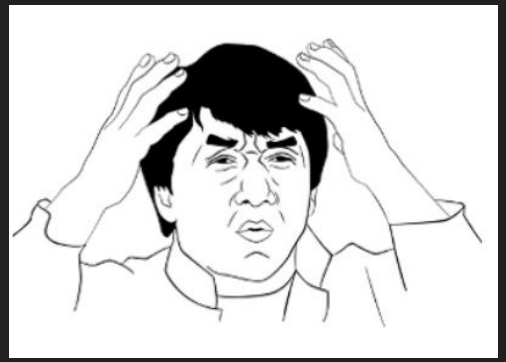
Apps



Which one?

Good? Bad?

Malware?

Malware Detection - Current Standard

Check Static Code & dynamic behaviour against patterns of malicious behaviour

New Threats?

Context Matters?

Context Matters?

- **An app sending text messages to raise money suspicious?**
- **An app that tracks location malicious?**
- **An app collects and sends contact info to server malicious?**

Yes, Context Matters!

- **An app sending text messages to raise money suspicious?** 

  Android uses this method as legitimate payment method for game features!!!

- **An app that tracks location malicious?** 

  A navigation or map application needs to use this feature!

- **An app collects and sends contact info to server malicious?** 

  Whatsapp!! This is exactly what it does upon initialization!!

Malware Detection - Current Standard

~~Does Static Code & dynamic behaviour match patterns of malicious behaviour?~~

Does the program behave as advertised?

CHABADA

CHecking App Behavior Again Description of Apps)

Alessandra Gola, Illaria Tavecchia, Florian Gross, Andreas Zeller
Saarland university, Germany

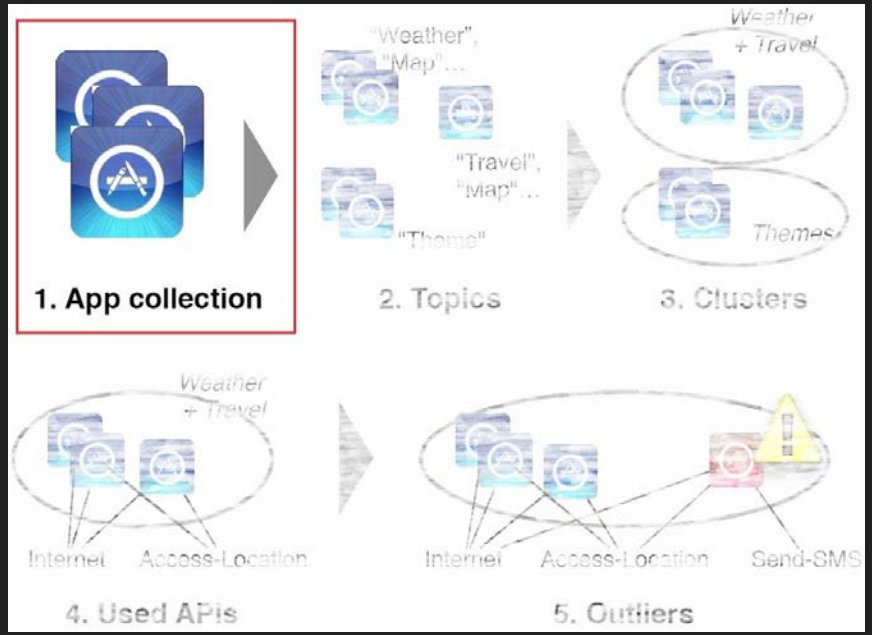
presenter name(s) removed for FERPA considerations

Contribution

- **CHABADA**
  - A new technique to detect malware by checking *implemented* behavior against *advertised* behaviour in the Android domain.

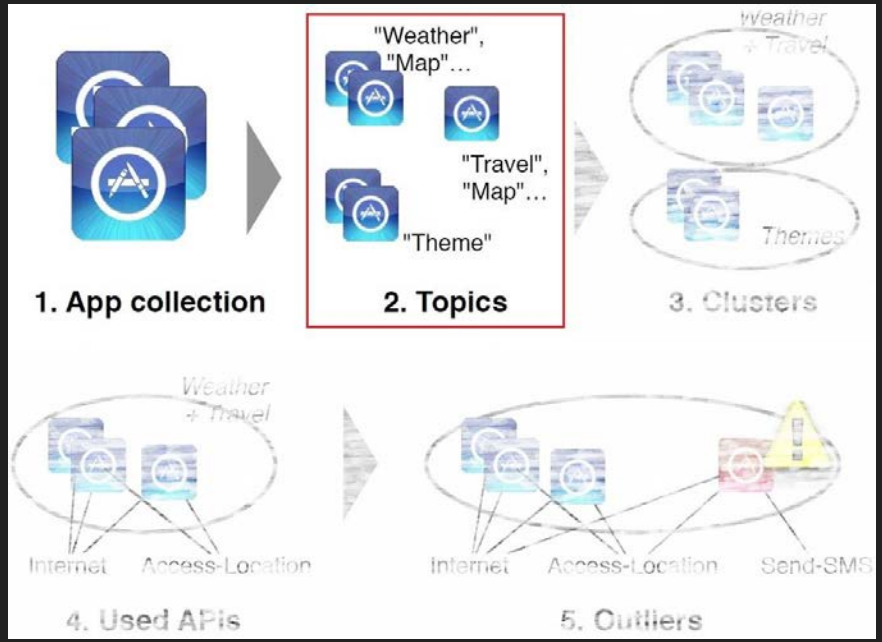
- **Evaluation of CHABADA**
  - Can this technique effectively identify anomalies (mismatches between description and behaviour)?
  - Can the technique be used to identify malicious Applications?

CHABADA Approach - Quick Overview



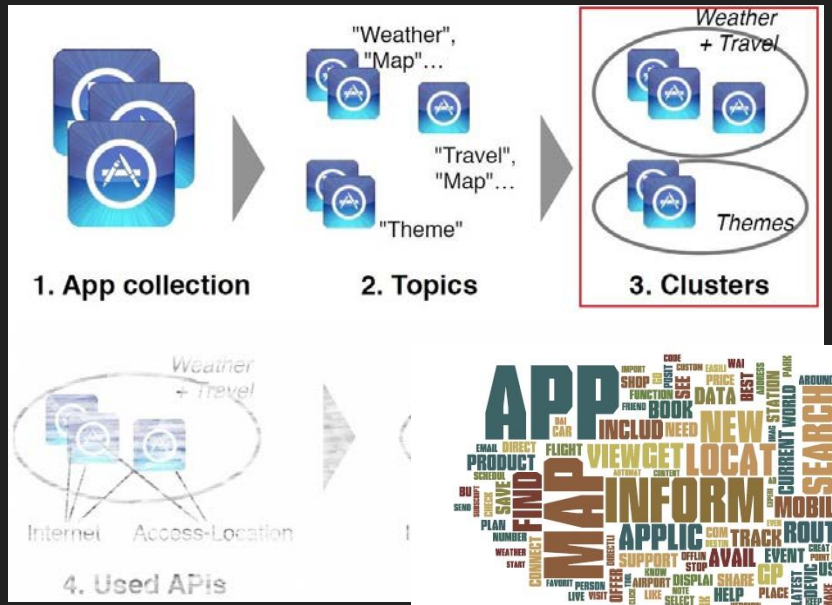
- Start with a collection of 22,500+ “GOOD” Android applications downloaded from Google Play store

CHABADA Approach - Quick Overview



- Using Latent Dirichlet Allocation (LDA) on app descriptions, identify the main topics for the each application.

CHABADA Approach - Quick Overview

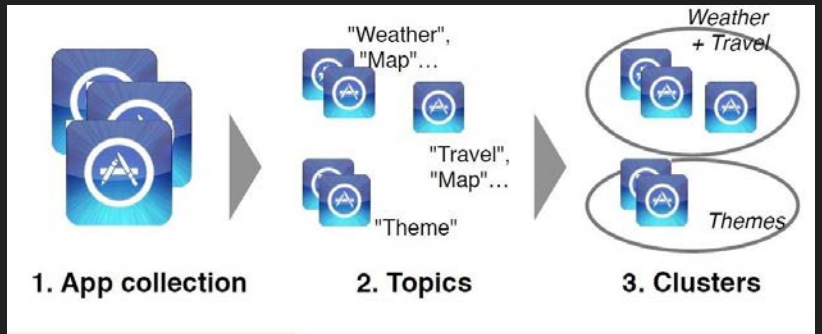


- Cluster the applications by related topics.

Figure 2: Words in app descriptions of the "navigation and travel" cluster. The bigger a word, the more app descriptions it appears in.

Figure 3: Words in "personalize" apps

CHABADA Approach - Quick Overview



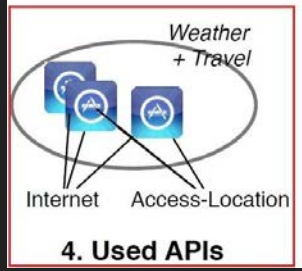
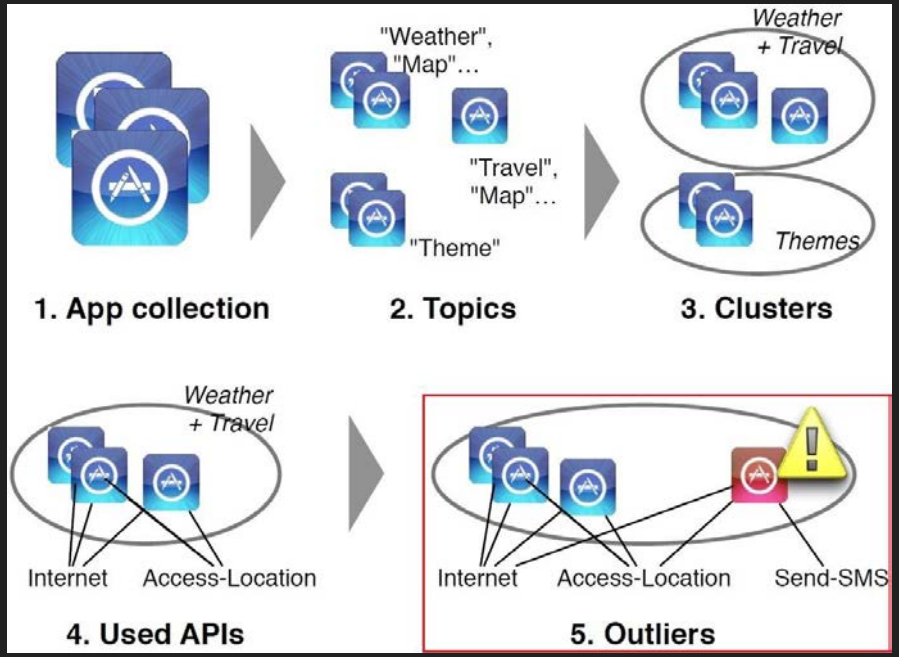


Figure 4: APIs, grouped by associated permission, used within the "navigation and travel" cluster. Standard behavior is to access the location ("ACCESS-FINE-LOCATION") and access the Internet ("ACCESS-NETWORK-STATE", "INTERNET").

- In each cluster, identify the APIs each app statically accesses.

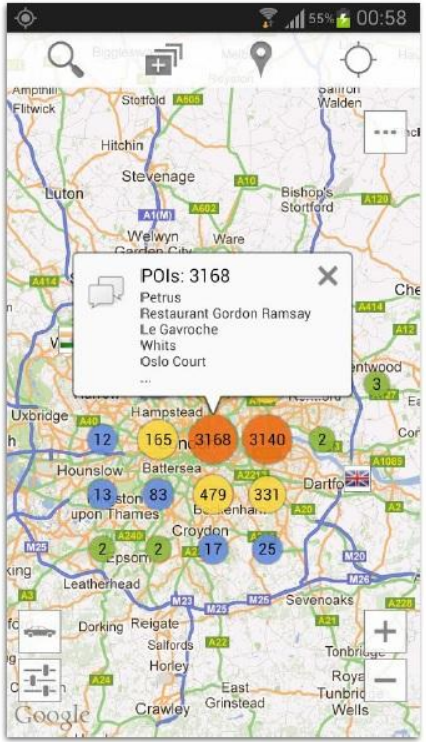
Figure 5: APIs for the "personalize" cluster. In contrast to Figure 4, these apps frequently access the SD card ("WRITE-EXTERNAL-STORAGE"), enable and disable new features ("CHANGE-COMPONENT-ENABLED-STATE"), but rarely, if ever, access the device location.

CHABADA Approach - Quick Overview



- Using unsupervised One-Class SVM anomaly classification, identify outliers with respect to API usage.

Example



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

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

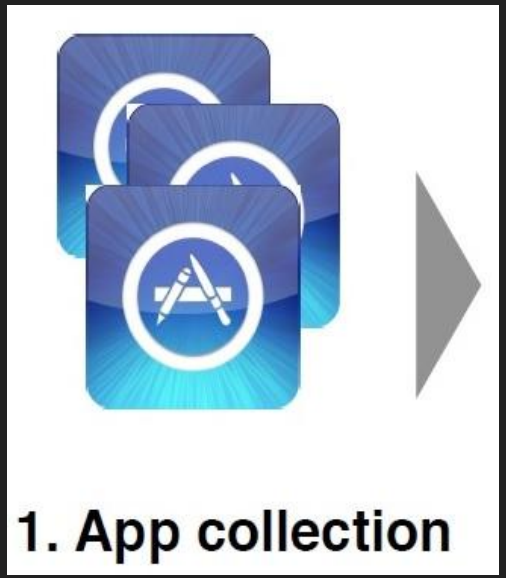
Figure 6: The app *London Restaurants Bars & Pubs +*, together with complete description and API groups accessed

Key Ideas

Step 1: Form the set of Android apps and features

- An automated script ran at regular intervals during the Winter and Spring of 2013 to download apps.
- For each of the 30 categories in the Google Play Store, the top 150 free apps in each category were downloaded.
- Total of 32,136 apps

Key Ideas



Step 1: Form the set of Android apps and features

- Several NLP methods used.
- All text not in English removed using Google’s Compact Language Detector.
- Stop words (e.g. “the”, “is”, “at”) removed.
- Stemming (reducing words to their root)
- Non-text items such as numerals, HTML tags, email addresses removed.
- All apps that do not use sensitive APIs (APIs that are governed by an Android permission setting)

Key Ideas

Step 2: Topic Modeling

- Uses a probabilistic model named *Latent Dirichlet Allocation (LDA)* on the app descriptions and identifies the main topics (e.g. “weather”) for each app.
- Using LDA, apps are assigned to one or more topics with certain probabilities.
- CHABADA chose 30 as the number of topics, same as categories in the Play Store.

Key Ideas

Step 2: Topic Modeling

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

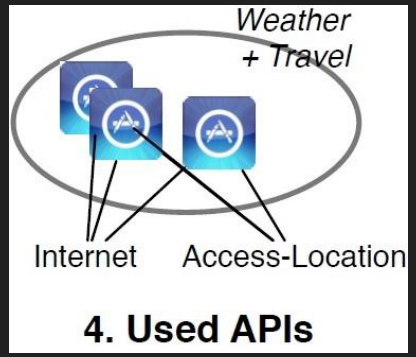
Application	$topic_1$	$topic_2$	$topic_3$	$topic_4$
app_1	0.60	0.40	—	—
app_2	—	—	0.70	0.30
app_3	0.50	0.30	—	0.20
app_4	—	—	0.40	0.60

Key Ideas

Step 3: Clustering

- Next step is to identify groups of applications with similar descriptions.
- Done using the K-means clustering algorithm.
- K-fold validation is used to find the best number of clusters.
- It uses the elements silhouette score for comparison.

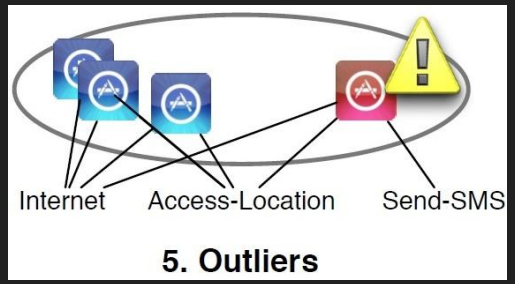
Key Ideas



Step 4: Identify the sensitive APIs used by the app

- Static API usage is considered as a proxy for behavior.
- For each app, the binary APK file is extracted with *apktool6*.
- API invocations extracted using smali disassembler.
- To obtain the set of sensitive APIs, the paper relies on the work of Felt et al.
- Why only use sensitive apis for analysis? Overfitting

Key Ideas



Step 5: Identify Outliers

- Using Machine Learning technique of One-Class SVM, outliers with respect to API usage identified.
- With the sensitive APIs as binary features, OC-SVM trained within each cluster to model which APIs are commonly in that cluster.
- For each cluster, produces a ranked list of apps where the top apps have the most abnormal API usage.

Evaluation

RQ1

Can this technique (CHABADA) effectively identify mismatches between description and behaviour in Android applications?

RQ2

Can CHABADA be used to identify malicious Android applications?

Evaluation - RQ1

- Perform k-fold validation 22521 apps that were grouped into 32 clusters
- The process uncovered outliers and the top 5 outliers in each cluster was selected for examination.
- Thus 160 total applications had been flagged by CHABADA as being suspicious were to be examined
- The examination had to be done manually by comparing app descriptions with actual source code

Evaluation - RQ1

- After the examination, the outliers were put into the 3 categories:

- Malicious apps: Apps who used sensitive APIs that were not advertised and more importantly these APIs were used against the interests of users

Evaluation - RQ1

- After the examination, the outliers were put into the 3 categories:

- Dubious apps : Used sensitive APIs that were not clearly advertised but their use of sensitive APIs did not go against the user's interest. Yahoo Mail was found to be among these because it uses the SMS API which was not advertised.

Evaluation - RQ1

- After the examination, the outliers were put into the 3 categories:

- Benign apps: Descriptions clearly matched the use of sensitive APIs in the source code.However apps that had inadequate descriptions on the Play Store also fell into this category

Evaluation - RQ1 Results

- As displayed in the table below, this examination resulted in identifying 42 outliers that were malware.This was a shocking 26% of the flagged applications. This was alarming because the researchers chose the top 150 downloaded apps from each category of the Google Play store

Table 5: Manual assessment of the top 5 outliers, per cluster and total.

Behavior	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	Total
Malicious	1	2	1	0	0	0	0	2	0	0	0	0	0	3	0	4	2	0	0	1	3	5	1	3	1	1	2	3	1	4	1	1	42 (26%)
Dubious	1	2	1	0	0	0	0	0	0	2	2	1	1	0	0	0	1	2	1	1	0	0	2	0	1	0	1	0	1	0	0	0	20 (13%)
Benign	3	1	3	5	5	5	5	3	5	3	3	4	4	2	5	1	2	3	4	3	2	0	2	2	3	4	2	2	3	1	4	4	98 (61%)

Evaluation - RQ2

- The One Class Support Vector Model which used in cases where there exists many samples of good data and fewer samples of anomalies or bad data.
- A large set of benign apps were used in this case. They consisted of apps that were not flagged in the first experiment and those that were flagged but were confirmed to be benign
- The One Class Support Vector Model was then going to be used as a classifier. It was trained on 90% of the benign apps

Evaluation - RQ2

- Then with the remaining 10% of benign apps, a set of 172 known malware were added to it. This was going to be the testing set. The OC-SVM was used as a classifier on this testing set.

- Essentially CHABADA, was presented with the problem of identifying malware without knowing previous malware patterns.

Evaluation - RQ2 Results

- The result as displayed in the table below showed that 56% of malicious applications were detected

Table 6: Checking APIs and descriptions within topic clusters (our approach)

	Predicted as malicious	Predicted as benign
Malicious apps	96.5 (56%)	75.5 (44%)
Benign apps	353.9 (16%)	1,884.4 (84%)

- The result may not seem very impressive compared to standard malware detectors but we have to bear in mind that standard malware detectors use known malware patterns whiles CHABADA is able to detect malware without knowing pre-existing malware patterns

Limitations

- Only free apps were used implying that results is biased towards app that rely on ads and in-app purchases to generate income

- App and malware bias: The sample used was from the top 150 downloads from each category on the Google Play Store. And as a matter of fact,this list of top downloads keeps changing. Thus, the selection of malware may not be representative of current threats.

- Sensitive APIs: The detection of sensitive APIs relies on a mapping provided by research that was published two years before the CHABADA experiment. Hence, the classification of sensitive APIs is likely to be obsolete

Conclusion

Even though CHABADA is not perfect, in practice CHABADA will work well by complementing standard malware detectors by specifically detecting unknown malware patterns.

Discussion Questions

Chabada works in Google Play domain. Can it perform just the same in other domains?

Discussion Questions

What was the main difference between the two experiments performed in evaluating the effectiveness of CHABADA?

Discussion Questions

How can CHABADA be used to make sure benign or harmless apps remain harmless?

Discussion Questions

CHABADA relies on evaluating static API usage, do you think this approach works well?

Discussion Questions

Can CHABADA categorically determine whether an app is good or malicious?