Mobile Security - Tutorial 2

Android M Permission Model

Brian Ricks

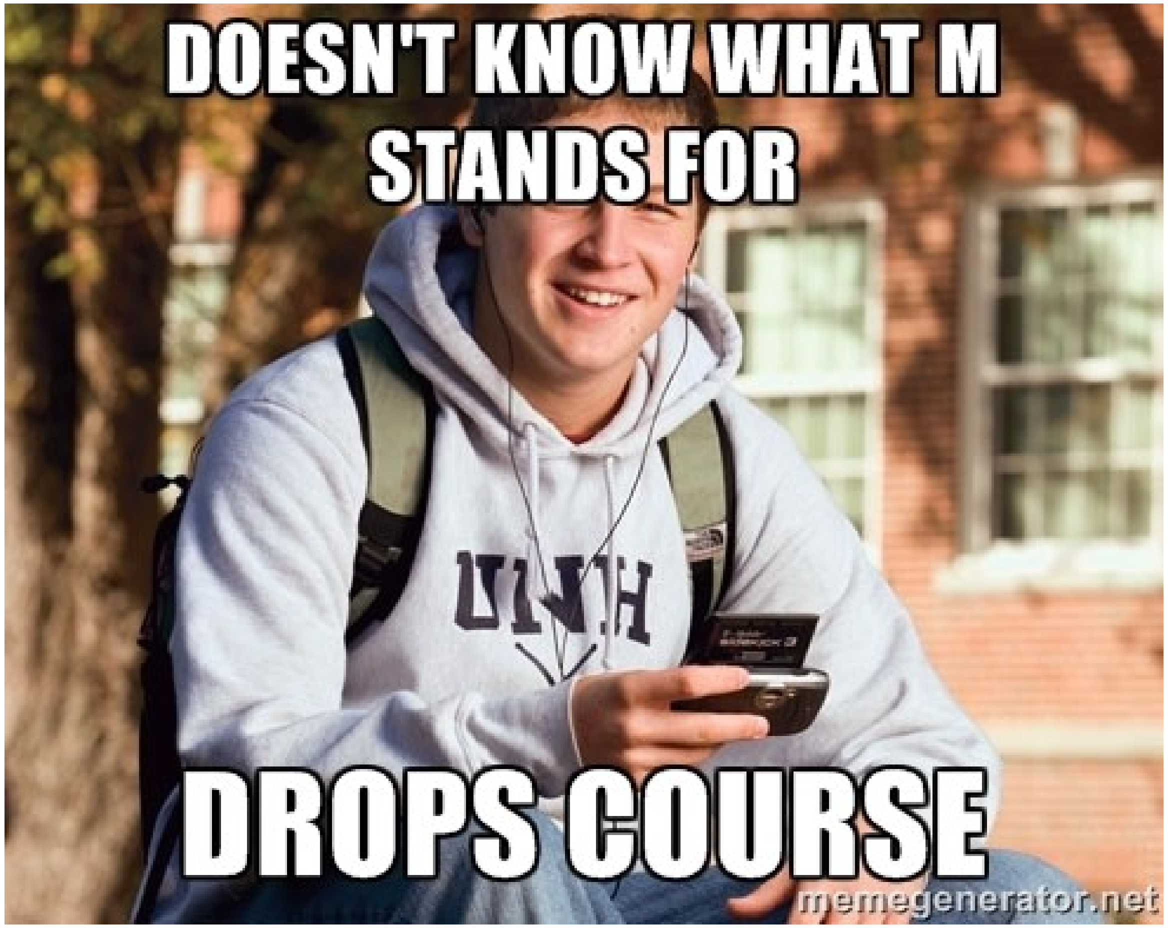
Fall 2015

Carnegie Mellon University

What are we doing?

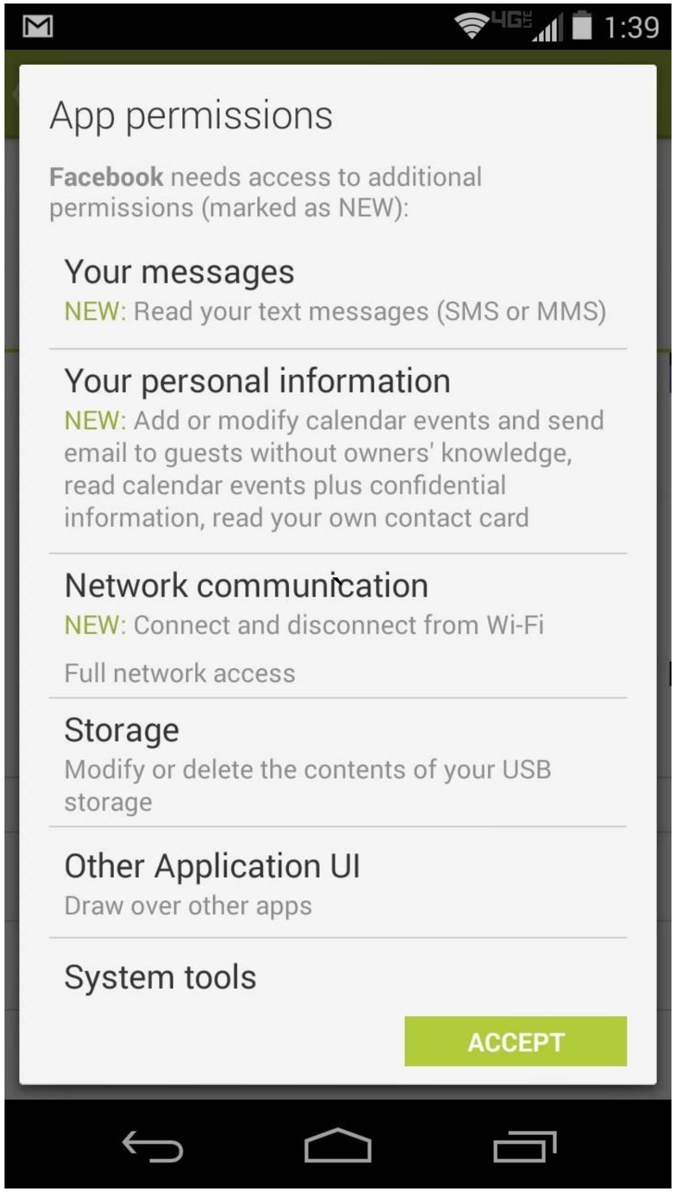
- Learn all about the Android M permission model!

Let's Begin!


DOESN'T KNOW WHAT M STANDS FOR
DROPS COURSE
memegenerator.net

A Little History

- Prior to M, Android used an *install time* permission model
  - All permissions had to be accepted by the user before the app could be installed



App permissions

Facebook needs access to additional permissions (marked as NEW):

Your messages
NEW: Read your text messages (SMS or MMS)

Your personal information
NEW: Add or modify calendar events and send email to guests without owners' knowledge, read calendar events plus confidential information, read your own contact card

Network communication
NEW: Connect and disconnect from Wi-Fi

Full network access

Storage
Modify or delete the contents of your USB storage

Other Application UI
Draw over other apps

System tools

A Little History

- This annoyed companies whose apps requested every permission under the sun

A Little History

- This also made it easy for app developers to include permissions the app did not need
  - Often times done by accident

The Solution?

- Runtime based permission model!
  - User is not prompted to accept any permissions at install time
  - User is prompted to accept individual permission *groups*, during runtime, *when they are needed*

Permission Groups

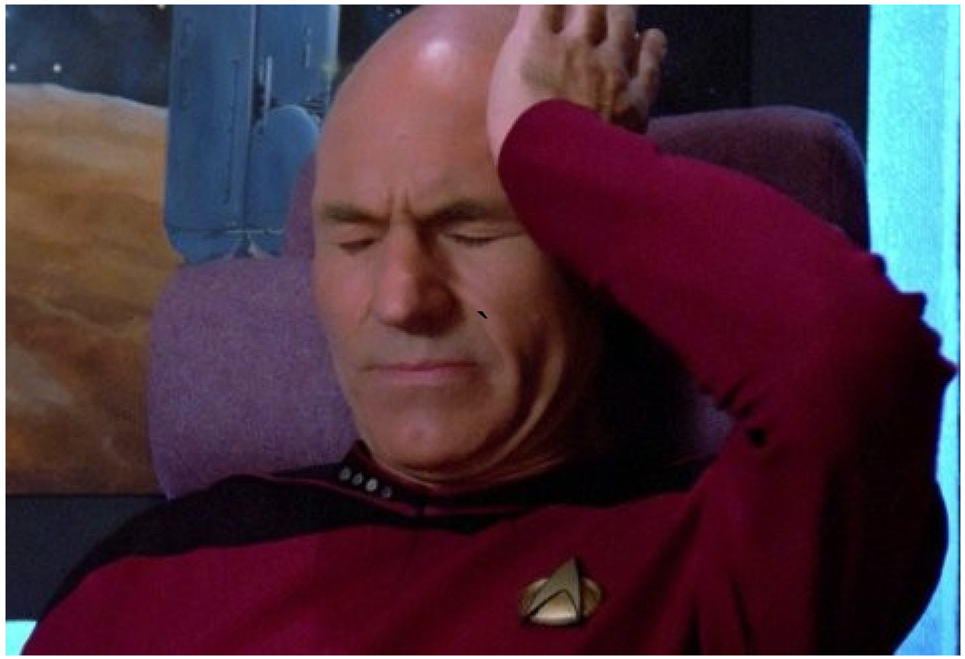
- Simply a group of related permissions
  - For example, if your app requests READ_CONTACTS, the user is prompted to accept/deny the contact list permission group (which includes WRITE_CONTACTS)
  - If your app then requests WRITE_CONTACTS sometime later, the permission is automatically granted (if the user granted the prior request)

Normal and Dangerous Permissions

- Normal Permissions: permissions declared in the app manifest which are automatically granted
- Dangerous Permissions: permissions declared in the app manifest which the user must accept
  - Prior to M: install time
  - M: runtime

Dangerous Permissions

- Wait, so you mean I need to declare my dangerous permissions in the manifest even though I request them at runtime?
  - Yes



Dangerous Permissions

- Why?
  - Backwards compatibility
    - Prior to M, all permissions were pulled from the Manifest for user granting at install time

Normal Permissions

- If normal permissions are always granted, why do we need to declare them?
  - These permissions still define actions which your app needs to take outside its sandbox

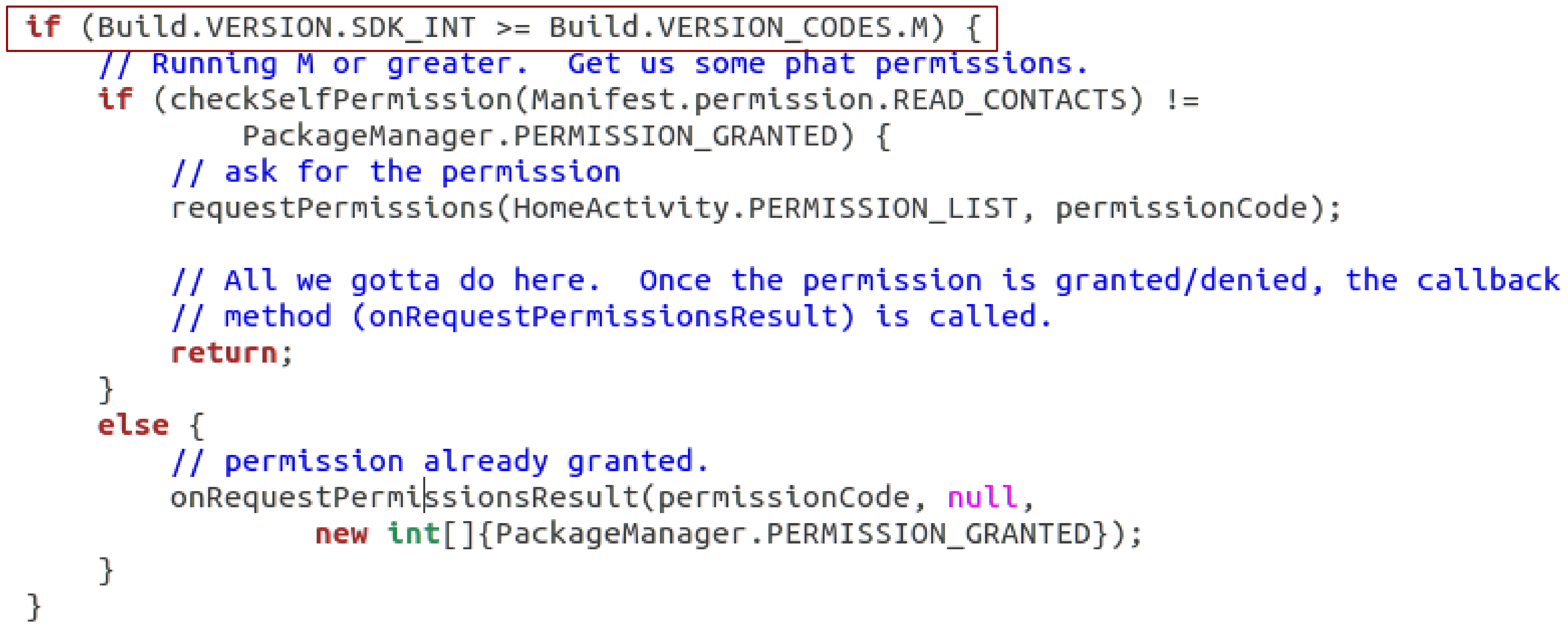
Normal Permissions

- In M, some dangerous permissions are now considered normal
  - INTERNET is the big one
    - All apps basically request this one

Implementing Runtime Permissions

- Let's use SuperAwesomeContacts as an example!!

Step 1



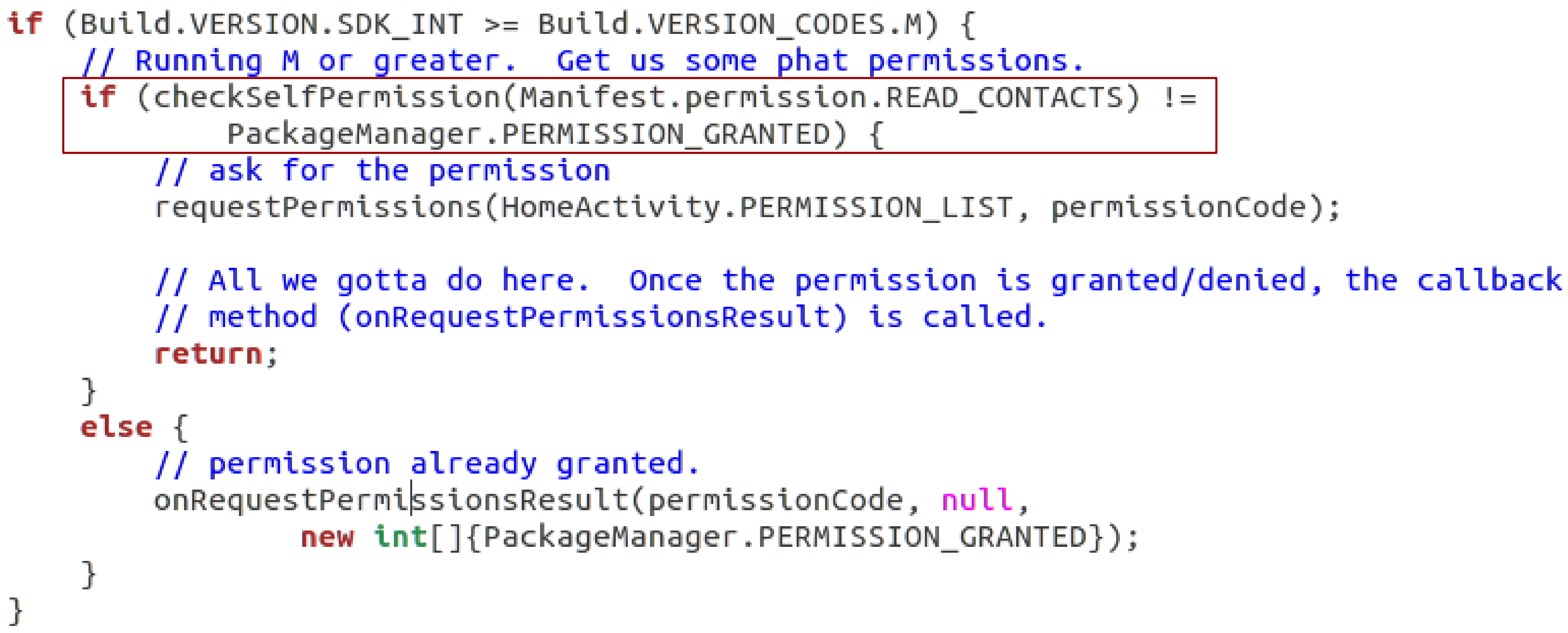
```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    // Running M or greater.  Get us some phat permissions.
    if (checkSelfPermission(Manifest.permission.READ_CONTACTS) !=
            PackageManager.PERMISSION_GRANTED) {
        // ask for the permission
        requestPermissions(HomeActivity.PERMISSION_LIST, permissionCode);

        // All we gotta do here.  Once the permission is granted/denied, the callback
        // method (onRequestPermissionsResult) is called.
        return;
    }
    else {
        // permission already granted.
        onRequestPermissionsResult(permissionCode, null,
                new int[]{PackageManager.PERMISSION_GRANTED});
    }
}
```

- Check to see what Android version the app is running on

Step 1

- We need to check, because unless our app's minimum API level is M (doubtful), we can't just assume that we will be dealing with the M permission model
  - CheckSelfPermission() is now included in ContextCompat, which provides a way to avoid the API check here.

Step 2



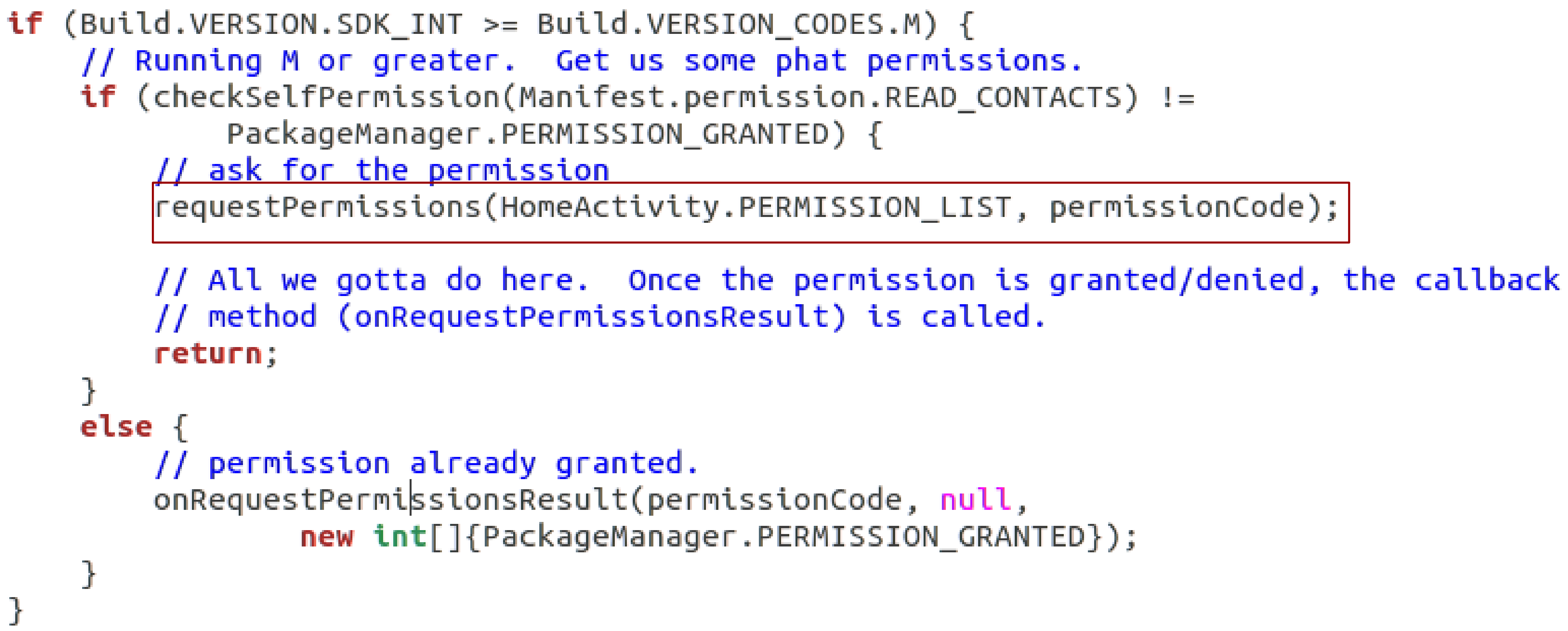
```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    // Running M or greater.  Get us some phat permissions.
    if (checkSelfPermission(Manifest.permission.READ_CONTACTS) !=
            PackageManager.PERMISSION_GRANTED) {
        // ask for the permission
        requestPermissions(HomeActivity.PERMISSION_LIST, permissionCode);

        // All we gotta do here.  Once the permission is granted/denied, the callback
        // method (onRequestPermissionsResult) is called.
        return;
    }
    else {
        // permission already granted.
        onRequestPermissionsResult(permissionCode, null,
                new int[]{PackageManager.PERMISSION_GRANTED});
    }
}
```

- Check to see if we were granted this permission already

Step 2

- If we have already been granted this permission earlier (say because this code was invoked earlier), then we can skip the permission request.
- Going back to Step 1, if you invoke ContextCompat.checkSelfPermission() instead, then on APIs < M, this method will still work as it's part of the support library.
  - Note, if you use this method, you must use the support library methods in subsequent steps!

Step 3



```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    // Running M or greater.  Get us some phat permissions.
    if (checkSelfPermission(Manifest.permission.READ_CONTACTS) !=
            PackageManager.PERMISSION_GRANTED) {
        // ask for the permission
        requestPermissions(HomeActivity.PERMISSION_LIST, permissionCode);

        // All we gotta do here.  Once the permission is granted/denied, the callback
        // method (onRequestPermissionsResult) is called.
        return;
    }
    else {
        // permission already granted.
        onRequestPermissionsResult(permissionCode, null,
                new int[]{PackageManager.PERMISSION_GRANTED});
    }
}
```

- If we have not been granted permission, then request it

Step 3

- If we have not been granted the permission, then request it.
  - If you are using the support library method to avoid Step 1, you must use ActivityCompat.requestPermissions() instead for this step.
- Once you request the permission, there is nothing else to do

Step 3

- Yep, this is an asynchronous process!
  - Once you request a permission, the requestPermissions() method returns.
    - Does not block!
  - The callback method (onRequestPermissionsResult()) is called once the user decides to accept or reject the permission
  - Therefore, *anything* can happen to your activity in the meantime!

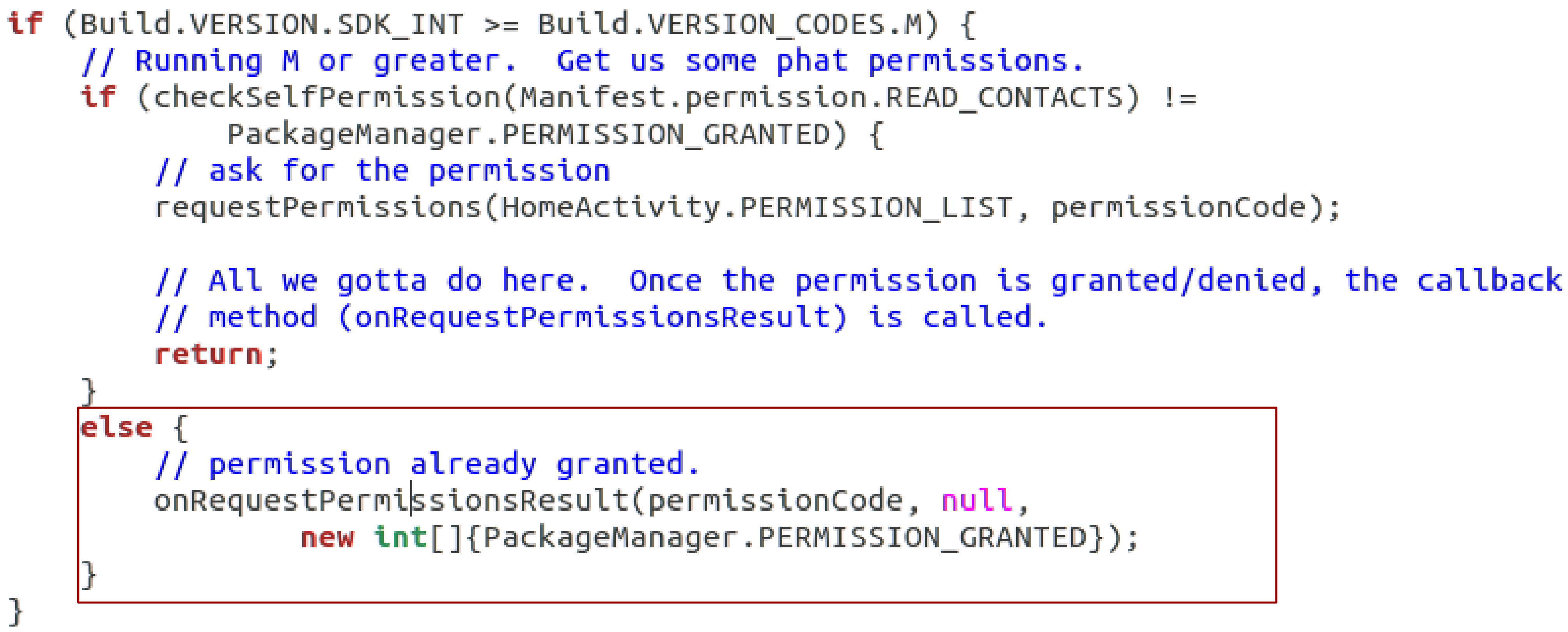
Step 3

- HomeActivity.PERMISSION_LIST = array of permissions we request here (just one – READ_CONTACTS)
  - While the input is an array, don't request permissions until you actually need them

Step 3

- permissionCode (request code) = unique developer-defined code to track a permission *request* through the permission request lifecycle
  - So when onRequestPermissionsResult() is called, you'll know which permission *request* you are dealing with
  - *Request* is in italics, why?
    - To emphasize that request != specific permission

Step 4



```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    // Running M or greater.  Get us some phat permissions.
    if (checkSelfPermission(Manifest.permission.READ_CONTACTS) !=
            PackageManager.PERMISSION_GRANTED) {
        // ask for the permission
        requestPermissions(HomeActivity.PERMISSION_LIST, permissionCode);

        // All we gotta do here.  Once the permission is granted/denied, the callback
        // method (onRequestPermissionsResult) is called.
        return;
    }
    else {
        // permission already granted.
        onRequestPermissionsResult(permissionCode, null,
                new int[]{PackageManager.PERMISSION_GRANTED});
    }
}
```

- Handle the condition that the permission has already been granted

Step 4

- It's very possible that our permission was already granted
  - If we need a permission when our activity is created, remember that our activity can be recreated at any time, like when we reorient our phone screens
  - It would be annoying if the user had to accept permissions each time

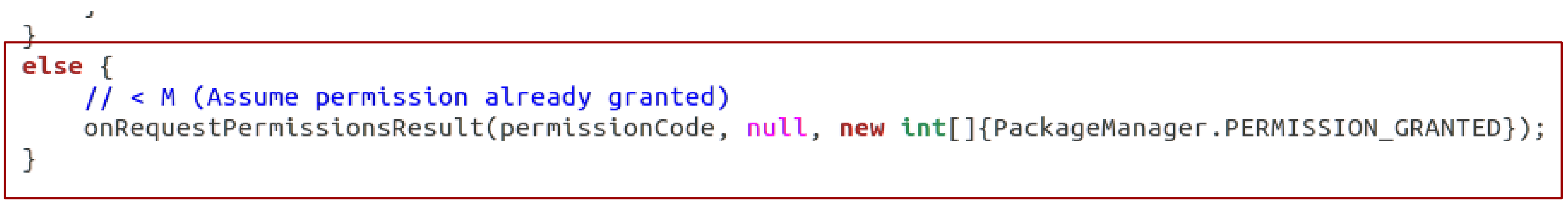
Step 4

- It's very possible that our permission was already granted
  - Therefore, once a user accepts, typically they won't be asked again to accept the same permission (group)
  - Thus, don't request the same permission redundantly! (hence Step 2)

Step 4

- Note that in SuperAwesomeContacts, if the permission was already granted, we simply call onRequestPermissionsResult() directly
  - This is usually not the best way to do this, but I was lazy and put a lot of code in onRequestPermissionsResult() which should have been in separate methods

Step 5



```
}
else {
    // < M (Assume permission already granted)
    onRequestPermissionsResult(permissionCode, null, new int[]{PackageManager.PERMISSION_GRANTED});
}
```

- Handle the condition that the API level is < M

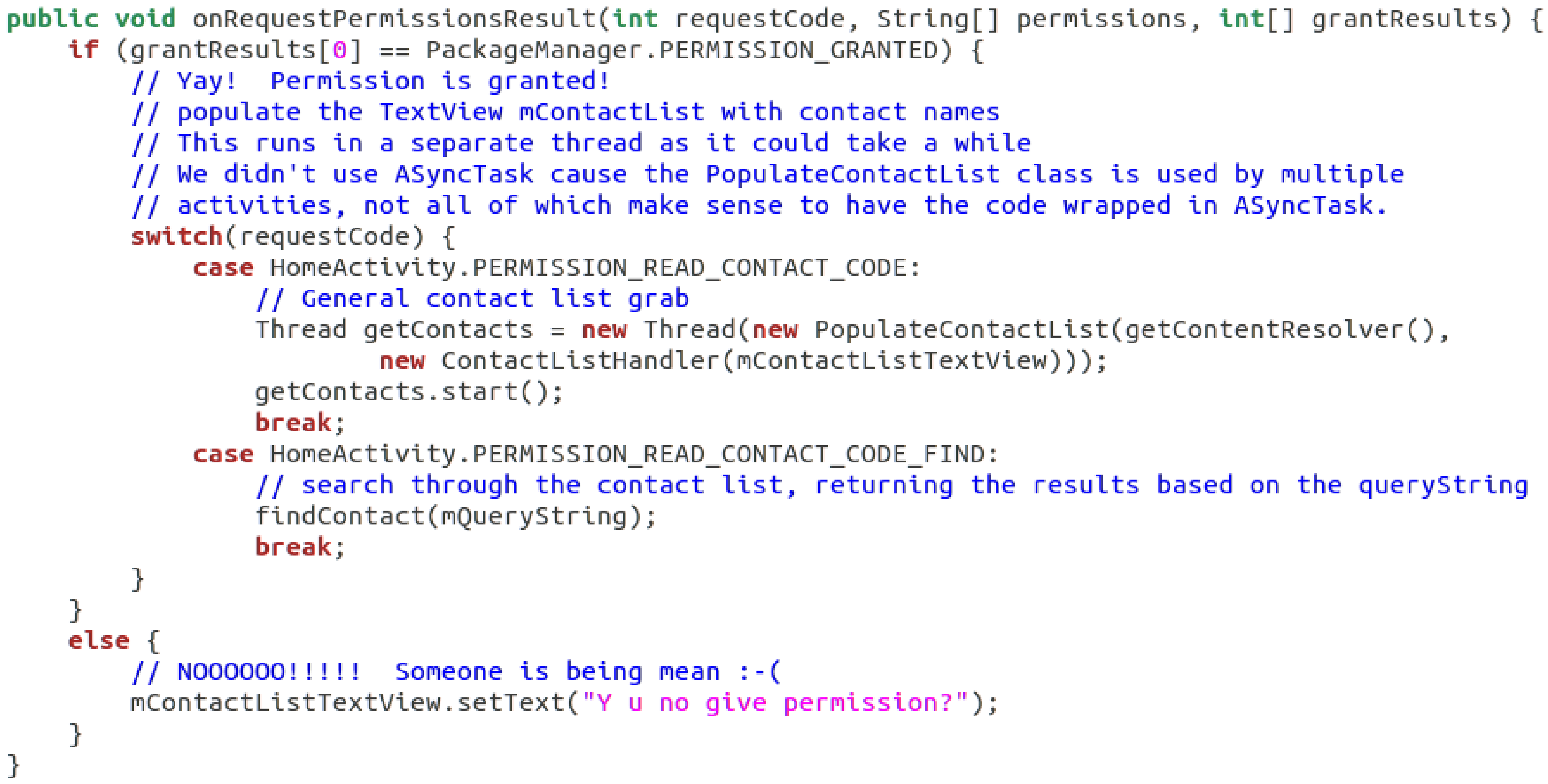
Step 5

- Providing you are not using the support library methods, if API version is < M, then handle as if you the permission was granted (should be, since it's install time < M)
  - Again, in SuperAwesomeContacts I call onRequestPermissionsResult() directly. Not best practice to do this.

Step 5

- What is the best practice then?
  - Instead of having code related to logic which requires the permission in your onRequestPermissionsResult() method, put it all in a separate method and call that method when needed
    - That way you can call that method directly if a permission was previously granted, and also call it from onRequestPermissionsResult() if the result was permission granted

Step 6



```java
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        // Yay!  Permission is granted!
        // populate the TextView mContactList with contact names
        // This runs in a separate thread as it could take a while
        // We didn't use ASyncTask cause the PopulateContactList class is used by multiple
        // activities, not all of which make sense to have the code wrapped in ASyncTask.
        switch(requestCode) {
            case HomeActivity.PERMISSION_READ_CONTACT_CODE:
                // General contact list grab
                Thread getContacts = new Thread(new PopulateContactList(getContentResolver(),
                        new ContactListHandler(mContactListTextView)));
                getContacts.start();
                break;
            case HomeActivity.PERMISSION_READ_CONTACT_CODE_FIND:
                // search through the contact list, returning the results based on the queryString
                findContact(mQueryString);
                break;
        }
    }
    else {
        // NOOOOOO!!!!!  Someone is being mean :-(
        mContactListTextView.setText("Y u no give permission?");
    }
}
```

- Handle permission request results

Step 6

- Handle the results of the permission request
  - Basic steps:
    - Check if permission was granted
      - Yes? Great! Check the request code so you know which permission request was granted
        - Do stuff with your new privilege
      - No? Well that sucks. Check the request code
        - Disable specific functionality related to this permission denial

Step 6

- Hey! SuperAwesomeContacts only requests one permission, but I see two request code cases!
  - Remember, the request code is for a specific request type. In SuperAwesomeContacts, we need the same permission (READ_CONTACTS) for both populating the contact textview and for searching contacts.

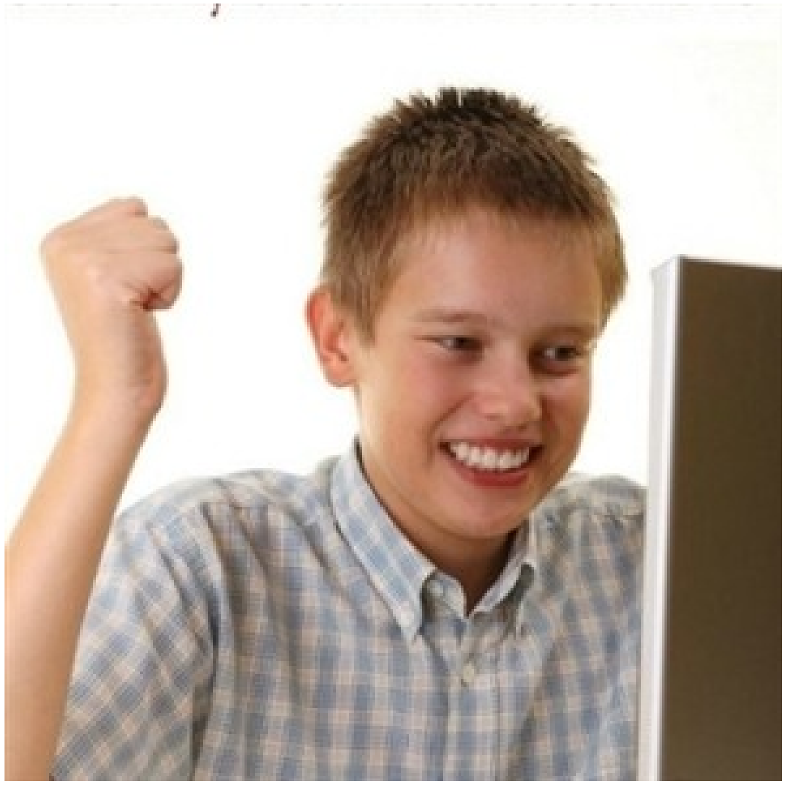
Stuff not in the example

- What if I really really want a user to accept my permission?
  - ShouldShowRequestPermissionRationale() returns true if the permission was denied in the past and the “Don't ask again” checkbox was *unchecked*
    - At this point you could display a dialog to the user explaining why you really really want that user to reconsider.
- If using the support library methods use ActivityCompat.shouldShowRequestPermissionRationale()

Some Additional Points

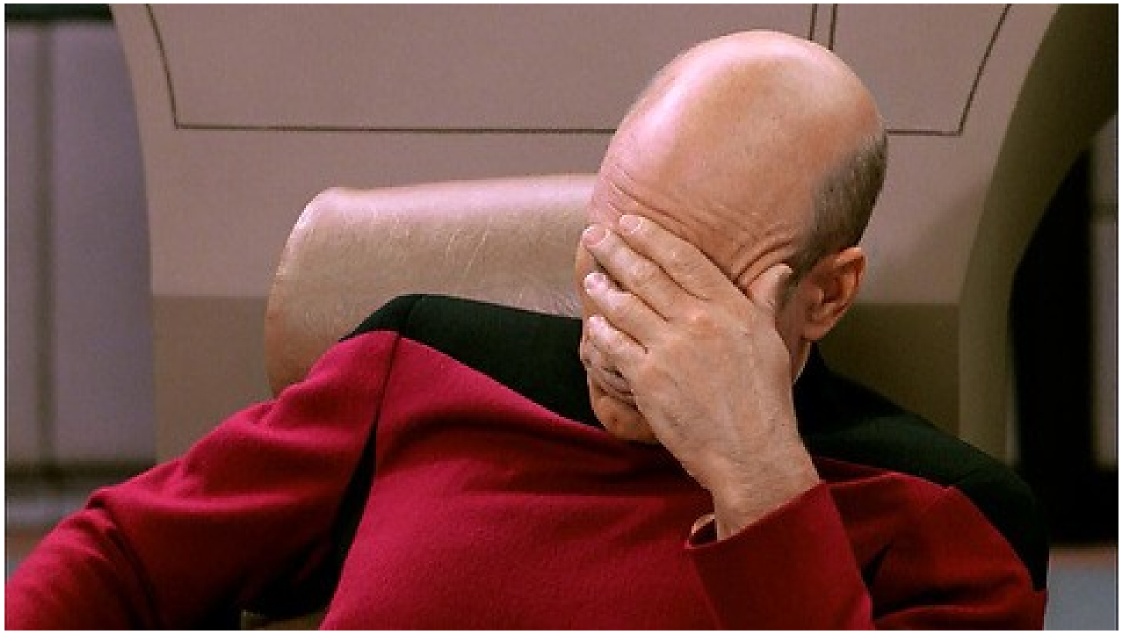
Fail #1

- I'll just bypass the security entirely by calling onRequestPermissionsResult() directly with results set to permission granted!



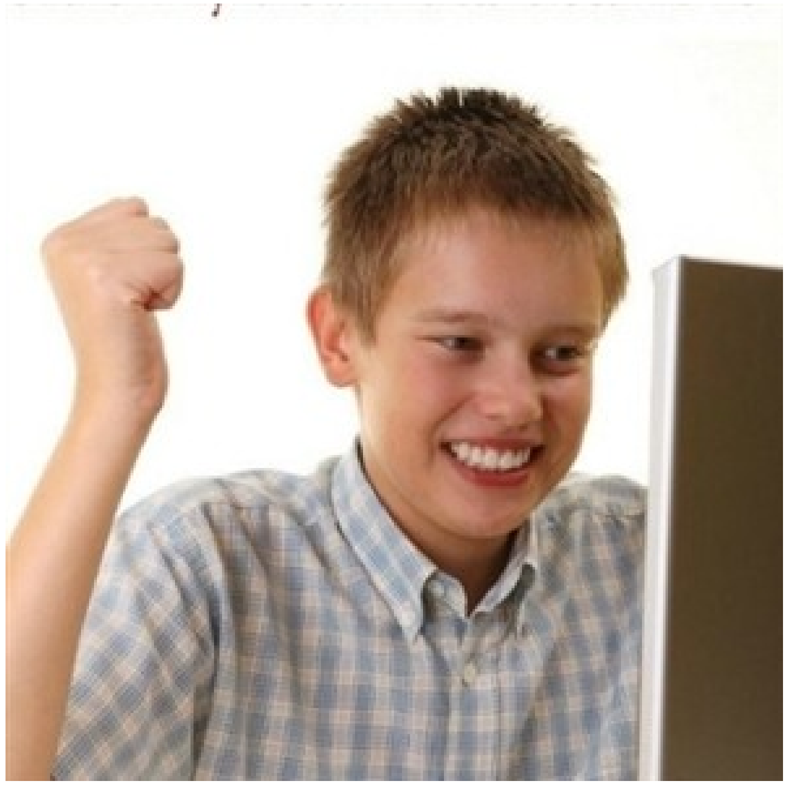
Fail #1

- Fail. That doesn't grant anything because no permissions were actually requested. Enjoy the SecurityExceptions :-D



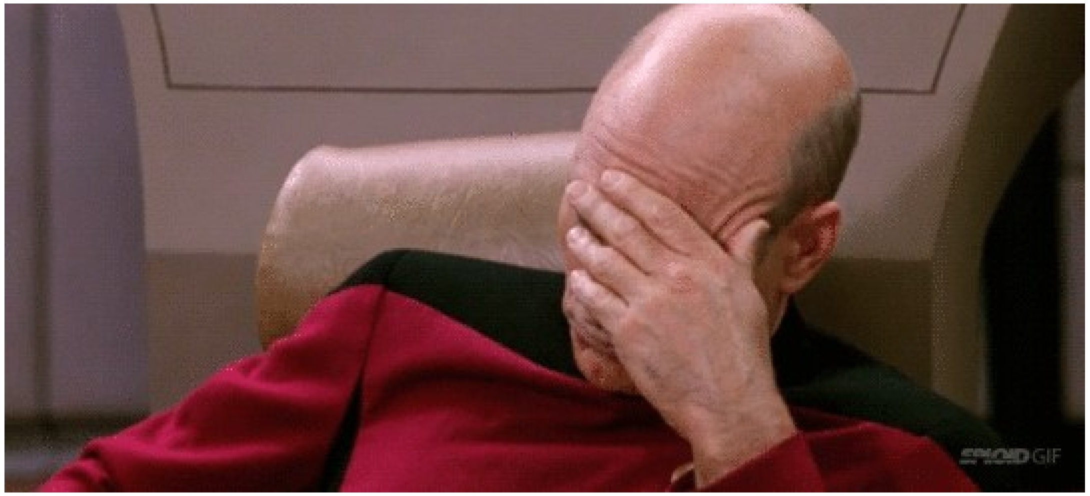
Fail #2

- I'll just request all my permissions when my app first launches!!



Fail #2

- Fail. Doing that will cause people not to like your app, and you want people to like your app... right???



The End

Disclaimer: Meme is in no way presented here to disrespect Dennis Ritchie or his legacy, only your mother.