Android Market Analysis with Activation Patterns

Peter Teufl, Stefan Kraxberger, Clemens Orthacker, Günther Lackner, Michael Gissing, Alexander Marsalek, Johannes Leibetseder, and Oliver Prevenhueber

University of Technology Graz, Institute for Applied Information Processing and Communications, Graz, Austria
{peter.teufl,stefan.kraxberger,clemens.orthacker, guenther.lackner}@iaik.tugraz.at

Abstract. The increasing market share of the Android platform is partly caused by a growing number of applications (apps) available on the Android market: by now (January 2011) roughly 200.000. This popularity in combination with the lax market approval process attracts the injection of malicious apps into the market. Android features a fine-grained permission system allowing the user to review the permissions an app requests and grant or deny access to resources prior to installation. In this paper, we extract these security permissions along other metadata of 130.211 apps and apply a new analysis method called Activation Patterns. Thereby, we are able to gain a new understanding of the apps through extracting knowledge about security permissions, their relations and possible anomalies, executing semantic search queries, finding relations between the description and the employed security permissions, or identifying clusters of similar apps. The paper describes the employed method and highlights its benefits in several analysis examples – e.g. screening the market for possible malicious apps that should be further investigated.

Keywords: Android Market, Activation Patterns, Machine Learning, Security Permissions, Android Malware, Anomaly Detection, Semantic Search, Unsupervised Clustering.

1 Introduction

As mobile operating systems start to spread from the classic smartphone platforms onto tablets and ultra mobile computers, they are experiencing a significant gain in market share. Consumer acceptance and therefore commercial success of a mobile operating system depends on several factors. Beside the quality and usability of the user interface, the availability of applications (apps) may be the most important feature demanded by the market. While traditional systems provided a preinstalled set of apps, modern solutions like Apple's iOS, Google's Android, RIM's Blackberry or Microsoft's Windows Phone 7 offer the possibility

R. Prasad et al. (Eds.): MOBISEC 2011, LNICST 94, pp. 1–12, 2012.
© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2012

to access and install a wide variety of apps from different genres, ranging from games to powerful business appliances.

While Apple enforces tight policies on software distributed via their App Store for iOS, regarding security and content, Google emphasizes a more *open* philosophy, providing many liberties to Android developers, distributing their products via the Android Market.

Newly developed iOS apps are thoroughly examined by Apple engineers to keep the platform as secure as possible. This approach sometimes limits the developers' access to hardware resources like GPS receivers, cell phone functionalities or integrated cameras. As recent events have shown, even this strict approach has loopholes and apps not in line with the policies can find their ways onto the customer's devices[1]. In order to confine the impact of such incidents, Apple supposedly implemented a *kill switch* to deactivate installed apps on all devices[2]. Although not much details are known, from a security point of view it makes sense to use such a component.

Apps submitted to the Android Market are rudimentarily checked but the process is not as strict as it is for the App Store. Google seems to pursue a *delete afterwards* strategy if apps have been found of low quality or malicious. Android implements a similar functionality to Apple's kill switch to remotely remove apps installed on customer devices[3].

Google introduced a fine grained permission system for their Android platform, allowing developers to precisely define the necessary resources and permissions for their products. The customer can decide during the installation whether she wants to grant or deny access to these requested resources such as the address book, the GPS subsystem or telephone functionalities[4].

This user centric process is sometimes challenging and inducing customers to accept whatever is requested by the app, opening potential loopholes for attackers. In order to gain a better understanding on how permissions are used throughout the Android Market, this paper presents our analysis of publicly available metadata of 130.211 apps in the Android Market. The extracted metadata is comprised of several features that are displayed when the user opens an app for installation. Among the security permissions, which were of primary interest, we have extracted the description, download count, price and the category of each app. For the analysis, we propose a sophisticated method based on *Activation Patterns* that allows us to answer a wide range of questions such as:

- *Extract all wall papers that have a non-typical combination of security permissions - meaning they are anomalies.*

[1] `http://news.cnet.com/8301-13579_3-10464021-37.html`

[2] `http://www.telegraph.co.uk/technology/3358134/Apples-Jobs-confirms-iPhone-kill-switch.html`

[3] `http://www.engadget.com/2010/06/25/google-flexes-biceps-flicks-android-remote-kill-switch-for-the/`

[4] The user can accept either all permissions and install the app or reject all permission by not installing the app. Accepting or rejecting just a subset of these permissions is not possible.

- *Is the usage of security permissions different in free/payed apps?*
- *Which permissions are typical when the term "navigation" is used within the description?*
- *What are the most relevant features when analyzing popular security apps?*
- *Cluster popular apps according to their description or security permissions.*

The paper first describes the Android permission system and discusses related work, then describes the *Activation Pattern* concept and finally shows how it is applied to the metadata of 130.211 Android Market apps.

2 Android Permission Mechanism

Android's security architecture ensures the isolation of apps from each other as well as from the system. Communication and resource sharing are subject to well-defined access restrictions. Android apps are executed in their own Linux process with their own unique user- and group ID (UID), which allows for protection of file system resources. Access restrictions on specific resources and functions are enforced via a fine-grained permission mechanism. Apps are allowed access to resources if they are granted the respective permissions by the user.

This isolation of apps, called sandboxing, is enforced by the kernel, not the Dalvik VM. Java as well as native apps run within a sandbox and are not allowed to access resources from other processes or execute operations that affect other apps.

Apps must declare required permissions for such resources within their app manifest file. These permissions are granted or denied by the user during the installation of the app. The user does not deny or grant permissions during the runtime of the app[5]. Permissions are enforced during the execution of the program when a resource or function is accessed, possibly producing an error if the app was not granted the respective permission. The Android system defines a set of permissions to access system resources such as for reading the GPS location, or for inter-app communication. Additionally, apps may define their own permissions that may be used by other apps.

3 Related Work

Toninelli et al. [9] have investigated current methods of specifying security policies for smartphones. From their assumption, that in mobile computing scenarios users will be required to manage the security on their own, they deduct that the foremost requirement must be to design a simple security model which allows mobile end users to understand their security decisions. Otherwise, this would lead users to define or accept security policies which they do not comply with or also to turn off troublesome security features. Thus, they introduced a semantic-based policy model solution as one step towards a usable security for

[5] There is no dynamic permission granting as with the Blackberry system.

smartphones. They assess the efficiency and practicality of their security model by applying it to typical security related use cases. They have first analyzed typical mobile use cases and derived critical requirements for the design of a usable access control model. Their proposed solution relies on the assumption that understandability of the policy model is a necessary condition for usability of the access control system.

Their approach relies on a semantic-based policy representation. Such a semantic-based approach improves the understanding of security policies, they argue, since users would be more aware of their implications. Although their work is not directly concerned with apps from the Android Market, their results can be applied to mobile computing and the smartphone community in particular. However, they also outline that the users tolerance to failure remains a crucial issue for a usable access control framework.

SMobile has done some research on the Android Market and its permission system. They have documented specific types of malicious apps and threats. In their latest paper [10] they have analyzed about 50,000 apps in the Android Market. They looked for apps which could be considered malicious or suspicious based on the requested permissions and some other attributes.

Their key findings are that a big number of apps, available from the market, are requesting permissions that have the potential of being misused to locate mobile devices, obtain arbitrary user-related data and putting the carrier networks or mobile device at risk. Although the Android Operating System and Android Market prompt users for permissions before the installation, users are usually not experienced in making decisions about the permissions they are allowing or more precisely what permissions an app should have. But most often users do not take the time or have not the proper knowledge of the security implications. The most important statement they make is that fundamental security concerns and increase in malicious apps can be related to poor decisions of the user. Their work was the most comprehensive security analysis of the Android Market to date. Their conclusion was that end-users need to make educated decisions regarding the apps they are installing and that third-party security technology could assist them in making better decisions. This was one of the motivations for us to make a more in-depth analysis and to provide an open-source framework for automated permission analysis.

4 Activation Patterns

The analysis of app permissions in the Android Market is based on the *Activation Patterns* framework that we have developed during the last three years. This framework has been applied successfully to a wide range of domains such as event correlation [7], text classification [8] or semantic web analysis [6]. The idea behind this technique is to transform a raw data vector containing arbitrary symbolic and real valued features into a pattern, which forms the basis for a wide range of subsequent analyses. This transformation process is depicted in Figure 1 which shows several processing layers that:

1. extract features and feature values from instances[6] and store the information as nodes in a semantic network [5],
2. represent relations between these feature values and the strength of these relations (e.g. defined by the number of co-occurences within a data-vector) as weighted links within this network,
3. apply spreading activation techniques [1] for each instance, which stimulate the network and spread the activation of selected nodes according to their links to other regions of the network,
4. and finally extract the activation values for each instance from the network and store them within a vector that we call the *Activation Pattern*.

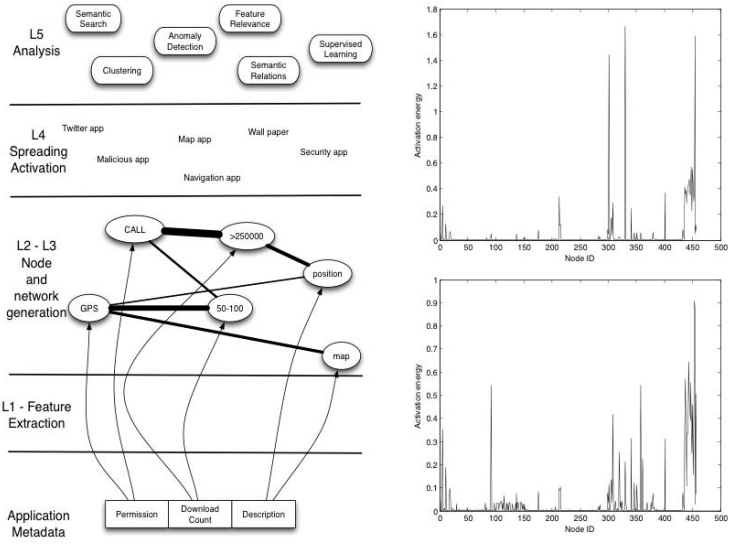

Fig. 1. Left: Layers for the *Activation Pattern* transformation and analysis. **Right:** Examples for two different *Activation Patterns*, the x-axis represents the nodes within the network, the y-axis represents the activation energy of these nodes after applying the spreading activation process to the activated nodes. These patterns are the basis for all subsequent analyses.

The generated patterns represent the activation values of different regions within the network that are activated due to different input stimuli (e.g. the feature values of an app). The similarity between two patterns can be calculated by distance measures (e.g. the cosine similarity) and expressed as a simple distance value.

[6] An instance is a data vector containing all feature values describing the instance. For the Android market, an app instance would be described by various features such as permissions, description terms or download count that have different feature values (e.g. different permissions).

This allows us to apply a wide range of standard machine learning algorithms and thereby cover various analyses with a single model:

Semantic *Search*: The distance between the *Activation Patterns* can be used to implement semantic search algorithms that retrieve semantically related instances. These search queries can also be used to specify certain feature values and find closely related patterns. *Example: Retrieve all description terms and permissions that are semantically related to the GPS permission.*

Feature *Relation*: The semantic network describes arbitrary relations between feature values. By activating one or more nodes (corresponding to feature values) within the semantic network, and spreading their activation via the links to the neighbors, we are able to extract details about the relations between various feature values and the strength of these relations. *Example: Which security permissions are strongly related to the term "GPS"?*

Feature *Relevance*: The relations within the semantic network are created according to the co-occurrence of feature values within the analyzed data set. The strength of these relations are represented by the associated weights within the network. Given a feature value that is represented by a node and the number of emerging/incoming links and their weights, we are able to deduce the importance of the information carried by the node. Nodes that are connected to a large number of other nodes typically do not add information for subsequent analysis processes. *Example: How relevant is the security permission for accessing the GPS when analyzing wall papers?*

***Anomaly* Detection:** The sum of all activation values within a pattern represents the activation energy of the whole pattern, which is a measurement for the response of the network. Anomalies can be detected in two ways: First, if features are combined in a non-typical way, the activation energy is lower than for normal combinations. Second, a large number of inputs (e.g. excessive usage of permissions) causes more activations and thereby higher total activation energies. The first anomaly detection method is more suitable when the number of feature values for each instance is constant. For the market analysis we concentrate on the second method, since the number of features varies from app to app. *Example: Find all wall paper apps that use non-typical permissions.*

***Typical Instances*:** An *Activation Pattern* is characterized by the activation values for the different feature values. By inspecting these values we can determine the strength of the activation and thereby the significance of the feature values. By combining several *Activation Patterns* with simple operations (e.g. mean, variance etc.) we gain knowledge about complete sets of patterns. *Example: What are the typical security permissions of free GPS apps?*

Unsupervised *Clustering*: Due to the transformation into *Activation Patterns* we can directly apply unsupervised clustering algorithms without the need to apply normalization and discretization strategies to the raw feature values. For this work we apply a simple Growing Neural Gas (GNG) algorithm [2] that is extended by the Minimum Description Length (MDL) criterion as used by the

Robust Growing Neural Gas (RGNG) algorithm [4]. This extension enables the algorithm to automatically detect the necessary model complexity, without the requirement to specify the number of clusters in advance[7]. *Example: Cluster all apps with a price larger that 10 Euros according to their description and employed security permissions.*

For a more detailed description of the *Activation Patterns* technique we refer to the appendix of [3].

5 Analyzing the Android Market with *Activation Patterns*

For our analyses we have extracted the metadata of 130.211 apps in December 2010[8]. All subsequent analyses have been conducted according to the following steps:

1. Apply an arbitrary filter to the app database (e.g. apps with certain permission, with a given download count, price or apps that are described with certain keywords).
2. Extract the features and their values from the remaining apps and apply the *Activation Pattern* transformation process.
3. Use the generated patterns for the analyses described in the previous section.

We have found several apps that have either a uncommon combination of permissions or make excessive use of permissions, however we must emphasize that having such permissions does not necessarily mean that the app abuses these permissions. Such abuses cannot be detected by the conducted analyses, but only by inspecting the code of the apps.

Due to space constraints we only highlight some prominent examples that demonstrate the capabilities of the *Activation Patterns* concept and refer the reader to our website where the complete analyses can be downloaded[9].

Q1: Retrieve apps that use the terms "hot" and "girl" in their description and find permission anomalies[10] ***(anomaly)****:* In contrast to the Apple App Store the Android Market allows apps with mature content. Similar to PC software or websites offering such content, we assume that such apps might be infected with trojans, spyware or are built deliberately in order to extract private information from the user.

[7] The Activation Patterns concept is not limited to the NG algorithm family – an arbitrary unsupervised algorithm can be applied to the patterns. Obviously, one could also apply supervised algorithms to the patterns for training a classifier, which is not shown in this paper.

[8] The app identifiers where collected from various web sites and the metadata itself was extracted from the Android Market via the android-market-api: `http://code.google.com/p/android-market-api/`

[9] `http://www.carbonblade.at/wordpress/research/android-market/`

[10] Hot Girls ALL Without Description.txt.

By applying the ***anomaly*** analysis to the patterns generated for the filtered apps, we are able to gain the following information: A large number of these apps just come with pictures that are displayed within the app or can be set as a wall paper. Some require a connection to the internet in order to grab pictures from web sites. The normal behavior and therefore the typical activation energy within a pattern is defined by this majority of apps. Anomalies deviate from this typical energy and are highlighted by two examples:

The first one is based on a group of apps that describe themselves as apps that "change the picture whenever the user receives an SMS with certain keywords". This description would suggest that the apps are required to have some permissions related to receiving and reading an SMS only, however they make excessive use of permissions related to writing SMSs, reading the contact data, accessing the internet, determining the user's position, using Google auth and many other additional permissions.

The second anomaly refers to an app that includes "hot puzzles and videos". However the app has access to the camera, is allowed to record audio, read and write contact data and has access to the GPS.

Q2: Is there a difference in the typical permissions when comparing free and payed apps with the terms "hot" and "girl" in their description[11] *(**typical instances**)?* In the second example we assume that free apps would be a better target for capturing private information, since they typically have a larger user base. After applying the filters we get the following results for the most active permissions, where the value within the parentheses represents the activation value. A higher value indicates a stronger activation within the network and therefore a higher significance of the corresponding feature value:

- **Payed (644 apps):** *internet* (0.74), *set wallpaper* (0.53), *get tasks* (0.45), *write external storage* (0.45), *get receive completed*[12] (0.42), *access network state* (0.28), *wake lock* (0.23),
- **Free (534 apps):** *internet* (2.95), *set wallpaper* (1.44), *read phone state* (1.19), *access network state* (1.15), *write external storage* (1.06), *access coarse location* (0.95), *access fine location* (0.64), *send sms* (0.46), *read contacts* (0.45), *wake lock* (0.42), *receive boot completed* (0.40), *receive sms* (0.38), *read sms* (0.37), *write sms* (0.37)

These results indicate a gap between free and payed apps with mature content. There might be several reasons for this difference: At first if these apps really include code that capture your private information than it would make more sense to include such code in free apps since they have a larger user base. The second explanation might be found in the light of various ad clients. Developers

[11] Hot Girls FREE Without Description.txt and Hot Girls PAYED Without Description.txt.

[12] This permission does not exist in the Android system. It might be a spelling mistake and therefore useless or a self defined permission that is used by multiple apps accessing each other. 88 of the 644 payed apps employ this permission.

often deploy free apps and generate revenue by using ad clients that display advertisements within the app. These ad clients tend to accumulate personal data and often need access to several permissions (see Q9 for more details). The third explanation might be that the developers simply added too many permissions and forgot to remove them. However, this does not seem likely since it would not explain the gap between free and payed apps. To determine what these apps really do, we need to go deeper, decompile these apps and inspect the code and all the calls made to Android APIs.

Q3: Extract popular (more than 5000-10000 downloads) apps with access fine or coarse location permissions and find permissions and terms that occur in the same semantical context as the term "GPS"[13] ***(search)**:* In this example we demonstrate the semantic search capability of the *Activation Patterns* concept. After applying the filter, we get 3204 apps with 5522 terms used for the description. We now execute a semantic search query for the term "GPS" yielding the following results, separated into terms and permissions:

- **Terms:** location, altitude, accuracy, coordinate, track, strength, position, sensor, range, program, technology, compass, satellite, longitude
- **Permissions:** *access fine/coarse location, internet, control location updates, access mock location, wake lock*, followed by permissions for accessing the camera, calling phones, receiving SMS etc.

The related terms are pretty obvious and show that the semantic search queries retrieve significant results. For the permissions it seems that most of the apps need to have access to the internet, prevent the phone from sleeping or dimming the screen (*wake lock*), or simulate a location update (*access mock location*). The last permission is needed especially during development in order to get a position in the emulator and probably was not removed after app deployment.

An example for retrieving semantically related instances would be a query that retrieves apps which do not contain the term "gps" but are semantically related to apps that have the term "gps" in their description (e.g. due to a description that contains "position" or "location"). Other interesting examples are the terms "wife" and "husband" which are closely related to the term "position". This semantic relation is created by apps that are used to spy on someone's wife/husband by tracking her/his location.

*Q4: How relevant are the feature values of the apps retrieved in Q3 "GPS" **(relevance)**?*

- **Most Relevant Feature Values:** The most relevant feature values are those that occur rarely. In case of the permissions the most relevant ones are either permissions for special purposes, permissions with a wrong spelling, or self-defined permissions.

[13] LOCATION - GPS With Description.txt.

– **Least Relevant Feature Values:** These are values that firstly occur within most of the analyzed apps and secondly that are randomly connected to other feature values. An example is the *internet* permission that is required by a large percentage of apps and is not correlated to other feature values – meaning it co-occurs randomly with those values.

Q5: How is the term "navigation" related to security permissions[14] ***(relation)****?* In this example we extract the semantic network links to security permission nodes emanating from the node "navigation" and use their weight to determine the strength of these relations: The following permissions are sorted according to the strength of their relation with "navigation": *access coarse/fine location, internet, read contacts, read phone state, write external storage, wake lock, access network state, call phone*. The contact and phone related permissions are typical required in order to control incoming calls/messages from the navigation app. The "navigation" term is also strongly related to the category "travel", high download counts, and GPS relevant terms similar to those of Q4.

Q6: Filter free apps that use the terms "wall" and "paper" in the description and have the permission set wallpaper. Find permission anomalies[15] ***(anomaly)****:* The biggest anomalies are caused by apps that replace the standard launcher of Android. Due to their functionality such apps require a large number of permissions and are therefore considered as anomaly. However, they are followed by several other interesting examples: One of these apps is called *FoxSaver* and describes itself as an app that allows you to browse photos from the website *foxsaver.com* and install them as wall paper. However, the app also has the *receive/read SMS, read contacts, access fine/coarse location* and *call phone* permissions. These additional permissions do not make sense when reading the description of the app.

Q7: Filter apps that have the permissions for reading, receiving and sending SMS messages but do not contain terms related to these permissions in their description (e.g. "sms", "message" etc.)[16] ***(anomaly)****:* The biggest anomalies are caused by apps related to Android development, security and backup. For most of these apps it makes sense to use the permissions, however there are other apps where the description does not match the required permissions: a large number of themes that are just described as "theme" with a certain keyword and several other apps do not state anything about messaging within their description (including games, fitness apps, travel guides etc.).

Q8: Cluster popular apps (more than 250.000 downloads) according to their description and security permissions[17] ***(clustering)****:* After applying the filter, 1079 apps remain that are grouped in the following categories (clusters):

[14] LOCATION - NAVIGATION With Description.txt.
[15] Wallpaper Without Description.txt.
[16] SMS PERMS Without Description.txt.
[17] DOWNLOAD COUNT - Only Description.txt and DOWNLOAD COUNT - Without Description.txt.

- **7 description clusters C1 (55)** ringtone apps; **C2 (64)** apps with German description[18]; **C3 (116)** social network apps; **C4 (339)** tools, widgets, games, browsers; **C5 (123)** translation, reference, books; **C6 (46)** music players, streaming; **C7 (336)** games.
- **8 permission clusters C1 (326)** games with a few permissions (*internet, access network state, read phone state)*; **C2 (31)** apps that have access to account data, contacts (e.g. social network related); **C3 (217)** mostly apps with internet access only (reference, games); **C4 (160)** also social network related, but with a bias to location related permissions; **C5 (99)** music and video apps, various permissions, but a strong activation of the *wake lock* permission, which is required to prevent the phone from sleeping; **C6 (59)** mostly ringtones apps with a strong activation of the *read phone state* permission. This permission is required by an app for recognizing that the phone is ringing; **C7 (126)** various apps that require a mixture of different permissions; **C8 (61)** phone and SMS related apps indicated by related permissions.

Especially, the permission cluster results are quite promising, since they allow us to gain knowledge on how permissions are typical used by various application categories and find outliers within a given category.

Q9: Identification and tracking of users: In order read out the unique ID of your smartphone, SIM ID, telephone number or cell ID, the permission *read phone state* is required[19]. In combination with the *internet* permission and possible the location related permissions[20], this enables an app to transmit information which allows user identification and tracking. 31865 of 130211 apps have these two permissions. For the 1079 apps that have more than 250.000 downloads, 362 have these two permissions. This corresponds to the results presented in an analysis of 101 popular apps, which focuses on private data sent to various companies (mostly related to advertisements)[21].

6 Conclusions and Outlook

In this paper we have applied the *Activation Patterns* concept to the Android Market apps. This new technique allows us to extract detailed knowledge about

[18] The distance between the German descriptions and the English ones is so large, that the 64 apps are only represented by one cluster. This could be changed by a more complex model.

[19] We refer to `http://developer.android.com/reference/android/telephony/TelephonyManager.html` for a detailed list of extractable information.

[20] The *read phone state* permission grants access to your current cell ID, which could already be used to determine the user's position if an appropriate database containing cell tower locations is available: e.g. `http://www.skyhookwireless.com/`. Therefore, the location can also be determined without the *fine and coarse location* permissions.

[21] `http://online.wsj.com/article/SB10001424052748704694004576020083703574602.html`

the apps and relations between the security permissions, description terms, download counts etc. Since, the Android Market share and therefore the number of apps are growing steadily we argue that the Android platform is an obvious target for malicious activities[22]. For this reason we deem it necessary to get a better understanding of the available apps, their employed security permissions and existing anomalies.

Now, that we have a solid basis for further analysis, we are planning several steps within the next months: The anomaly detection part can be used to screen the market for possible malicious apps, which are then subject to a more detailed analysis. We are currently devising a system that is capable of performing an in depth app analysis.

References

1. Crestani, F.: Application of spreading activation techniques in information retrieval. Artificial Intelligence Review 11(6), 453–482 (1997)
2. Fritzke, B.: A growing neural gas learns topologies. Advances in Neural Information Processing Systems (1), 1211–1216 (2005)
3. Lackner, G., Teufl, P., Weinberger, R.: User Tracking Based on Behavioral Fingerprints. In: Heng, S.-H., Wright, R.N., Goi, B.-M. (eds.) CANS 2010. LNCS, vol. 6467, pp. 76–95. Springer, Heidelberg (2010)
4. Qin, A.K., Suganthan, P.N.: Robust growing neural gas algorithm with application in cluster analysis. Neural Netw. 17(8-9), 1135–1148 (2004)
5. Quillian, M.R.: Semantic Memory, vol. 2, ch.10, pp. 227–270. MIT Press (1968)
6. Teufl, P., Lackner, G.: RDF Data Analysis with Activation Patterns. In: Und Hermann Maurer, K.T. (ed.) Proceedings of the 10th International Conference on Knowledge Management and Knowledge Technologies, iKNOW 2010 Graz Austria, Journal of Computer Science (2010)
7. Teufl, P., Payer, U., Fellner, R.: Event correlation on the basis of activation patterns. In: Proceedings of the 18th Euromicro Conference on Parallel Distributed and NetworkBased Processing, PDP 2010, pp. 631–640 (2010)
8. Teufl, P., Payer, U., Parycek, P.: Automated Analysis of e-Participation Data by Utilizing Associative Networks, Spreading Activation and Unsupervised Learning. In: Macintosh, A., Tambouris, E. (eds.) ePart 2009. LNCS, vol. 5694, pp. 139–150. Springer, Heidelberg (2009)
9. Toninelli, A., Montanari, R., Lassila, O., Khushraj, D.: What's on Users' Minds? Toward a Usable Smart Phone Security Model. IEEE Pervasive Computing 8(2), 32–39 (2009)
10. Vennon, T., Stroop, D.: Android Market: Threat Analysis of the Android Market (2010)

[22] ...and also legal apps that identify and track you due to advertisements.