Checkmarx

CHECKMARX SECURITY RESEARCH

Android Camera App – Gaining Control Without Permissions

Table of Contents

About the Checkmarx Security Research Team

The Checkmarx Security Research Team is committed to providing organizations with actionable insight to support their efforts of building more secure software. Producing a variety of threat research reports, the Checkmarx Team helps all technology users better understand the most prominent software and application issues impacting today's digital world. We all have a responsibility to build software security into everything we deliver, and the Checkmarx Security Research Team is at the forefront of this mission.

Brief Description

It is possible for any application, without specific permissions, to control the Google Camera app developed for Android and force it to take photos and / or record videos, even if the phone is locked and the screen is turned off.

After disclosing these findings to Google, they shared the report with other Android manufacturers, and Samsung confirmed the vulnerabilities existed in their smartphones as well. According to Google, additional OEMs also confirmed the flaws, expanding the impact to hundreds-of-millions of Android users worldwide.

For this report, we've documented our findings stemming from our tests on Pixel 2 and Pixel 3 devices. The vulnerabilities detailed below – CVE-2019-2234 – are relevant to all Google phone models.

Details

Google smartphones have a camera application provided by the com.google.android.GoogleCamera package. Our team decided to analyze the application to search for any vulnerabilities.

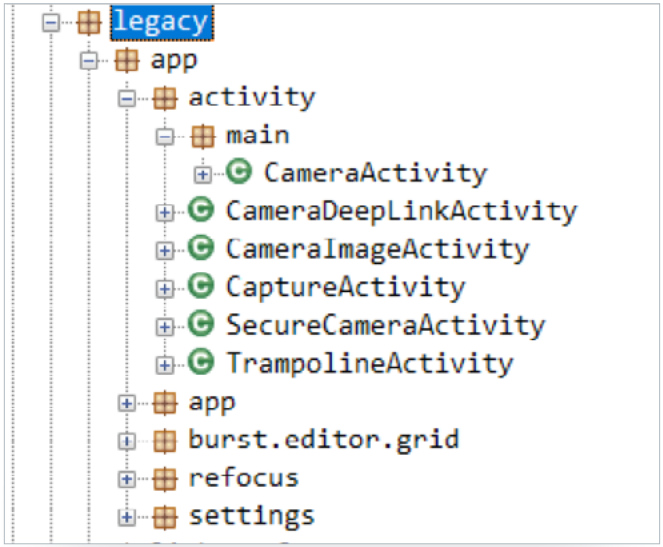
The Google Camera app has the following exported activities, which can be called from any other application with no permissions:

```
com.google.android.apps.camera.legacy.app.activity.main.CameraActivity
com.android.camera.CameraLauncher
com.android.camera.CameraActivity
com.android.camera.activity.CaptureActivity
com.android.camera.VideoCamera
com.android.camera.CameraImageActivity
com.android.camera.CameraVideoShortcutActivity
com.android.camera.CameraDeepLinkActivity
com.android.camera.SecureCameraActivity
com.google.android.apps.camera.legacy.app.settings.CameraSettingsActivity
com.google.android.apps.camera.legacy.app.refocus.ViewerActivity
com.google.android.apps.camera.photobooth.activity.PhotoboothActivity
com.google.android.libraries.social.licenses.LicenseMenuActivity
```

With so many unprotected activities, we mapped each one to the corresponding class and tested them individually to find out if there was any insecure code path that could be leveraged into discovering a vulnerability.

These activities resolve to the following classes:

```
com.google.android.apps.camera.legacy.app.activity.main.CameraActivity
com.google.android.apps.camera.legacy.app.activity.CaptureActivity
com.google.android.apps.camera.legacy.app.activity.CameraImageActivity
com.google.android.apps.camera.legacy.app.activity.CameraDeepLinkActivity
com.google.android.apps.camera.legacy.app.activity.SecureCameraActivity
com.google.android.apps.camera.legacy.app.settings.CameraSettingsActivity
com.google.android.apps.camera.legacy.app.refocus.ViewerActivity
com.google.android.apps.camera.photobooth.activity.PhotoboothActivity
com.google.android.libraries.social.licenses.LicenseMenuActivity
```



In addition, we can easily see that the following actions are mapped:

```
android.media.action.IMAGE_CAPTURE:
com.google.android.GoogleCamera/com.android.camera.activity.CaptureActivity
```

Action: **“android.media.action.IMAGE_CAPTURE”**
Category: **“android.intent.category.DEFAULT”**

```
android.media.action.IMAGE_CAPTURE_SECURE:
com.google.android.GoogleCamera/com.android.camera.SecureCameraActivity
```

Action: **“android.media.action.IMAGE_CAPTURE_SECURE”**
Category: **“android.intent.category.DEFAULT”**

android.intent.action.MAIN:
com.google.android.GoogleCamera/com.android.camera.CameraLauncher

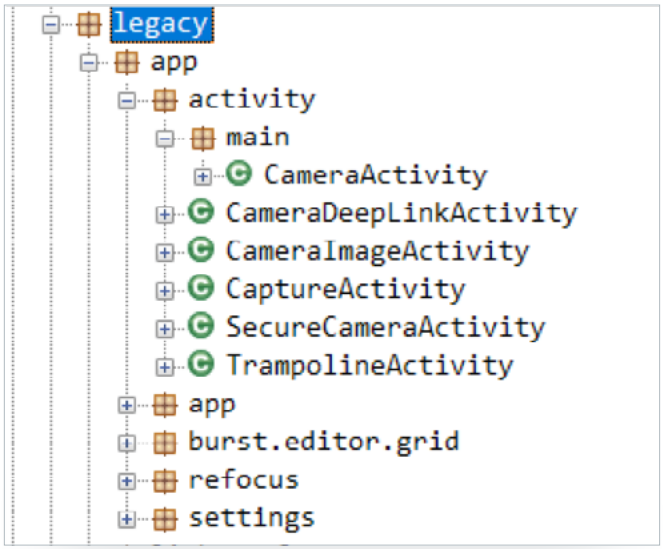
Action: **“android.intent.action.MAIN”**
Category: **“android.intent.category.DEFAULT”**
Category: **“android.intent.category.LAUNCHER”**
com.google.android.GoogleCamera/com.android.camera.CameraActivity
Action: **“android.intent.action.MAIN”**
Category: **“android.intent.category.DEFAULT”**

android.media.action.STILL_IMAGE_CAMERA:
com.google.android.GoogleCamera/com.android.camera.CameraImageActivity

Action: **“android.media.action.STILL_IMAGE_CAMERA”**
Category: **“android.intent.category.DEFAULT”**
Category: **“android.intent.category.VOICE”**

android.media.action.STILL_IMAGE_CAMERA_SECURE:
com.google.android.GoogleCamera/com.android.camera.SecureCameraActivity

Action: **“android.media.action.STILL_IMAGE_CAMERA_SECURE”**
Category: **“android.intent.category.DEFAULT”**
Category: **“android.intent.category.VOICE”**



```
android.media.action.VIDEO_CAPTURE:
com.google.android.GoogleCamera/com.android.camera.VideoCamera
```

Action: **"android.media.action.VIDEO_CAPTURE"**
Category: **"android.intent.category.DEFAULT"**

```
android.media.action.VIDEO_CAMERA:
com.google.android.GoogleCamera/com.android.camera.CameraVideoShortcutActivity
```

Action: **"android.media.action.VIDEO_CAMERA"**
Category: **"android.intent.category.DEFAULT"**
Category: **"android.intent.category.VOICE"**

Testing the different actions, one can note the following:

a. Invoking the android.media.action.VIDEO_CAMERA action starts the Google Camera and it immediately starts to record a video

b. Invoking the android.media.action.VIDEO_CAPTURE action does NOT make the Google Camera start to record a video

c. Any other action opens the Google Camera app in Photo mode, but does NOT take a photo.

After gathering and testing the actions that could be invoked, we searched the code for extras that could be used to influence the application behavior.

By dissecting the many classes, there are some interesting extras that should be noted:

android.intent.extra.USE_FRONT_CAMERA - allows the user to select the front camera (or back camera if absent)

android.intent.extra.TIMER_DURATION_SECONDS - allows the camera to have a timer before taking a photo (3 seconds minimum, hardcoded)

extra_turn_screen_on - obscure flag, likely used with wearables that turns on the screen of the device

Vulnerabilities Discovered

By manipulating the specific actions and intents, an attacker can now control the Google Camera app to take photos and/or record videos through a rogue application that has no permissions to do so. Additionally, we found that certain attack scenarios enable malicious actors to circumvent various storage permission policies, giving them access to stored videos and photos, as well as GPS metadata embedded in photos to locate the user by taking a photo or video and parsing the proper EXIF data.

Explanation & Why is this an Issue?

It is known that Android camera applications usually store their photos and videos on the SD card. Since photos and videos are sensitive user information, for an application to access them, it needs special permissions: storage permissions.

Unfortunately, storage permissions are very broad and they provide access to the entire SD card. In addition, there are a large number of applications with legitimate use-cases to access storage that have no special interest in photos and videos.

The ability for an application to get input from the camera, microphone, and GPS location is also considered highly invasive, and AOSP even created a specific permission group for these permissions.

By leveraging the issue described above, an application that has access to storage not only has access to past photos and videos (which already had, by permission design, nothing new there), but also has a way to access newly taken photos and videos by abusing the Google Camera app exported components.

This means that a rogue application can take photos and/or videos without specific camera permissions, and it only needs storage permissions to take things a step further and fetch photos and videos after being taken. Additionally, if the location is enabled in the camera app, the rogue application also has a way to access the current GPS position of the phone and user.

This is not a desired behavior, since the Google Camera app should not be allowed to be fully controlled by an external app, circumventing the camera/mic/GPS permissions that the user is trusting the Android OS to enforce.

In fact, this 'feature' defeats the purpose of requesting/granting apps the **`android.permission.CAMERA`**, **`android.permission.RECORD_AUDIO`** and even **`android.permission.ACCESS_FINE_LOCATION`**, **`android.permission.ACCESS_COARSE_LOCATION`** **`permissions.`**

Proof of Concept (PoC)

Verification Using adb

The easiest way to verify this issue is via adb.

To force a video to be taken, issue the following command:

```
$ adb shell am start-activity -n
com.google.android.GoogleCamera/com.android.camera.CameraActivity --ez
extra_turn_screen_on true -a android.media.action.VIDEO_CAMERA --ez
android.intent.extra.USE_FRONT_CAMERA true
```

To force a photo to be taken, issue this command instead:

```
$ adb shell am start-activity -n
com.google.android.GoogleCamera/com.android.camera.CameraActivity --ez
extra_turn_screen_on true -a android.media.action.STILL_IMAGE_CAMERA --ez
android.intent.extra.USE_FRONT_CAMERA true --ei
android.intent.extra.TIMER_DURATION_SECONDS 3
```

Note that both commands can be issued without any problems even if the phone is in a locked state.

Verification with PoC App

We developed a mock weather application as a PoC that will make the same calls as shown above. The mock weather app was shared with Google and Samsung. In addition, we developed an advanced PoC that uses several stealth features to prevent the user from seeing that photos and videos were being taken.

We've recorded a short video of the PoC which can be seen here. This video demonstrates how we used the mock weather app to bypass permissions to take photos and videos that includes geolocation information. We also demonstrate how we can retrieve photos and videos to the C&C server, gathering the GPS history from images in the SD card. Finally, we were able to auto-record both sides of phone calls and more, in a discreet way without the user knowing it.

Fingerprint

```
$ adb shell getprop ro.build.fingerprint
google/blueline/blueline:9/PQ3A.190605.003/5524043:user/release-keys
$ adb shell dumpsys package com.google.android.GoogleCamera|grep -E “Package|version|time”
Packages:
       Package [com.google.android.GoogleCamera] (ce5cc83):
       pkg=Package{27a2e00 com.google.android.GoogleCamera}
       versionCode=49456750 minSdk=28 targetSdk=28
       versionName=6.2.030.244457635
       timeStamp=2019-05-09 16:02:33
       installerPackageName=com.android.vending
      signatures=PackageSignatures{c09c639 version:3, signatures:[e3ca78d8], past signatures:[]}
       Package [com.google.android.GoogleCamera] (3a4f47e):
       pkg=Package{72eaddf com.google.android.GoogleCamera}
       versionCode=44450387 minSdk=28 targetSdk=28
       versionName=6.0.012.208106668
       timeStamp=2009-01-01 08:00:00
       signatures=PackageSignatures{fb78cf5 version:0, signatures:[], past signatures:[]}
```

Google's Response

"We appreciate Checkmarx bringing this to our attention and working with Google and Android partners to coordinate disclosure. The issue was addressed on impacted Google devices via a Play Store update to the Google Camera Application in July 2019. A patch has also been made available to all partners."

Mitigation Recommendation

For proper mitigation and as a general best practice, ensure you update all applications on your device.

Timeline of Disclosure

- Jul 4, 2019 – Submitted a vulnerability report to Android's Security team at Google
- Jul 4, 2019 – Google confirmed receiving the report
- Jul 4, 2019 – A PoC "malicious app" was sent to Google
- Jul 5, 2019 – A PoC video of an attack scenario was sent to Google
- Jul 13, 2019 – Google set the severity of the finding as "Moderate"
- Jul 18, 2019 – Sent further feedback to Google
- Jul 23, 2019 – Google raised the severity of the finding to "High"
- Aug 1, 2019 – Google confirms our suspicion that the vulnerabilities may affect other Android smartphone vendors and issues CVE-2019-2234
- Aug 18, 2019 – Multiple vendors were contacted regarding the vulnerabilities
- Aug 29, 2019 – Samsung confirmed they are affected
- Nov 2019 – Both Google and Samsung approved the publication

Note: This publication was coordinated with Google and Samsung after their confirmation of a fix being released. Please refer to Google for information regarding the fixed version of the Android OS and Google Camera app.

Final Words

The professionalism shown by both Google and Samsung does not go unnoticed. Both were a pleasure to work with due to their responsiveness, thoroughness, and timeliness.

This type of research activity is part of the Checkmarx Security Research Team's ongoing efforts to drive the necessary changes in software security practices among vendors that manufacture consumer-based smartphones and IoT devices, while bringing more security awareness amid the consumers who purchase and use them. Protecting privacy of consumers must be a priority for all of us in today's increasingly connected world.

About the Checkmarx Security Research Team

The Checkmarx Security Research Team is committed to providing organizations with actionable insight to support their efforts of building more secure software. Producing a variety of threat research reports, the Checkmarx Team helps all technology users better understand the most prominent software and application issues impacting today's digital world. We all have a responsibility to build software security into everything we deliver, and the Checkmarx Security Research Team is at the forefront of this mission.

Software = Security

About Checkmarx

Checkmarx is the global leader in software security solutions for modern enterprise software development. Checkmarx delivers the industry's most comprehensive Software Security Platform that unifies with DevOps and provides static and interactive application security testing, software composition analysis, and developer AppSec awareness and training programs to reduce and remediate risk from software vulnerabilities. Checkmarx is trusted by more than 40 percent of the Fortune 100 and half of the Fortune 50. Learn more at www.checkmarx.com.