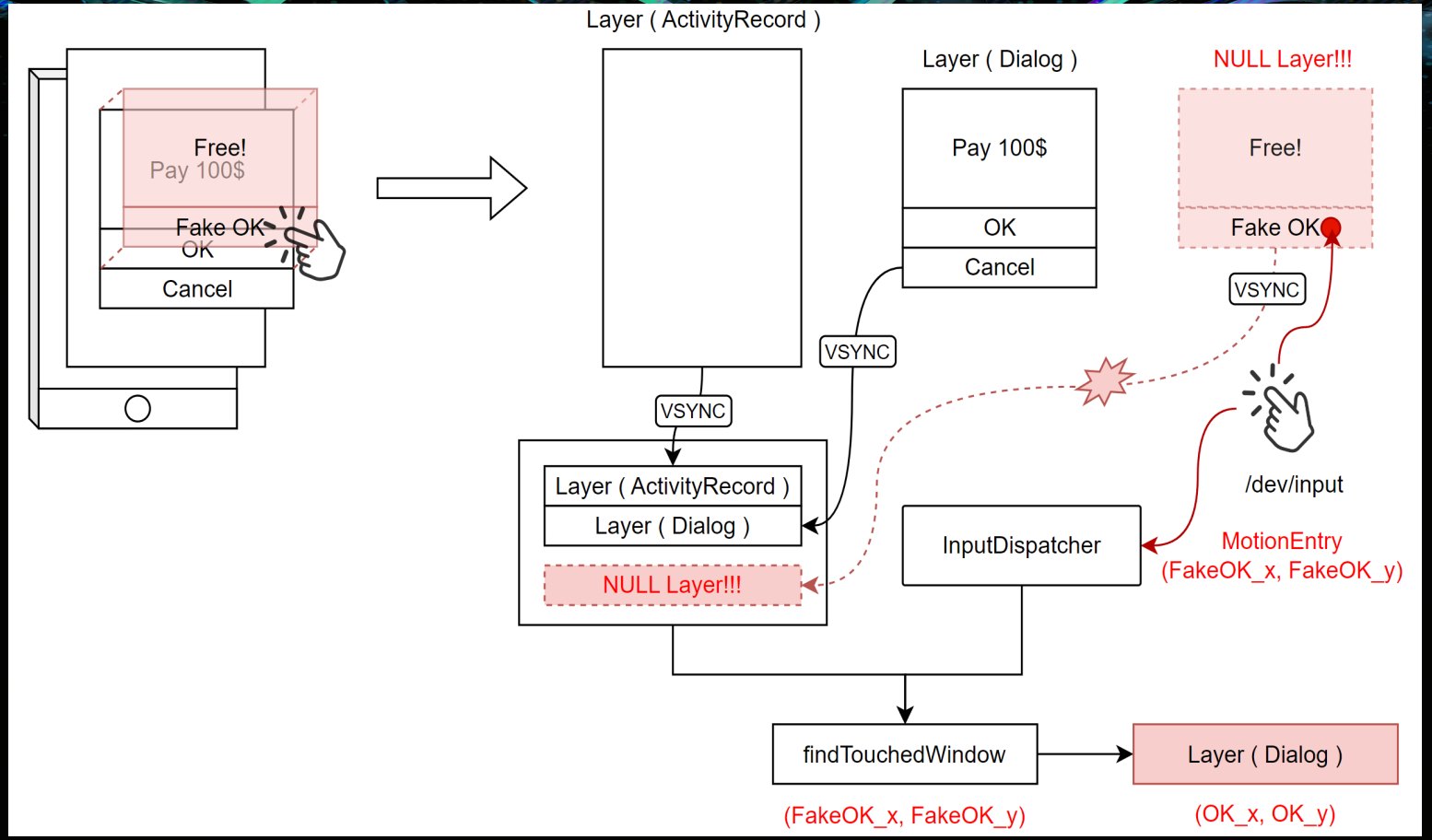
InputDispatcher

- InputDispatcher will search from toppest window
  - Skip WATCH_OUTSIDE_TOUCH flag window

```
libinputflinger.so`android::inputdispatcher::InputDispatcher::findTouchedWindowAtLocked:
0x721bd7067c <+152>: add    x25, x26, #0x10
0x721bd70680 <+156>: mov    x0, x25
0x721bd70684 <+160>: bl     0x721bdafdb0   ; symbol stub for: android::gui::WindowInfo::isSpy() const
(lldb)  p (*(int*)($x26+0x10+32)&1)==1?(*(char**)($x26+0x10+32+16)):((char*)($x26+0x10+32+1))
(char *) 0xb4000074714cf150 "629027f ScreenDecorOverlayBottom"
(lldb)  p (*(int*)($x26+0x10+32)&1)==1?(*(char**)($x26+0x10+32+16)):((char*)($x26+0x10+32+1))
(char *) 0xb4000072f14a4ed0 "f4d9dab ScreenDecorOverlay"
(lldb)  p (*(int*)($x26+0x10+32)&1)==1?(*(char**)($x26+0x10+32+16)):((char*)($x26+0x10+32+1))
(char *) 0xb4000074914b54c1 "StrictModeFlash"
(lldb)  p (*(int*)($x26+0x10+32)&1)==1?(*(char**)($x26+0x10+32+16)):((char*)($x26+0x10+32+1))
(char *) 0xb40000749149ea81 "fac065f NavigationBar0"
(lldb)  p (*(int*)($x26+0x10+32)&1)==1?(*(char**)($x26+0x10+32+16)):((char*)($x26+0x10+32+1))
(char *) 0xb4000074914a8d41 "2b5ddd3 StatusBar"
(lldb)  p (*(int*)($x26+0x10+32)&1)==1?(*(char**)($x26+0x10+32+16)):((char*)($x26+0x10+32+1))
(char *) 0xb4000074714bf250 "a93e7f6 (1-1)-Window{18dc4ab alpha=1.0 fl=}"
(lldb)  p (*(int*)($x26+0x10+32)&1)==1?(*(char**)($x26+0x10+32+16)):((char*)($x26+0x10+32+1))
(char *) 0xb4000072e147b5d0 "bb50efe com.google.android.apps.nexuslauncher/com.google.android.apps.nexuslauncher.NexusLauncherActivity"
```

Difference

- InputDispatcher only search where you can touch
  - **it don’t care wether you can see it or not**
- SurfaceFlinger only “show” your window
  - **It don’t care wether you can touch it ot not!**
- **Difference lead to BUG!**



VSYNC Drive SF to SHOW

- SF will init a Scheduler to dispatch Vsync signal
- When signal comes
  - Scheduler will pack current frame time
  - Calling MessageQueue::vsyncCallback

```
mScheduler = std::make_unique<Scheduler>(static_cast<ICompositor&>(*this),
                                         static_cast<ISchedulerCallback&>(*this), features,
                                         std::move(modulatorPtr));
// ...
mScheduler->initVsync(mScheduler->getVsyncSchedule()->getDispatch(),
                      *mFrameTimeline->getTokenManager(), configs.late.sfWorkDuration);
```

```
void Scheduler::onFrameSignal(ICompositor& compositor, VsyncId vsyncId,
                              TimePoint expectedVsyncTime) {
    const TimePoint frameTime = SchedulerClock::now();

    if (!compositor.commit(frameTime, vsyncId, expectedVsyncTime)) {
```

VSYNC Drive SF to SHOW

- SF will init a Scheduler to dispatch Vsync signal
- When signal comes
  - Scheduler will pack current frame time
  - Calling MessageQueue::vsyncCallback
  - **SF pack all window data in current frame**
  - **Send them to InputDispatcher!**

```
void SurfaceFlinger::updateInputFlinger(VsyncId vsyncId, TimePoint frameTime) {
    // ...
    if (mUpdateInputInfo) {
        mUpdateInputInfo = false;
        updateWindowInfo = true;
        buildWindowInfos(windowInfos, displayInfos);
    }
    // ...
    BackgroundExecutor::getInstance().sendCallbacks({[...]() {
        ATRACE_NAME("BackgroundExecutor::updateInputFlinger");
        if (updateWindowInfo) {
            mWindowInfosListenerInvoker
                    ->windowInfosChanged(gui::WindowInfosUpdate{std::move(windo
                                                                std::move(displa
                                                                vsyncId.value, f
```

VSYNC Drive SF to SHOW

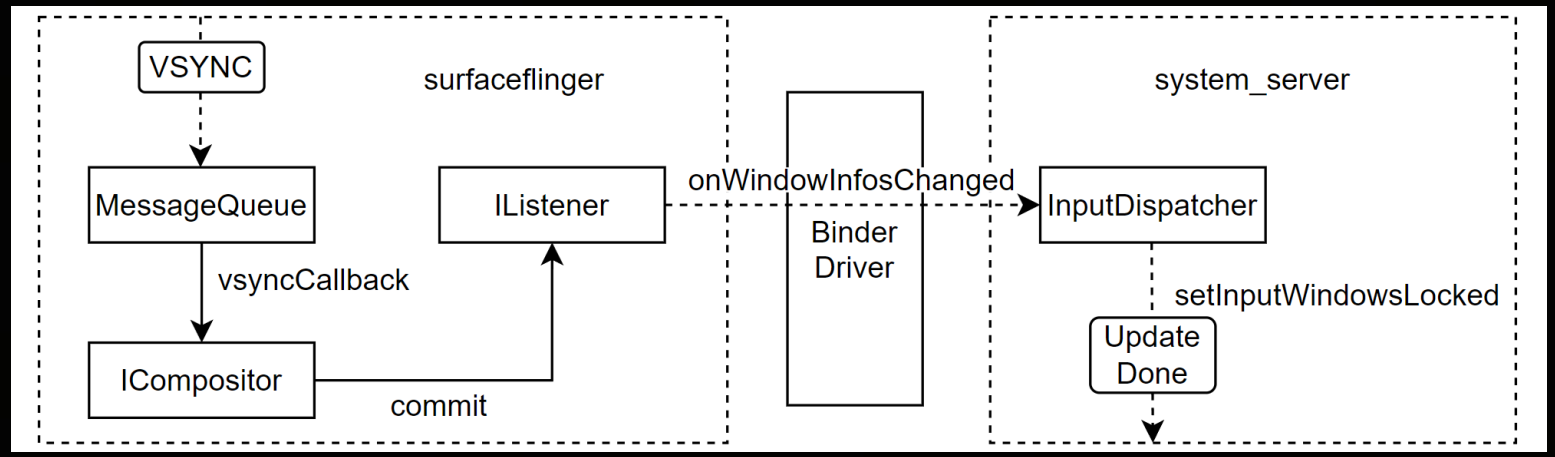
- We can control our own Window
- Means we can control what data SF send to InputDispatcher!!

```
void SurfaceFlinger::buildWindowInfos(std::vector<WindowInfo>& outWindowInfos,
                                      std::vector<DisplayInfo>& outDisplayInfos) {
    // ...
        mDrawingState.traverseInReverseZOrder([&](Layer* layer) {
            if (!layer->needsInputInfo()) return;
            const auto opt =
                    mFrontEndDisplayInfos.get(layer->getLayerStack())
                            .transform([](const frontend::DisplayInfo& info) {
                                return Layer::InputDisplayArgs{&info.transform, info.isSecure};
                            });

            outWindowInfos.push_back(layer->fillInputInfo(opt.value_or(Layer::InputDisplayArgs{})));
        });
    }
```

SurfaceFlinger#buildWindowInfos

InputDispatcher ONEWAY interface

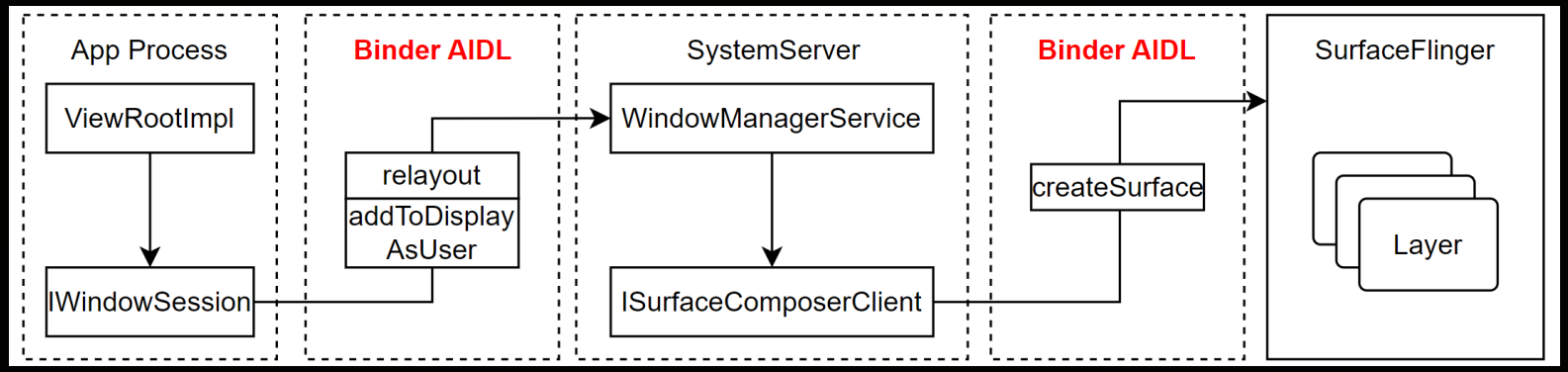
- SF IPC to InputDispatcher::onWindowInfosChanged
- BpWindowInfosListener::onWindowInfosChanged is ONEWAY!!!!!




```
.text:0000000000159CA4 21 00 80 52          MOV     W1, #1
.text:0000000000159CA8 24 00 80 52          MOV     W4, #1  ; ::android::IBinder::FLAG_ONEWAY
.text:0000000000159CAC 08 00 40 F9          LDR     X8, [X0]
.text:0000000000159CB0 08 15 40 F9          LDR     X8, [X8,#0x28]
.text:0000000000159CB4 00 01 3F D6          BLR     X8
```

POC

- Create 600kb Data window is workable!
- When SystemServer IPC to SF, use NON-ONEWAY interface!

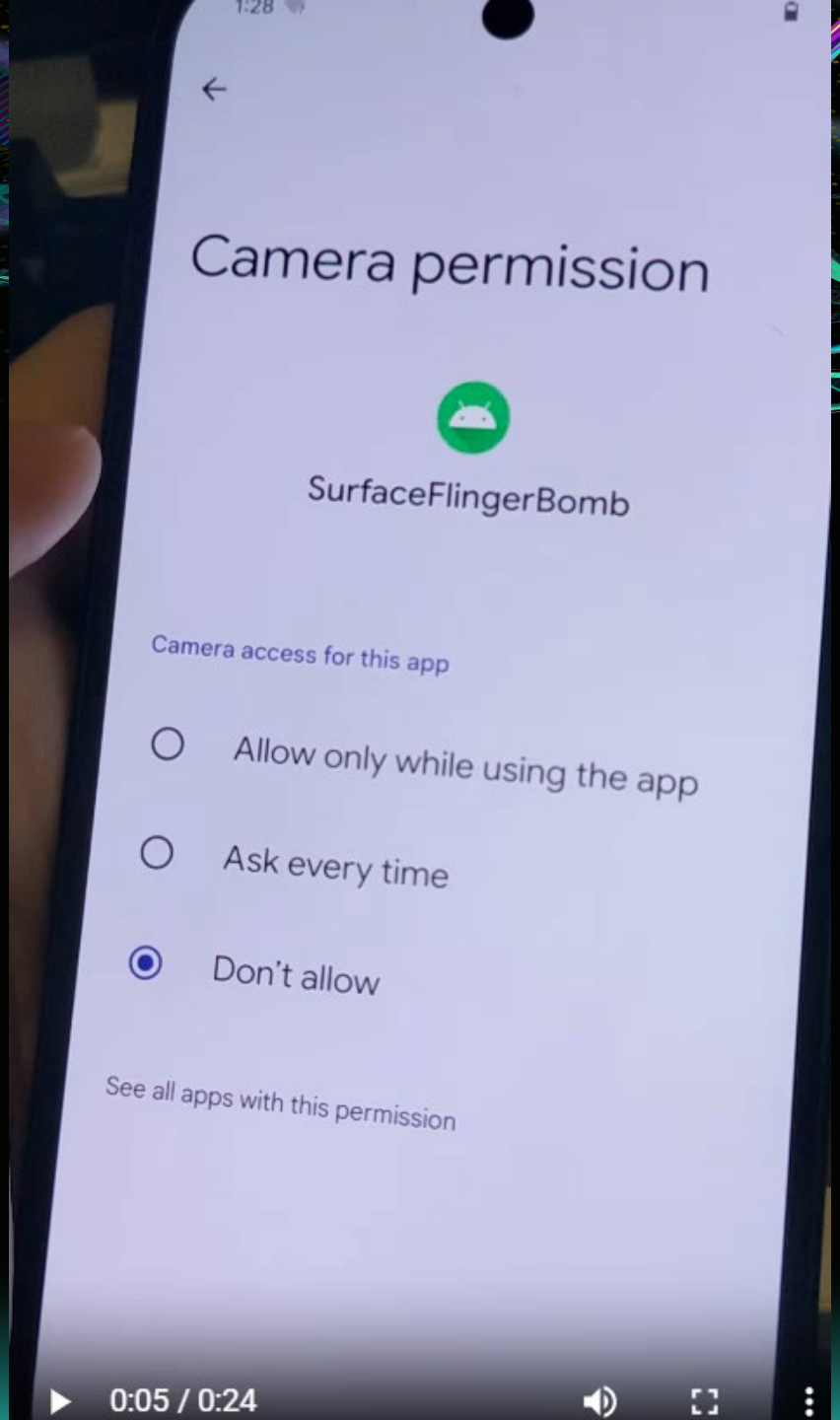



```
public int relayout(IWindow iWindow, WindowManager.LayoutParams layoutParams, int i,
    Parcel parcelObtain = Parcel.obtain(asBinder());
    Parcel parcelObtain2 = Parcel.obtain();
    // ...
```

```
public int addToDisplayAsUser(IWindow iWindow, WindowManager.LayoutParams layoutParams,
    Parcel parcelObtain = Parcel.obtain(asBinder());
    Parcel parcelObtain2 = Parcel.obtain();
    // ...
    try {
        this.mRemote.transact(2, parcelObtain, parcelObtain2, 0);
```

```
tain, parcelObtain2, 0);
```

DEMO


1:28
Camera permission
SurfaceFlingerBomb
Camera access for this app
Allow only while using the app
Ask every time
Don't allow
See all apps with this permission
0:05 / 0:24

Wrong Focus

- Bypass ARIS, Attack WMShell, Abusing Binder IPC…
- All we want to do is cover target SurfaceControl with our own
- What if …

Wrong Focus

- Bypass ARIS, Attack WMShell, Abusing Binder IPC…
- All we want to do is cover target SurfaceControl with our own
- What if … **We can directly control target SurfaceControl??**

Poisoning SurfaceControl

- Window related with SurfaceControl
- SurfaceControl related with Transaction
  - View size, alpha, visbility, animation…

```
public static class Transaction implements Closeable, Parcelable {
    /**
     * @hide
     */
    public static final NativeAllocationReg
            Transaction.class.getClassLoade
            nativeGetNativeTransactionFinal
    /**
     * @hide
     */
    public long mNativeObject;

    private final ArrayMap<SurfaceControl,
    private final ArrayMap<SurfaceControl,
             new ArrayMap<>();

    Runnable mFreeNativeResources;
    private static final float[] INVALID_CO
```

```
public final class SurfaceControl implements Parcelable {
    private static final String TAG = "SurfaceControl";

    private static native long nativeCreate(SurfaceSession session, String name,
            int w, int h, int format, int flags, long parentObject, Parcel metadata)
            throws OutOfResourcesException;
    private static native long nativeReadFromParcel(Parcel in);
    private static native long nativeCopyFromSurfaceControl(long nativeObject);
    private static native void nativeWriteToParcel(long nativeObject, Parcel out);
    private static native long nativeGetNativeSurfaceControlFinalizer();
    private static native void nativeDisconnect(long nativeObject);
    private static native void nativeUpdateDefaultBufferSize(long nativeObject, int width, int height);

    private static native long nativeMirrorSurface(long mirrorOfObject);
    private static native long nativeCreateTransaction();
    private static native long nativeGetNativeTransactionFinalizer();
    private static native void nativeApplyTransaction(long transactionObj, boolean sync);
    private static native void nativeMergeTransaction(long transactionObj,
            long otherTransactionObj);
```

Poisoning SurfaceControl

- Control target Activity's SurfaceControl by ActivityOptions

```
public ActivityOptions(Bundle opts) {
    super(opts);

    mPackageName = opts.getString(KEY_PACKAGE_NAME);
    try {
        mUsageTimeReport = opts.getParcelable(KEY_USAGE_TIME_REPORT, PendingIntent.class);
    } catch (RuntimeException e) {
        Slog.w(TAG, e);
    }
    mLaunchBounds = opts.getParcelable(KEY_LAUNCH_BOUNDS, android.graphics.Rect.class);
    mAnimationType = opts.getInt(KEY_ANIM_TYPE, ANIM_UNDEFINED);
    switch (mAnimationType) {
        case ANIM_CUSTOM:
            mCustomEnterResId = opts.getInt(KEY_ANIM_ENTER_RES_ID, 0);
            mCustomExitResId = opts.getInt(KEY_ANIM_EXIT_RES_ID, 0);
            mCustomBackgroundColor = opts.getInt(KEY_ANIM_BACKGROUND_COLOR, 0);
            mAnimationStartedListener = IRemoteCallback.Stub.asInterface(
                    opts.getBinder(KEY_ANIM_START_LISTENER));
            break;
```

Poisoning SurfaceControl

- Control target Activity's SurfaceControl by ActivityOptions

```
public ActivityOptions(Bundle opts) {
    super(opts);

    mPackageName = opts.getSt
    try {
        mUsageTimeReport = op
    } catch (RuntimeExceptio
        Slog.w(TAG, e);
    }
    mLaunchBounds = opts.getB
    mAnimationType = opts.get
    switch (mAnimationType) {
        case ANIM_CUSTOM:
            mCustomEnterResId
            mCustomExitResId
            mCustomBackgroun
            mAnimationStarte
                    opts.getB
            break;
```

```
void applyOptionsAnimation() {
    if (DEBUG_TRANSITION) Slog.i(TAG, "Applying options for " + this);
    if (mPendingRemoteAnimation != null) {
        mDisplayContent.mAppTransition.overridePendingAppTransitionRemote(
                mPendingRemoteAnimation);
        mTransitionController.setStatusBarTransitionDelay(
                mPendingRemoteAnimation.getStatusBarTransitionDelay());
    } else {
        if (mPendingOptions == null) {
            return;
        } else if (mPendingOptions.getAnimationType() == ANIM_SCENE_TRANSITION) {
            // Scene transition will run on the client side, so just notify transition
            // controller but don't clear the animation information from the options since they
            // need to be sent to the animating activity.
            mTransitionController.setOverrideAnimation(
                    AnimationOptions.makeSceneTransitionAnimOptions(), null, null);
            return;
        }
        applyOptionsAnimation(mPendingOptions, intent);
    }
    clearOptionsAnimationForSiblings();
}
```

ActivityRecord#applyOptionsAnimation

Poisoning SurfaceControl

- Control target Activity's SurfaceControl by ActivityOptions

```
public ActivityOptions(Bundle opts) {
    super(opts);
    mPackageName = opts.getSt
    try {
        mUsageTimeReport = op
    } catch (RuntimeExceptio
        Slog.w(TAG, e);
    }
    mLaunchBounds = opts.getB
    mAnimationType = opts.get
    switch (mAnimationType) {
        case ANIM_CUSTOM:
            mCustomEnterResId
            mCustomExitResId
            mCustomBackgroun
            mAnimationStarte
                    opts.getB
            break;
```

```
void applyOptionsAnimation() {
    if (DEBUG_TRANSITION) Slog.i(TAG, "Applying options for " + this);
    if (mPendingRemoteAnimation != null) {
    } e
    }
    cle
}
```

```
private static void applyTransformation(long time, SurfaceControl.Transaction t,
        SurfaceControl leash, Animation anim, Transformation transformation, float[] matrix,
        Point position, float cornerRadius, @Nullable Rect immutableClipRect) {
    anim.getTransformation(time, transformation);
    if (position != null) {
        transformation.getMatrix().postTranslate(position.x, position.y);
    }
    t.setMatrix(leash, transformation.getMatrix(), matrix);
    t.setAlpha(leash, transformation.getAlpha());

    final Rect clipRect = immutableClipRect == null ? null : new Rect(immutableClipRect);
    Insets extensionInsets = Insets.min(transformation.getInsets(), Insets.NONE);
    if (!extensionInsets.equals(Insets.NONE) && clipRect != null && !clipRect.isEmpty()) {
        // Clip out any overflowing edge extension
        clipRect.inset(extensionInsets);
        t.setCrop(leash, clipRect);
    }
```

DefaultTransitionHandler#applyTransformation

Poisoning SurfaceControl

- Control target Activity's SurfaceControl by ActivityOptions

```
public ActivityOptions(Bundle opts) {
    super(opts);
    mPackageName = opts.getSt
    try {
        mUsageTimeReport = op
    } catch (RuntimeExceptio
        Slog.w(TAG, e);
    }
    mLaunchBounds = opts.getB
    mAnimationType = opts.get
    switch (mAnimationType) {
        case ANIM_CUSTOM:
            mCustomEnterResId
            mCustomExitResId
            mCustomBackgroun
            mAnimationStarte
                    opts.getB
            break;
```

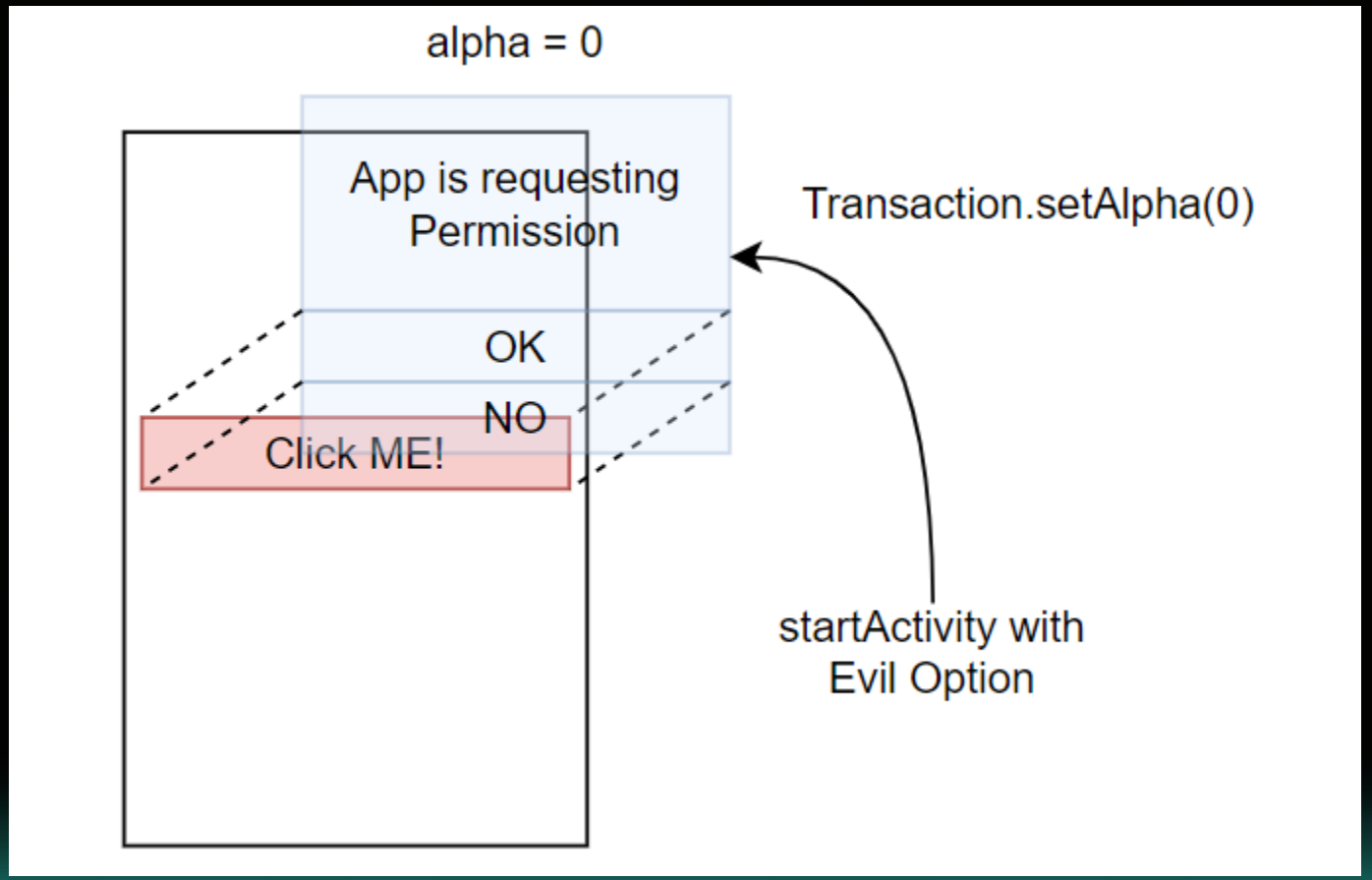
```
void applyOptionsAnimation() {
    if (DEBUG_TRANSITION) Slog.i(TAG, "Applying options for " + this);
    if (mPendingRemoteAnimation != null) {
    } e
    }
    cle
}
```

```
private static void applyTransformation(long time, SurfaceControl.Transaction t,
        SurfaceControl leash, Animation anim, Transformation transformation, float[] matrix,
        Point position, float cornerRadius, @Nullable Rect immutableClipRect) {
    anim.getTransformation(time, transformation);
    if (position != null) {
        transformation.getMatrix().postTranslate(position.x, position.y);
    }
    t.setMatrix(leash, transformation.getMatrix(), matrix);
    t.setAlpha(leash, transformation.getAlpha());

    final Rect clipRect = immutableClipRect == null ? null : new Rect(immutableClipRect);
    Insets extensionInsets = Insets.min(transformation.getInsets(), Insets.NONE);
    if (!extensionInsets.equals(Insets.NONE) && clipRect != null && !clipRect.isEmpty()) {
        // Clip out any overflowing edge extension
        clipRect.inset(extensionInsets);
        t.setCrop(leash, clipRect);
    }
```

DefaultTransitionHandler#applyTransformation

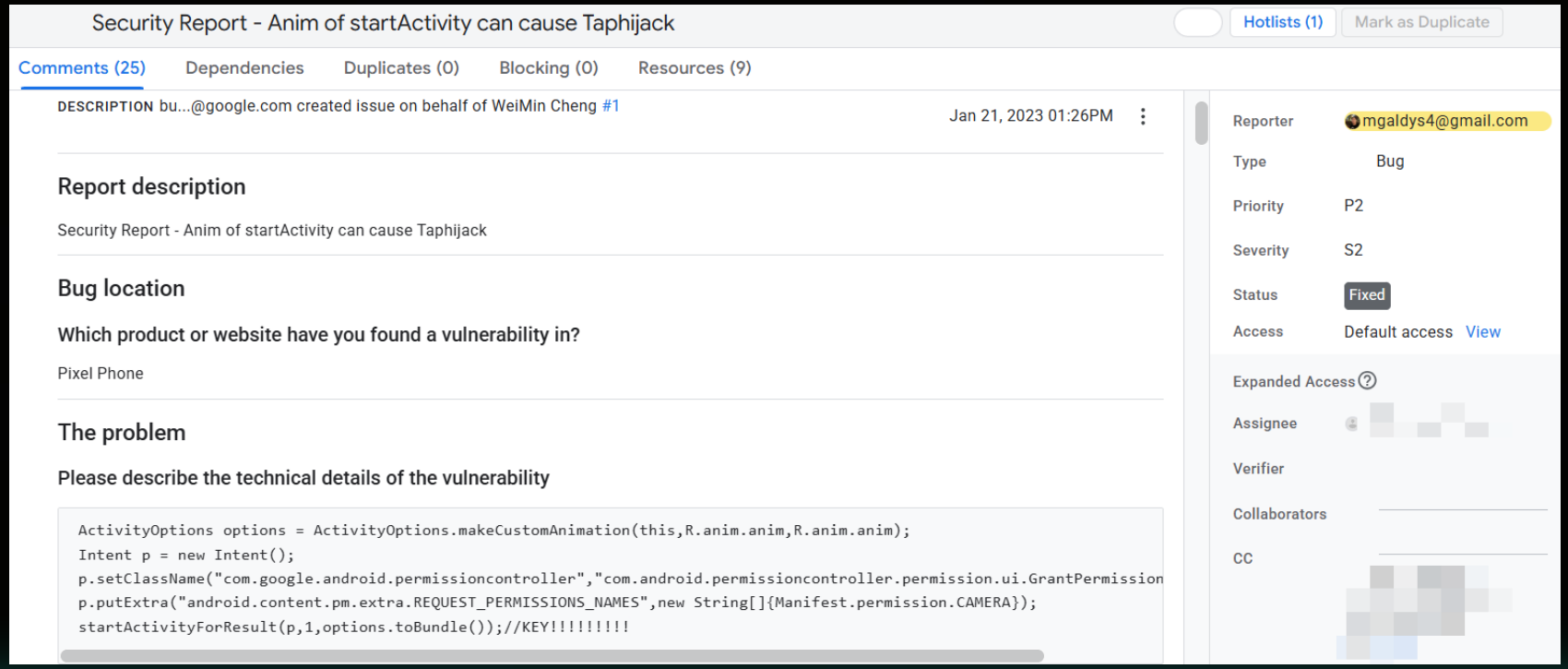
Poisoning SurfaceControl

- Control target Activity's SurfaceControl by ActivityOptions

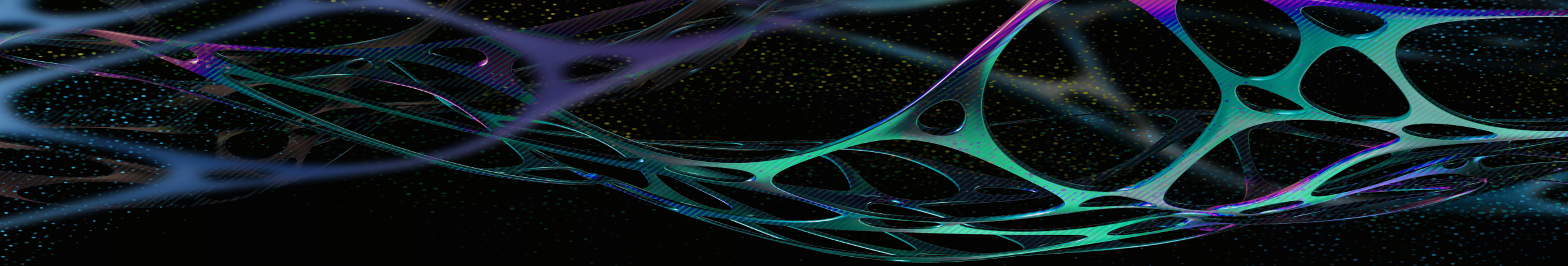


Timeline

- Found it in 2021
  - I was a sophomore at the time, not doing security research, don’t know it is a vuln.
- Report it in 2023
- Fix at Dec 20, 2025
- CVE-2025-48621



EXP CHAIN



Thank You