UQAC
Université du Québec
à Chicoutimi

Université du Québec à Chicoutimi (UQAC)

Hiver 2014

Cours 8INF857 - Sécurité informatique

Projet final

La faille humaine des smartphones

Présenté par

Mr Sylvain STOESEL

Mr Florian BOUCHOT

PROFESSEUR: Mr Sylvain HALLÉ

Remis le 22/04/2014

Sommaire

Introduction

Dans le cadre du cours de sécurité en maîtrise informatique à l'UQAC, nous avons mis en place notre projet, qui consiste à étudier les permissions d'Android vis-à-vis des applications. Le but est de montrer, à l'aide d'une démonstration, comment il est possible, avec les smartphones Android, de récupérer une quantité impressionnante d'informations (et ses limites) sur le téléphone, l'identité du propriétaire, sa vie privée…

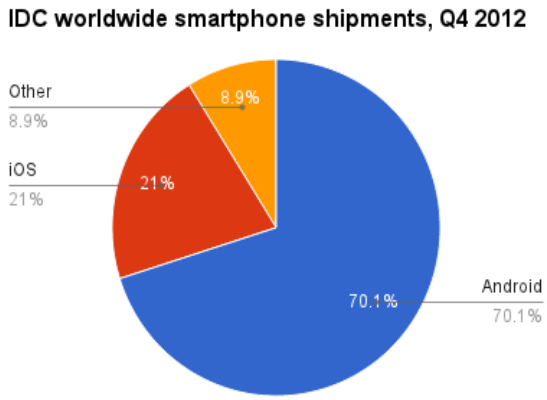
Nous étudierons les problèmes de sécurité que peuvent alors poser la possession de ces informations par un tiers : vol de numéros de cartes bancaires, *phishing*…

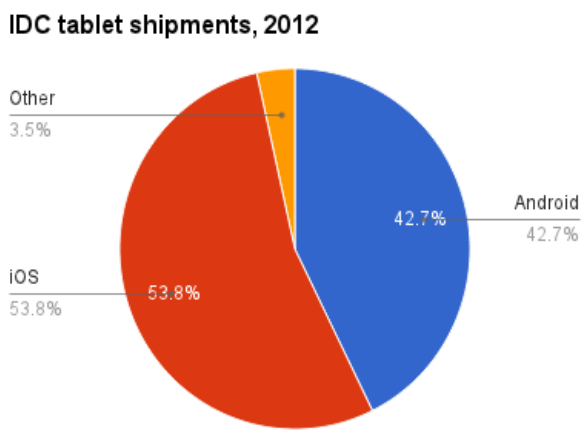
I. Etude sur les comportements des utilisateurs vis-à-vis des problèmes de sécurité et de vie privée

Dans le cadre des tenants et aboutissants du projet de sécurité effectué, il est intéressant de connaître certains chiffres sur android, pour ce rendre compte de l'ampleur que peu prendre l'application développée.

1) Statistiques android[1]

Actuellement, la plateforme android représente plus de 900 millions d'appareils. C'est le système d'exploitation mobile le plus utilisé au monde, largement devant Apple avec IOS.

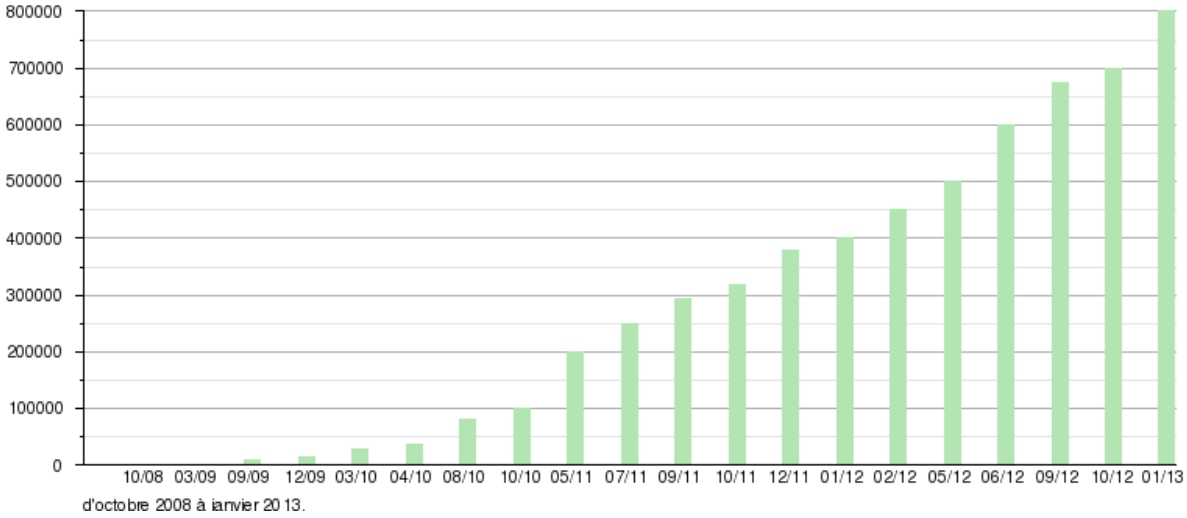




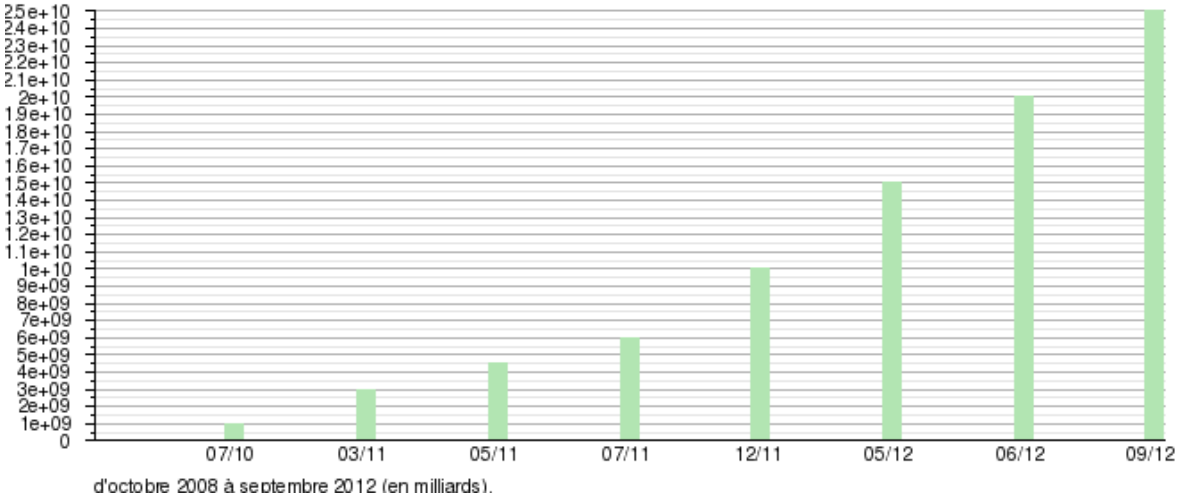
[1] http://techland.time.com/2013/04/16/ios-vs-android/

Et pourtant les systèmes de sécurité ne sont pas équivalents (voir III).

Les applications (tablettes et smartphones) sont disponibles via Google Play (play.google.com). Actuellement, le Google Play Store contient plus de 800 000 applications, et est considéré comme la plus grosse boutique d'applications au monde.



Il y a également plus de 1 300 000 activations de terminaux Android par jour.



2) Statistiques des permissions

"*When you download an app from the Android Google Play store, it will prompt you to accept the permissions it requests from your device. Most people do not pay attention and simply download the app. This is a bad idea. Left unchecked, app permissions can open your device to possible data theft, spam and malware.*" Dan Rowinski (08/20/2012)

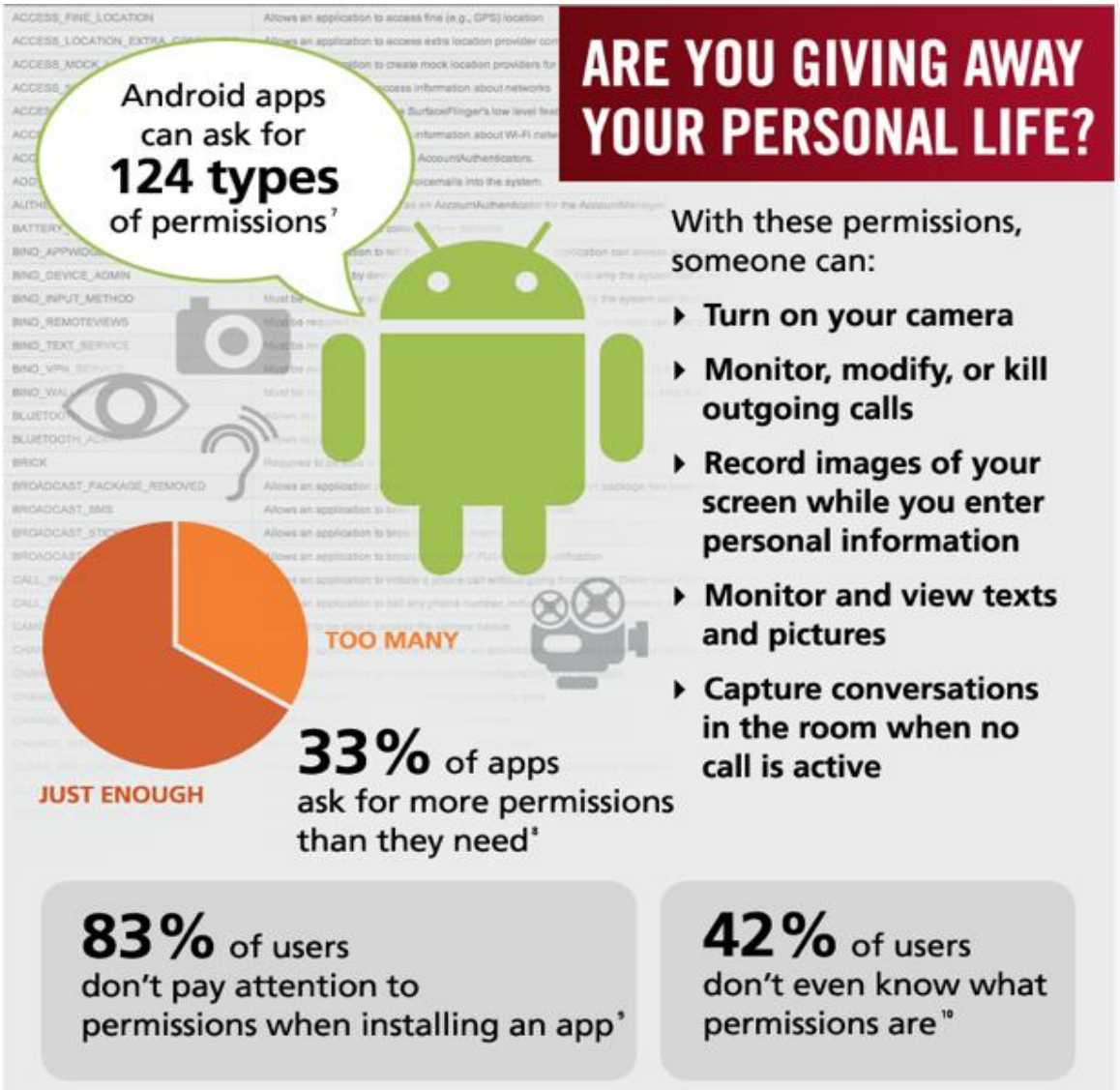
Les permissions sont les autorisations demandées par l'application que l'on veut télécharger, pour avoir accès à certains composants et/ou informations de l'appareil mobile. Avant le téléchargement de l'application, les permissions apparaissent à l'utilisateur, celui-ci peu accepter et télécharger l'application, ou refuser, et l'application ne sera donc pas installée sur l'appareil.

Une application peu demander jusqu'à 124 permissions différentes. Il faut savoir, d'après une étude de UC Berkeley Electrical Engineering and Computer Sciences department, environ **33%** des applications Android demandent plus de permissions qu'elles en ont besoin pour fonctionner.

Du point de vue utilisateurs :

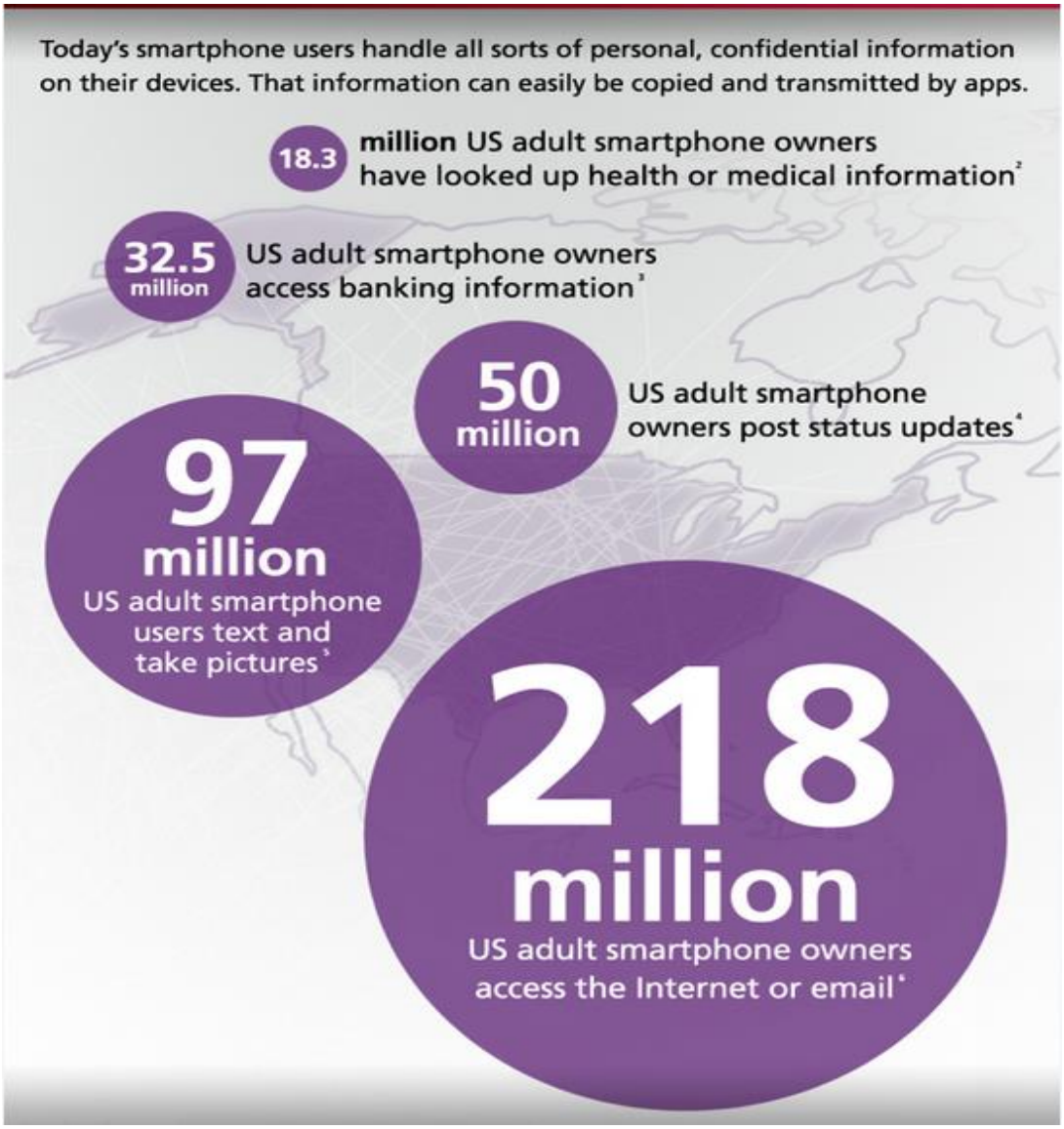
- **83%** des utilisateurs font attentions aux permissions quand ils installent une application

- **42%** d'entre eux ne savent pas ce que sont les permissions.



Le département de Berkeley a mené une étude, où il demandait à des utilisateurs android de retourner ce qu'ils comprenaient des permissions, et à quoi elles servaient. 97% des utilisateurs n'ont pas pu identifier à quoi servaient l'ensemble des permissions demandées.

3) Types d'informations divulguées



Parmi les informations les plus habituellement présentes sur le téléphone d'un utilisateur, on retrouve les mails, les SMS et l'historique Internet. On retrouve régulièrement aussi les photos de l'utilisateur, ainsi que ses statuts Facebook ou Twitter. Enfin, moins fréquemment, les téléphones contiennent des informations bancaires voire médicales.
Toutes ces informations, la plupart du temps présentes en clair dans la mémoire du téléphone, sont potentiellement divulgables.

II. Mise en pratique de l'application, tenants et aboutissants

Notre application a pour but de récupérer un certain nombre d'informations stockées sur le téléphone, et de les envoyer à un serveur via requête HTTP. Par la suite nous détaillerons l'ensemble des étapes.

1. Fonctionnement de l'application

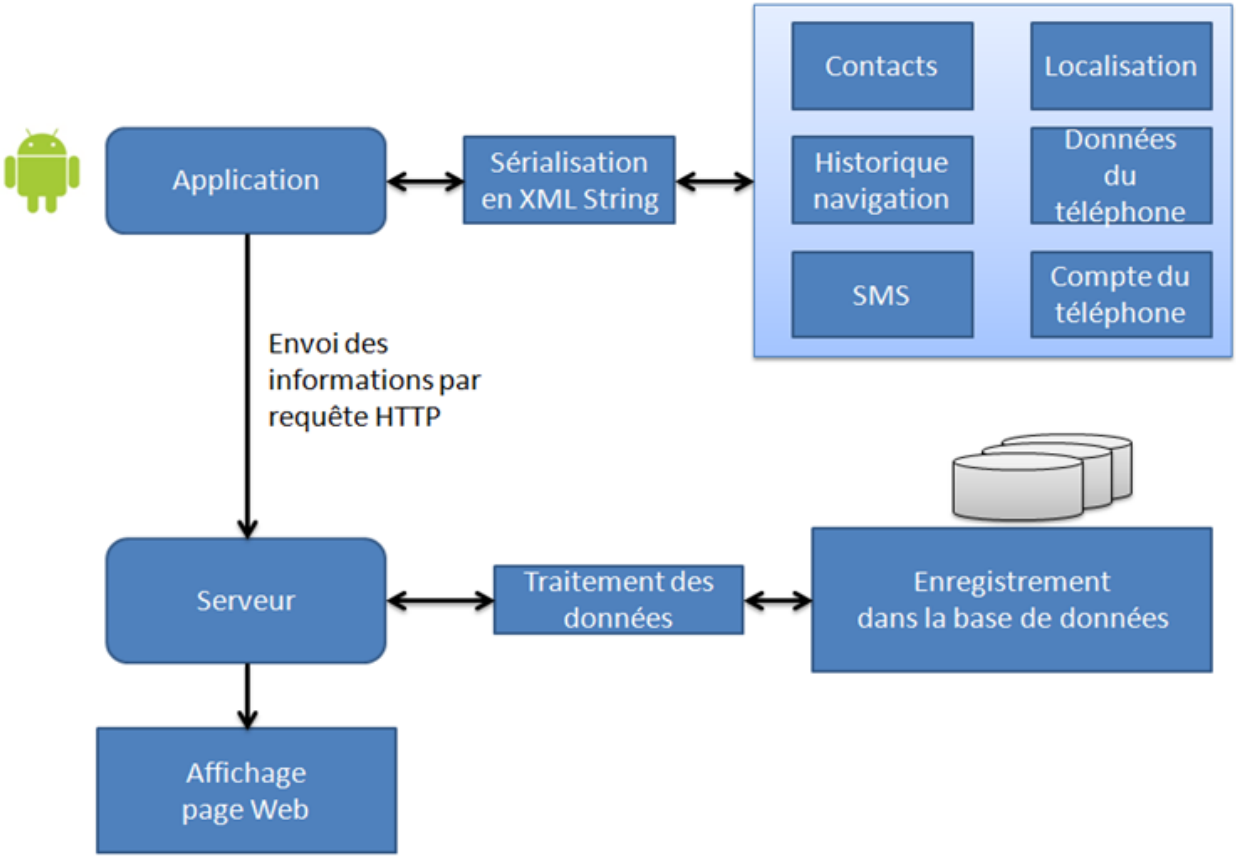


Schéma explicatif du fonctionnement de l'application

Une fois l'application installée, et les permissions accordées par l'utilisateur, l'application va chercher à son lancement l'ensemble des informations disponibles sur le téléphone (contacts, SMS…)

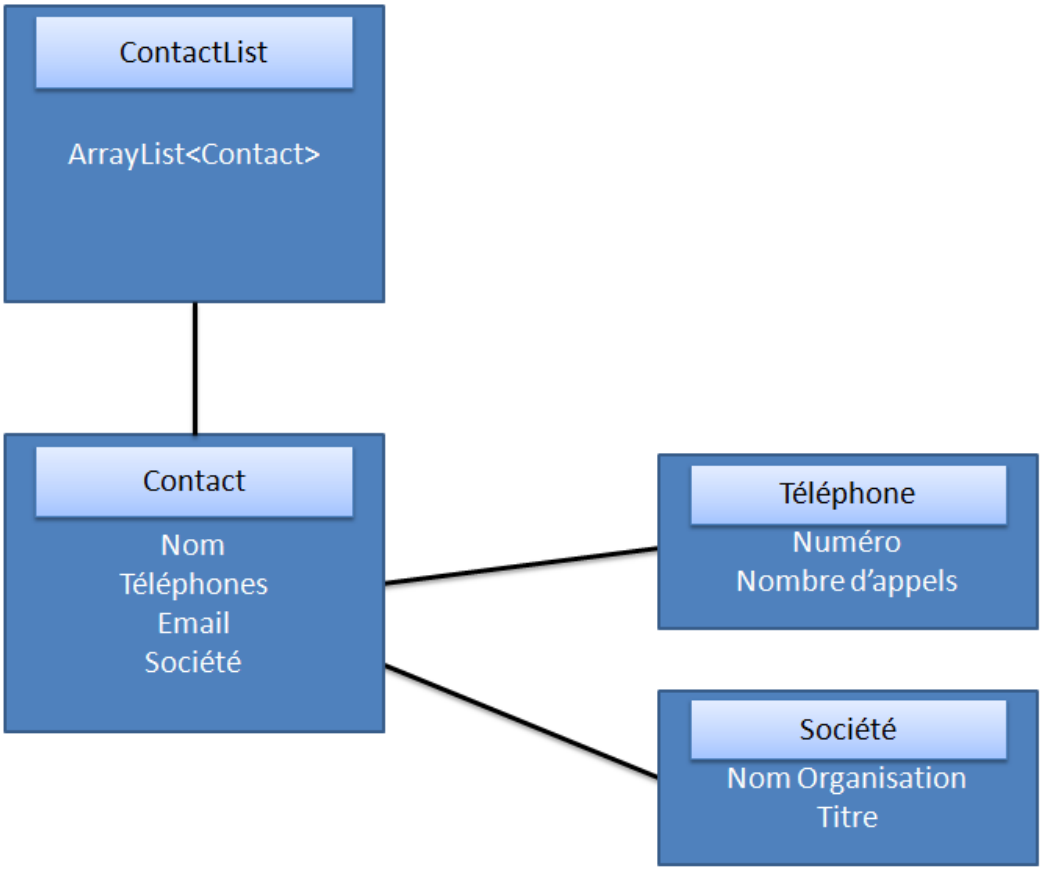
Ces informations sont ensuite sérialisées au format XML, dans une chaîne de caractères de grande taille.

Par la suite les informations sont envoyées au serveur, qui va travailler sur la chaîne de caractères pour identifier les informations, et les rentrer dans une base de données créée pour cette application.

Une fois toute ces étapes effectuées, une URL s'affiche sur l'écran du téléphone, qui renvoie vers un site web ou toutes les informations récupérées sont triées et visibles.

2. Récupération d'informations

a. Récupérations des contacts et de leurs informations



Pour pouvoir avoir accès au contact, l'application à besoin d'une permission spécifique :

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

Une fois cette permission mis en place, il faut accéder aux contacts, et retourner l'ensemble des informations possibles. Nous avons décidé de récupérer certaine informations sur les contacts :

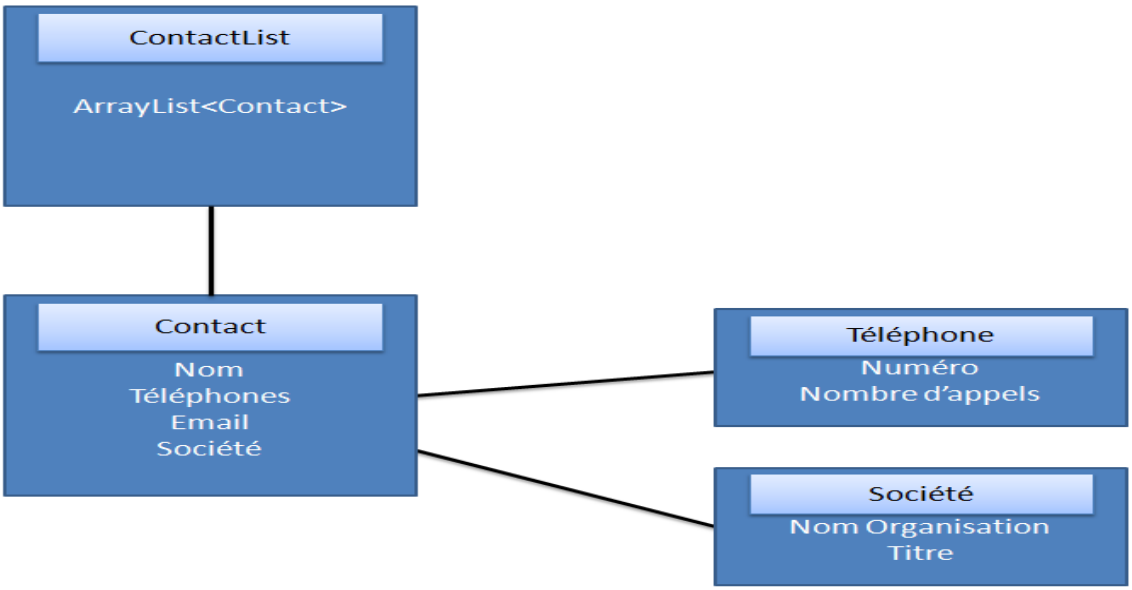


Schéma de récupération des contacts

On accède aux données via un curseur qui pointe directement sur la base de données d'android, vers la classe ContactsContacts, où sont stockés l'ensemble des informations sur les contacts.

```
ContentResolver cr = context.getContentResolver();

Cursor cursor = cr.query(ContactsContract.Contacts.CONTENT_URI,null, null, null, null);
```

De la on récupère le nom, les numéros de téléphones, les emails, et la société de chaque contact. On stock l'ensemble des informations dans la classe ListContact.

Dans l'activité principale, on met en place un thread pour récupérer ces informations et les sérialisés.

a. Récupérations de la localisation

Pour récupérer la localisation d'un appareil android, il y a deux choix :

- **GPS, avec la permission**

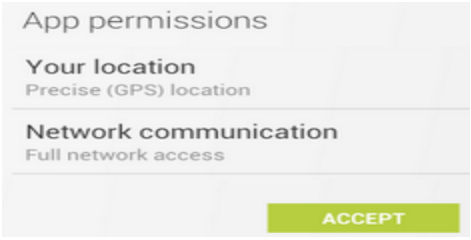
```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Si le GPS est activé, on peu récupérer les coordonnées géographiques de l'appareil (longitude et latitude). Les coordonnées sont vraiment précises.

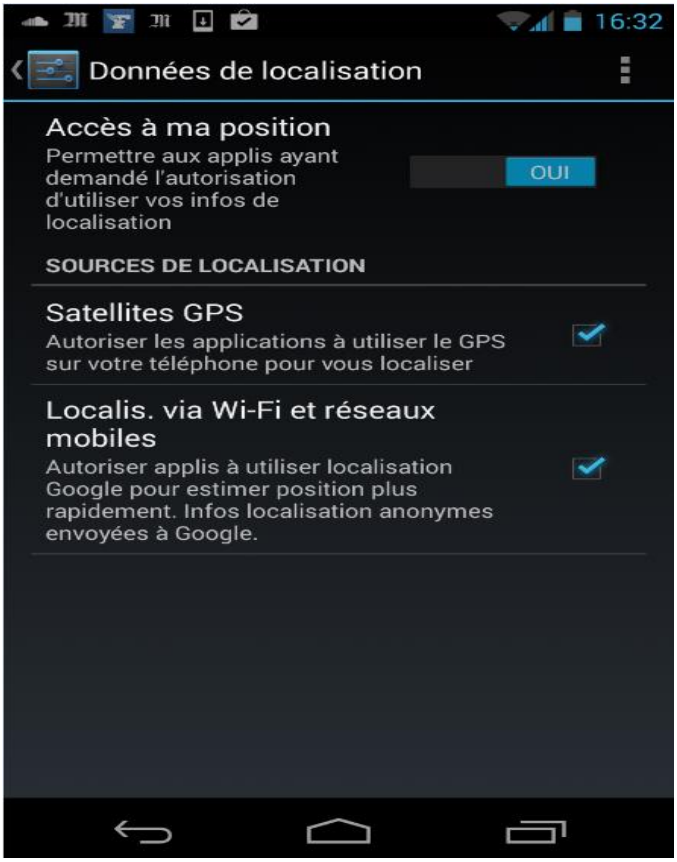
-**Wifi et réseaux mobiles**

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

Si la localisation est activée, on peu récupérer les coordonnées géographiques (longitude et latitude) de l'antenne relais du réseau mobile, ou la position de la box Wifi. Les coordonnées récupérées sont moins précises qu'avec le GPS.



Pour pouvoir avoir accès à ces informations, il est nécessaire que l'utilisateur ait activé l'accès à sa position.



Dans l'application, on test les deux possibilités. Tout d'abord on regarde si le GPS est disponible, si oui, alors on retourne la localisation GPS, sinon on test la localisation par wifi et réseaux mobiles, si c'est disponible on retour la localisation, sinon on retourne des cordonnées négatives.

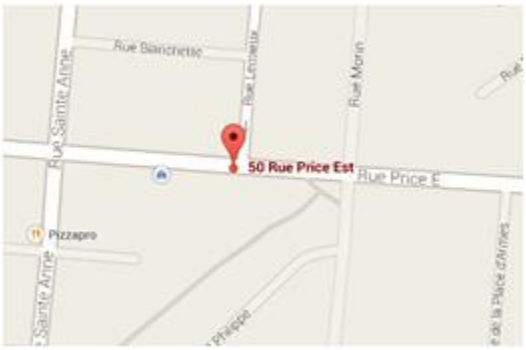
```
public void getBestLocalisation(Context context){

    if(getGPS(context))

    {}
    else
    _getLocation(context);

}
```

On arrive à récupérer des données précises, que l'on peut rentrer sur Google Maps, qui nous retourne notre position exacte lors de l'utilisation de l'application.



Exemple de localisation récupérée

b. *Récupérations des informations téléphones*

Pour récupérer les informations du terminal mobile, et de l'utilisateur, il est nécessaire d'avoir les permissions suivantes :

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

→ seulement pour l'utilisateur

Les informations du téléphone n'ont pas besoin de permission spécifique, et sont récupérables facilement via la classe **android.os.Build.** De là on récupère les informations sur le téléphone

```
public void setAllPhoneDeviceInformations(){
    this.device= android.os.Build.MODEL;
    this.hardware=android.os.Build.HARDWARE;
    this.manufacturer=android.os.Build.MANUFACTURER;
    this.product=android.os.Build.PRODUCT;
    this.User=android.os.Build.USER;

}
```

Comme le modèle, l'hardware, le constructeur, le produit et l'utilisateur.

Pour récupérer les informations de l'utilisateur, il faut passer par l'**AccountManager**, qui permet d'avoir accès à l'ensemble des comptes du téléphones (facebook, google+...)

```
final AccountManager manager = AccountManager.get(context);
final Account[] accounts = manager.getAccountsByType("com.google");
```

De la on peut récupérer l'identifiant, l'email … des différents comptes stockés sur le téléphone.

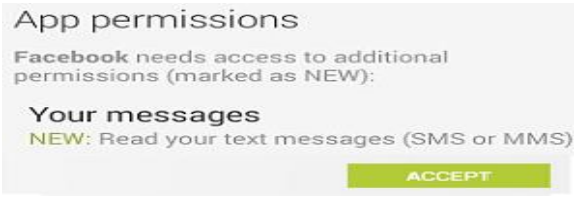
c. *Récupérations des SMS*

L'ensemble des SMS sont stockés dans une table spécifique

```
Uri message = Uri.parse("content://sms/");
```

Pour avoir accès à cette table, il faut avoir la permission suivante :

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



Une fois cette permission mise en place, on peut récupérer la date, le nom du contact et son numéro, et le message en clair de chaque SMS. On utilise la même technique que pour contact, on créer un cursor qui parcourt l'ensemble des SMS et nous renvoi les informations voulues.

```
public void getAllSms(Context context) {

    ArrayList<SMS> smslist= new ArrayList<SMS>();

    Uri message = Uri.parse("content://sms/");
    ContentResolver cr = context.getContentResolver();
    Cursor c = cr.query(message, null, null, null, null);
    int totalSMS = c.getCount();
    if (c.moveToFirst()) {
        for (int i = 0; i < totalSMS; i++) {

            SMS sms= new SMS();


            sms.setContactNumber(c.getString(c.getColumnIndexOrThrow("address")));

            sms.setMessage(c.getString(c.getColumnIndexOrThrow("body")));

            sms.setConversationID(c.getInt(c.getColumnIndexOrThrow("thread_id")));

            smslist.add(sms);
            c.moveToNext();
        }
    }
    setSmsList(smslist);
    c.close();

}
```

Ici, on récupère le numéro de téléphone, le message, et l'id de la conversation. L'id de conversation sert à déterminer quel SMS appartient à quelle conversation. De plus on récupère la date d'envoi des messages pour pouvoir trier les SMS dans notre base de données.

d. Récupérations des historique de navigation web

L'ensemble des historiques de navigation sont stockés sur le téléphone, la permission suivante est nécessaire pour pouvoir y accéder :

```
<uses-permission android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
```

Une fois cette permission mise en place, on créer un curseur qui pointe vers le dossier où sont stockés les historiques, et on les parcours pour en ressortir, ici, le titre de la page, et l'URL.

```
public void getBrowserHist(Context context)  {
    ArrayList<BrowserPage> browserlist= new ArrayList<BrowserPage>();
    ContentResolver cr = context.getContentResolver();
    Cursor mCur = cr.query(Browser.BOOKMARKS_URI,
            Browser.HISTORY_PROJECTION, null, null, null);
    mCur.moveToFirst();
    if (mCur.moveToFirst() && mCur.getCount() > 0) {
        while (mCur.isAfterLast() == false) {

        BrowserPage browserPage=new BrowserPage(mCur.getString(
                Browser.HISTORY_PROJECTION_TITLE_INDEX), mCur.getString(Browser.HISTORY_PROJECTION_URL_INDEX));
        browserlist.add(browserPage);
            mCur.moveToNext();
       }
        setBrowserPageList(browserlist);
  }
}
```

e. Envoi d'informations au serveur

Une fois toutes les informations collectées, elles sont transformées en une chaîne de caractère XML, que l'on envoie au serveur via une requête **HTTP POST**.

Le serveur récupère cette chaîne et parse le XML contenu.

```
if (isset($_POST["xml"])){
    $strxml = $_POST["xml"];

    $ALLXML = new DOMDocument;
    $ALLXML->loadXML($ALLXMLSTR);
```

En fonction du contenu trouvé dans le XML, des tuples sont rajoutés dans la BDD relationnelle (MySQL).

Par exemple, pour l'enregistrement de l'utilisateur, la fonction procède comme suit :

```
function creerUtilisateur($XML, $bdd){
    $nom1 = $XML->getElementsByTagName('OwnerAccount')->item(0)->getElementsByTagName('name')->item(0)->nodeValue;
    $adresse1 = "";
    $gmail1 = $XML->getElementsByTagName('OwnerAccount')->item(0)->getElementsByTagName('email')->item(0)->nodeValue;

    if ($XML->getElementsByTagName('OwnerAccount')->length>1){
        $nom2 = $XML->getElementsByTagName('OwnerAccount')->item(1)->getElementsByTagName('name')->item(0)->nodeValue;
        $adresse2 = "";
        $gmail2 = $XML->getElementsByTagName('OwnerAccount')->item(1)->getElementsByTagName('email')->item(0)->nodeValue;
    }
    else{
        $nom2 = "";
        $adresse2 = "";
        $gmail2 = "";
    }

    $randomString = substr(str_shuffle("0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"), 0, 32);

    $requete = "insert into UTILISATEUR(NOM1, NOM2, ADRESSE1, ADRESSE2,
                    COMPTEMAIL1, COMPTEMAIL2, DATEENREGISTREMENTBDD, CHAINEUNIQUE)
                    values ('$nom1', '$nom2', '$adresse1', '$adresse2',
                    '$gmail1', '$gmail2', now(), '$randomString')";

    $sql = $bdd->exec($requete);
    if ($sql>0)
        return $bdd->lastInsertId();
    return -1;
}
```

f. La base de données côté serveur

Schéma entité-association

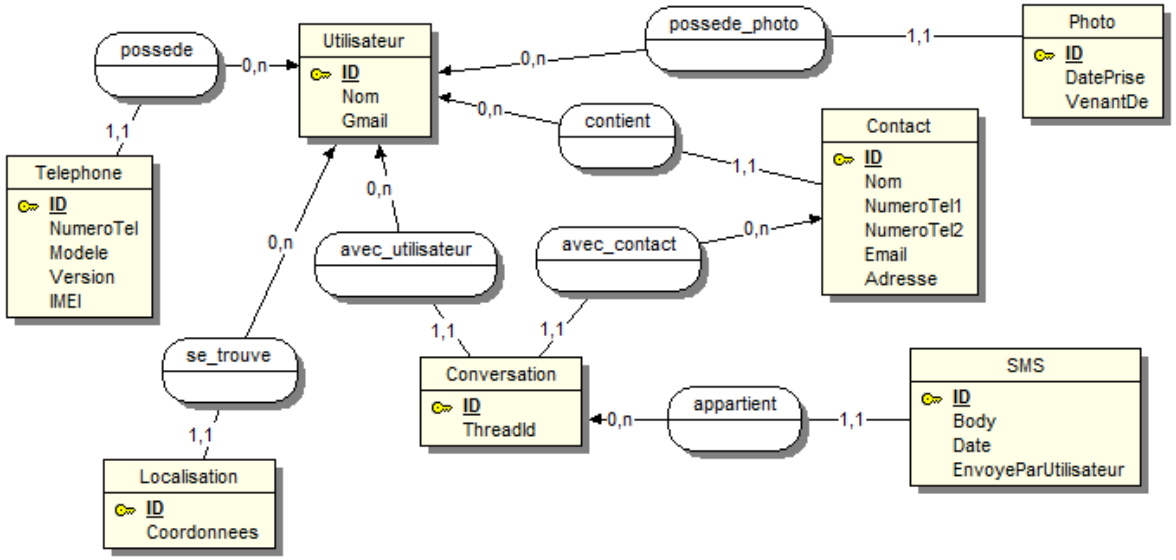
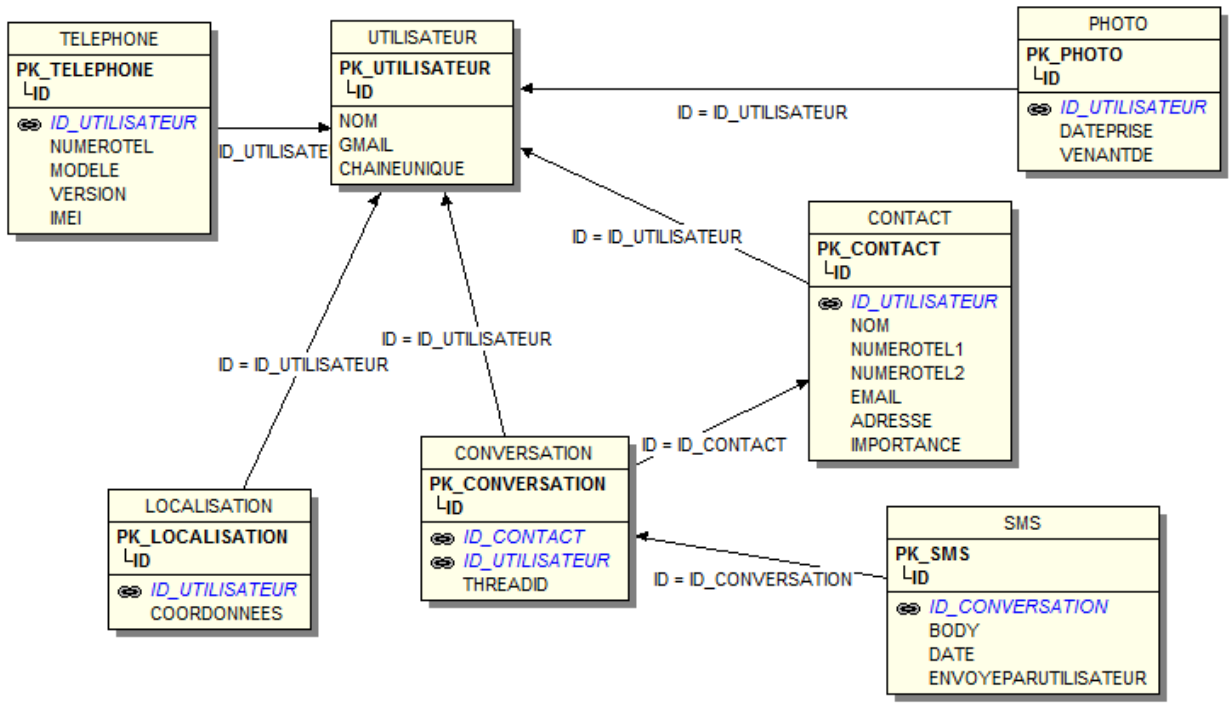


Schéma des tables SQL



Une fois l'ensemble du fichier XML parsé, des tuples sont normalement rajoutés dans chacune des tables ci-dessus.

g. Affichage des informations pour l'utilisateur

A la création d'un utilisateur dans la base de données, une chaîne unique et aléatoire et générée, puis enregistrée dans la table UTILISATEUR (voir fonction PHP plus haut).
Cette chaîne aléatoire permet d'identifier de manière unique un utilisateur. Elle est renvoyée au client (l'application Android) une fois l'enregistrement des informations effectué.
Un utilisateur qui souhaite voir ses propres données doit fournir au serveur cette chaîne unique pour s'identifier. Ainsi, un utilisateur ne peut voir que ses propres données, grâce à cet identifiant retourné par le serveur.
Pour voir les informations d'un autre utilisateur, il doit connaître la chaîne unique de cet autre utilisateur, ce qui n'est normalement pas possible. Bien entendu, notre application ne sert pas à publier à la vue de tous ces informations ; elles doivent rester confidentielles à l'utilisateur concerné.
Nous avons gardé un utilisateur dans la BDD, accessible par ce lien :
http://uqac.netii.net/show.php?uid=OyoEWQwR9j376X8xCDiuAMKBkmvl2V0s
(Ici, l'identifiant unique généré est ***OyoEWQwR9j376X8xCDiuAMKBkmvl2V0s***).

Les informations sont alors visibles dans une page HTML :

III. Les risques potentiels des applications mobiles

1. Le partage d'information privée

Le fait que un certain nombre d'utilisateurs ne savent pas ce qu'est une permissions, ou ne les lisent pas, rend d'autant plus facile le "vol" d'informations sur les terminaux mobiles Android. On a montré précédemment qu'il n'était pas compliqué de récupérer les informations d'un téléphone avec les permissions, et de les envoyer à un tiers.

On peut alors faire plusieurs choses avec ces informations :

- Voler et dévoiler les informations privées d'une personne, sachant que l'on récupère toutes ses conversations avec ses amis, sa famille, et l'ensemble de ses proches

- Etudier les habitudes de l'utilisateur en faisant du forage de données sur son historique d'URL, ces localisations… on peut facilement récupérer un profil type de client pour faire du marketing.

- Certaines permissions permettent de mettre en marche le téléphone, et /ou de prendre des photos à l'insu de l'utilisateur. On peut alors espionner facilement l'utilisateur en écoutant ses conversations, ou augmenter drastiquement ses factures de téléphones en téléphonant à des services.

2. Création de fausses identités

Certaines permissions permettent de créer de fausses identités sur le téléphone. On peut par exemple facilement créer un contact factice, avec un nom, prénom, numéro de téléphone… et créer des conversations avec cette personne factice sur le téléphone.

C'est une intrusion énorme dans la sphère privée de l'utilisateur, car d'après certaines études, 98% des SMS sont lus, donc on peut facilement faire de la publicité ciblé, que l'on sait lue par l'utilisateur, en créant de faux contacts.

Il est aussi possible, avec la récupération de l'ensemble des emails stockés sur le téléphone, d'abuser de la confiance des utilisateurs. Avec un simple script VBS, on peut envoyer des emails de la part d'un des contacts que l'on a récupéré, à l'utilisateur. Cette pratique permet de passer la barre de la confiance de l'utilisateur. Par exemple, si on a identifié que "Alex" est le contact avec lequel l'utilisateur à le plus d'interactions (Nombre de SMS, d'appels…). On peut envoyer un mail de la part d'Alex à l'utilisateur pour lui faire télécharger par exemple des malwares.

3. Comparaison avec IOS et Windows Phone

Il y a plusieurs points communs entre les trois systèmes :

- Les mêmes types de permissions sont accordés aux applications sur les différents OS (lecture des SMS, récupération des contacts…)
- Une fois l'application installée, le même processus de vol d'informations détaillées plus haut peut être mis en place sur les trois différents systèmes d'exploitation.

Néanmoins, certaines différences notables sont présentes :

- Android permet à l'utilisateur de savoir clairement quelles sont les permissions que demandent les applications et qui leur sont accordées, lors du téléchargement et avant l'installation sur le téléphone.
- A l'inverse, Apple ne montre jamais les permissions accordés à l'utilisateur, car Apple effectue des tests sur l'ensemble des applications disponibles sur l'Apple Store, que ce soit en termes de bugs, de compatibilité, et d'intrusion dans la vie privée des utilisateurs. Il en va de même pour Windows phone.
- Android permet l'installation de sources différentes du Play Store, ce qui laisse une plus grande liberté aux utilisateurs, mais qui augmente les risques d'exécution de programmes malveillants.

Conclusion

Ce projet nous a vraiment permis de mettre en place une application, pour vérifier l'accessibilité aux données d'un téléphone Android, et de mettre avant la faille, qu'est l'humain, dans ce système de permissions.
Nous avons voulu montrer, et au final créer une application de sensibilisation pour les personnes qui ne regardent jamais les permissions.

A terme l'application développée pourra être mise sur le **PlayStore** de Google ; pour cela, il reste à améliorer la rapidité des accès aux informations du téléphone, et à effectuer du *refactoring* sur le code, l'application ayant été construite de manière expérimentale. De plus, notre programme n'est pas encore totalement exempt de *bugs*. Enfin, la partie web, avec l'affichage des données récupérées sur le site, se doit également d'être davantage perfectionnée.

Nous tenons à ajouter que nous même avons été impressionnés de voir nos propres informations envoyées aussi facilement sur le serveur. Les voir s'afficher dans un navigateur, sur un autre écran que notre téléphone – là où nous avions l'habitude de les voir – nous a montré à quel point les limites de la vie privée ont été repoussé depuis l'avènement des smartphones.

Références

Chiffres sur android et les mobiles :
http://techland.time.com/2013/04/16/ios-vs-android/

Informations sur les permissions
http://www.androidcentral.com/look-application-permissions
http://www.makeuseof.com/tag/the-seven-deadly-android-permissions-how-to-avoid-the-sin-of-slothful-preparedness/
http://lifehacker.com/5679471/how-can-i-tell-if-an-android-app-is-malware
https://support.doo.net/hc/en-us/articles/201939407-How-does-doo-for-Android-use-permissions-and-for-what-
http://www.makeuseof.com/tag/app-permissions-work-care-android/

Code :
http://stackoverflow.com/