FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

ContrOWL

An Android Security App

Bachelor Thesis

submitted: October 2012

by: Marcel Hrnecek

Student ID Number: 21340224

Department of Computer Science
Friedrich-Alexander-University Erlangen-Nuremberg
D – 91058 Erlangen
Internet: `http://www1.informatik.uni-erlangen.de`

Abstract

Android offers developers an extensive Application Programming Interface (API) that includes access to valuable aspects of the android platform and user data. The access to these privacy- and security-relevant parts is controlled by permissions which have to be granted to start the installation process.

Android uses intents for inter- and intra-application communication. One way to use intents is as a broadcast to inform interested apps of changes or events. To get these intents apps can use the intent filter of their broadcast receivers.

Permissions and intents are both used as gateway for malicious applications[1] and are a crucial aspect of security. This Bachelor Thesis studies the quantity of permissions and filtered intents of broadcast receivers of malware and non-malware apps. The results have been compared in order to form a rating algorithm based on an app's permissions and received intents.

The results and the rating algorithm have then been implemented in the security app "ContrOWL" which was developed in the course of this Bachelor Thesis.

[1] Consecutively always Android applications and abbreviated by app

Contents

List of Figures

List of Tables

1. Introduction

Mobile devices have been drastically improved in the past few years and as a result they are more and more used and replace the conventional desktop computer at home. The downside to this fast development is the lack of security, new possibilities for attackers and missing safety awareness of many users. To countervail this movement this Bachelor Thesis tries to improve the security on mobile devices and help users in their daily handling with a vast amount of mobile apps.

Android is a popular platform for mobile app developers because of its unrestricted market, open source, and extensive API. For these reasons Android has been chosen as a basis for this Bachelor Thesis. The access to privacy- and security-relevant parts is controlled by permissions. Before an app can be installed the user has to grant the permissions which are requested from the installer of the app at the beginning of the installation process. Otherwise the installation is canceled.

Many apps are over privileged which means they ask for more permissions than they actually need (1). This could be by accident or on purpose to harm the user of the app or seek some valuable data. Either way it can lead to security risks.

Beside permissions malicious apps can also use intents to harm the user. Android uses intents for inter- and intra-application communication. One way to use intents is as a broadcast to inform interested apps of changes or events. An app can register a broadcast receiver for certain intents, decide by itself what priority it has in receiving this intent, and even consume it so other receivers will never be informed about the changes or events. A malware app can for example register itself for the SMS_RECEIVED[1] intent with the highest priority. SMS_RECEIVED is than broadcasted once after a Short Message Service (SMS) message is received. The malware app can then catch this event before the

[1]Full package name is android.provider.Telephony.SMS_RECEIVED

system, read it, and not allow it to be sent to other registered apps. The user will never notice that he received a SMS message while the malicious app can read all incoming messages.

Some broadcasted intents can only be received when an app holds a certain permission. In this way permissions and intents are both used as a gateway for malicious apps and are a crucial aspect of security. For the example above the permission RECEIVE_SMS[2] is needed.

To find out what permissions and broadcast intents are preferably used by malicious apps we studied the quantities of these permissions and intents in malware and non-malware apps. Therefore a sample collection of about 10.000 malware and 25.000 non-malware apps has been investigated. This huge collection of sample apps has been provided by the Department of Computer Science (Friedrich-Alexander-University Erlangen Nuremberg). We developed scripts to search in this sample collection for permissions and intents filtered by intent-filters of broadcast receivers.

The results have then been compared in order to form a rating algorithm first for the single permissions and broadcast intents. Based on these results in a second step individual ratings for the apps are calculated. The aim was to give a high rating for permissions and broadcast intents which are preferably used by malware and accordingly a high overall rating for apps using these permissions and intents. Needless to say the aim for non-malicious permissions, intents and apps is a low rating.

Various rating algorithms have been tested on the sample collection to find the best match for this declared aims.

On base of this study the security app "ContrOWL" has been developed. The key features of the app are:

- List of top permissions and broadcast intents used by malware
- Ranking, rating and description of these permissions and intents
- Finding apps on the device which use these permissions and intents
- List of all installed apps on the device and their overall rating

[2]Full package name is android.permission.RECEIVE_SMS

- Permissions and intents used by an app
- Possibility to remove apps

Many more features, like a service which runs in the background and immediately informs the user after the installation of a new app about its rating, are desirable but haven't yet been implemented due to the limited extend of this Bachelor Thesis.

2. Study on Quantity of Permissions and Intent-Filters of BroadcastReceivers

The aim of this study is to search in a collection of malicious and non-malicious apps for their permissions and intent filters of broadcast receivers. Afterwards the quantity of the results will be compared. We found that particular permissions and filtered intents in malware apps are noticeable different from non-malware apps.

For this purpose the Department of Computer Science (Friedrich-Alexander-University Erlangen-Nuremberg) provided a collection of sample apps. We downloaded the 25994 non-malicious sample apps on 08.05.2012 and the 9968 malicious apps on 12.08.2012 form this collection (2).

2.1. Quantity of Permissions

Some apps ask for permissions, which they don't really need. This can be to leave room for future updates, by accident, to harm or to spy on the user.

The malware apps described above are such apps and it is of interest to take a look at their permissions and compare them to normal non-harmful apps.

2.1.1. Prerequisites

For the purpose of reading the permissions, filtering and sorting them a Windows Power Shell[1] script has been developed (see Listing 2.1). To get permissions Android Asset Packaging Tool[2] (aapt) with the parameters "d permissions filename"[3] has been executed. Most apps have more than one permission and to

[1] This is Microsoft's task automation framework, consisting of a command-line shell and associated scripting language built on top of, and integrated with the .NET Framework (3).
[2] A SDK-tool of android that allows to view, create and update apk files
[3] d[ump] the permissions from file.apk

find any pattern or popular combinations of permissions, pairs of two and three have been filtered as well.

```
$market = get-childitem "...path..." -recurse |
        where{$_.name -like "*.apk"}
$hashVals = @{}

foreach($file in $market)
{
  # ... get $fileItem ...
  $ApPerm = aapt d permissions $fileItem
  $hashSemiVals = @{}

  # Purging Double Permissions:
  foreach($entry in $ApPerm)
  {
    if($entry -ne $NULL)
    {
      $hashSemiVals[$entry]++
    }
  }

  foreach($entry in $hashSemiVals.Keys)
  {
    $hashVals[$entry]++
  }
}

# ... write $hashVals to file ...
```

Listing 2.1: Code snippet to find single permissions

Table 2.1.: Top permissions of (25994) Google Market apps

Rank	Permission	Quantity
1	INTERNET	22272
2	ACCESS_NETWORK_STATE	13721
3	WRITE_EXTERNAL_STORAGE	8936
4	READ_PHONE_STATE	8653
5	ACCESS_COARSE_LOCATION	6064
17	SEND_SMS	957

2.1.2. Results

Table 2.1 shows very common permissions in the top. INTERNET[4] for example is used by over 85% of non-malware apps, so there is no reason to be very suspicious about this permission.

Permissions like SEND_SMS[5] which have a lower quantity and are only used by less than 4% of the Google Market sample apps could be more dangerous. There is another reason why this permission is critical, but this is shown later in Section 2.3.

Table 2.2 is of the same kind as Table 2.1 but only for malicious apps. Some permissions have a similar percentage as in Table 2.1 (e.g. INTERNET with 89%). Others have a noticeable different percentage (e.g. SEND_SMS with 40% 10 times as many). This indicates that these permissions are often exploited for malicious behavior.

Table 2.3 shows pairs of two permissions often found in the sample of malware apps.

Table 2.4 shows paris of three permissions often found in the sample of malware apps.

[4]Full package name is android.permission.INTERNET
[5]Full package name is android.permission.SEND_SMS

Table 2.2.: Top permissions of (9968) malware apps

Rank	Permission	Quantity
1	INTERNET	8827
2	READ_PHONE_STATE	5515
3	ACCESS_NETWORK_STATE	4910
4	SEND_SMS	4010
5	WRITE_EXTERNAL_STORAGE	3835
11	ACCESS_COARSE_LOCATION	1920

Table 2.3.: Top permission pairs of two in (9968) malware apps

Rank	Permission pairs	Quantity
1	INTERNET + READ_PHONE_STATE	5423
2	INTERNET + ACCESS_NETWORK_STATE	4875
3	INTERNET + SEND_SMS	3729
4	INTERNET + WRITE_EXTERNAL_STORAGE	3721
5	ACCESS_NETWORK_STATE + READ_PHONE_STATE	3263

Table 2.4.: Top permission pairs of three in (9968) malware apps

Rank	Permission triple	Quantity
1	INTERNET + READ_PHONE_STATE + ACCESS_NETWORK_STATE	3187
2	INTERNET + READ_PHONE_STATE + SEND_SMS	3064
3	INTERNET + READ_PHONE_STATE + WRITE_EXTERNAL_STORAGE	2858
4	INTERNET + ACCESS_NETWORK_STATE + WRITE_EXTERNAL_STORAGE	2722
5	SEND_SMS + RECEIVE_SMS + INTERNET	2275

2.2. Quantity of Intent-Filters of BroadcastReceivers

Just like permissions it can be dangerous for the user when malicious apps filter for particular broadcasted intents. To investigate this matter this study also searched in the sample app collection for the quantity of certain intent filters in the malware and the non-malware scope. To narrow the results down to the relevant intents only action intents have been checked.

2.2.1. Prerequisites

This time it was trickier to get the results because the output of this command is not a list with the wanted intents but all the entries of the app's Android Manifest[6] and the challenge was to filter for the right intents. The script was built on the base of a few sample apps and debugged against all malware-apps until no errors occurred. The result was Listing 2.2.

```
1  $market = get-childitem "...path..." -recurse |
2        where{$_.name -like "*.apk"}
3  $hashVals = @{}
4  foreach($file in $market)
5  {
6    # ... get $fileItem ...
7    $AppMF = aapt d xmltree $fileItem AndroidManifest.xml
8
9    # Searching for Action Intents of BroadcastReceiver:
10   foreach($entry in $AppMF)
11   {
12     if($entry -ne $NULL)
13     {
14       $entryTrim = $entry.Trim()
15       while(($entryTrim.Length -ge 11) -and ($entryTrim.
             substring(0,11) -eq "E: receiver"))
16       {
```

[6]The manifest presents essential information about the app to the Android system (4)

```
17          [void]$foreach.moveNext()
18          $entryTrim = $foreach.current.Trim()
19          while(($entryTrim.substring(0,2) -eq "A:") -or (
                $entryTrim.substring(0,2) -eq "C:"))
20          {
21            [void]$foreach.moveNext()
22            if($foreach.current -eq $NULL){break}
23            $entryTrim = $foreach.current.Trim()
24          }
25          while(($entryTrim.Length -ge 16) -and ($entryTrim.
                substring(0,16) -eq "E: intent-filter"))
26          {
27            [void]$foreach.moveNext()
28            $entryTrim = $foreach.current.Trim()
29            while($entryTrim.substring(0,2) -eq "A:")
30            {
31              [void]$foreach.moveNext()
32              if($foreach.current -eq $NULL){break}
33              $entryTrim = $foreach.current.Trim()
34            }
35            while(($entryTrim.Length -ge 9) -and ($entryTrim
                .substring(0,9) -eq "E: action"))
36            {
37              [void]$foreach.moveNext()
38              #FOUND! -> write into table
39              $entryTrim = $foreach.current.Trim()
40              $entryTrim = $entryTrim.substring(29)
41              $intI = $entryTrim.IndexOf("Raw")-3
42              if($intI -ne -1) { $entryTrim = $entryTrim.
                    substring(0,$intI) }
43              if($entryTrim.Length -ge 70) { $entryTrim =
                    $entryTrim.substring(0,70)}
44              $hashVals[$entryTrim]++
45
```

```
46                [void]$foreach.moveNext()
47                if($foreach.current -eq $NULL){break}
48                $entryTrim = $foreach.current.Trim()
49              }
50              while(((($entryTrim.Length -ge 7) -and (
                    $entryTrim.substring(0,7) -eq "E: data")) -or
                    (
51              (($entryTrim.Length -ge 11) -and ($entryTrim.
                    substring(0,11) -eq "E: category")))
52              {
53                [void]$foreach.moveNext()
54                [void]$foreach.moveNext()
55                if($foreach.current -eq $NULL){break}
56                $entryTrim = $foreach.current.Trim()
57  } } } } } }
58  # ... write $hashVals to file ...
```

Listing 2.2: Code snippet to find intents of broadcast receivers

2.2.2. Results

Table 2.5 It is noticeable, that the action intents of broadcast receivers are less used like permissions. The most frequent one, INSTALL_REFERRER[7], is used only in about 12% of the Google Market sample apps. Others, further down of the list like PHONE_STATE[8] are even used only in about 0.8%.

Table 2.6 shows that malware apps also use less intents of broadcast receivers than permissions but the difference is not so big as it is in non-malware apps. The most frequent intent BOOT_COMPLETED[9] is used by almost 32% of the malicious sample apps.

[7]Full package name is com.android.vending.INSTALL_REFERRER
[8]Full package name is android.intent.action.PHONE_STATE
[9]Full package name is android.intent.action.BOOT_COMPLETED

Table 2.5.: Top BroadcastReceiver intents of (25994) Google Market apps

Rank	Action Intent	Quantity
1	INSTALL_REFERRER	3099
2	APPWIDGET_UPDATE	2398
3	BOOT_COMPLETED	2068
4	REGISTRATION	609
4	RECEIVE	609
12	PHONE_STATE	209

Table 2.6.: Top BroadcastReceiver intents of (9968) malware apps

Rank	Action Intent	Quantity
1	BOOT_COMPLETED	3260
2	PHONE_STATE	799
3	SMS_RECEIVED	763
4	INSTALL_REFERRER	690
5	APPWIDGET_UPDATE	624
13	REGISTRATION	123

2.3. Interpretation of the results

Only permissions will be discussed in this chapter. However intents filtered by broadcast receivers have the same outcome when it comes to their interpretation.

One way to look at the results is to compare top ranked malware permissions to the percentage of the same permissions in the list of non-malware apps.

INTERNET for example has rank 1 in both lists and their percentage is almost equal. The only conclusion to make here is that this permission is generally very common and the user doesn't need to worry that it is at the top of the malware list.

SEND_SMS on the other hand is on rank 17 of non-malware and rank 4 of malware apps. As described before its frequency among non-malware apps is only about 4%, but about 40% of malicious apps use this permission. That is 10 times as many and a lot of other permissions show similar or even worse ratios. It should raise the suspicion of the user if such a permission is demanded by an app and it is recommended to check whether this app really needs that permission to achieve its purpose.

The conclusion on this approach is that permissions which have a significant higher percentage in the malware list than in the non-malware list are critical and apps using these permissions to be treated carefully. This can be easily calculated by dividing the percentage of the malware permission by the non-malware permission. The result can be seen as rating and the higher the rating the more likely it is that this permission indicates malicious behavior of its app. One example is presented above with the SEND_SMS permission where 10 is the rating ($\frac{40\%}{4\%} = 10$). Another example is INSTALL_PACKAGES[10] with a percentage of 11.8% among malicious and only 0.54% among non-malicious apps. The rating of this permission is almost 22 and for that even more critical than SEND_SMS.

An other way to look at the result is the combination of permissions (see Table 2.3 and Table 2.4) and take them as a basis to find possible malware apps. This approach is similar to the one for single permissions but instead of looking at

[10]Full package name is android.permission.INSTALL_PACKAGES

perfect matches between the lists and the app it is statistically better, because of the vast amount of possibilities, just to find similarities.

As stated before the same interpretation and calculation of the rating apply for broadcast intents.

2.4. Rating for an app

To give an overall evaluation about an app a rating algorithm that takes the above conclusions in consideration has been developed. The basic idea is to average over the ratings of the single permissions the app uses and the intents its broadcast receivers filter for. To counter the effect that a single permission or intent with a high rating carries no weight between a lot of other permissions or intents with low ratings, the ratings are squared. For better understanding it makes sense to give the user a minimal (0) and a maximal rating (100).

Different ideas of rating algorithms for the purpose to give a high rating for malicious and a low rating for non-malicious apps have been tested on the sample apps. The best result has been achieved by this version:

- PR = permission rating of a permission used by this app
- n = number of the app's rated permissions
- IR = intent rating of an intent filtered by this app's broadcast receivers
- m = number of the app's rated intents
- APR = app's overall permission rating
- AIR = app's overall intent rating
- SR = semi result
- $Rating$ = final result. Rating for this app

$$APR = \sum^{n} \frac{4 * PR^2}{n} \text{ with } n > 0$$

$$AIR = \sum^{m} \frac{4 * IR^2}{m} \text{ with } m > 0$$

$$SR = (APR + AIR) * (\frac{1}{2} \text{ if } APR \wedge AIR \neq 0)$$

$$Rating = min\{100; SR\}$$

The test result of this algorithm on the sample apps is an average rating of about 15 for the Google Market and about 54 for the malware apps. This doesn't mean that all apps with a rating under 15 are non-malicious and above 54 malicious. It gives the user an indication what apps are more likely malware. Especially apps with a rating of 50 and above deserve more detailed investigation.

Only apps with at least one permission or intent filtered by a broadcast receiver are taken into consideration, which are 90% of the Google Market and 95% of the malware apps.

3. The security app ContrOWL

In order to implement the research results and the rating algorithm in a working prototype the security app "ContrOWL" has been developed. It helps the user to better understand permissions and intents filtered by broadcast receivers. It also supports the user to estimate the security and safety risks of all installed apps on the device and gives the opportunity to delete an app when it is considered dangerous.

3.1. Features of ContrOWL

These are the key features of "ContrOWL" and will be discussed subsequently in detail:

- List of top permissions and broadcast intents used by malware
- Ranking, rating and description of these permissions and intents
- Finding apps on the device which use these permissions and intents
- List of all installed apps on the device and their overall rating
- Permissions and intents used by an app
- Possibility to remove apps

The app starts with a splash screen to fill the gap until the app is fully loaded. This is necessary as it takes some time, depending on how many apps are installed on the device, till it fetches all installed apps and calculates their rating.

After the loading process the app shows four tabs with the first one activated per default. This is the "Help tab" containing some basic information about "ContrOWL" and a brief explanation of its key features (see Figure 3.1).

The "Permission tab" (see Figure 3.2a) lists the 50 most used permissions among malicious apps. Three numbers are displayed next to the name of a permission.

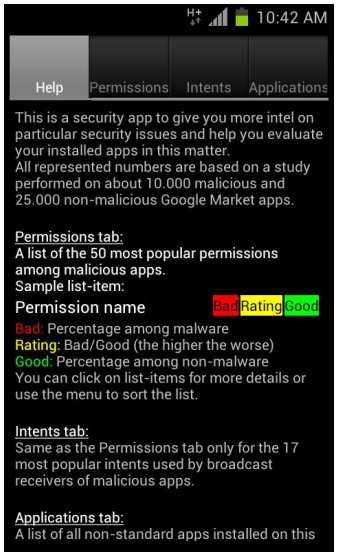
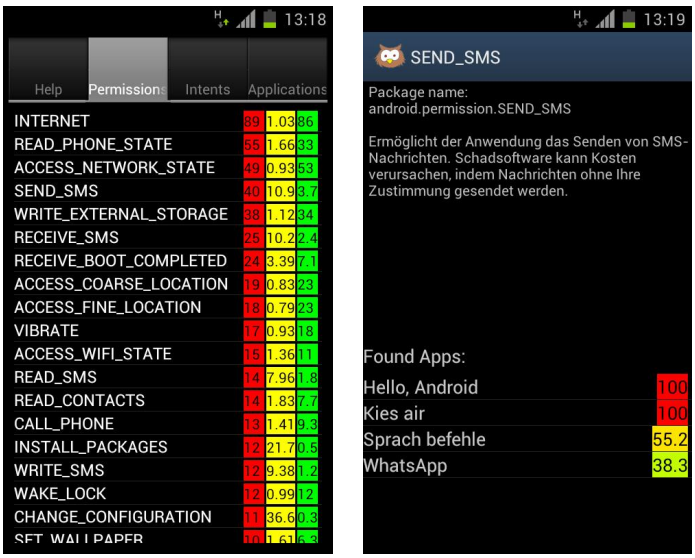


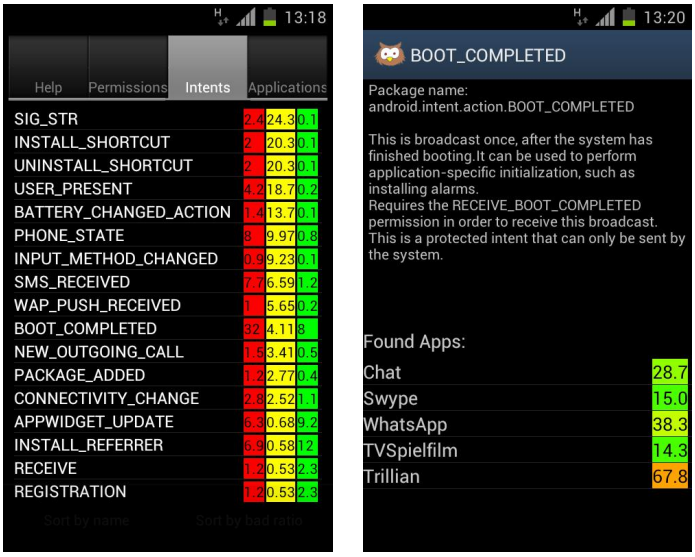
Figure 3.1.: Help tab.

The number on the left side (in the red box) represents the percentage of the corresponding permission among malicious apps. The number on the right side (in the green box) describes the percentage among non-malicious apps. The central number in the yellow box is the calculated rating.



(a) Permissions tab. (b) Permission details.

Figure 3.2.: Permissions tab of ContrOWL



(a) Intents tab. (b) Intent details.

Figure 3.3.: Intents tab of ContrOWL

By using the menu button the user has the possibility to sort this list alphabetically, by the percentage of malware as well as of non-malware, and by the rating. In Figure 3.2a the list is sorted by percentage of malicious apps.

Clicking on a list item leads to its details (see Figure 3.2b). This includes the full package name, a more detailed description, and a list of all installed apps using this permission. In Figure 3.2b this description is in German because it is loaded dynamically from the Android platform and therefore in the language of the device on which "ContrOWL" is installed.

The "Intents tab" (see Figure 3.3a) lists the 17 most used intent filters of broadcast receivers among malicious apps. It is structured in the same way as the "Permissions tab". The possibility to sort the list items is also implemented in the menu button. In Figure 3.3a the list is sorted by the rating.

The user can click on a list item just like in the "Permissions tab" and gets the same kind of information (see Figure 3.3b). This time the description is English since it is static due to the lack of intent descriptions on the Android platform.

The "Applications tab" (see Figure 3.4) is a list of all non standard apps the user has installed on his device. Each list item consists of an app name and its calculated overall rating. The rating is colored by its value from green for 0 to

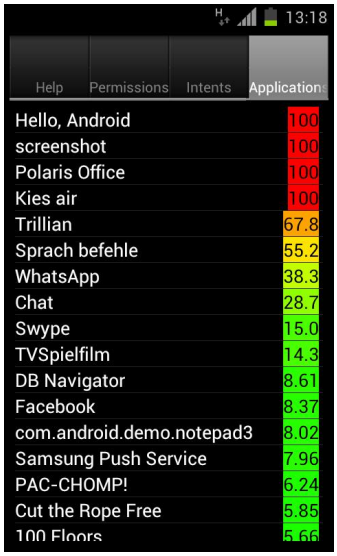
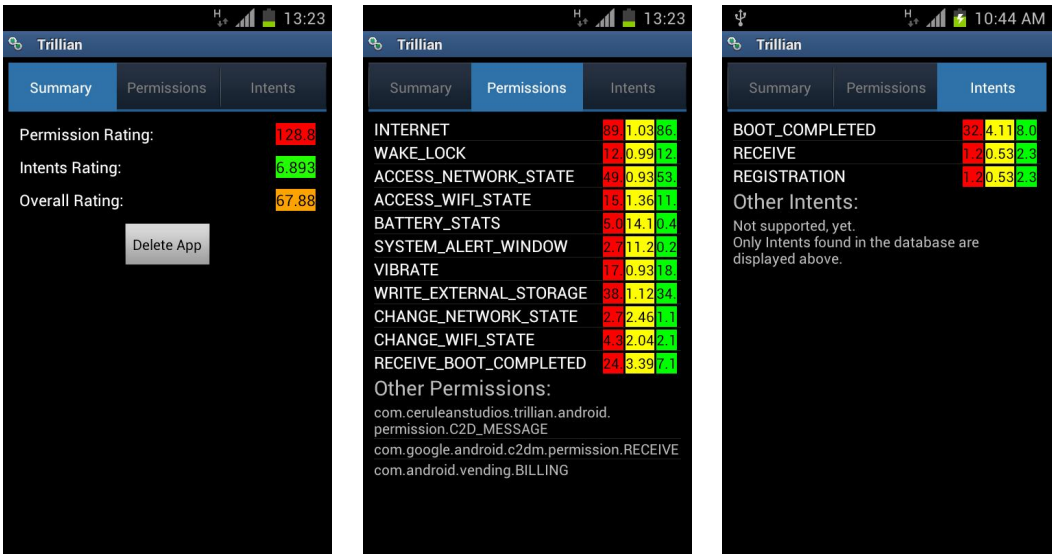


Figure 3.4.: Applications tab.

red for 100. By default the list is sorted descending ordered by the rating of the apps but can also be sorted alphabetically by using the menu button.

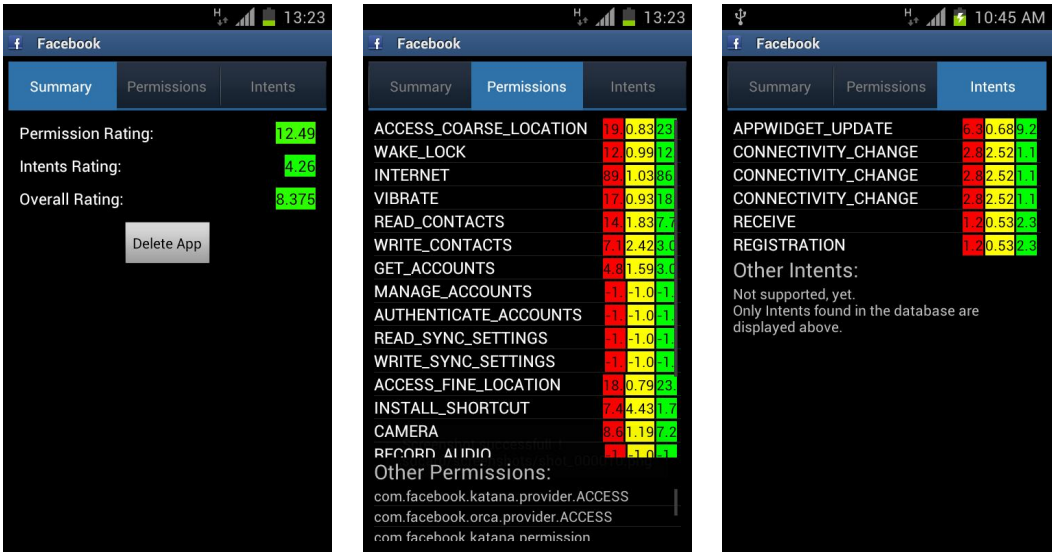
Each list item can by opened to get more details. The details page is subdivided in summary, permissions, and intents (see Figure 3.5 and 3.6). In summary the user can see the rating for this app based only on permissions, intents, or the combined overall rating. The user has the possibility to delete the app at this point if he sees it fit in consideration of its ratings. The permissions section is further divided into normal android permissions in the top and other permissions (e.g. app specific) in the bottom. The intents section has the same structure but the "other intents" part is not filled with data since the Android platform doesn't support a possibility to realize this feature. Figure 3.5 shows details of an app with a high rating. It indicates that this app should be further investigated.

Figure 3.6 on the other hand shows details of a less suspicious app with a lower rating even though it has more permissions. This is because most of them have a low rating and some even no rating at all. Permissions with no rating are marked with "-1". These are permissions which belong to the Android standard but don't belong to the top 50 permissions used by malware. They are used by only 2% and less of malicious apps and therefore not of interest for this study.



(a) Summary details. (b) Permission details. (c) Intent details.

Figure 3.5.: Details of Trillian app



(a) Summary details. (b) Permission details. (c) Intent details.

Figure 3.6.: Details of Facebook app

3.2. Implementation of the features

This chapter describes some of the code developed to realize the features shown in Section 3.1.

When "ContrOWL" is started (see Listing 3.1) it first loads all non standard installed apps (see line 1-5), scans them for their permissions and intents filtered by their broadcast receivers (see line 9+10), and calculates their ratings on base of the scan results (see line 11-15).

```
1  List<PackageInfo> packs = pm.getInstalledPackages(0);
2    for (PackageInfo packInfo : packs) {
3          //Pre-installed System-Apps
4          if ((packInfo.applicationInfo.flags &
               ApplicationInfo.FLAG_SYSTEM) != 0) {
5                continue; }
6          String name = packInfo.applicationInfo.loadLabel(
             pm).toString();
7          String pkgName = packInfo.packageName;
8          float permRating = calculatePermRating(pkgName);
9          float intRating = calculateIntRating(name);
10         float rating = permRating + intRating;
11         if(permRating != 0 && intRating != 0)
12           rating /= 2;
13         if(rating > 100)
14           rating = 100F;
15 ... }
```

Listing 3.1: Gets all non standard apps installed on the device and calculates their ratings

As you can see in Listing 3.1 the method calculatePermRating(String) has been called (see Listing 3.2). This method gets all permissions of an app and calculates a part of the rating based on the found permissions. For demonstration purposes some variable declarations and try catch blocks are left out the code snippet.

```
1  pkgInfo = mCtx.getPackageManager().getPackageInfo(pkgName,
       PackageManager.GET_PERMISSIONS);
2  ...
3  if(pkgInfo != null && pkgInfo.requestedPermissions != null
       ){
4    for(String perInfo : pkgInfo.requestedPermissions){
5      if(perInfo.startsWith("android.permission.") ||
           perInfo.startsWith("com.android.launcher.permission
           .") || perInfo.startsWith("com.android.browser.
           permission.")){
6        Cursor curs = mPermDbHelper.fetchPermByPkgName(
             perInfo);
7              if(curs.getCount() != 0){
8                rating += curs.getFloat(curs.
                   getColumnIndexOrThrow(PermsDbAdapter.
                   KEY_RATING)) * curs.getFloat(curs.
                   getColumnIndexOrThrow(PermsDbAdapter.
                   KEY_RATING));
9                ratingCount++;
10       }
11       curs.close();
12       ...
13 } } }
14 if(ratingCount != 0)
15   rating = 4 * rating / ratingCount;
16 return rating;
17 }
```

Listing 3.2: calculatePermRating(String pkgName)

In line 1-5 all requested permissions of the app with the package name "pkgName" are loaded, checked, and iterated. "mCtx" is the context[1] of the activity[2] instantiating the class from the shown code snippet(s) (AppsDbAdapter.java). In line

[1]Interface to global information about an application environment (5)
[2]An activity is an app component that provides a screen with which users can interact in order to do something (6)

6-8 the found permission and its rating is loaded from "ContrOWL"s database[3]. In line 8-18 all found ratings are used to calculate the overall rating of the app and return it by this method.

The same approach is used to find all apps using a certain permission. All apps are loaded from the database and then checked for their permissions. When there is a match the app is added to the list.

```
1  PackageManager pm = mCtx.getPackageManager();
2  Cursor mIntentsCurs = mIntDbHelper.fetchAllIntents();
3  mIntentsCurs.moveToFirst();
4  do{
5    Intent intent = new Intent(intentPkgName);
6    List<ResolveInfo> activities = pm.
       queryBroadcastReceivers(intent, 0);
7    for (ResolveInfo ri : activities) {
8      if(appName.equals(ri.loadLabel(pm).toString())){
9        //Get rating from current intent and calculate
10     }
11   }
12 }
13 while(mIntentsCurs.moveToNext());
14 //Calculate rest
```

Listing 3.3: Code snippet of how to get intents filterd by the broadcast receiver of an app

It is not possible on Android 4.0 to get intents filtered by the broadcast receiver of an app directly. But it is possible to ask for apps which are registered for certain broadcast intents. This is used in Listing 3.3 and also narrows down the found broadcast intents of an app to those which are saved in the database of "ContrOWL". Lines 2-4 and 13 are used to get all intents saved in the database and iterates them. In lines 5 and 6 all apps are requested which registered a broadcast receiver for the given intent. Afterwards lines 7 and 8 iterate the found apps and search for a match. The comments[4] are standing for the logic

[3]"ContrOWL" has its own SQLite database with a table for permissions, intents, and apps
[4]Green text after // which is not compiled

for the rating calculation which is very similar to Listing 3.2 and therefore left out. In "ContrOWL" Listing 3.3 is divided in two parts due to performance reasons. First all apps for certain broadcast intents are searched and saved in an array list. This list is then used for each app to find its broadcast intents. The performance enhancement from this approach shows that the method queryBroadcastReceivers(intent, int) slows down the app when used frequently.

The description of a permission is saved in the Android platform and can be loaded dynamically (see Listing 3.4). This description is normally in the language of the device.

```
1 PackageManager pm = mCtx.getPackageManager();
2 //pkgName = package name of permission
3 CharSequence csPermissionInfoDescription = pm.
    getPermissionInfo(pkgName, 128).loadDescription(pm);
```

Listing 3.4: Code snippet of how to get the permission description

In order to delete an app "ContrOWL" creates an ACTION_DELETE intent with the URI[5] of the app's package name and starts an activity to do the job (see Listing 3.5). When the app has been successfully deleted the "ContrOWL" database is updated by removing this app.

```
1 //pkgName = package name of app
2 Uri packageURI = Uri.parse("package:" + pkgName);
3 Intent uninstallIntent = new Intent(Intent.ACTION_DELETE,
    packageURI);
4 startActivityForResult(uninstallIntent, 0);
```

Listing 3.5: Code snippet of how to delete an app

[5] A Uniform Resource Identifier that identifies an abstract or physical resource (7)

4. Summary, Conclusions, and Further Work

This Bachelor Thesis examines the permissions and intents of broadcast receivers of malicious and non-malicious Android apps and determines the differences and similarities between them.

The Study: To investigate this matter a study on about 10.000 malware and 25.000 non-malware apps has been performed. The results showed that some permissions and broadcast intents have a much higher relative quantity among malicious apps than among non-malicious apps. Based on this findings a rating algorithm was developed which can be used to determine how likely an app is malware or non-malware. The algorithm has also been tested on the app collection mentioned above to see whether it calculates proper ratings.

The test shows that there is still room for improvement as there are many more possibilities to analyze the sample collection and contribute the results to the rating algorithm. Pairs of two and three permissions have already been investigated in this study but haven't yet been implemented in the algorithm. Other possibilities would be to also analyze pairs of two and three of broadcast intents and combination of permissions and intents. The problem is the vast amount of possible combinations and a pattern matching algorithm would be needed to find a good hit-rate on malicious apps and a low false-positive rate on non-malicious apps.

The rating itself could use some more tweaks as well to get a higher hit-rate on malicious apps and a lower false-positive rate on non-malicious apps. This could be done with some changes on the algorithm and with more input from further studies.

In conclusion this study is based on empirical observations and intuition which means no code analysis on malware and non-malware apps has been performed. On these grounds the results have a pure statistical nature and therefore the rating is an information which indicates the likeliness of malicious behavior. There is

no guarantee that an app with certain permissions and intents is in fact malware or not. The success rate however is for this early state of the algorithm already very satisfying: Average rating of about 15 for Google Market and about 54 for the malware apps.

ContrOWL: In order to implement the research results and the rating algorithm in a working prototype the security app "ContrOWL" has been developed. With this app the user gets all information on permissions and broadcast intents gathered in this study as well as the permissions, intents, and calculated ratings of his apps.

"ContrOWL" has only been tested on the device of the author so far. 34 non-standard apps are installed on this device and none of them are malware. Three have a rating of 100, one of 67.8, and one of 55.2 which makes 5 suspicious apps and a false-positive rate of about 15%. Considering the findings of the study this is an expected outcome.

To give a better insight of how well "ContrOWL" performs it would need a feature which collects usage data from users. These would be the user's apps ratings and the input of the user whether he considers them to be valid or not. These statistics could also be used as advice for other users to make better decisions about their apps.

At the moment the user has to open "ContrOWL" manually and look how his apps are rated. A good feature here would be a background service which prompts the user immediately after the installation of an app about its rating and other details. This would not only improve the likeliness of the use of "ContrOWL" but also the security since a recently installed app has less time to do harm.

In summary the study and the security app "ContrOWL" already provide a good insight on permissions and intents filtered by broadcast receivers. "ContrOWL" also improves the security aspect on devices where it is installed. Nevertheless both the study and the app could be improved as suggested above and provide still room for further work.

5. Acknowledgements

I want to thank Prof. Felix Freiling for making it possible for me to work on this subject and write this Bachelor Thesis.

I want to especially thank Michael Spreitzenbarth for supervising my work, friendly support and professional help.

I also want to thank my girlfriend Kristin Rudolph for coming up with the app name “ContrOWL” and my good friend Hannes Stadler for proofreading.

Bibliography

1 A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android Permissions Demystified," 2011. [Online]. Available: http://www.cs.berkeley.edu/~afelt/android_permissions.pdf

2 mobilesandbox.org, "Android Mobile Sandbox," 2012. [Online]. Available: http://mobilesandbox.org

3 Microsoft, "Windos PowerShell," 2012. [Online]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=7217

4 Android, "Android Manifest File," 2012. [Online]. Available: http://developer.android.com/guide/topics/manifest/manifest-intro.html

5 ——, "Android Context," 2012. [Online]. Available: http://developer.android.com/reference/android/content/Context.html

6 ——, "Android Activity," 2012. [Online]. Available: http://developer.android.com/guide/components/activities.html

7 ——, "Android Uniform Resource Identifier," 2012. [Online]. Available: http://developer.android.com/reference/java/net/URI.html

8 J. Burns, "Mobile Application Security On Android," 2009. [Online]. Available: http://www.blackhat.com/presentations/bh-usa-09/BURNS/BHUSA09-Burns-AndroidSurgery-PAPER.pdf

9 Android, "Android Developers," 2012. [Online]. Available: http://developer.android.com/develop/index.html

10 J. Oberheide, "A look at a modern mobile security model," 2009. [Online]. Available: http://jon.oberheide.org/files/cansecwest09-android.pdf

11 L. P. Software, "How to be safe, avoid viruses, and find trusted apps," 2011. [Online]. Available: http://alostpacket.com/2010/02/20/how-to-be-safe-find-trusted-apps-avoid-viruses/

12 C. Orthacker, P. Teufl, S. Kraxberger, G. Lackner, M. Gissing, A. Marsalek, J. Leibetseder, and O. Prevenhueber, "Android security permissions - can we trust them?" nA. [Online]. Available: https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=57576

13 T. Vidas, N. Christin, and L. F. Cranor, "Curbing android permission creep," nA. [Online]. Available: http://www.andrew.cmu.edu/user/nicolasc/publications/VCC-W2SP11.pdf

14 Android, "Android security overview," 2012. [Online]. Available: http://source.android.com/tech/security/

Appendix

A. First class of appendices

A.1. Complete list of rated intents and permissions

Intents of Broadcast Receiver	**Malware**	**Non-Malware**	
Package	Percentage	Percentage	Rating
android.intent.action.BOOT_COMPLETED	32,7046549	7,95568208	4,110855
android.intent.action.PHONE_STATE	8,01565008	0,8040317	9,969321
android.provider.Telephony.SMS_RECEIVED	7,654494382	1,161806571	6,588441
com.android.vending.INSTALL_REFERRER	6,922150883	11,921982	0,580621
android.appwidget.action.APPWIDGET_UPDATE	6,260032103	9,225205817	0,678579
android.intent.action.USER_PRESENT	4,173354735	0,223128414	18,70382
android.net.conn.CONNECTIVITY_CHANGE	2,768860353	1,100253905	2,516565
android.intent.action.SIG_STR	2,437800963	0,1	24,37801
com.android.launcher.action.INSTALL_SHORTCUT	2,036516854	0,1	20,36517
com.android.launcher.action.UNINSTALL_SHORTCUT	2,036516854	0,1	20,36517
android.intent.action.NEW_OUTGOING_CALL	1,534911717	0,45010387	3,410128
android.intent.action.BATTERY_CHANGED_ACTION	1,374398074	0,1	13,74398
com.google.android.c2dm.intent.RECEIVE	1,233948636	2,34284835	0,526687
com.google.android.c2dm.intent.REGISTRATION	1,233948636	2,34284835	0,526687
android.intent.action.PACKAGE_ADDED	1,21388443	0,438562745	2,767869
android.provider.Telephony.WAP_PUSH_RECEIVED	1,043338684	0,184657998	5,650114
android.intent.action.INPUT_METHOD_CHANGED	0,922953451	0,1	9,229535

Figure A.1.: The complete list of all rated intents filtered by broadcast receiver

Permissions:	**Malware**	**Non-Malware**	
Package	Percentage	Percentage	Rating
android.permission.INTERNET	88,55337079	85,68131107	1,03352
android.permission.READ_PHONE_STATE	55,32704655	33,28845118	1,662049
android.permission.ACCESS_NETWORK_STATE	49,2576244	52,78525814	0,93317
android.permission.SEND_SMS	40,22873194	3,681618835	10,92691
android.permission.WRITE_EXTERNAL_STORAGE	38,47311396	34,37716396	1,119147
android.permission.RECEIVE_SMS	25	2,431330307	10,28244
android.permission.RECEIVE_BOOT_COMPLETED	23,97672552	7,070862507	3,390919
android.permission.ACCESS_COARSE_LOCATION	19,26163724	23,32846041	0,825671
android.permission.ACCESS_FINE_LOCATION	18,03772071	22,63599292	0,79686
android.permission.VIBRATE	17,09470305	18,28498884	0,934904
android.permission.ACCESS_WIFI_STATE	14,90770465	10,9332923	1,363515
android.permission.READ_CONTACTS	14,17536116	7,732553666	1,833206
android.permission.READ_SMS	13,93459069	1,750403939	7,960786
android.permission.CALL_PHONE	13,04173355	9,252135108	1,409592
android.permission.WAKE_LOCK	12,01845907	12,02585212	0,999385
android.permission.INSTALL_PACKAGES	11,7776886	0,542432869	21,71271
android.permission.WRITE_SMS	11,50682183	1,227206278	9,376437
android.permission.CHANGE_CONFIGURATION	10,98515249	0,300069247	36,60872
android.permission.SET_WALLPAPER	10,10232745	6,270677849	1,611042
android.permission.CAMERA	8,617576244	7,22089713	1,193422
com.android.launcher.permission.INSTALL_SHORTCUT	7,403691814	1,669616065	4,434368
android.permission.WRITE_CONTACTS	7,142857143	2,954527968	2,417597
android.permission.GET_TASKS	6,611155698	5,293529276	1,248913
android.permission.READ_LOGS	5,617977528	1,86966223	3,004809
com.android.browser.permission.READ_HISTORY_BOOKMARKS	5,607945425	0,530891744	10,56326
android.permission.WRITE_SETTINGS	5,447431782	3,843194583	1,417423
android.permission.BATTERY_STATS	5,006019262	0,353927829	14,14418
android.permission.GET_ACCOUNTS	4,785313002	3,004539509	1,592694
android.permission.WRITE_APN_SETTINGS	4,725120385	0,242363622	19,496
android.permission.RECEIVE_WAP_PUSH	4,624799358	0,1	46,24799
android.permission.EXPAND_STATUS_BAR	4,293739968	0,169269831	25,36624
android.permission.CHANGE_WIFI_STATE	4,273675762	2,096637686	2,038347
android.permission.RESTART_PACKAGES	4,163322632	2,173578518	1,915423
android.permission.MODIFY_AUDIO_SETTINGS	3,962680578	1,750403939	2,263866
android.permission.DELETE_PACKAGES	3,832263242	0,184657998	20,7533
com.android.launcher.permission.UNINSTALL_SHORTCUT	3,782102729	0,469339078	8,058359
android.permission.WRITE_CALENDAR	3,752006421	2,242825267	1,672893
android.permission.READ_CALENDAR	3,691813804	2,400553974	1,537901
android.permission.MOUNT_UNMOUNT_FILESYSTEMS	3,551364366	2,146649227	1,654376
android.permission.BLUETOOTH	3,360754414	0,557821036	6,02479
com.android.browser.permission.WRITE_HISTORY_BOOKMARKS	2,849117175	0,234669539	12,14098
android.permission.CHANGE_NETWORK_STATE	2,738764045	1,11179503	2,463371
android.permission.RECEIVE_MMS	2,738764045	0,253904747	10,78658
android.permission.SYSTEM_ALERT_WINDOW	2,718699839	0,242363622	11,21744
com.android.launcher.permission.READ_SETTINGS	2,42776886	0,126952374	19,12346
android.permission.STATUS_BAR	2,407704655	0,323151496	7,450699
android.permission.CONTROL_LOCATION_UPDATES	2,357544141	0,250057706	9,428
android.permission.READ_FRAME_BUFFER	2,237158909	0,1	22,37159
com.android.launcher.permission.WRITE_SETTINGS	2,227126806	0,1	22,27127
android.permission.SET_WALLPAPER_HINTS	2,106741573	0,153881665	13,69066

Figure A.2.: The complete list of all rated permissions

Eidesstattliche Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Ich bin ferner damit einverstanden, dass meine Arbeit zum Zwecke eines Plagiatsabgleichs in elektronischer Form anonymisiert versendet und gespeichert werden kann. Mir ist bekannt, dass von der Korrektur der Arbeit abgesehen werden kann, wenn die Erklärung nicht erteilt wird.

Ort, Datum

Marcel Hrnecek