MWR LABS

Security Advisory

Screen Capture via UI Overlay in MediaProjection

13/11/2017

Software	Android Open Source Project (AOSP)
Affected Versions	5.0 to 7.1.2
CVE Reference	N/A
Author	Amar Menezes
Severity	High
Vendor	Google
Vendor Response	Patched in Android 8.0

Description:

Google introduced the MediaProjection service to the Android Framework in Android 5.0. This gave Android application developers the ability to capture screen contents and/or record system audio. Prior to android 5.0 application developers required their application's to run with root privileges or sign their applications with the device's release keys in order to use system protected permissions to capture screen contents.

With MediaProjection, application developers no longer need root privileges, nor do they require to sign their applications with the device's release keys. Furthermore, there are no permissions that are required to be declared in the AndroidManifest.xml in order to use the MediaProjection service.

To use the MediaProjection service, an application would simply have to request access to this system Service via an Intent. Access to this system Service is granted by displaying a SystemUI pop-up that warns the user that the requesting application would like to capture the user's screen.

It was discovered that an attacker could overlay this SystemUI pop-up which warns the user that the contents of their screen would be captured, with an arbitrary message to trick the user into granting the attacker's application the ability to capture the user's screen.

Impact:

This vulnerability would allow an attacker to capture the user's screen should the user tap on the SystemUI pop-up that has been overlayed by the attacker with an arbitrary message.

The lack of specific android permissions to use this API makes it harder to determine if an application uses the MediaProjection service.

This vulnerability is particularly severe since the SystemUI pop-up is launched within the context of the attacker's application making it possible for an attacker to detect the pop-up and draw an overlay without the user noticing.

Cause:

The primary cause of this vulnerability is due to the fact that affected Android versions are unable to detect partially obscured SystemUI pop-ups. This allows an attacker to craft an application to draw an overlay over the SystemUI pop-up which would lead to the elevation of the application's privileges (i.e. would allow it to capture the user's screen).

Furthermore, the SystemUI pop-up is the only access control mechanism available that prevents the abuse of the MediaProjection service. An attacker could trivially bypass this mechanism by tapjacking this pop-up using publicly known methods to grant their applications the ability to capture the user's screen.

Interim Workaround:

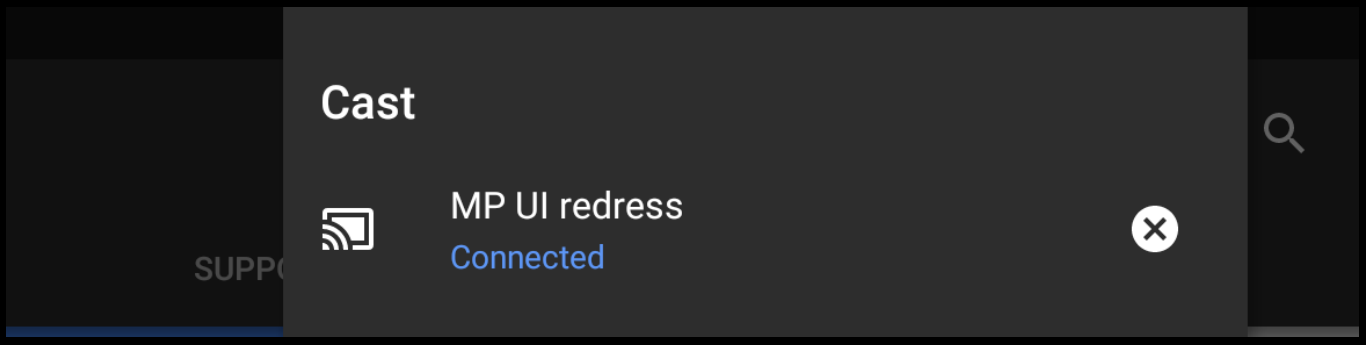
This vulnerability has currently only been patched in Android 8.0. However, due to the issue of version fragmentation within the Android ecosystem, there are a number of Android devices that can't upgrade to Android 8.0 or no longer receive updates from device vendors that would still be vulnerable. According to the Android developer dashboard as of 02-Oct-17, approximately 77.5% of active android devices are still vulnerable to this particular attack[1].

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.6%
4.1.x	Jelly Bean	16	2.3%
4.2.x		17	3.3%
4.3		18	1.0%
4.4	KitKat	19	14.5%
5.0	Lollipop	21	6.7%
5.1		22	21.0%
6.0	Marshmallow	23	32.0%
7.0	Nougat	24	15.8%
7.1		25	2.0%
8.0	Oreo	26	0.2%

Data collected during a 7-day period ending on October 2, 2017.
Any versions with less than 0.1% distribution are not shown.

However, this attack is not entirely undetectable. When an application gains access to the MediaProjection Service, it generates a Virtual Display which activates the screencast icon in the notification bar. Should users

see a screencast icon in their devices notification bar, they should investigate the application/process currently running on their devices. An example of which is shown below:



Solution:

This vulnerability has been addressed in Android 8.0 and Android users are advised to update to Android 8.0. It is unclear if Google plans to release a patches for older affected versions of Android. However, should a security patch for older affected versions of Android be released at a later date, users should update their devices.

Android application developers can defend against this attack by enabling the FLAG_SECURE layout parameter via the application's WindowManager. This would ensure that the content of the applications windows are treated as secure, preventing it from appearing in screenshots or from being viewed on non-secure displays. An example of how the FLAG_SECURE parameter could be added to an applications Activities is shown below:

```
public class ApplicationActivity extends AppCompatActivity {


  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);


    getWindow().setFlags(LayoutParams.FLAG_SECURE,
                         LayoutParams.FLAG_SECURE);



    /* application code follows */
  }
}
```

Technical details

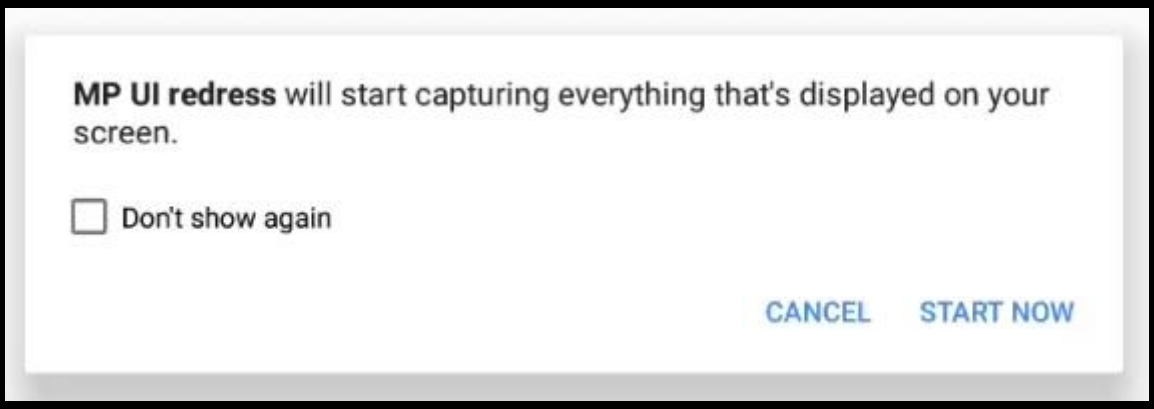
In order to use the MediaProjection service an application developer would create a *MediaProjectionManager* instance that connects to the system service MEDIA_PROJECTION_SERVICE. An example of which is shown below:

```
mProjectionManager = (MediaProjectionManager)
getSystemService(Context.MEDIA_PROJECTION_SERVICE);
```

Once a MediaProjectionManager object has been instantiated the application developer would then generate an Intent to request for screen capture permission. An example of which shown below:

```
int REQUEST_CODE = 1000;
startActivityForResult(mProjectionManager.createScreenCaptureIntent(), REQUEST_CODE);
```

Launching this Intent triggers the SystemUI pop-up that warns the user of the application that all activity on their screen is going to be recorded. The following screenshot is an example of this pop-up:



The user's response to this pop-up can be detected using *onActivityResult* function. A typical example of which is shown below:

```
public void onActivityResult(int requestCode, int resultCode, Intent data) {
        mMediaProjectionCallback = new MediaProjectionCallback();
        mMediaProjection = mProjectionManager.getMediaProjection(resultCode, data);
        mMediaProjection.registerCallback(mMediaProjectionCallback, null);
        mVirtualDisplay = createVirtualDisplay();
        mMediaRecorder.start();
    }
```

Application developers can reliably control the UI feedback loop since the application developer controls when the intent is launched to generate the SystemUI pop-up and can detect user supplied input to this pop-up within the *onActivityResult()*. An attacker could leverage this to draw an overlay over the SystemUI pop-up.

In order to do that an attacker would have to first draw an overlay before launching the Screen Capture intent. An example of how this could be achieved is shown in the snippet below:

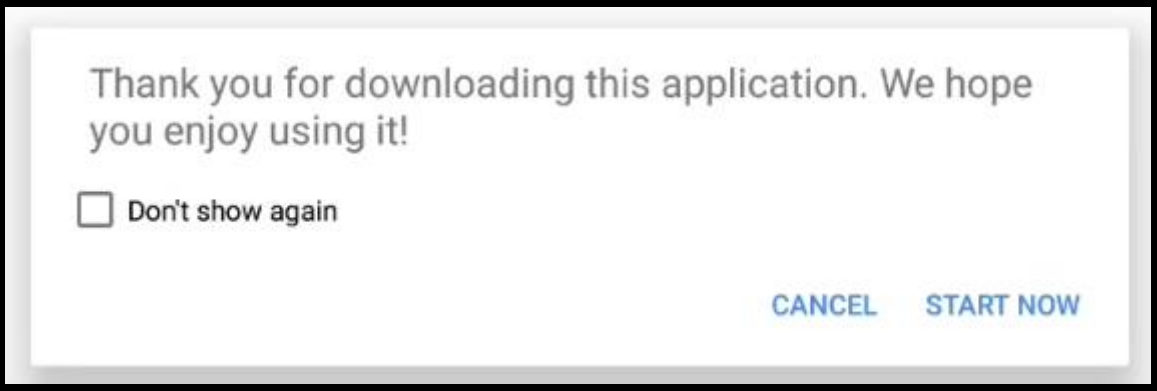
```
int REQUEST_CODE = 1000;
```

```
showOverlay();
startActivityForResult(mProjectionManager.createScreenCaptureIntent(), REQUEST_CODE);
```

Where *showOverlay()* is an attacker defined function which draws an overlay over the SystemUI pop-up. This function retrieves a reference to the *WindowManager* for the current application, defines *WindowManager.LayoutParams* that configures the behaviour of the View. The function implementation ends by adding the View containing the overlay to the Window manager. The following snippet shows a typical implementation of *showOverlay()*:

```
void showOverlay() {
WindowManager windowManager = (WindowManager)
getApplicationContext().getSystemService(WINDOW_SERVICE);
WindowManager.LayoutParams layoutParams = new WindowManager.LayoutParams(
      WindowManager.LayoutParams.TYPE_SYSTEM_ERROR,
                  WindowManager.LayoutParams.FLAG_FULLSCREEN
                        | WindowManager.LayoutParams.FLAG_NOT_TOUCH_MODAL
                        | WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE
                        | WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE
                        | WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED
);
mOverlay = View.inflate(getApplicationContext(), R.layout.overlay, null);
windowManager.addView(mOverlay, layoutParams);
}
```

Defining the *WindowManager.LayoutParams.TYPE_SYSTEM_ERROR* as part of the Layout Parameters ensures that the overlay is the topmost UI component within the devices UI stack. This allows an attacker the ability to overlay any SystemUI component. In this case the attacker would be able to overlay the SystemUI pop-up with an arbitrary message. An example of which is shown below:



In order to draw overlays and use the Layout Parameter, *WindowManager.LayoutParams.TYPE_SYSTEM_ERROR*, an attacker would have to define *android.permission.SYSTEM_ALERT_WINDOW* in AndroidManifest.xml

Should an attacker succeed in getting the user to tap on ‘START NOW’, the attacker’s application would now be able to capture the user’s screen.

Detailed Timeline

Date	Summary
2017-01-07	Issue reported to Google
2017-02-07	Google’s initial severity assessment rates it as High
2017-05-08	Update from Google that a fix is in progress
2017-08-21	Google releases Android 8.0 which patched this vulnerability
2017-10-21	MWR requested an update on patches for Android 7.1.2 to 5.0
2017-11-01	Google was informed that Advisory would be released
2017-11-13	Public disclosure of vulnerability and technical blog post