ANDROID

HTTP, Permissions, and intents

COS216
AVINASH SINGH
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF PRETORIA

HTTP

- Older Android SDKs did not have native support for HTTP (AJAX-like) requests
  - Request could only be made through Apache on Android
- Before the new SDK came out, Android programmers had to use third-party libraries
  - Like OkHttp or Volley
- The new Android SDK now has full support for HTTP requests

HTTP

- Do not use the Apache client in Android: DefaultHttpClient
  - Cumbersome to use
  - Deprecated a few versions ago
- Rather use the new HttpURLConnection class

HTTPURLCONNECTION

- The HttpURLConnection class
  - (Relatively) easy to use and maintained by Google
  - Supports cookies
  - Supports HTTP authentication
  - Supports redirection through proxies
  - Supports IPv6
  - Supports optional request caching (just like browsers cache requests/files)

https://developer.android.com/reference/java/net/HttpURLConnection

HTTPURLCONNECTION

- Requires permission to access the internet
- Recall that permission are added to the AndroidManifest
- To access the internet

```
<uses-permission android:name="android.permission.INTERNET" />
```

- Additionally, if you want to check the network connection (e.g. connected/disconnected)

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

HTTPURLCONNECTION

- A simple GET request

```
URL url = new URL("http://api.satoshi.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
try
{
   InputStream in = new BufferedInputStream(connection.getInputStream());
   readStream(in);
}
finally
{
   connection.disconnect();
}
```

HTTPURLCONNECTION

- A POST request

```
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
try
{
   connection.setDoOutput(true);
   connection.setChunkedStreamingMode(0);
   OutputStream out = new BufferedOutputStream(connection.getOutputStream());
   writeStream(out);
   InputStream in = new BufferedInputStream(connection.getInputStream());
   readStream(in);
}
finally
{
   connection.disconnect();
}
```

HTTPURLCONNECTION

- Set additional HTTP headers

```
connection.setRequestProperty("Content-Type", "application/json");
```

HTTPURLCONNECTION

- Login via HTTP

```
Authenticator.setDefault(new Authenticator()
{
    protected PasswordAuthentication getPasswordAuthentication()
    {
        return new PasswordAuthentication(username, password.toCharArray());
    }
});
```

HTTPURLCONNECTION

- Use cookies if your really have to

```
CookieManager cookieManager = new CookieManager();
CookieHandler.setDefault(cookieManager);
HttpCookie cookie = new HttpCookie("ApiKey", "KeyMcKeyFace");
cookie.setDomain("satoshi.com");
cookie.setPath("/");
cookie.setVersion(0);
cookieManager.getCookieStore().add(new URI("http://statoshi.com"), cookie);
```

OKHTTP

- Full-fledged HTTP third-party library for Java
- This library is most commonly used on Android
- Clean and easy to use interface
- To use the library
  - Either download the precompiled JAR file and add it as a dependency through Maven or Gradle
  - Or download the source code and manually compile it with your app

http://square.github.io/okhttp

OKHTTP

- Simple GET request in OkHttp

```
OkHttpClient client = new OkHttpClient();

String run(String url) throws IOException
{
        Request request = new Request.Builder().url(url).build();
        Response response = client.newCall(request).execute();
        return response.body().string();
}
```

VOLLEY

- An alternative to OkHttp and HttpUrlConnection is Volley
- Started as a standalone project
  - Now supported by and integrated into Android
- Easier to use than HttpUrlConnection
  - Also has integrated JSON parsing

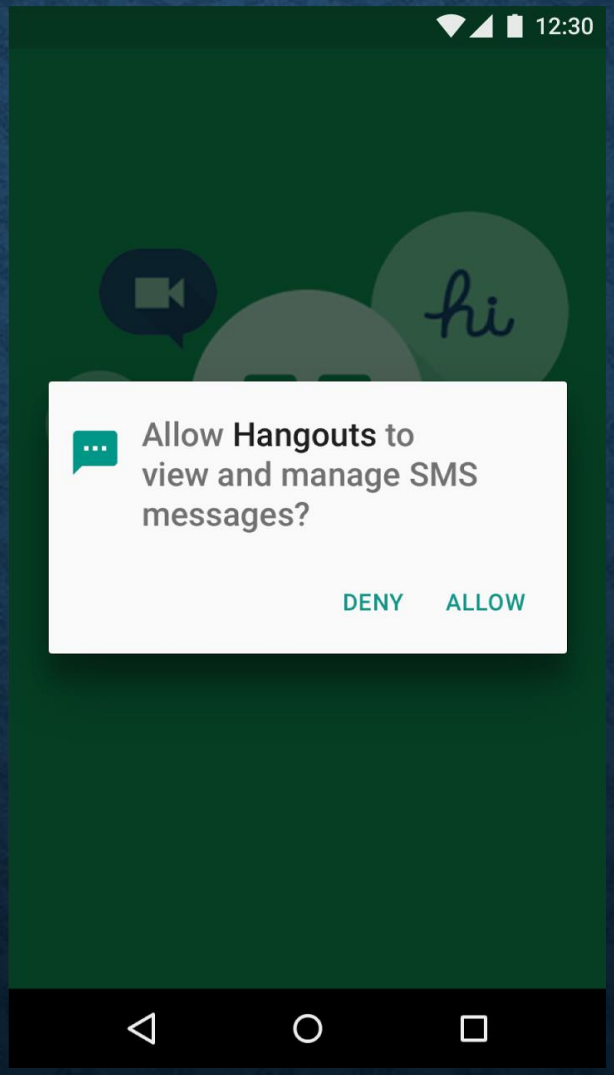
https://developer.android.com/training/volley/

https://github.com/google/volley

PERMISSIONS

- Android apps need permissions for actions that can be used for malicious purposes
  - Access the internet
  - Accessing GPS
  - Access photos in the gallery
  - Reading and writing files on external storage
  - Searching contacts
  - Installing external apps
  - Many more …
- These permissions are added to the AndroidManifest

PERMISSIONS



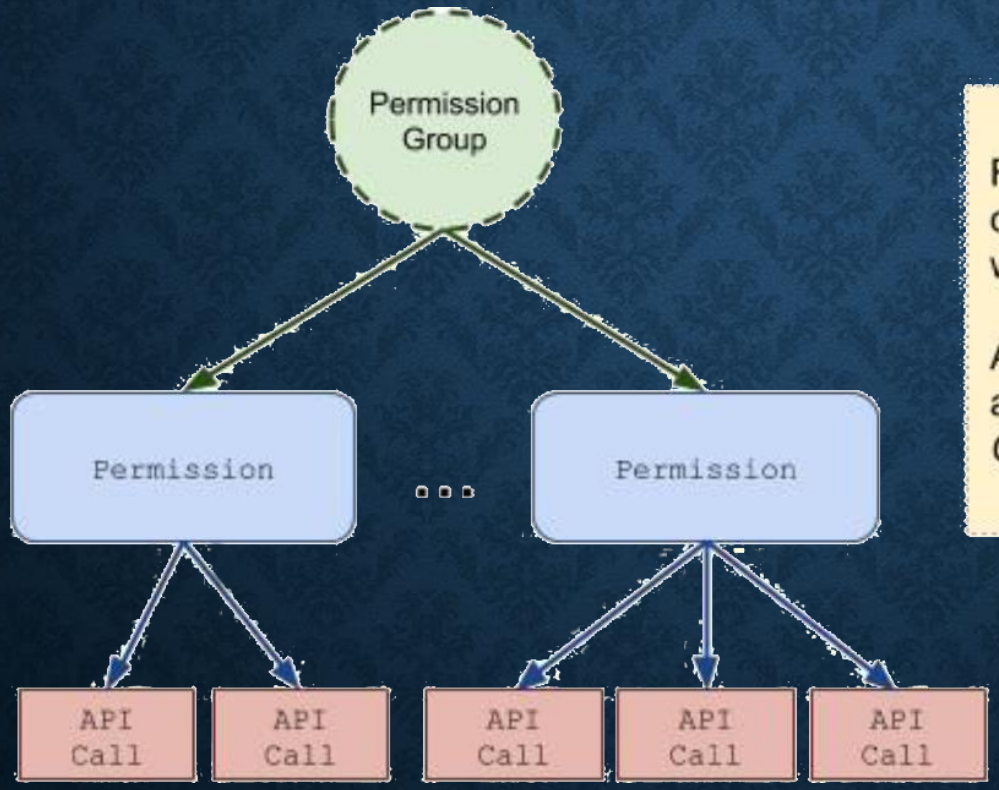
PERMISSION CATEGORIES

- System permissions fall into two categories
- Normal Permissions
  - Actions that do not directly risk the user's privacy
  - These permissions are automatically granted by Android to the app
- Dangerous Permissions
  - Actions that might cause the violation of the user's privacy
  - These permissions must be explicitly granted by the user
  - Will show a confirmation popup dialog to the user the first time the permission is required

PERMISSION GROUPS

- Permissions are organized into permission groups
- If access is granted to the permission group, all permissions inside that group is granted
- Permission popups show to the user are typically permission groups
  - If individual permissions would have to be granted, the user would get a lot of popups which would be annoying and user-unfriendly

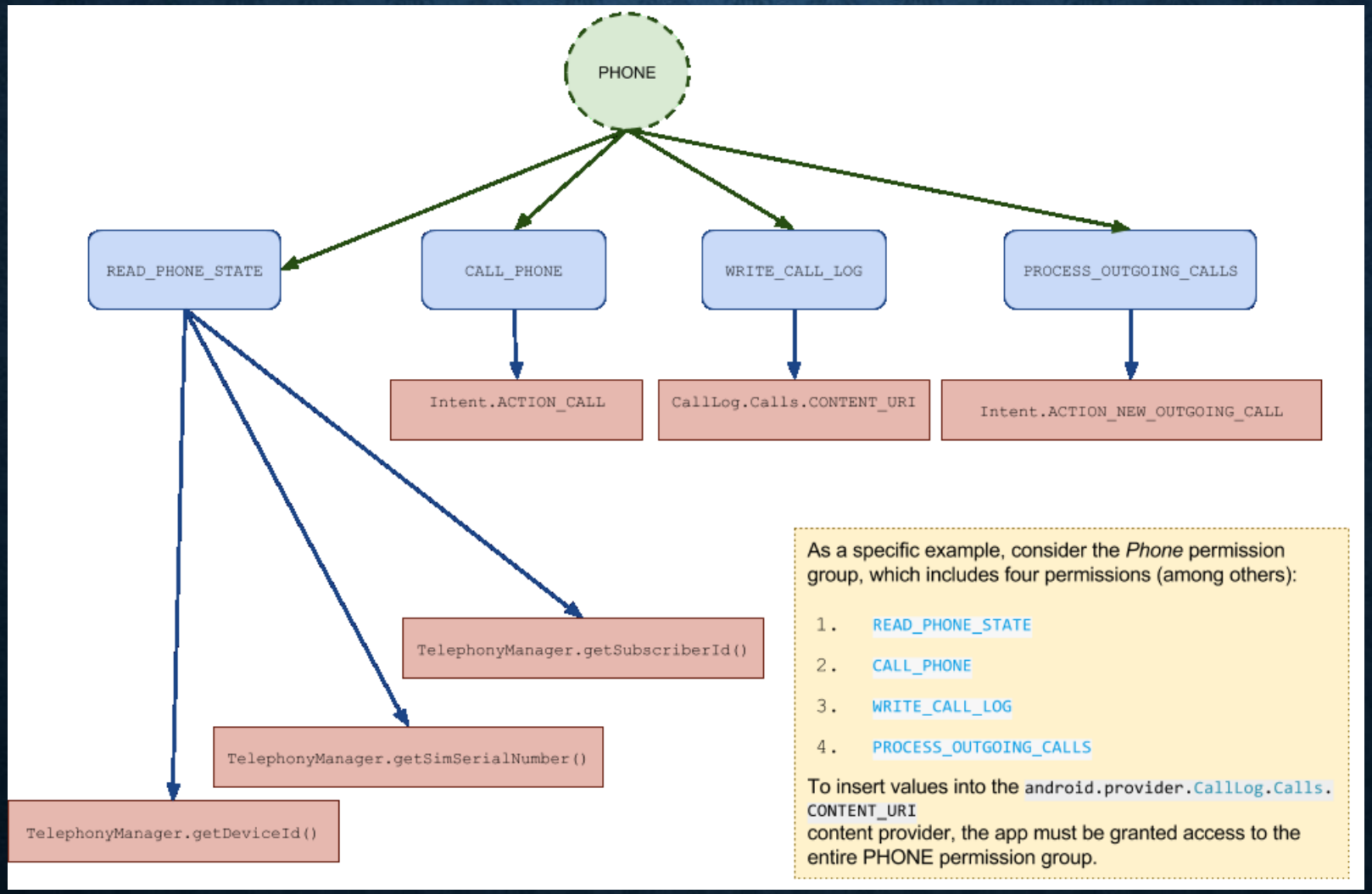
PERMISSION GROUPS



Related API Calls and access to object properties are associated with a specific permission request.

And related permission requests are rolled up into a *Permissions Group*.

PERMISSION GROUPS



As a specific example, consider the *Phone* permission group, which includes four permissions (among others):

1. `READ_PHONE_STATE`
2. `CALL_PHONE`
3. `WRITE_CALL_LOG`
4. `PROCESS_OUTGOING_CALLS`

To insert values into the `android.provider.CallLog.Calls.CONTENT_URI` content provider, the app must be granted access to the entire PHONE permission group.

PERMISSION RESTRICTIONS

- Only use the permissions if absolutely necessary for your app
  - There might be a different way to what you want that does not require permissions
- Pay attention to permissions required by libraries
  - When you include libraries, your app might require those permissions as well, depending on which functions you call in the library
- Be transparent with your permissions
  - Clearly indicate to the user which permissions are required and why, add those details to your readme, app about section, or website
- Make system accesses explicit
  - Provide continues indications when you access certain capabilities (eg camera) so that the user knows that you are not collecting data surreptitiously

PERMISSION DECLARATION

- Every app runs in a limited sandbox
- If apps use resources outside the app, they need the appropriate permission
- The permissions are declared in the AndroidManifest

```
<manifest
        xmlns="http://schemas.android.com/apk/res/android"
        package="com.satoshi.app">

        <uses-permission android:name="android.permission.SEND_SMS" />

        <application>
                ...
        </application>
</manifest>
```

PERMISSION RUNTIME

- Since Android 6, apps can request permissions during runtime
- If an app requires dangerous permissions, it has to check those permissions every time it executes the operation
- The user can revoke the permissions at any time, even if they were granted previously

PERMISSION RUNTIME

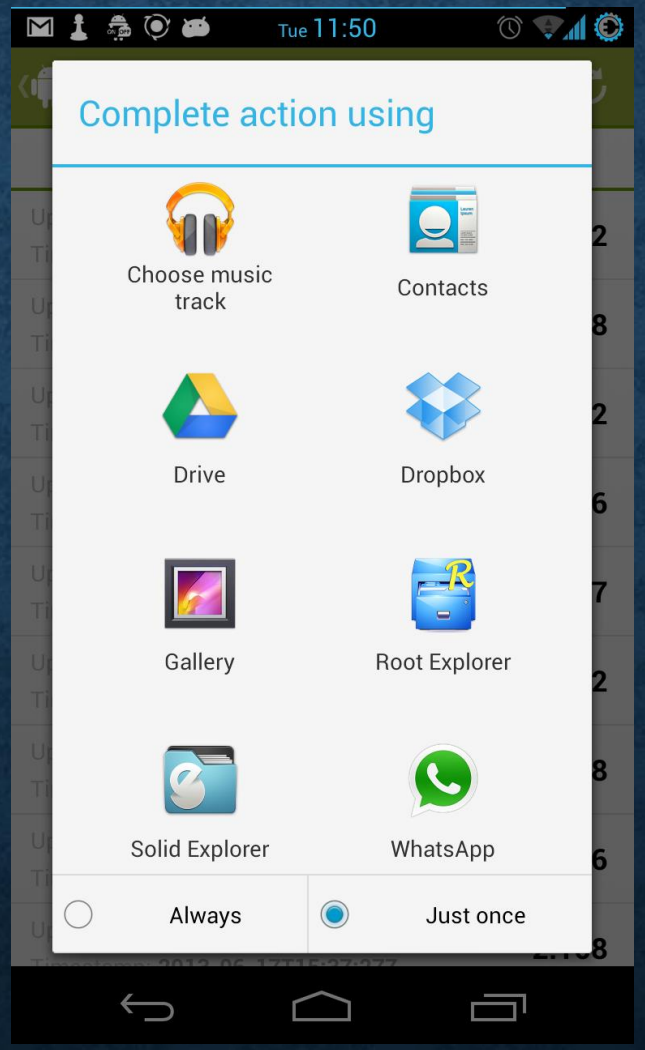
```
int per = ContextCompat.checkSelfPermission(
        currentActivity, Manifest.permission.SEND_SMS);

if(per == PackageManager.PERMISSION_GRANTED)
{
        // Already has permissions
}
else
{
        // Need to request permission
}
```

INTENTS

- To navigate between activities in your app, you have to use **explicit intents**
- When you want an external app to perform an action, you need **implicit intents**
- Implicit intents do not declare the class name of the component to start, but rather declare an action to perform
- Implicit intents often also include data associated with the action
- Implicit intents require an installed app to handle the request
  - If more than one app can handle the intent, a popup will ask the user which app to use (either use the app only once, or always use the selected app for those requests)

INTENTS


Tue 11:50
Complete action using
Choose music track
Contacts
Drive
Dropbox
Gallery
Root Explorer
Solid Explorer
WhatsApp
Always
Just once

INTENTS

- Calling a number

```
Uri number = Uri.parse("tel:0124205263");
Intent callIntent = new Intent(Intent.ACTION_DIAL, number);
```

INTENTS

- View a map

```
Uri location = Uri.parse("geo:0,0?q=1600+Union+Building,+Pretoria");
Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);

// Or provide coordinates
// z parameter is the zoom level

Uri location = Uri.parse("geo:-25.739472, 28.211819?z=14");
Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);
```

INTENTS

- View a website

```
Uri webpage = Uri.parse("https://coinmarketcap.com");
Intent webIntent = new Intent(Intent.ACTION_VIEW, webpage);
```

ANDROID