Michael Harkess
EC 601 - Project 1 Lit Review
0918/2022

Introduction

Millions of people have Android devices and therefore are utilizing apps from the Google Play store and other third-party stores. As such, it is important to prevent malicious applications from being accessible in the app store. Having a program that can detect malicious code in apps would help mitigate the risk of a mobile app user being subjected to a potential risk in the security of their device. In addition to detecting malicious applications, such a program can also classify benign applications into the app store categories they would be published under.

The best method to achieve both of those goals is to combine the current major methods of analyzing applications into a single analysis model. In doing so, this unified approach covers a greater amount of threats that may harm the user's devices and data. As such, the hybrid model needs to further develop to encompass more current and upcoming threats that pose a risk to the data of users.

Other Research

There have been different approaches in academia to combat this issue. For example, to attempt a resolution to benign apps, some researchers developed an automated classification program to classify apps into their corresponding app store categories. There are two primary methods that facilitate the classification of apps: Static and Dynamic analysis. Static analysis studies various aspects of an application's code to determine an outcome. Some of those aspects include data flow and control flow. Dynamic analysis is the opposite of static analysis in that it studies aspects of an application during runtime including: network traffic and code execution paths.

One group of researchers used a classification model centered around static analysis and another focused on making a classification system combining the static and dynamic methodologies. For the static analysis classification system, researchers decided that applications can be split into two major categories, each with their own subcategories. The two primary categories were Games and Non-Games (Wang). Then the subcategories were genres of either category (e.g Trivia is a subcategory of games and Health is a subcategory of non-games). This subdivision of categories is logical as that would be the classification companies like Google are using when running their app stores.

The results of this type of classification were that the initial split between Games and Non-Games had a high accuracy rate while the subcategories had much lower accuracy rates, a difference of about 35% (Wang). The subcategories also had very lopsided preferences for categorization. An example would be the Casual/Card category of the Games section. This category had the highest number of game apps classified as this category and had a high accuracy score of 93% (Wang). However, the difference in accuracy between this category and all other Game subcategories was at least 56% (Wang). This could be due to the reason that many apps on the app store belong to more than one category, making it harder to classify apps into a single category. Also, many genres of apps borrow elements of other categories without being a part of that category. The biggest example of this is the Casual subcategory under Games as this category reels in many users. By borrowing elements of the Casual category, applications can keep their users engaged with minimal effort and gameplay. This means that the classifier may classify an app as Casual even though that may not be the developer's intention for their application's genre.

The other team of researchers focused on a classification combining Static and Dynamic analysis. The methods of static analysis used by this team consists of a more standardized approach (not dividing between Games and Non-Games), but use very similar algorithms to classify applications (e.g. Random Forest, K-Nearest Neighbor). However, this team uses Dynamic Analysis to enhance the classification capabilities of Static Analysis. Using the methodology, the team looked at the permissions application requested of the operating system and analyzed the compiled Java and DEX files that are present in the application's APK. Their discovery was that certain combinations of permissions, such as reading phone state and sending messages have a higher risk of the application being malicious (Singh 219). Permissions such as the ones previously mentioned can grab personal data such as the device's phone number and IMEI number then transfer that data to a third party via text (Singh 219). They also used a "bugging" method where data was inserted into the application code between the conversion from Java to DEX in the native language used as an intermediary between the two file types (Singh 219). This added data recorded call logs from the target device.

Dynamic Analysis is able to detect more inconspicuous malicious activity than static analysis based upon the experiment described previously. However, the technique to add extra code in between the Java to DEX conversion also has other implications. If malicious code can be inserted during that process, then code pertaining to methodologies such as Dynamic Analysis can also be inserted. Having a Dynamic Analysis program implemented automatically during the conversion process allows companies that maintain mobile device ecosystems (e.g Google, Apple) helps to protect user data from future app updates that may contain malicious code (some applications do not update from the app store and instead from their own third party servers).

This makes the user device better protected from threats and allows users to trust companies that run the app stores.

Both of the classification methodologies described above have their own merits and demerits. The static method is much easier to implement and faster to execute than the other method. In contrast, the dynamic method takes longer to execute, more difficult to implement but is more thorough with detecting anomalies in the application. However, the dual method allows for greater security and more robust methods of classifying applications. A hybrid model allows for the strengths of both approaches to be used, allowing for a greater protection of user data. The dual method would also be better for large scale application, namely for usage by institutions. This is because any change in behavior in the application that is deemed malicious can be caught immediately and notify the institution of the change in behavior. This allows for apps to be monitored on devices, to ensure the safety of the user.

After reviewing some of the methods researchers are using to classify benign and malignant applications, I can see some ways that it could be approved upon. For the instance of classifying apps with multiple categories, it might be possible to run the app through the classification algorithm again except without the category that it was classified into previously. Another idea would be to utilize the consensus method (multiple classification algorithms classifying the same application) to decide if an app gets more than one category (e.g If there are 5 classification algorithms and there are 2 categories the algorithms decided on, the application would get those categories tagged onto it). But as with most automated tasks, there should be an option for a manual review as an automated application may not get every classification correct, in order to ensure the integrity of the platforms applications are published on.

Significance

This topic is useful because it can help make the process of vetting mobile applications easier and keep the users of those apps safer. This form of mobile application vetting lets companies who run application stores automatically check if a mobile app is malicious, saving them time (from checking manually) and cost (hiring a third party to vet applications). If the application is developed to encompass common threats in compiled mobile apps, more time can be used to develop techniques to prevent other forms of malicious exploits in apps on the app store. In addition, techniques such as Dynamic Analysis provides “real-time” protection from malicious code from running on a user device.

Conclusion

These detection methodologies are being used currently by app stores such as Google Play, to vet apps being accessed through their storefront. Consumers can keep the same level (or greater level) of trust towards the app store as apps are being vetted for maliciousness. Along with App stores using the code analyzer to distinguish between malicious and benign apps being uploaded to their platform and “real-time” protection applications, companies that have many teams of developers to produce apps can also use the analyzer to make sure that their final product does not have a static security issue (or can be patched if a potential vulnerability is found). To stay on top of threats that may arise in the future from applications, programs such as Play Protect that use the hybrid model need to be developed and enhanced further. In doing so, the mobile app ecosystem that companies maintain won’t be compromised and users can maintain the level of trust they have in those companies.

Works Cited

Singh, Pooja, Pankaj Tiwari, and Santosh Singh. "Analysis of Malicious Behavior of Android Apps." *Procedia Computer Science* 79 (2016): 215-20. Web

Wang, Wei, et al. "Detecting Android Malicious Apps and Categorizing Benign Apps with Ensemble of Classifiers." *Future Generation Computer Systems* 78 (2018): 987-94. Web.