Android Malware

Jennifer Lindgren and Hampus Eriksson
{jenli414, hamer848}@student.liu.se
Supervisor: Alireza Mohammadinodooshan
Linköping University, Sweden

***Abstract*—With over two billion Android users and millions of available applications, detecting malware before it is installed is essential. This report identifies application attributes that may indicate if an application is malicious or not. The results are based on a feature selection done using Principal Component Analysis (PCA) and random forest methods on APK data. The data was collected from a popular Chinese marketplace, *Anzhi*, using a Python web crawler, and from parsing the downloaded APK files with a third-party Python library, APK parse3.**

The study shows that with a data set of about 28 000 APKs, including 1 500 that were classified as malicious by more than 30 % of the virus scans provided by VirusTotal, the selected features in the report extracted from APK metadata are not enough to determine whether an APK is malicious or not. However, analyzing more intricate features, larger data sets, data sets containing more malware or using other analysis methods may yield better results.

I. Introduction

Android has over two billion users, as Google announced during their 2017 Google I/O conference. Spread across different markets, there are millions of applications available to these users [1]. On Google Play alone, there were over two million applications in 2018. While this means that many market options, and an immense amount of functionality, is available to the users, it also means that the threat of malware in Android Application Package (APK) files can not be ignored. In March 2018 alone, 420 795 new Android malware were discovered [2]. It is also worth noting that only 5.2 % of all Android users ran the current Android version equipped with the latest security updates in July 2018, 11 months after its release. In fact, one third of all Android devices run on outdated operating systems that no longer receive security updates and few use effective virus protection.

One security measure that has been taken to prevent malware is the Android permission system. It requires the user to grant permissions to the application before it can perform certain tasks [3]. For example, special permission is required before accessing the phone storage or using the camera. However, this does not necessarily stop the malicious applications as users tend to blindly grant the required permissions.

There clearly exists a need for malware to be detected before it is installed. This study identifies a set of attributes that are obtainable before installing the application and determines whether they could indicate if an APK file is malicious or not. The attributes were derived using Principle Component Analysis (PCA) and random forest on application data. The data was collected directly from the marketplace *Anzhi* using a web crawler, and from parsing the APK files downloaded from there.

This report introduces the reader to the relevant topics in Section II. Section III describes the limitations of the study as well as how it was conducted. Section IV contains the results from the study as well as an analysis of those results. Section V contains a discussion of the results and the method of the study. Section VI and Section VII contain related work, an analysis of the results and conclusions.

II. Background

In order to understand the contents of this report, short introductions to relevant concepts are presented in this section. The Android Application Package (APK) file is explained in the first subsection. The second subsection describes web crawling, the following two cover Principal Component Analysis (PCA) and random forest and the final two describe k-nearest neighbor and R^2.

A. The Android Application Package File

Android applications are compiled and packaged into Android application package (APK) files [4]. APK is an archive file format similar to ZIP. An overview of the main components is shown in Figure 1. This section will only explain the Android manifest, as it is the only component that is relevant to this study.

The Android manifest contains important information about the application such as names of Java packages and libraries, required permissions and minimum Android API level [4]. It also contains information about activities, services, receivers and providers [5]. An activity is an entry point for an application to interact with the user. A service is an entry point used to keep an application running in the background. A receiver enables the system to notify the application of events even when the application is not running. A content provider can be used to manage application data that other applications can query and/or modify.

The manifest played an important role in this study as required permissions as well as the number of activities, services, receivers and providers were aspects that were considered when looking for malicious patterns in Android malware.

B. Web Crawling

Web crawling allows users to extract information from the web by downloading web page content and querying it to

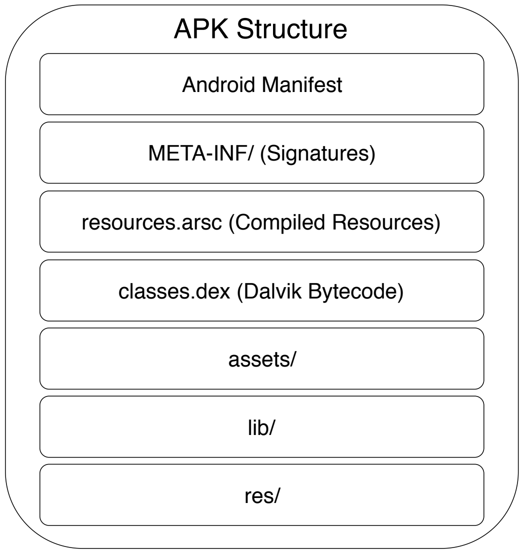

Fig. 1. APK Structure.

retrieve desired information [6]. The system is often called a web crawler, a robot, or a spider. Web crawling can be utilized in a variety of software, some examples are web search engines, web archiving and web data mining.

C. Principal Component Analysis

Principal Component Analysis (PCA) can be used to reduce the dimensions of large data sets while preserving most of the variation [7]. By doing so, the features that have the largest impact will be extracted. PCA achieves this by using the covariance matrix of the data to compute the eigenvectors and eigenvalues and keeping only the eigenvectors with the largest eigenvalues.

D. Random Forest

Decision trees can be used to create models for data in both classification problems where the outcome variable is discrete and in regression problems where the outcome variable is continuous [8]. It is a popular method in for example spam filtering and medical diagnosis.

Decision trees have a tree structure where the leaf nodes represent classifications and the nodes above them represent decisions [8]. The goal is to achieve a result as accurate as possible using as few decisions as possible. The method is useful when looking for features that have a large impact on the classification result, as not only the outcome is visible, but all decisions that lead to it. The resulting decision nodes will contain the features with the largest impact on the data.

In order to improve the results, a large number of trees can be generated randomly to determine the most popular classification [9]. There are many ways to generate the trees, but a common element is that for each tree, a random vector is independently generated and used to get a classifier. When a large number of trees have been generated, the most popular classifiers can be identified. This method is called random forest.

E. K-nearest Neighbour

K-nearest neighbour (KNN) is a classifier that uses previously labeled observations to assign new observations to the class with the most similar labeled observations [10]. The k in k-nearest neighbour represents how many of the most similar observations it should consider when determining the label.

K-nearest neighbour can be used to evaluate the feature selection performed by random forest or using a PCA model. An example of this is dividing the available data into training and test sets and do training using the features selected by the other models. The trained k-nearest neighbour model is used to label the test data. The accuracy of the result will show how robust the model is.

F. Coefficient of Determination (R^2)

The coefficient of determination, R-squared (R^2), is a well-known method for measuring the goodness-of-fit in a regression model. The measure ranges between $[0, 1]$ where a higher value of R^2 corresponds to a better model fit [11].

R^2 is calculated with a handful sums of squares [12]. Specifically, these are *the residual sum of squares* ($\mathrm{SSD}_{\mathrm{res}}$), *the model sum of squares* ($\mathrm{SSD}_{\mathrm{mod}}$) and *the total sum of squares* ($\mathrm{SSD}_{\mathrm{tot}}$). Each definition are presented in Equation 1.

$$\begin{aligned} \mathrm{SDD}_{\mathrm{res}} &= \sum_{i=1}^{n}(y_i - \hat{\mu_i})^2 \\ \mathrm{SSD}_{\mathrm{mod}} &= \sum_{i=1}^{n}(\hat{\mu_i} - \bar{y})^2 \\ \mathrm{SSD}_{\mathrm{tot}} &= \sum_{i=1}^{n}(y_i - \bar{y})^2. \end{aligned} \tag{1}$$

$\bar{y}$ represents the mean value of the response variable y, $\hat{\mu_i}$ denotes the fitted value and lastly y_i denotes the actual observation.

$$R^2 = \frac{\mathrm{SDD}_{\mathrm{mod}}}{\mathrm{SSD}_{\mathrm{tot}}} = 1 - \frac{\mathrm{SDD}_{\mathrm{res}}}{\mathrm{SSD}_{\mathrm{tot}}} \tag{2}$$

Equation 2 defines how R^2 is calculated with the aforementioned sum of squares.

III. Method

This section describes how the study was carried out and is divided into three parts. First, the limitations of the project are discussed and then the theoretical and practical methods are presented.

A. Limitations

Due to a number of factors, some of which stated by the supervisor and some found when reading previous work during the research phase, there was a need to define limitations of the study. One of the first factors that had to be taken into consideration was the fact that the web crawling needed to be throttled to some degree. Otherwise, the project could have been put to a halt by an IP ban from Anzhi or VirusTotal. Web sites, including those used in this study, reside on servers that are in charge of handling incoming requests by clients. Servers can only handle a certain number of requests at a time, and there are specific attacks that leverage this fact [13]. Therefore servers need to protect themselves, usually by banning the attacker. The crawlers used in this project are able to operate at very high speed and thus, not restricting them would most likely have resulted in a ban. Furthermore, the VirusTotal API has stated in their guidelines that a regular user is only allowed to make four requests per minute [14].

While researching previous work done on the subject, the group found that other research groups have used more than one app market. It was also common to see a large data set of APK information used for the analysis. Due to the time frame of this project there was no room to handle data from more than one market. To understand how to scrape and collect information from one market would be time consuming enough. Nor could it be expected that this project would obtain such a large data set as previous studies. This was both a consequence of the given time frame, and of the fact that the scraping had to be throttled.

B. Theoretical Methods

The first step in the project involved reading previous work in order to get a better understanding of the subject and what could be relevant to investigate. It included examining what attributes may be correlated to the maliciousness of an app, which markets appear to host malicious apps, and how the information should be collected.

Selecting an App Market

When searching for a suitable market place, the most important factor was that it actually had to host malicious apps. Large marketplaces such as Google Play have extensive virus protection and so they were not good choices for this project. After looking at previous work, it was decided that the marketplace Anzhi should be used, which had a good amount of malware found previously [15].

Evaluating Web Crawling Frameworks

In order to fetch data from the app market the group needed to decide on a framework to parse HTML documents. Prior to the project, the group had no experience with web scraping. Thus an evaluation of different frameworks and libraries was needed. The two candidates were *Beautiful Soup* and *Scrapy*. Both were found to be popular choices when it came to web scraping, however having some important differences. Beautiful Soup was found to be a powerful tool to extract data from HTML documents, while it lacked functionality to fetch the documents themselves or further process the data [16] [17]. On the contrary, Scrapy was a fully-fledged framework that could retrieve HTML documents from a given website, extract relevant data and process it further [18]. Since the group was looking for a complete framework that could fetch website HTML and work with said data, it was decided that Scrapy should be used for the project. Additionally, Scrapy was built with *Twisted*, a networking framework that operates asynchronously [19]. That meant that the group could take advantage of multiprocessing of data, without having to implement it themselves.

Selecting relevant attributes

When deciding what attributes about the APKs to analyze, previous work was again looked at. The goal was to include as many features as possible, if they could easily be gathered by a web crawler or by the chosen APK tool, APK parse3, they would be gathered. Some features were gathered as sums and others as binaries. For example the total number of permissions were stored as a sum but the top 20 most common permissions found in malware by other authors were also filtered out as binaries. i.e. 1 if they are requested and 0 if not [20]. The chosen features originated from two sources, the marketplace web page and the APK file.

From the market:

- APK ID
- APK URL
- Number of versions
- Number of days since last update
- Rating (1-5)
- Number of downloads
- Number of comments

From the APK file:

- APK file size
- Needs Internet permission
- Needs read phone state permission
- Needs access network state permission
- Needs write external storage permission
- Needs access WiFi state permission
- Needs read SMS permission
- Needs receive boot completed permission
- Needs write SMS permission
- Needs send SMS permission
- Needs receive SMS permission
- Needs vibrate permission
- Needs access coarse location permission
- Needs read contacts permission
- Needs wake lock permission
- Needs access fine location permission
- Needs call phone permission
- Needs change WiFi state permission
- Needs write contacts permission
- Needs write APN settings permission
- Needs kill background processes permission
- Number of permissions
- Number of activities
- Number of services
- Number of receivers
- Number of providers
- Number of components (the sum of four above)
- Number of libraries
- Indication of being malware (% based on VirusTotal scan)

Methods of Analyzing the Data

To analyze the data when it had been collected, the group wanted to use more than one method for feature selection. The requirements on the methods were that feature selection could be done and that the features involved could be identified. The methods random forest and Principal Component Analysis (PCA) were selected as they both fulfill the requirements but use slightly different approaches. For PCA, a requirement that was set was that the at least 80 % of the variance in the data had to be covered by the selected components.

To compare the results of the two feature selections performed, a third method had to be used. The group decided to use k-nearest neighbor for this. The selected components from the two methods would be used to classify the data and the robustness of each result would be determined.

C. Practical Methods

When it was clear what should be collected from the market and with what framework, the task of collecting the information began. This section describes how market data was scraped and how APK data was extracted.

Gathering ID and URLs

The first step of the data collection was to scrape the marketplace for IDs and URLs associated with all available APKs. The ID is a unique number given by Anzhi to each APK that can be used to send PHP requests to download or get information about the APK. The URL collected is the URL that leads to the application page containing information to be collected later.

The IDs and URLs were collected using a Scrapy crawler that visited all pages containing APK listings on Anzhi. This was achieved by iterating through the categories that are represented by numbers in the URL. For each category page, the crawler was able to navigate through all pages and then proceed to the next category when all pages had been parsed. About 220 000 pairs of IDs and URLs were collected.

Gathering Marketplace Information

The IDs and URLs that were gathered previously were now used by a second crawler to visit each of the APK URLs. The rating (1-5) was gathered by analyzing the element showing 1-5 stars. The number of downloads was shown as a number followed by the Chinese sign for e.g. thousand or million, this information was used to store the number of downloads in the database. In order to get data about comments, the URL was slightly modified resulting in the source code for the comments being accessible. From that, the number of comments could be extracted. Finally, a PHP-request was made using the ID to get information about versions. The total number of versions and number of days since last update was stored in the database. All 220 000 IDs and URLs were used resulting in the same amount of marketplace information gathered.

Downloading and Parsing the APKs

The final piece of software used in the data collection utilized the APK ID to send a PHP-request to Anzhi to download each APK. After downloading the APK, it was sent to VirusTotal for scanning. While waiting for the results, the APK was parsed using APK parse3 to get the remainder of APK information about permissions, components and libraries. The file size was also collected. After fetching the VirusTotal results, all data was stored in the database and the next APK was parsed.

This step in the data collection process was the main bottleneck. It wasn't an option to run this software at full speed as that would have overloaded the servers of both Anzhi and VirusTotal. As previously mentioned, VirusTotal state in their guidelines that a maximum of four requests per minute and API key can be made. In this project, a total of eight API keys were used spread across four Raspberry Pi units, meaning the software could be run at a maximum pace of 32 APKs/min. However, some of the units were used for the crawlers initially, so the average pace was lower than that. A total of about 30 000 APKs were downloaded and parsed in this step.

Analysis Using Random Forest and PCA

Once the data collection was up and running, the focus of the project shifted to data analysis. Since the data was spread across several Raspberry Pi units, their data had to be merged before analysis could be performed. The columns in the database was also spread across several databases because some features had been gathered by the crawler and some by the parser as described in Section III-C.

Once the data set was complete, analysis began by using a random forest library in R. From the forest, the increase in mean squared error (MSE) when removing each of the components was derived. This described the impact of each of the components on the classification, which was important when determining what attributes could indicate malware.

Another analysis was performed using PCA to decrease the dimensionality while still covering a majority of the variance in the data. From this analysis, new components containing several features were derived. Further examination of each component was done to see how much features contributed to them.

The results from the two analyses were evaluated by creating a K-Nearest Neighbour (KNN) model and deriving the R^2 value for each of the them. The final R^2 value was calculated from the test data predictions that the models made and the actual test data values. Test data was not introduced to the model during training, that way R^2 could be seen as an evaluation of the model on new data. The optimal k for the models were found with a 10-fold cross-validation on the training data. Additionally, both KNN models were be created with the same number of predictors. The difference was that the model based on random forest used the top number of variables in the importance plot, while the PCA-based model used components. This led to a more fair comparison between the two feature selection models. The number of components needed to achieve 80 % variation coverage in PCA dictated how many predictors were used in the KNN models.

IV. Results and Analysis

This section will present malware scan results, as well as the Principal Component Analysis (PCA) and random forest made in R from the collected data set. A brief analysis of the result will also be presented.

A. *Malware in the Collected Data*

Out of the 28 237 APKs scanned, 1 457 were classified as malware in more than 30 % of the virus reports provided by VirusTotal, which corresponds to 5 %.

B. *Random Forest*

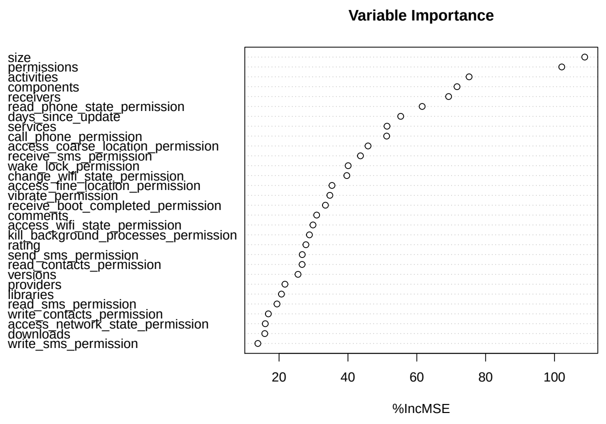

Fig. 2. Variable importance as measured by effect on mean squared error.

The results from the random forest show the importance of each variable by describing for each component how the mean squared error (MSE) changes if it is permuted (%IncMSE). A high value of %IncMSE corresponds to a high variable importance in the classification. The results are shown in Figure 2. The attributes that affect MSE the most, by more than 100 %, are *size* and *number of permissions*. There is a slight gap between those two attributes and the rest, but other significant attributes that affect MSE by more than 60 % are number of receivers, components and activities, as well as the read phone state permission.

C. *Principal Component Analysis*

The principal component analysis generated components affected by multiple attributes to different degrees. The variance coverage for each of the components is illustrated in Figure 3. 16 of the PCA components are required to cover 82 % of the variance in the data. The dimensionality of the data can thus be reduced from 34 to 16 while retaining 80 % of the variance.

The largest component, which covers 25 % of the variance, is affected by many different attributes, where the individual contributions are not notably high. This means that many variables will affect the classification. The features with the largest contributions were number of permissions (6.4 %), number of components (5.5 %), the access WiFi state permission (5.4 %), change WiFi state permission (5.3 %), access fine location permission (5.3 %), number of activities (5.2 %), the read phone state permission (5.2 %) and access coarse location permission (5.0 %). It is worth noting that the second component is mostly affected by features relating to SMS permissions, read, write, send and receive SMS permissions and read and write contacts permissions, which contribute with 8-10 % each. The fifth component seems to be especially affected by market-related data, number of comments (18 %), number of downloads (18 %) and number of versions (11 %).

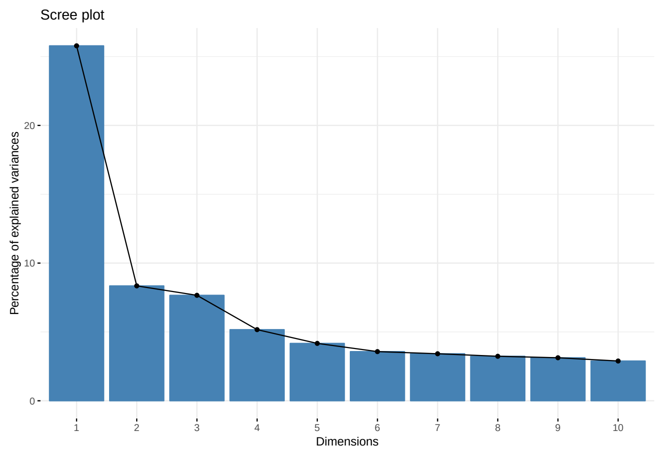

Fig. 3. Scree plot of the principal components.

D. *Evaluation Using K-nearest Neighbour*

Running the 10-fold cross-validation for each of the feature selection models yielded the best k such that $k = 5$.

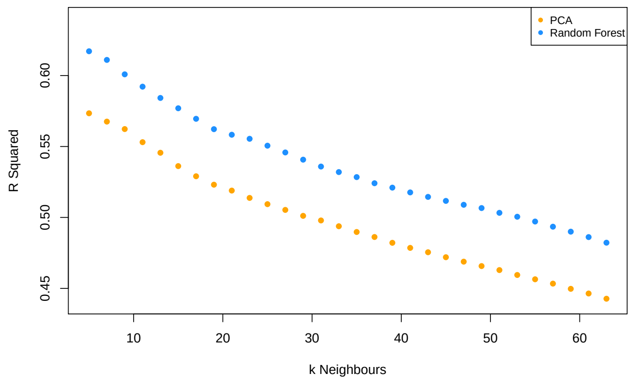

Fig. 4. R^2 value for each tested k in knn cross-validation.

Observing Figure 4, each R^2 value for each k tested during cross-validation of the KNN models can be seen. The values decrease with increasing k. The optimal R^2 values for each KNN model can be observed in Table I.

TABLE I
R^2 VALUES FOR THE KNN MODELS.

PCA	Random Forest
0.5733959	0.6170968

V. Discussion

This section will discuss the results presented in the previous section and what they indicate. It will also evaluate the methods used and discuss if there may be better ways to perform the study. Finally, it includes a discussion of how limitations such as time and knowledge may have affected the results.

A. Malware Data Results

While one could argue that the fact that only 5 % of the APKs collected were classified as malware by more than 30 % of the virus scanners is positive in general, it is negative for this study. Less malware data means less chance of creating a good model for feature selection. If a study similar to this one is to be performed in the future, an overall larger data set or just a higher percentage of malicious APKs would most likely be preferable.

B. Results from the Analysis

The results from the model validation using R^2 indicate that both the random forest and PCA models perform equally and both have a quite high R^2. However, it could be better. There are two possible reasons as to why the models are not performing optimally. One could argue that the models are simply bad, and that the supplied data is not the problem. On the other hand, the models could be performing well; the problem is that the ground truth, the data itself, is insufficient. This project made two independent feature selection models with roughly the same outcome. Thus, it is more plausible that the data supplied during feature selection is not enough to then make a good regression model with the results.

C. Indications from the Results

What can be said about the random forest result is that it indicates that permissions, file size and components affect the classification the most. This can be seen in Figure 2. Out of the component types, activities, services, receivers and providers, the number of providers has the lowest impact while the other three all have quite high importance. Activities and services provide entry points for the malware to interact with the user and also keep the application running in the background. Receivers enable the malware to be aware of events occurring when the application is not running. It is reasonable that these things are beneficial to malicious software, while content providers that manage application data and help other applications seem less useful.

It is harder to pinpoint specific features from the PCA results, as the component that represents the largest portion of the data variation contains a mix of many features. However, it is interesting how the second largest component is affected by SMS related permissions and the fifth by market related data. Perhaps this is of some importance.

As stated, both models could have performed better if it were not for the data collected. What this essentially means is that the features that the PCA and random forest models had to choose from are not enough to indicate what is malware or goodware.

D. Impact of Limitations and Possible Improvements

The study was performed during a relatively short period of time, about 2 months. The knowledge the group members were able to obtain within the domain and the size of the data set that could be collected during that time were both limited. If there had been more time, the results could probably have been improved by collecting more data and more features. There may also exist better methods for analyzing the data. Finally, this study mainly focuses on the manifest part of the APK file. As shown in Figure 1, there are more attributes that could be studied. One could also study the different activities, services, providers and receivers more closely.

VI. Related Work

In other studies, authors have used similar methods to collect data and find attributes that indicate malware. However, one big difference is that other authors have studied much larger datasets than the one in used in this project. They usually studied more features. Martín et al. studied a set of 100 000 apps and over 20 000 features, compared to our smaller dataset of 28 237 apps and 34 selected features [21]. The results are however not completely different from the ones derived from this project. Martín et al. conclude that permissions play a big role in their classification which was also one of our conclusions.

Ignacio et al. also performed a study similar to ours, and their top 15 features also included features such as number of permissions, days since last update and file size [22]. It is worth noting that their top features were developer and certificate issuer reputation, accounting for the percentage of applications that each developer and certificate issuer have labeled as malware, which was not something we considered in this project.

VII. Conclusions

The main goal of the study was to investigate the effectiveness of meta data features in telling if an APK is malicious or not and examine the presence of Android malware. From the analysis performed on the collected data, the conclusion is that the features studied can not be used to determine if an APK is malicious or not. Regarding the presence of malware, the marketplace targeted in this study, Anzhi, had a malware rate of 5 %, ultimately a low percentage. Perhaps studying more intricate attributes of APKs, larger data sets or data sets containing more malware samples would yield more fruitful results. Trying different analysis methods in the latter steps of the study might also improve the results.

References

[1] Statista, "Number of apps available in leading app stores," 2018. [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/ [Accessed: 2019-04-12].

[2] AV-TEST, "The av-test security report 2017/2018: The latest analysis of the it threat scenario," 2018. [Online]. Available: https://www.av-test.org/en/news/the-av-test-security-report-20172018-the-latest-analysis-of-the-it-threat-scenario/ [Accessed: 2019-04-12].

[3] M. H. H. G. Daniel Arp, Michael Spreitzenbarth and K. Rieck, "Drebin: Effective and explainable detection of android malware in your pocket," 02 2014.

[4] M. Lubyová, "The security of android apks," 2017. [Online]. Available: https://is.muni.cz/th/f43em/thesis.pdf [Accessed: 2019-04-12].

[5] Android, "Application fundamentals." [Online]. Available: https://developer.android.com/guide/components/fundamentals [Accessed: 2019-04-12].

[6] C. Olston and M. Najork, "Web crawling," *Foundations and Trends® in Information Retrieval*, vol. 4, no. 3, pp. 175–246, 2010. [Online]. Available: http://dx.doi.org/10.1561/1500000017

[7] X. L. Tsuhan Chen, Yufeng Jessie Hsu and W. Zhang, "Principle component analysis and its variants for biometrics," in *Proceedings. International Conference on Image Processing*, vol. 1, Sep. 2002.

[8] K. Chumachenko, "Machine learning methods for malware detection and classification." 2017. [Online]. Available: http://urn.fi/URN:NBN:fi:amk-201703103155 [Accessed: 2019-04-12].

[9] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[10] Z. Zhang, "Introduction to machine learning: k-nearest neighbors," *Annals of translational medicine*, vol. 4, no. 11, Jun 2016. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4916348/pdf/atm-04-11-218.pdf

[11] S. Menard, "Coefficients of determination for multiple logistic regression analysis," *The American Statistician*, vol. 54, no. 1, pp. 17–24, 2000. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00031305.2000.10474502

[12] T. Tjur, "Coefficients of determination in logistic regression models—a new proposal: The coefficient of discrimination," *The American Statistician*, vol. 63, no. 4, pp. 366–372, 2009. [Online]. Available: https://doi.org/10.1198/tast.2009.08210

[13] Cloudflare, "What is a denial-of-service attack?" [Online]. Available: https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/ [Accessed: 2019-04-18].

[14] VirusTotal, "Frequently asked questions." [Online]. Available: https://www.virustotal.com/en/faq/ [Accessed: 2019-04-18].

[15] F. K. Y. T. E. S. M. A. T. Y. B. S. Yuta Ishii, Takuya Watanabe and T. Mori, "Understanding the security management of global third-party android marketplaces," in *Proceedings of the 2Nd ACM SIGSOFT International Workshop on App Market Analytics*, ser. WAMA 2017. ACM, 2017, pp. 12–18. [Online]. Available: http://doi.acm.org.e.bibl.liu.se/10.1145/3121264.3121267

[16] L. Richardson, "Beautiful soup documentation," 2015. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/#beautiful-soup-documentation [Accessed: 2019-05-02].

[17] K. Lopuhin, "Scrapy frequently asked questions," 2017. [Online]. Available: http://docs.scrapy.org/en/latest/faq.html [Accessed: 2019-05-02].

[18] H. Yi, "Scrapy overview," 2018. [Online]. Available: https://docs.scrapy.org/en/latest/intro/overview.html [Accessed: 2019-05-02].

[19] A. Patel, "Event-driven networking," 2019. [Online]. Available: https://docs.scrapy.org/en/latest/topics/architecture.html#event-driven-networking [Accessed: 2019-05-02].

[20] C. S. Yang Wang, Jun Zheng and S. Mukkamala, "Quantitative security risk assessment of android permissions and applications," in *Data and Applications Security and Privacy XXVII*. Springer Berlin Heidelberg, 2013, pp. 226–241.

[21] M. A. J. K. G. G. Guillermo Suarez-Tangil, Santanu Dash and L. Cavallaro, "Droidsieve: Fast and accurate classification of obfuscated android malware," 03 2017.

[22] A. M. Ignacio Martín, José Alberto Hernández and A. Guzmán, "Android malware characterization using metadata and machine learning techniques," 07 2018.