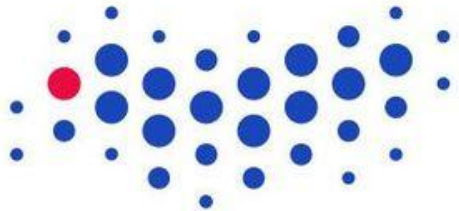
Практикум на ЭВМ.

Программирование под Android

Занятие 7



BroadcastReceiver

BroadcastReceiver

- Обмен сообщениями между приложениями/внутри приложения

BroadcastReceiver

```
public final class SimpleReceiver extends BroadcastReceiver {
    @Override
    @MainThread
    public void onReceive(Context context, Intent intent) {
        ...
    }
}
```

Подписывание на события

```
<application>
    ....
    <receiver android:name=".SimpleReceiver">
        <intent-filter>
            <action android:name="ru.ifmo.android_2016.SIMPLE_ACTION"/>
        </intent-filter>
    </receiver>
    ...
</application>
```

Приложение будет запущено при необходимости.

Подписывание на события

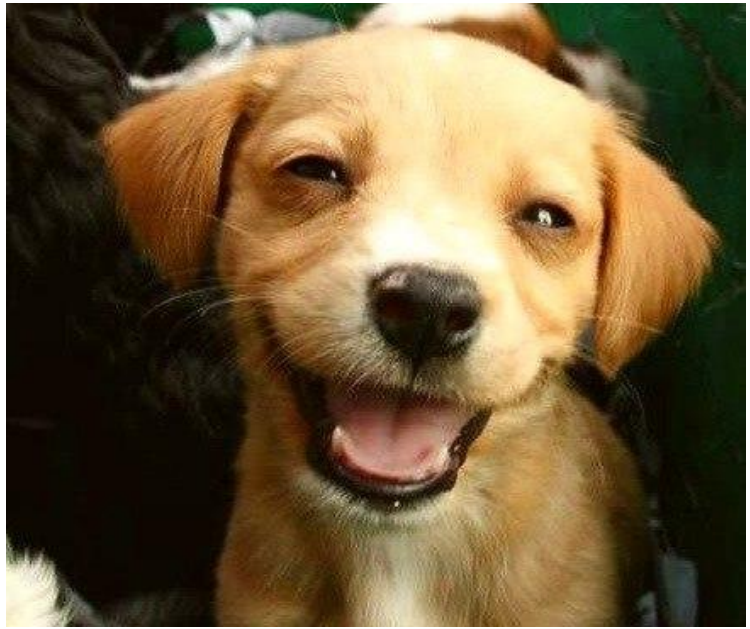
```
receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        ....
    }
};

IntentFilter intentFilter = new IntentFilter(ACTION);
registerReceiver(receiver, intentFilter);


unregisterReceiver(receiver);
```

Жизненный цикл

- onReceive



Отправка событий

```
Intent intent = new Intent(ACTION);

intent.putExtra(...);


context.sendBroadcast(intent);
```

Упорядоченные события

```
sendOrderedBroadcast(intent, null);


intentFilter.setPriority(IntentFilter.SYSTEM_HIGH_PRIORITY);


@Override
public void onReceive(Context context, Intent intent) {
    abortBroadcast();
}
```

Безопасность

- Кто получит:
  - `intent.setPackage(packageName)`
  - `sendBroadcast(intent, permissionName);`

```
<receiver
    android:name=".broadcast.SimpleReceiver"
    android:permission="permissionName">
```

- Кто может отправить:

```
<receiver
     android:name=".broadcast.SimpleReceiver"
     android:exported="false">
```

!Odnoklassniki / OL-55107

Перехват входящих сообщений пользователя

Edit | Comment | Assign | More | Return to development

Details

Type:	Bug	Status:	Resolved
Priority:	Major	Resolution:	Done
Labels:	bugbounty security		
Bug Environment:	Эксплуатация		
Project affected:	Приложение Android		

Description

Любое стороннее андроид приложение может перехватывать входящие сообщения без ведома пользователя.

Во вложении исходный код и APK простого приложения, которое демонстрирует уязвимость. Для наглядной демонстрации выполнить следующие шаги:

- Установить официальное приложение Одноклассники под андроид, запустить и залогиниться. Закрыть приложение.
- Установить демо приложение OkMessageSpy.apk, запустить и закрыть
- Отправить текущему пользователю сообщения в личной переписке в Одноклассниках – при этом на устройстве будут появляться уведомления о входящих сообщениях.
- Запустить OkMessageSpy – он покажет все перехваченные входящие сообщения

Примечание: если в момент получения входящего сообщения был открыт экран переписки в официальном приложении Одноклассники, то это сообщение не перехватывается.

Перехват возможен из-за того, что при получении пуша с входящим сообщением, приложение Одноклассники создает броадкаст с action=ru.ok.android.action.NOTIFY, а в extras кладет текст сообщения. Любому стороннему приложению достаточно подписаться на этот броадкаст и свободно получать его. Функциональность приложения Одноклассники при этом никак не нарушается. Кроме текста сообщения в extras кладется разная техническая информация (ID переписки, ID отправителя и пр.) Точно таким же образом перехватываются уведомления о подарках, о новых комментариях в обсуждениях, о входящих видео-звонках и т.п.

Уязвимость может быть легко обнаружена хакерами, для этого достаточно распаковать APK файл официального приложения и посмотреть манифест. Броадкаст там явно задекларирован.

LocalBroadcastManager

- Работает в рамках одного процесса
- Нет упорядоченных событий

```
LocalBroadcastManager.getInstance(this).
        registerReceiver(receiver, intentFilter);
```

Service

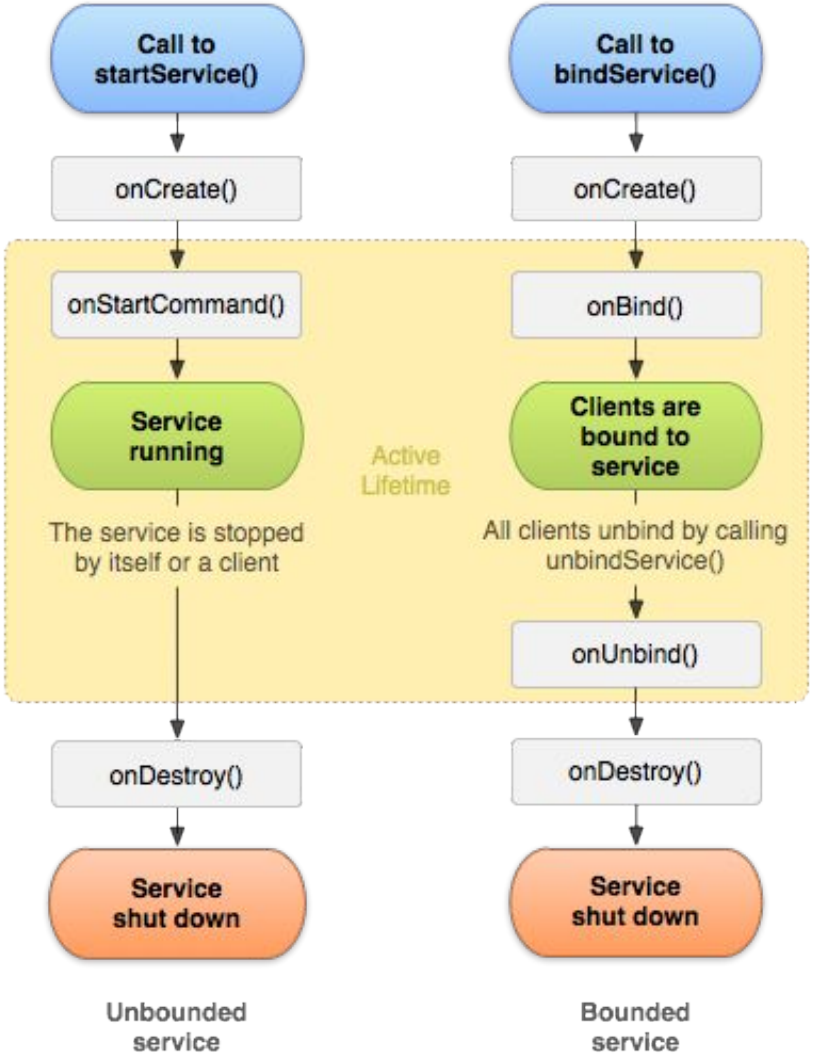
Service

- Не связан с интерфейсом
- Повышает приоритет процесса
- Может быть перезапущен системой
- Межпроцессное взаимодействие (IPC)
- Может находиться в отдельном процессе

AndroidManifest.xml

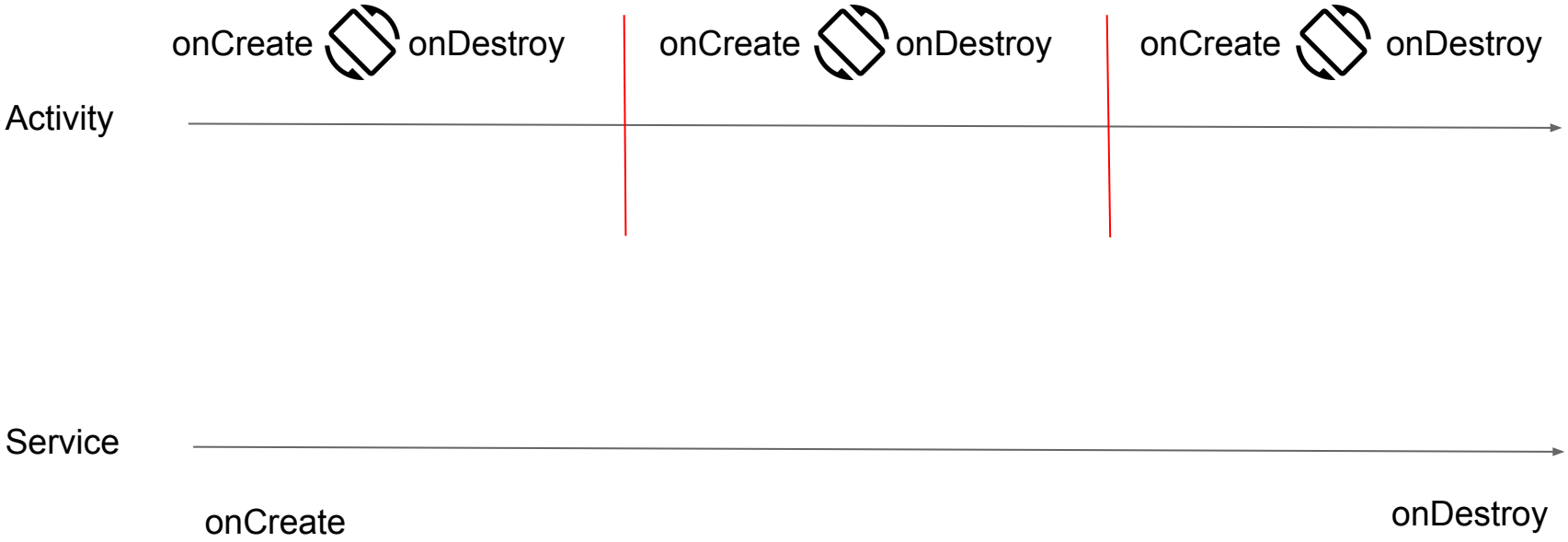
```
<application/>
    ...
    <service android:name=".SimpleThreadService"/>
    ...
</application>

public final class SimpleService extends Service {
    ...
}
```



Главный поток

Жизненный цикл



onStartCommand

```
startService(new Intent(this, SimpleThreadService.class));
```

```
@Override
public int onStartCommand(
        Intent intent, int flags, int startId) {
    return START_REDELIVER_INTENT;
}
```

onStartCommand

- START_NOT_STICKY
- START_STICKY
- START_REDELIVER_INTENT

stopSelf

```
private void onTaskDone() {
    stopSelf();
}
```

```
private void onTaskDone(int startId) {
    stopSelf(startId);
}
```

IntentService

```
public final class ThreadService extends IntentService {
    @Override
    protected void onHandleIntent(Intent intent) {
        // process intent here
    }
}
```

- Вызывает stopSelf
- Выполняет задачи последовательно

Foreground service

```
public final void startForeground(
        int id, Notification notification)
```

```
public final void stopForeground(boolean removeNotification)
```

Приоритеты процессов

- На переднем плане (foreground)
  - Activity (onResume)
  - Foreground Service
  - Выполняется BroadcastReceiver.onReceive, Service.onCreate/onStartCommand/onDestroy
  - Сервис, с которым взаимодействует Activity с переднего плана
- Видимые (visible)
  - Activity (onPause)
  - Сервис, с которым взаимодействует видимая Activity
- Содержит сервис (startService)
- Фоновые (background)
  - Activity (onStop)
- Пустые

Вопросы?