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 .................................................. v

List of Tables .................................................... vi

1. Introduction ................................................... 1

2. Study on Quantity of Permissions and Intent-Filters of BroadcastReceivers ................................. 4
   2.1. Quantity of Permissions ..................................... 4
       2.1.1. Prerequisites ........................................... 4
       2.1.2. Results ................................................. 6
   2.2. Quantity of Intent-Filters of BroadcastReceivers .................................................. 8
       2.2.1. Prerequisites ........................................... 8
       2.2.2. Results ................................................. 10
   2.3. Interpretation of the results .................................. 12
   2.4. Rating for an app ........................................... 13

3. The security app ContrOWL .................................... 15
   3.1. Features of ContrOWL ....................................... 15
   3.2. Implementation of the features ............................... 20

4. Summary, Conclusions, and Further Work ....................... 24

5. Acknowledgements ............................................. 26

Bibliography .................................................... 27
Appendix

A. First class of appendices
   A.1. Complete list of rated intents and permissions
List of Figures

3.1. Help tab of ContrOWL ........................................ 16
3.2. Permissions tab of ContrOWL ............................... 16
3.3. Intents tab of ContrOWL ..................................... 17
3.4. Applications tab of ContrOWL .............................. 18
3.5. Details of Trillian app ....................................... 19
3.6. Details of Facebook app ..................................... 19

A.1. Complete list of rated intents ......................... 30
A.2. Complete list of rated permissions .................. 31
List of Tables

2.1. Top permissions of (25994) Google Market apps . . . . . . . . . 6
2.2. Top permissions of (9968) malware apps . . . . . . . . . . . . . 7
2.3. Top permission pairs of two in (9968) malware apps . . . . . . . 7
2.4. Top permission pairs of three in (9968) malware apps . . . . . . 7
2.5. Top BroadcastReceiver intents of (25994) Google Market apps . . 11
2.6. Top BroadcastReceiver intents of (9968) malware apps . . . . . . . 11


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

\(^{1}\text{Full package name is android.provider.Telephony.SMS RECEIVED}\)
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.

```bash
$market = get−childitem "...path..." −recurse |
  where{$_.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
2.1. Quantity of Permissions

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

<table>
<thead>
<tr>
<th>Rank</th>
<th>Permission</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTERNET</td>
<td>22272</td>
</tr>
<tr>
<td>2</td>
<td>ACCESS_NETWORK_STATE</td>
<td>13721</td>
</tr>
<tr>
<td>3</td>
<td>WRITE_EXTERNAL_STORAGE</td>
<td>8936</td>
</tr>
<tr>
<td>4</td>
<td>READ_PHONE_STATE</td>
<td>8653</td>
</tr>
<tr>
<td>5</td>
<td>ACCESS_COARSE_LOCATION</td>
<td>6064</td>
</tr>
<tr>
<td>17</td>
<td>SEND_SMS</td>
<td>957</td>
</tr>
</tbody>
</table>

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 pairs of three permissions often found in the sample of malware apps.

\(^4\)Full package name is android.permissionINTERNET

\(^5\)Full package name is android.permissionSEND_SMS
### 2.1. Quantity of Permissions

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

<table>
<thead>
<tr>
<th>Rank</th>
<th>Permission</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTERNET</td>
<td>8827</td>
</tr>
<tr>
<td>2</td>
<td>READ_PHONE_STATE</td>
<td>5515</td>
</tr>
<tr>
<td>3</td>
<td>ACCESS_NETWORK_STATE</td>
<td>4910</td>
</tr>
<tr>
<td>4</td>
<td>SEND_SMS</td>
<td>4010</td>
</tr>
<tr>
<td>5</td>
<td>WRITE_EXTERNAL_STORAGE</td>
<td>3835</td>
</tr>
<tr>
<td>11</td>
<td>ACCESS_COARSE_LOCATION</td>
<td>1920</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Rank</th>
<th>Permission pairs</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTERNET + READ_PHONE_STATE</td>
<td>5423</td>
</tr>
<tr>
<td>2</td>
<td>INTERNET + ACCESS_NETWORK_STATE</td>
<td>4875</td>
</tr>
<tr>
<td>3</td>
<td>INTERNET + SEND_SMS</td>
<td>3729</td>
</tr>
<tr>
<td>4</td>
<td>INTERNET + WRITE_EXTERNAL_STORAGE</td>
<td>3721</td>
</tr>
<tr>
<td>5</td>
<td>ACCESS_NETWORK_STATE + READ_PHONE_STATE</td>
<td>3263</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Rank</th>
<th>Permission triple</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTERNET + READ_PHONE_STATE + ACCESS_NETWORK_STATE</td>
<td>3187</td>
</tr>
<tr>
<td>2</td>
<td>INTERNET + READ_PHONE_STATE + SEND_SMS</td>
<td>3064</td>
</tr>
<tr>
<td>3</td>
<td>INTERNET + READ_PHONE_STATE + WRITE_EXTERNAL_STORAGE</td>
<td>2858</td>
</tr>
<tr>
<td>4</td>
<td>INTERNET + ACCESS_NETWORK_STATE + WRITE_EXTERNAL_STORAGE</td>
<td>2722</td>
</tr>
<tr>
<td>5</td>
<td>SEND_SMS + RECEIVE_SMS + INTERNET</td>
<td>2275</td>
</tr>
</tbody>
</table>


2.2. Quantity of Intent-Filters of Broadcast Receivers

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 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.

```
$market = get-childitem "...path..." -recurse |
    where{$_.name -like "*.apk"}
$hashVals = @{}
foreach($file in $market)
{
    # ... get $fileItem ...
    $AppMF = aapt d xmltree $fileItem AndroidManifest.xml
    # Searching for Action Intents of BroadcastReceiver:
    foreach($entry in $AppMF)
    {
        if($entry -ne $NULL)
        {
            $entryTrim = $entry.Trim()
            while((($entryTrim.Length -ge 11) -and ($entryTrim.substring(0,11) -eq "E:receiver"))
            {
```

\[\text{6\}The manifest presents essential information about the app to the Android system (4)\]
2.2. Quantity of Intent-Filters of BroadcastReceivers

```java
[ void ] $foreach . moveNext ( )
$entryTrim = $foreach . current . Trim ( )
while ( ( $entryTrim . substring ( 0 , 2 ) − eq "A:" ) − or ( $entryTrim . substring ( 0 , 2 ) − eq "C:" ) )
{
    [ void ] $foreach . moveNext ( )
    if ( $foreach . current − eq $NULL ){ break }$entryTrim = $foreach . current . Trim ( )
}
while ( ( $entryTrim . Length − ge 16 ) − and ( $entryTrim . substring ( 0 , 16 ) − eq "E: intent-filter" ) )
{
    [ void ] $foreach . moveNext ( )
    $entryTrim = $foreach . current . Trim ( )
    while ( ( $entryTrim . substring ( 0 , 2 ) − eq "A:" ) )
    {
        [ void ] $foreach . moveNext ( )
        if ( $foreach . current − eq $NULL ) { break }$entryTrim = $foreach . current . Trim ( )
    }
    while ( ( $entryTrim . Length − ge 9 ) − and ( $entryTrim . substring ( 0 , 9 ) − eq "E: action" ) )
    {
        [ void ] $foreach . moveNext ( )
        # FOUND! -> write into table
        $entryTrim = $foreach . current . Trim ( )
        $entryTrim = $entryTrim . substring ( 29 )
        $intI = $entryTrim . IndexOf ( "Raw" ) − 3
        if ( $intI − ne − 1 ) { $entryTrim = $entryTrim . substring ( 0 , $intI ) }
        if ( $entryTrim . Length − ge 70 ) { $entryTrim = $entryTrim . substring ( 0 , 70 ) }
        $hashVals [ $entryTrim ] ++
```
2.2 Quantity of Intent-Filters of Broadcast Receivers

Listing 2.2: Code snippet to find intents of broadcast receivers

```java
[ void ] $foreach . moveNext ()
if ($foreach . current -eq $NULL){break}
$entryTrim = $foreach . current . Trim ()
}
while ( ( ($entryTrim . Length -ge 7 ) -and ( $entryTrim . substring ( 0 , 7 ) -eq "E: data") ) -or

(( $entryTrim . Length -ge 11 ) -and ($entryTrim . substring ( 0 , 11 ) -eq "E: category") ))
{
[ void ] $foreach . moveNext ()
[ void ] $foreach . moveNext ()
if ($foreach . current -eq $NULL){break}
$entryTrim = $foreach . current . Trim ()
}
# ... .write $hashVals to file ... 
```

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, is used only in about 12% of the Google Market sample apps. Others, further down of the list like PHONE_STATE 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 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

<table>
<thead>
<tr>
<th>Rank</th>
<th>Action Intent</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INSTALL_REFERRER</td>
<td>3099</td>
</tr>
<tr>
<td>2</td>
<td>APPWIDGET_UPDATE</td>
<td>2398</td>
</tr>
<tr>
<td>3</td>
<td>BOOT_COMPLETED</td>
<td>2068</td>
</tr>
<tr>
<td>4</td>
<td>REGISTRATION</td>
<td>609</td>
</tr>
<tr>
<td>4</td>
<td>RECEIVE</td>
<td>609</td>
</tr>
<tr>
<td>12</td>
<td>PHONE_STATE</td>
<td>209</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Rank</th>
<th>Action Intent</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BOOT_COMPLETED</td>
<td>3260</td>
</tr>
<tr>
<td>2</td>
<td>PHONE_STATE</td>
<td>799</td>
</tr>
<tr>
<td>3</td>
<td>SMS_RECEIVED</td>
<td>763</td>
</tr>
<tr>
<td>4</td>
<td>INSTALL_REFERRER</td>
<td>690</td>
</tr>
<tr>
<td>5</td>
<td>APPWIDGET_UPDATE</td>
<td>624</td>
</tr>
<tr>
<td>13</td>
<td>REGISTRATION</td>
<td>123</td>
</tr>
</tbody>
</table>
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
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

\[ APR = \sum_{n}^{} \frac{4 \times PR^2}{n} \quad \text{with } n > 0 \]

\[ AIR = \sum_{m}^{} \frac{4 \times IR^2}{m} \quad \text{with } m > 0 \]
2.4. Rating for an app

\[ SR = (APR + AIR) \times \left( \frac{1}{2} \text{ if } APR \land AIR \neq 0 \right) \]

\[ Rating = \min\{100; SR\} \]

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.
3.1. Features of ContrOWL

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
3.1. Features 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
3.1. Features of ContrOWL

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.
3.1. Features of ContrOWL

<table>
<thead>
<tr>
<th>Summary</th>
<th>Permissions</th>
<th>Intents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating:</td>
<td>89.8</td>
<td>4.4</td>
</tr>
<tr>
<td>Rating:</td>
<td>9.10</td>
<td></td>
</tr>
<tr>
<td>Rating:</td>
<td>87.68</td>
<td></td>
</tr>
</tbody>
</table>

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

Figure 3.5.: Details of Trillian app

<table>
<thead>
<tr>
<th>Summary</th>
<th>Permissions</th>
<th>Intents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating:</td>
<td>72.2</td>
<td>6.20</td>
</tr>
<tr>
<td>Rating:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rating:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(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).

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.
3.2. Implementation of the features

```java
    pkgInfo = mCtx.getPackageManager().getPackageInfo(pkgName, PackageManager.GET_PERMISSIONS);
    ...
    if (pkgInfo != null && pkgInfo.requestedPermissions != null) {
        for (String perInfo : pkgInfo.requestedPermissions) {
            if (perInfo.startsWith("android.permission.")) ||
                perInfo.startsWith("com.android.launcher.permission.") || perInfo.startsWith("com.android.browser.permission.")) {
                Cursor curs = mPermDbHelper.fetchPermByPkgName(perInfo);
                if (curs.getCount() != 0) {
                    rating += curs.getFloat(curs.getColumnIndexOrThrow(PermsDbAdapter.KEY_RATING)) * curs.getFloat(curs.getColumnIndexOrThrow(PermsDbAdapter.KEY_RATING));
                    ratingCount++;
                }
            }
        }
        if (ratingCount != 0)
            rating = 4 * rating / ratingCount;
    } return rating;
```

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)
3.2. Implementation of the features

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.

```java
PackageManager pm = mCtx.getPackageManager();
Cursor mIntentsCurs = mIntDbHelper.fetchAllIntents();
mIntentsCurs.moveToFirst();
do{
    Intent intent = new Intent(intentPkgName);
    List<ResolveInfo> activities = pm.
        queryBroadcastReceivers(intent, 0);
    for (ResolveInfo ri : activities) {
        if (appName.equals(ri.loadLabel(pm).toString())){
            //Get rating from current intent and calculate
        }
    }
} while (mIntentsCurs.moveToNext());
//Calculate rest
```

Listing 3.3: Code snippet of how to get intents filtered 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
3.2. Implementation of the features

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 queryBroadcast Receivers(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 Package Manager 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 URI5 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


Appendix
A. First class of appendices

A.1. Complete list of rated intents and permissions

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

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

<table>
<thead>
<tr>
<th>Permissions:</th>
<th>Malware</th>
<th>Non-Malware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Percentage</td>
<td>Percentage</td>
</tr>
<tr>
<td>android.permission INTERNET</td>
<td>88,55337079</td>
<td>85,6831107</td>
</tr>
<tr>
<td>android.permission.READ_PHONE_STATE</td>
<td>55,32704655</td>
<td>33,28845118</td>
</tr>
<tr>
<td>android.permission.ACCESS_NETWORK_STATE</td>
<td>49,2576244</td>
<td>52,7852814</td>
</tr>
<tr>
<td>android.permission.SEND_SMS</td>
<td>40,22873194</td>
<td>3,681618835</td>
</tr>
<tr>
<td>android.permission.WRITE_EXTERNAL_STORAGE</td>
<td>38,47311396</td>
<td>34,37716396</td>
</tr>
<tr>
<td>android.permission.RECEIVE_SMS</td>
<td>25</td>
<td>2,431330307</td>
</tr>
<tr>
<td>android.permission.RECEIVE_BOOT_COMPLETED</td>
<td>23,97672552</td>
<td>7,070862507</td>
</tr>
<tr>
<td>android.permission.ACCESS_COARSE_LOCATION</td>
<td>19,26163724</td>
<td>23,32846041</td>
</tr>
<tr>
<td>android.permission.ACCESS_FINE_LOCATION</td>
<td>18,03772071</td>
<td>22,63599292</td>
</tr>
<tr>
<td>android.permission.VIBRATE</td>
<td>17,09470305</td>
<td>18,28498884</td>
</tr>
<tr>
<td>android.permission.ACCESS_WIFI_STATE</td>
<td>14,90770465</td>
<td>10,9332923</td>
</tr>
<tr>
<td>android.permission.READ_CONTACTS</td>
<td>14,17536116</td>
<td>7,732553666</td>
</tr>
<tr>
<td>android.permission.READ_SMS</td>
<td>13,93459069</td>
<td>1,750403939</td>
</tr>
<tr>
<td>android.permission.CALL_PHONE</td>
<td>13,04173355</td>
<td>9,252135108</td>
</tr>
<tr>
<td>android.permission.WAKE_LOCK</td>
<td>12,01845907</td>
<td>12,02585212</td>
</tr>
<tr>
<td>android.permission.INSTALL_PACKAGES</td>
<td>11,777688</td>
<td>0,542432869</td>
</tr>
<tr>
<td>android.permission.WRITE_SMS</td>
<td>11,50682183</td>
<td>1,22706278</td>
</tr>
<tr>
<td>android.permission.CHANGE_CONFIGURATION</td>
<td>10,98512549</td>
<td>0,300069247</td>
</tr>
<tr>
<td>android.permission.SET_WALLPAPER</td>
<td>10,10232745</td>
<td>6,270677849</td>
</tr>
<tr>
<td>android.permission.CAMERA</td>
<td>8,617576244</td>
<td>7,22089713</td>
</tr>
<tr>
<td>com.android.launcher.permission.INSTALL_SHORTCUT</td>
<td>7,40391814</td>
<td>1,669610606</td>
</tr>
<tr>
<td>android.permission.WRITE_CONTACTS</td>
<td>7,142857143</td>
<td>2,954527968</td>
</tr>
<tr>
<td>android.permission.GET_TASKS</td>
<td>6,611155698</td>
<td>5,293529276</td>
</tr>
<tr>
<td>android.permission.READ_LOGS</td>
<td>5,67977528</td>
<td>1,86966223</td>
</tr>
<tr>
<td>com.android.browser.permission.READ_HISTORY_BOOKMARKS</td>
<td>5,807945425</td>
<td>0,530891744</td>
</tr>
<tr>
<td>android.permission.WRITE_SETTINGS</td>
<td>5,447431782</td>
<td>3,843194853</td>
</tr>
<tr>
<td>android.permission.BATTERY_STATS</td>
<td>5,06019262</td>
<td>0,33972829</td>
</tr>
<tr>
<td>android.permission.GET_ACCOUNTS</td>
<td>4,78331002</td>
<td>3,004539509</td>
</tr>
<tr>
<td>android.permission.WRITE_APN_SETTINGS</td>
<td>4,725120385</td>
<td>0,242363622</td>
</tr>
<tr>
<td>android.permission.RECEIVE_WAP_PUSH</td>
<td>4,624799358</td>
<td>0,1</td>
</tr>
<tr>
<td>android.permission.EXPAND_STATUS_BAR</td>
<td>4,29379968</td>
<td>0,16926831</td>
</tr>
<tr>
<td>android.permission.CHANGE_WIFI_STATE</td>
<td>4,273675762</td>
<td>2,09662623</td>
</tr>
<tr>
<td>android.permission.RESTART_PACKAGES</td>
<td>4,163322632</td>
<td>2,175785181</td>
</tr>
<tr>
<td>android.permission.MODIFY_AUDIO_SETTINGS</td>
<td>3,962680578</td>
<td>1,750403939</td>
</tr>
<tr>
<td>android.permission.DELETE_PACKAGES</td>
<td>3,832263242</td>
<td>0,184657998</td>
</tr>
<tr>
<td>android.permission.UNINSTALL_SHORTCUT</td>
<td>3,782102729</td>
<td>0,409339078</td>
</tr>
<tr>
<td>android.permission.WRITE캘리</td>
<td>3,750062421</td>
<td>2,242825267</td>
</tr>
<tr>
<td>android.permission.READCALENDAR</td>
<td>3,69183044</td>
<td>2,400539974</td>
</tr>
<tr>
<td>android.permission.MOUNT_UNMOUNT_FILESYSTEMS</td>
<td>3,551364366</td>
<td>2,146649227</td>
</tr>
<tr>
<td>android.permission.BLUETOOTH</td>
<td>3,360754414</td>
<td>0,557821036</td>
</tr>
<tr>
<td>com.android.browser.permission.WRITE_HISTORY_BOOKMARKS</td>
<td>2,849117175</td>
<td>0,234669539</td>
</tr>
<tr>
<td>android.permission.CHANGE_NETWORK_STATE</td>
<td>2,738764045</td>
<td>1,1179503</td>
</tr>
<tr>
<td>android.permission.ACCESS_LOCATION_STATE</td>
<td>2,738764045</td>
<td>0,235904747</td>
</tr>
<tr>
<td>android.permission.SYSTEM_ALERT_WINDOW</td>
<td>2,718699839</td>
<td>0,242363622</td>
</tr>
<tr>
<td>com.android.launcher.permission.READ_SETTINGS</td>
<td>2,427768862</td>
<td>0,126951874</td>
</tr>
<tr>
<td>android.permission.STATUS_BAR</td>
<td>2,407704655</td>
<td>0,323151496</td>
</tr>
<tr>
<td>android.permission.CONTROL_LOCATION_UPDATES</td>
<td>2,35754414</td>
<td>0,250057706</td>
</tr>
<tr>
<td>android.permission.READ_FRAME_BUFFER</td>
<td>2,237158909</td>
<td>0,1</td>
</tr>
<tr>
<td>com.android.launcher.permission.WRITE_SETTINGS</td>
<td>2,227126806</td>
<td>0,1</td>
</tr>
<tr>
<td>android.permission.SET_WALLPAPER_HINTS</td>
<td>2,106741573</td>
<td>0,153881665</td>
</tr>
</tbody>
</table>

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