Mining Apps for Anomalies

Andreas Zeller
Saarland University, Saarbrücken, Germany

Joint work with Alessandra Gorla, Ilaria Tavecchia, Vitalii Avdiienko, Konstantin Kuznetsov, Eric Bodden, Steven Arzt, Siegfried Rasthofer, Isabelle Rommelfanger, and Florian Gross
Saarbrücken
Saarbrücken

1700 BSc + MSc students
375 PhD students
200 Researchers (post PhD)
8 New buildings since 2001

10 ERC Grant holders
6 Leibniz Awardees
4 ACM Fellows
1 Software Engineer
Specifications

removeChild

\[ \text{XML} \text{Element} \]

\( \text{child}?: \text{XML\_ELEMENT} \)

\( \text{child}?: \in \text{enumerateChildren} \)

\( \text{child}?: \neq \text{null} \)

\( \text{enumerateChildren}' = \text{enumerateChildren} \setminus \text{child}? \)

\( \text{getChildrenCount}' = \text{getChildrenCount} - 1 \)
Specifications

public class XMLElement implements IXMLElement, Serializable {
    // The name.
    private String name;

    // The child elements.
    private Vector children;

    // Returns an enumeration of all child elements.
    public Enumeration enumerateChildren() {
        // more methods and attributes...
    }

    // Returns the number of children.
    public int getChildrenCount() {
        // more methods and attributes...
    }

    // Removes a child element.
    public void removeChild(XMLElement child) {
        // more methods and attributes...
    }
}

Figure 1: The XMLElement class from the NanoXML parser

This is precisely what our proposed approach produces:

Given a program, we automatically produce a high-level specification.

In the Z specification language, the mined specification for `removeChild()` is shown in Figure:

```
XMLElement child
XMLElement child ⇔ enumerateChildren(child)
getChildrenCount = getChildrenCount - 1
```

Figure 2: Mined specification for `removeChild` as set forth in this proposal

Note how the specification captures two important preconditions not stated in the documentation—

- `child` be a child of the target node,
- `child` be non-null.

Both properties are essential for generating test cases for instance. The postconditions precisely describe the effect of `removeChild()` and could be used as test oracles or as a base for program synthesis.

1d.3 State of the Art

1d.3.1 Static Analysis

How does one obtain a specification like this? Static analysis takes the program code and infers properties. The `removeChild()` code indeed reveals some insights:

From this code, any static analysis can easily deduce precondition:

```
child ⇔ null
```

But how would...
Specifying Correctness

This is precisely what our proposed approach produces: Given a program, we automatically produce a high-level specification. In the Z specification language, the mined specification for `removeChild` is shown in Figure:

```
removeChild
\DeltaXMLElement
child? : XML_ELEMENT

child? ∈ enumerateChildren
class? ≠ null
enumerateChildren' = enumerateChildren \ child?
getChildrenCount' = getChildrenCount - 1
```

Note how the specification captures two important preconditions not stated in the documentation—that `child` be a child of the target node and that `child` be non-null. Both properties are essential for generating test cases. The postconditions precisely describe the effect of `removeChild` and could be used as test oracles or as a base for program synthesis.
Unknown error

Was this information helpful?
Normality
Mining Normality

public class XMLElement implements IXMLElement, Serializable {
    // The name.
    private String name;

    // The child elements.
    private Vector children;

    // Returns an enumeration of all child elements.
    public Enumeration enumerateChildren() {
        // more methods and attributes...
    }

    // Returns the number of children.
    public int getChildrenCount() {
        // more methods and attributes...
    }

    // Removes a child element.
    public void removeChild(XMLElement child) {
        // more methods and attributes...
    }
}

Figure 1: The XMLElement class from the NanoXML parser

This is precisely what our proposed approach produces:

Given a program, we automatically produce a high-level specification. In the Z specification language, the mined specification for `removeChild()` is shown in Figure:

```z
removeChild
    XMLElement child
      ⇔ enumerateChildren
      ⇔ child = null
    enumerateChildren = enumerateChildren \ child
    getChildrenCount = getChildrenCount - 1
```

Figure 2: Mined specification for `removeChild` as set forth in this proposal

Note how the specification captures two important preconditions not stated in the documentation—

- that `child` be a child of the target node
- that `child` be non-null

Both properties are essential for generating test cases. The postconditions precisely describe the effect of `removeChild()` and could be used as test oracles or as a base for program synthesis.

1d.3 State of the Art

1d.3.1 Static Analysis

How does one obtain a specification like this? Static analysis takes the program code and infers properties. The `removeChild()` code indeed reveals some insights:

From this code, any static analysis can easily deduce precondition:

- `child = null`

But how would (a) Executable Program (b) Specification (c) Test...
Outliers
London Restaurants

Looking for a restaurant, a bar, a pub or just to have fun in London? Search no more! This application has all the information you need:

• You can search for every type of food you want: french, british, chinese, indian etc.
• You can use it if you are in a car, on a bicycle or walking
• You can view all objectives on the map
• You can search objectives
• You can view objectives near you
• You can view directions (visual route, distance and duration)
• You can use it with Street View
• You can use it with Navigation

Keywords: london, restaurants, bars, pubs, food, breakfast, lunch, dinner, meal, eat, supper, street view, navigation

Also sends out account info
Also sends out mobile phone number
Also sends out your device ID
What is malicious?

Also sends out **account info**
Also sends out **mobile phone number**
Also sends out **your device ID**
What is normal?

- “London Restaurants” is a “travel” app
- For “travel” apps, sending account info is *abnormal*
- For “messaging” apps, this is far more likely
1. INTRODUCTION

How do we know a program does what it claims to do? After clustering with respect to their API usage, we identify outliers that use a specific pattern or not; it is uncommon for that cluster.

In all the examples above, the user would be informed and asked for authorization before any questionable behavior. It is the advertised behavior that is questionable or downright malicious.

The question thus is not whether the behavior of an app matches the advertised. As a proxy for the advertised behavior of an app, we use its natural language description from the Google Play Store. As a proxy for the implemented behavior, we use the set of Android application programming interfaces (APIs) that are used from within the app binary. The key idea is to associate descriptions and APIs used, grouped by related per-

ABSTRACT

Specifically, our approach takes five steps, illustrated in Figure 1 and detailed later in the paper:

1. App collection
2. Topics
3. Clusters
4. APIs
5. Outliers

In this paper, we attempt to check known malware patterns.

Rhythm pattern in Jazz.
1. INTRODUCTION

2. Topics

3. Clusters

4. APIs

5. Outliers
In this paper, we attempt to check whether a program does what it claims to do. After clustering applications with unadvertised behavior, we identify several anomalies: “This ‘weather’ application accesses the current location. Applied on a set of 22,500+ Android applications, our prototype identified several anomalies; additionally, it flagged 56% of novel malware as such, without requiring any known malware patterns.

Starting from a collection of “good” apps, we identify their description topics to form clusters of related apps. For each cluster, we identify the APIs used, grouped by related permissions. Specifically, our approach takes five steps, illustrated in Figure 1:

1. **App collection**: Collect a set of “good” apps from the Google Play Store.
2. **Topics**: Identify the topics associated with each app based on its description.
3. **Clusters**: Group the apps into clusters based on their topics.
4. **APIs**: Identify the APIs used by each cluster.
5. **Outliers**: Identify any outliers that use APIs in ways that are questionable or downright malicious.

The question thus is not whether the behavior of an app matches its advertised. Instead, it is whether the advertised representation of the app’s behavior is accurate. As a proxy for the advertised behavior of an app, we use its natural language description from the Google Play Store. As a proxy for its implemented behavior, we use the set of Android application programming interfaces (APIs) that are used from within the app binary. The key idea is to associate descriptions and APIs, and check whether the APIs used are consistent with the advertised behavior.

Checking App Behavior Against App Descriptions

1. **INTRODUCTION**
   - **Rhythm pattern in Jazz.**
   - **Feature of another app:**
   - **instance, behavior considered malicious in one app may well be a very much depends on the current context.
   - **any specification on what makes behavior beneficial or malicious.**
   - **gram behavior will be beneficial or malicious.**
   - **The problem is that new attacks, as it is hard to define in advance whether some pro-

3. **Clusters**
   - **Internet**
   - **Access-Location**
   - **“Weather”, “Map”…**
   - **“Travel”, “Map”…**
   - **“Theme”**
   - **Weather + Travel**
   - **Themes**

4. **APIs**
   - **Internet**
   - **Access-Location**
   - **Send-SMS**

5. **Outliers**
   - **Internet**
   - **Access-Location**
   - **Send-SMS**
App Collection

• Source: Google Play Store

• Downloaded top 150 apps + metadata from each of the 30 categories

• Time frame: Winter to Spring 2013

• Total: 32,136 apps

• Data package available on Web site
Stemming

looking for a restaurant, a bar, a pub or just to have fun in london? search no more! this application has all the information you need:

• you can search for every type of food you want: french, british, chinese, indian etc.
• you can use it if you are in a car, on a bicycle or walking
• you can view all objectives on the map
• you can search objectives
• you can view objectives near you
• you can view directions (visual route, distance and duration)
• you can use it with street view
• you can use it with navigation
keywords: london, restaurants, bars, pubs, food, breakfast, lunch, dinner, meal, eat, supper, street view, navigation
looking for a restaurant, a bar, a pub or just to have fun in London? search no more! this application has all the information you need:

- you can search for every type of food you want: french, british, chinese, indian etc.
- you can use it if you are in a car, on a bicycle or walking
- you can view all objectives on the map
- you can search objectives
- you can view objectives near you
- you can view directions (visual route, distance and duration)
- you can use it with street view
- you can use it with navigation

keywords: London, restaurants, bars, pubs, food, breakfast, lunch, dinner, meal, eat, supper, street view, navigation
Stemming

look london restaur search bar pub just applic fun
inform can search need everi type food want french
british chines indian etc car bicycl walk
can us can view object map visual rout
can search object search can view distanc
durat can view direct object near
can us street view can us navig
keyword london restaur bar pub food view
breakfast lunch dinner meal eat supper street navig
Topic Analysis

- Eliminated all apps with \( \leq 10 \) words, now 22,521 apps
- Want to discover the *topics* that occur in a collection of unlabeled text
- A *topic* consists of a cluster of words that frequently occur together
- Used *Latent Dirichlet Allocation* (LDA) to identify 30 topics
2.5 Finding the Best Number of Clusters

As an example, Table 2 shows four applications

tics with a certain probability. What we need, though, is to identify

One of the challenges with K-means is to estimate the number

In this context, we use applications as the elements to be clus-

Topic modeling can only assign application descriptions to top-

Assigned Name

<table>
<thead>
<tr>
<th>Id</th>
<th>Assigned Name</th>
<th>Most Representative Words (stemmed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>“personalize”</td>
<td>galaxi, nexu, device, screen, effect, instal, customis</td>
</tr>
<tr>
<td>1</td>
<td>“game and cheat sheets”</td>
<td>game, video, page, cheat, link, tip, trick</td>
</tr>
<tr>
<td>2</td>
<td>“money”</td>
<td>slot, machine, money, poker, currenc, market, trade, stock, casino coin, finance</td>
</tr>
<tr>
<td>3</td>
<td>“tv”</td>
<td>tv, channel, countri, live, watch, germani, nation, bbc, newspap</td>
</tr>
<tr>
<td>4</td>
<td>“music”</td>
<td>music, song, radio, play, player, listen</td>
</tr>
<tr>
<td>5</td>
<td>“holidays” and religion</td>
<td>christmas, halloween, santa, year, holiday, islam, god</td>
</tr>
<tr>
<td>6</td>
<td>“navigation and travel”</td>
<td>map, inform, track, gps, navig, travel</td>
</tr>
<tr>
<td>7</td>
<td>“language”</td>
<td>language, word, english, learn, german, translat</td>
</tr>
<tr>
<td>8</td>
<td>“share”</td>
<td>email, ad, support, facebook, share, twitter, rate, suggest</td>
</tr>
<tr>
<td>9</td>
<td>“weather and stars”</td>
<td>weather, forecast, locate, temperatur, map, city, light</td>
</tr>
<tr>
<td>10</td>
<td>“files and video”</td>
<td>file, download, video, media, support, manage, share, view, search</td>
</tr>
</tbody>
</table>
Cluster with the corresponding probabilities of belonging to topics. If we treat the applications as the elements to be identified, K-means identifies one centroid, thus identifying clusters.

### Table 2: Four applications and their likelihoods of belonging to topics

<table>
<thead>
<tr>
<th>App</th>
<th>Id</th>
<th>0.50</th>
<th>0.40</th>
<th>0.30</th>
<th>0.20</th>
</tr>
</thead>
<tbody>
<tr>
<td>“design and art”</td>
<td>13</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“food and recipes”</td>
<td>14</td>
<td>0.50</td>
<td>0.40</td>
<td>0.30</td>
<td></td>
</tr>
<tr>
<td>“personalize”</td>
<td>15</td>
<td>0.60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“health”</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“travel”</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“kids and bodies”</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“ringtones and sound”</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“game”</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“search and browse”</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“battle games”</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“settings and utils”</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“sports”</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“wallpapers”</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“connection”</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“policies and ads”</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“popular media”</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“puzzle and card games”</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 3: The list of clusters that were identified for the four applications

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Important Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>weather and stars, fans, shows, gangnam, top, bieber</td>
</tr>
<tr>
<td>2</td>
<td>weather and stars, fans, shows, gangnam, top, bieber</td>
</tr>
<tr>
<td>3</td>
<td>games, puzzle and card games, popular media, music, song, radio</td>
</tr>
<tr>
<td>4</td>
<td>games, puzzle and card games, popular media, music, song, radio</td>
</tr>
</tbody>
</table>

Note: The usage of topics is crucial for understanding the context of applications. The table above shows the most representative words (stemmed) for each topic. These words are key indicators of the themes and functionalities that the applications offer. The clusters we identified pretty much differ from the categories based on topics instead of clustering plain descriptions. The clusters (potentially even one per app) and thus being very specific. It is in the appropriate cluster. If the value is close to 1, it means that the element is the measure of how closely the element is matched to other elements within its cluster and how loosely it is matched to the other elements of the neighboring clusters. When the value of the silhouette of an element is close to 1, it means that the element is the measure of how closely the element is matched to other elements of the neighboring clusters. When the value of the silhouette of an element is close to 1, it means that the element is the measure of how closely the element is matched to other elements of the neighboring clusters.
London Restaurant Topics

look london restaur search bar pub just applic fun inform can search need everi type food want french british chines indian etc car bicycl walk can us can view object map visual rout can search object search can view distanc durat can view direct object near can us street view can us navig keyword london restaur bar pub food view breakfast lunch dinner meal eat supper street navig “navigation and travel” (59.8%) “food and recipes” (19.9%) “travel” (14.0%)
CHABADA

1. App collection

2. Topics

"Weather", "Map"...
"Travel", "Map"...
"Theme"

3. Clusters

Weather + Travel
Themes

4. APIs

Internet
Access-Location

5. Outliers

Internet
Access-Location
Send-SMS
Checking App Behavior Against App Descriptions

CHABADA

1. App collection
2. Topics
   - "Weather"
   - "Map"
   - "Travel"
   - "Theme"
3. Clusters
4. APIs
   - Internet
   - Access-Location
5. Outliers
   - Internet
   - Access-Location
   - Send-SMS

ABSTRACT

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Starting from a collection of “good” apps, we identify their natural language description from the Google Play Store. As a proxy for the advertised behavior of an app, we use the set of APIs that is advertised. Checking whether a program does what it claims to do is a long-standing problem for computer users, too. Whenever we install a new app, we run the risk of the app being “malware”—that is, to act against the interests of its users.

In the mobile world, for instance, behavior considered malicious in one app may well be a very much depends on the current context. Any specification on what makes behavior beneficial or malicious will be beneficial or malicious. The problem is that new attacks, as it is hard to define in advance whether some pro...

In all the examples above, the user would be informed and asked for authorization before any questionable behavior. It is the uncommon for that cluster (5).

Starting from a collection of “good” Android apps by their description topics, we identify outliers that use a specific pattern or not; it is uncommon for that cluster (5).

Specifically, our approach takes five steps, illustrated in Figure 1 and detailed later in the paper: (1) Starting from a collection of “good” Android apps downloaded from the Google Play Store. (2) We extract the natural language description topics (2) to form clusters of related apps (3). For each cluster, we identify the APIs (4) that are used from within the app binary. The key idea is to associate descriptions and implemented behavior, we use the set of Android application programming interfaces (APIs) that are used.

• An application that takes all of your contacts and sends them an SMS message to your premium number? Definitely suspicious.
• An app that tracks your current position is malicious? Not if it is a navigation app, a trail tracker, or a map application.
• An app that sends a text message to a premium number to raise money is suspicious? Maybe, but on Android, this is a known malware pattern. It flagged 56% of novel malware as such, without requiring any known malware patterns.
• An app that sends messages thus becomes an anomaly; likewise, a “weather” application accesses the user’s current location. Applied on a set of 22,500+ Android applications, our prototype identified several anomalies; additionally, our approach takes five steps, illustrated in Figure 1 and detailed later in the paper: (1) Starting from a collection of “good” Android apps downloaded from the Google Play Store. (2) We extract the natural language description topics (2) to form clusters of related apps (3). For each cluster, we identify the APIs (4) that are used from within the app binary. The key idea is to associate descriptions and implemented behavior, we use the set of Android application programming interfaces (APIs) that are used. Figure 1: Detecting applications with unadvertised behavior.
Clustering

- Want to identify groups of applications that are similar according to their descriptions.
- Used K-Means to identify such clusters
- Used elements silhouette to identify best number K of clusters
# Clusters

<table>
<thead>
<tr>
<th>Id</th>
<th>Assigned Name</th>
<th>Size</th>
<th>Most Important Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>“sharing”</td>
<td>1,453</td>
<td>share (53%), settings and utils, navigation and travel</td>
</tr>
<tr>
<td>2</td>
<td>“puzzle and card games”</td>
<td>953</td>
<td>puzzle and card games (78%), share, game</td>
</tr>
<tr>
<td>3</td>
<td>“memory puzzles”</td>
<td>1,069</td>
<td>puzzle and card games (40%), game (12%), share</td>
</tr>
<tr>
<td>4</td>
<td>“music”</td>
<td>714</td>
<td>music (58%), share, settings and utils</td>
</tr>
<tr>
<td>5</td>
<td>“music videos”</td>
<td>773</td>
<td>popular media (44%), holidays and religion (20%), share</td>
</tr>
<tr>
<td>6</td>
<td>“religious wallpapers”</td>
<td>367</td>
<td>holidays and religion (56%), design and art, wallpapers</td>
</tr>
<tr>
<td>7</td>
<td>“language”</td>
<td>602</td>
<td>language (67%), share, settings and utils</td>
</tr>
<tr>
<td>8</td>
<td>“cheat sheets”</td>
<td>785</td>
<td>game and cheat sheets (76%), share, popular media</td>
</tr>
<tr>
<td>9</td>
<td>“utils”</td>
<td>1,300</td>
<td>settings and utils (62%), share, connection</td>
</tr>
<tr>
<td>10</td>
<td>“sports game”</td>
<td>1,306</td>
<td>game (63%), battle games, puzzle and card games</td>
</tr>
<tr>
<td>11</td>
<td>“battle games”</td>
<td>953</td>
<td>battle games (60%), game</td>
</tr>
<tr>
<td>ID</td>
<td>Assigned Name</td>
<td>Size</td>
<td>Most Important Topics</td>
</tr>
<tr>
<td>----</td>
<td>-------------------------------------</td>
<td>------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>19</td>
<td>“sports”</td>
<td>580</td>
<td><strong>sports</strong> (62%), share, popular media</td>
</tr>
<tr>
<td>20</td>
<td>“files and videos”</td>
<td>679</td>
<td><strong>files and videos</strong> (63%), share, settings and utils</td>
</tr>
<tr>
<td>21</td>
<td>“search and browse”</td>
<td>363</td>
<td><strong>search and browse</strong> (64%), game, puzzle and card games</td>
</tr>
<tr>
<td>22</td>
<td>“advertisements”</td>
<td>380</td>
<td><strong>policies and ads</strong> (97%)</td>
</tr>
<tr>
<td>23</td>
<td>“design and art”</td>
<td>978</td>
<td><strong>design and art</strong> (48%), share, game</td>
</tr>
<tr>
<td>24</td>
<td>“car games”</td>
<td>449</td>
<td><strong>cars</strong> (51%), game, puzzle and card games</td>
</tr>
<tr>
<td>25</td>
<td>“tv live”</td>
<td>500</td>
<td><strong>tv</strong> (57%), share, navigation and travel</td>
</tr>
<tr>
<td>26</td>
<td>“adult photo”</td>
<td>828</td>
<td><strong>photo and social</strong> (59%), share, settings and utils</td>
</tr>
<tr>
<td>27</td>
<td>“adult wallpapers”</td>
<td>543</td>
<td><strong>wallpapers</strong> (51%), share, kids and bodies</td>
</tr>
<tr>
<td>28</td>
<td>“ad wallpapers”</td>
<td>180</td>
<td><strong>policies and ads</strong> (46%), wallpapers, settings and utils</td>
</tr>
<tr>
<td>29</td>
<td>“ringtones and sound”</td>
<td>662</td>
<td><strong>ringtones and sound</strong> (68%), share, settings and utils</td>
</tr>
<tr>
<td>30</td>
<td>“theme wallpapers”</td>
<td>593</td>
<td><strong>wallpapers</strong> (90%), holidays and religion, share</td>
</tr>
<tr>
<td>31</td>
<td>“personalize”</td>
<td>402</td>
<td><strong>personalize</strong> (86%), share, settings and utils</td>
</tr>
<tr>
<td>32</td>
<td>“settings and wallpapers”</td>
<td>251</td>
<td><strong>settings and utils</strong> (37%), <strong>wallpapers</strong> (37%), personalize</td>
</tr>
</tbody>
</table>
“Personalize” Cluster

FREE LAUNCHER DOWNLOAD THEME
“Travel” Cluster
In the mobile world, for instance, behavior considered malicious in one app may well be a very much depends on the current context. Any specification on what makes behavior beneficial or malicious will be beneficial or malicious. The problem is that new attacks, as it is hard to define in advance whether some pro-patterns of malicious behavior. However, this will not help against by checking static code and dynamic behavior against predefined the interests of its users.

we run the risk of the app being "malware"—that is, to act against a problem for computer users, too. Whenever we install a new app, standing problem for developers. Unfortunately, it now has become known malware patterns.

"Messaging" app would typically not be expected to access the current location. Applied on a set of 22,500+ Android applications, "messaging" app that sends messages thus becomes an anomaly; likewise, a "weather" application accesses the current location. Weather and Map applications typically do not access and could be expected to access the current location. A "travel" app that does so could be expected to access the current location. A "weather" app that accesses the current location could be expected to access the current location. A "travel" app that does not access the current location could be expected to access the current location. As a proxy for the advertised behavior of an app, we use the set of Android application programming interfaces (APIs) that are used from within its natural language description from the Google Play Store. As a proxy for its implemented behavior, we use the set of Android APIs that are used from within its natural language description from the Google Play Store. As a proxy for its implemented behavior, we use the set of Android APIs that are used from within its natural language description from the Google Play Store.

Our domain is Android apps, starting from a collection of "good" apps (1), we identify their API usage. The key idea is to associate descriptions and app binary. The key idea is to associate descriptions and key idea is to associate descriptions and API usages (1). As a proxy for the advertised behavior of an app, we use the set of Android APIs that are used from within its natural language description from the Google Play Store. As a proxy for its implemented behavior, we use the set of Android APIs that are used from within its natural language description from the Google Play Store. As a proxy for its implemented behavior, we use the set of Android APIs that are used from within its natural language description from the Google Play Store.

"Themes"... Weather + Travel

"Map"... Themes

"Theme"... Themes

"Travel"... Themes

"Map"... Themes

"Weather"... Themes

"Internet"... Themes

"Access-Location"... Themes

"Send-SMS"... Themes

"Internet"... Themes

"Access-Location"... Themes

"Send-SMS"... Themes

"Internet"... Themes

"Access-Location"... Themes

"Send-SMS"... Themes

"Internet"... Themes

"Access-Location"... Themes

"Send-SMS"... Themes

CHABADA

1. App collection

2. Topics

3. Clusters

4. APIs

5. Outliers
1. App collection

2. Topics

3. Clusters

4. APIs

5. Outliers
API Analysis

• For each APK, we identified the APIs used

• Used simple static analysis

• Only considered sensitive APIs which would be governed by permissions
London Restaurants

android.net.ConnectivityManager.getActiveNetworkInfo()
android.webkit.WebView()

INTERNET
GET-ACCOUNTS
ACCESS-WIFI-STATE
ACCESS-NETWORK-STATE
ACCESS-FINE-LOCATION
READ-PHONE-STATE
VIBRATE

android.net.NetworkInfo.isConnectedOrConnecting()
android.net.ConnectivityManager.getAllNetworkInfo()
“Travel” Cluster

Description

Permissions of APIs used
“Personalize” Cluster

Description

Permissions of APIs used
CHABADA

1. App collection

2. Topics

3. Clusters

4. APIs

5. Outliers

1. App collection

2. Topics

3. Clusters

4. APIs

5. Outliers
Checking App Behavior Against App Descriptions

Ilaria Tavecchia
Alessandra Gorla
Florian Gross
Andreas Zeller

CHABADA

1. App collection
2. Topics
3. Clusters
4. APIs
5. Outliers
“Travel” Cluster

Description

Permissions of APIs used

ACCESS-FINE-LOCATION
ACCESS-NETWORK-STATE
INTERNET
“Travel” Cluster

Permissions of APIs used

London Restaurants

Permissions of APIs used
London Restaurants

api

-- An Outlier in the “Travel” Cluster
Outlier Analysis

- In each cluster, identified outliers through *one-class support vector machine* (OC-SVM)
- Features of each APK: a vector of *(Sensitive API, #call sites)*
London Restaurants

- android.net.ConnectivityManager.getActiveNetworkInfo()
- android.webkit.WebView()
- java.net.HttpURLConnection.connect()
- android.app.NotificationManager.notify()
- java.net.URL.openConnection()
- android.telephony.TelephonyManager.getDeviceId()
- android.location.LocationManager.getBestProvider()
- android.telephony.TelephonyManager.getLine1Number()
- android.net.wifi.WifiManager.isWifiEnabled()
- android.accounts.AccountManager.getAccountsByType()
- android.net.wifi.WifiManager.getConnectionInfo()
- android.location.LocationManager.getLastKnownLocation()
- android.location.LocationManager.isProviderEnabled()
- android.location.LocationManager.requestLocationUpdates()
- android.net.NetworkInfo.isConnected()
- android.net.ConnectivityManager.getAllNetworkInfo()

→ Identified as Outlier
In this paper, we attempt to check app behavior against app descriptions. We describe CHABADA, a prototype system that uses natural language descriptions of apps from the Google Play Store to check app behavior against the app's description. Our approach takes five steps:

1. **App collection**: Starting from a collection of “good” apps, we identify their natural language description from the Google Play Store.

2. **Topics**: We extract topics from the app descriptions, such as “Weather”, “Map”, “Travel”, “Theme”, “Messaging”.

3. **Clusters**: We group apps with similar topics to form clusters.

4. **APIs**: We analyze the APIs used by each app, such as Internet, Access-Location, Send-SMS.

5. **Outliers**: We identify apps that use APIs in a way that is uncommon for their cluster, flagging them as potential anomalies.

Specifically, our approach is designed to detect apps that send text messages to premium numbers, access contact information, or steal location. As a proxy for the advertised behavior of an app, we use the set of Android APIs that are used from within the app binary. The key idea is to associate descriptions and APIs, and then identify anomalies that deviate from the expected behavior.}

1. INTRODUCTION

2. Topics

3. Clusters

4. APIs

5. Outliers

CHABADA stands for CHecking App Behavior Against Descriptions.
Evaluation: Outliers

• Can our technique effectively identify anomalies (i.e., mismatches between description and behavior) in Android apps?

• Manually checked top 5 outliers in each cluster (160 total)

• 26% showed covert behavior using sensitive APIs that acts against the interest of its users.
What makes an outlier?

- Ad frameworks (apploving, airpush)
- Dubious behavior (UNO, WICKED, Yahoo!)
- Uncommon behavior (SoundCloud)
- Benign outliers (Mr. Will’s Stud Poker)
Evaluation: Malware

- Can our technique be used to identify malicious Android applications?

- In each cluster, trained OC-SVM on 90% of “benign” apps

- Used TF-IDF as classifier on sets with remaining “benign” apps and 173 known malware apps

Malware recognition rate >80%
Úlfar Erlingsson
Which sensitive APIs does the device ID flow to?

- Benign Apps:
  - Network + SMS: 1%
  - Intent: 38%
  - Log: 60%

- Malicious Apps:
  - Network + SMS: 37%
  - Intent: 6%
  - Log: 57%
MUDFLOW

App 1

App 2

App 3

LOG 1

LOG 2

SMS 2

Outlier Detector

Training

Outlier Detection

Malware recognition rate > 86%

d = 0.76
Outliers

Detect Outliers for UI Elements!
Tripwolf

A serious travel app
Registration

Join Tripwolf

E-mail

JOIN TRIPWOLF

Already have an account?

SIGN IN
APIs used

- startActivity
- Handler
- Bundle
- TelephonyManager
- Handler
- Bundle
- SharedPreferences
- EditText
- EditText
- Uri

JOIN TRIPWOLF

SIGN IN
Outliers

The "Join Tripwolf" button transmits the current location to the Tripwolf servers.

This is unusual for "join" buttons
Backstage

1. App Collection
2. Mining GUI Elements
3. Context and APIs
4. Cluster Analysis
5. Outlier Detection
Backstage

1. App Collection
2. Mining GUI Elements
3. Context and APIs
4. Cluster Analysis
5. Outlier Detection
Mining GUI Elements

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:layout_width="fill_parent"
android:layout_height="fill_parent">
<fragment android:id="@+id/fragment"
class="uinomaly.fragmentclass"..//>
<Button android:id="@+id/buttonOK"
android:text="@string/buttonOK"
android:onClick="xmlDefinedOnClick"
style="@style/okButtonStyle"/>
<ImageButton android:id="@+id/imageButtonPrint" ...
android:src="@drawable/print_button"
android:contentDescription="@string/printText"/>
</LinearLayout>
```
Mining GUI Elements

<Button android:id="@+id/buttonOK"
    android:text="@string/buttonOK"
    android:onClick="xmlDefinedOnClick"
    style="@style/okButtonStyle"/>

<ImageButton android:id="@+id/imageButtonPrint" ... 
    android:src="@drawable/print_button"
    android:contentDescription="@string/printText" />

Icon file

Alternate Text

Text shown

Callback
1. App Collection
2. Mining GUI Elements
3. Context and APIs
4. Cluster Analysis
5. Outlier Detection
Backstage

1. App Collection
2. Mining GUI Elements
3. Context and APIs
4. Cluster Analysis
5. Outlier Detection
Mining APIs

• To identify APIs called, Backstage uses a static analysis built on top of Soot

• Builds a call graph starting with APIs defined in layout file

• Collects all reachable Android APIs
Context Sensitivity

View.OnClickListener myClick = new View.OnClickListener() {
    public void onClick(View v) {
        switch (v.getId()) {
        case R.id.ok_button:
            // action if button is the okButton
            break;
        
        case R.id.cancel_button:
            // action if button is the cancelButton
            break;
        }
    }
};
Mining Text

- Extracted all labels from all UI elements
- Static analysis includes labels set dynamically
- Extracted all text from surrounding screens (activities)
1. App Collection

2. Mining GUI Elements

3. Context and APIs

4. Cluster Analysis

5. Outlier Detection
Backstage

1. App Collection
2. Mining GUI Elements
3. Context and APIs
4. Cluster Analysis
5. Outlier Detection
Semantic Distance

- Clustering by LDA (as in CHABADA) failed: Too little text on mobile UIs
- Took us a year to realize that
- Instead, now use *semantic distance* using Google Words2Vec
- Words2Vec trained from 100 billion words
Clustering

- Used k-means to cluster all labels into 250 concepts with low semantical distance
agree · abort · account · account info · achievement · activate · activation · activity · add · add content · add email · add list · add photo · address · admin · agree · agreement · album · alert · alphabet · amazon · amount · answer · app · apply · appointment · apps · architecture · archive · attach · audio · authenticate · authorize · average · baby · back · background · backup · badge · bangalore · barbie · barcode · baseball · bath · beauty · bedroom · begin · birth · block · bluetooth · board · broadcast · build · business · buy · bypass · cache · calculator · calendar · call · calorie · camera · campus · cap · card · cardio · career · celsius · challenge · change · chapter · chart · check · checkout · cheer · choose · city · claim · clean · clear · click · clock · cloud · code · colombia · come · comment · commentary · connect · contact · continue · contribution · coupon · cpu · create · create account · credit · credit card · custom · customer · customize · cycle · data · day · deal · debug · decline · default · delete · demo · departure · deposit · description · desire · destination · detail · device · dictionary · do · download · draw · edit · edit account · editor · electron · email · enable · enter · error · examination · execute · export · facebook · fax · feedback · fiction · file · fill · find · folder · follower · friend · gallery · google play · handoff · health · hello · image · import · information · install · instrument · internet · invoice · itinerary · jupiter · keyboard · launch · league · license · list · location · log · login · map · meal · merge · message · mild · mode · news · next · notification · ok · open · order · panorama · password · payment · paypal · people · permission · phone · photo · picture · play · please · power usage · premium · prev · price · privacy · profile · project · projector power · pushup · quiz · redeem · register · reminder · report · reset · retry · roster · rule · save · save account · scan · scanner · search · send · setting · share · shopping · show · shutter · skip · sms · space · stay · store · submit · subscription · sync · taxi · term · test · theme · ticket · tip · title · twitter · unlock · update · upgrade · upload · url · user · vehicle · vehicle · version · view · virus · voice · wallpaper · website · weight · workout · zone

D. In-Concept Outliers

label instead. If

the threshold of 0.5, i.e.,

\[ v_p \]

“login” concept, Figure 7 shows the word cloud for the “share”

each word included.
members of overall classification, a two-stage setup first introduced with label instead. We repeat the above process mixing the original label vec-

This gives us a concept, and Figure 8 for the "shopping" concept. For each phrase surrounding a element, we now determine its

As illustrated in Figure 9, we then use the resulting distance serves as an...
Share
Shopping

- shopping
- start
- list
- dollar
- depot
- amazon
- target
- basket
- hancock
- sprouts
- stein
- harris
- keep
- save
- buy
- hhgregg
- yes
- about
- joe
- teeter
- cvs
- stop
- shop
- target
- ace
- walmart
- lowes
- cart
- general
- macy
- office
- bath
- bed
- home
- sears
- gap
- continue
- trader
- family
- shopper
- market
- go
- old
- winn
- kmart
- beyond
1. App Collection

2. Mining GUI Elements

3. Context and APIs

4. Cluster Analysis

5. Outlier Detection

"Sign up" cluster

"Join Tripwolf" "E-Mail"
"Already have an account?"

JOIN TRIPWOLF

startActivity

LocationManager

JOIN TRIPWOLF

startActivity

LocationManager
Classifying per Context

Outlier Detection

Training

Outlier Detection

AndroidHttpClient
NetworkInfo
Resources

AndroidHttpClient
SQLiteDatabase
ConnectivityManager

AndroidHttpClient
AccountManager
Resources

d = 0.76
Overall Anomalies

Training

Classifying

Classifier

Button

...
Outliers

JOIN TRIPWOLF

Handler
Bundle
startActivity
TelephonyManager
Handler
Bundle
startActivity
SharedPreferences
EditText

E-mail

sign in

JOIN TRIPWOLF

✔

✘
Evaluation

How well does Backstage discover UI anomalies?
“Label replace” mutation: assign a GUI element a different label – e.g. “Guides” is replaced by “Open” or “Print”

“Label crossover” mutation: swap labels of two GUI elements – e.g. “Welcome” gets “Special Offers” label and vice versa
Our analysis that induce noise. Our static analysis may over-
this kind of data, and thus must speculate.

Threats to validity. In terms of "Search with Google" and "Search with Bing" will use the
them. But if this distance is low,
correct label is crucial. If this distance is high, as indicated
alternative anomaly thresholds.

Higher precision for lower recall and vice versa by choosing
a bit more than two out of three
mutations fare even slightly better, with 76%. For "crossover"
the precision is 75%, meaning that three out of four
many false positives. For "random" mutations (Table V),
that is, if the
such mistake. All these results should be interpreted from the

Our work is related to three central strands of work, and in
detector
approximate and report
elements reported by
BACKSTAGE
Input
Total
Correct
Mutant

Total
4469
5686
10155

Precision = 75%
Recall = 67%
Accuracy = 73%
Specificity = 79%

Label Crossover Mutations

<table>
<thead>
<tr>
<th>Input</th>
<th>Classified as</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Abnormal</td>
<td>Normal</td>
<td></td>
<td>Total</td>
<td></td>
</tr>
<tr>
<td>Mutant</td>
<td>TP = 2290</td>
<td>FN = 2475</td>
<td>4765</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correct</td>
<td>FP = 1026</td>
<td>TN = 4121</td>
<td>5147</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>3316</td>
<td>6596</td>
<td>9912</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Precision = 69%
Recall = 48%
Accuracy = 65%
Specificity = 80%
Figures

- Backstage detects abnormal UI elements with an accuracy of 73–75%
- First machine learning approach to detect UI anomalies
- Mined 87,100 UI Elements in 12,000 apps
- 5 GB data set publicly available, with UI elements, labels, context, APIs…
App Mining

- For 100,000s of apps:
- Gather *descriptions*
- Gather *metadata*
- Gather *code and UI features*
- Find what is *common* and what is *uncommon*
Mining Apps for Anomalies
Andreas Zeller
Saarland University, Saarbrücken, Germany
Joint work with Alessandra Gorla, Ilaria Tavecchia, Vitalii Avdiienko, Konstantin Kuznetsov, and Florian Gross

http://www.st.cs.uni-saarland.de/appmining/