CMPS 392: Android Malware Classification

Team Members

Mohamad Houdeib

Hadi Haidar

Description:

The goal is being able to classify APK's as being malicious (containing some malware) or safe. Our dataset is provided on Kaggle by Professor Frank Breitinger.

Implementation wise, we will use files with correctly labeled APK's (Malicious or Safe) to train our Machine Learning model and then use test files to see if our models works as it should.

Feature Engineering

As a first step, we dealt with array like features, which are permissions, intents, API's & Strings.

Permissions

First, we needed to get the set of all permission used in the dataset. An example of the format of a permission is **android.permission.READ_PHONE_STATE**. We have chosen to disregard the first part, "android.permissions" and only look at the last part "READ_PHONE_STATE".

We will explain this choice through an example. In the training set, we have found an APK that references this permission **ru.android.apps.permission.C2D_MESSAGE** while another one referenced this one **de.mcoins.fitplay.permission.C2D_MESSAGE**. Notice that **both** these permissions refer to the cloud to device messaging permission (C2D_MESSAGE), but the first part is different. In order not to treat those two permissions as different ones, we only looked at the last part.

Therefore, in the set of all permissions used in our training set, we found 863 different permissions.

One approach we could have followed is converting each of those into a one-hot feature.

However, doing this does not scale with 6336 training examples, since we have to respect the VC dimension tradeoff of $10d_{vc} \leq N$.

So instead of adding all the permissions as features, we went and figured out which are the most dominant ones among the Malicious APK's.

Here is what we found, the image below shows the 53 most popular permissions used by attackers, as well as how many times each permission was referenced by Malicious and Safe APK's.

```
Malicious: 1913  Safe: 1266     Permission: READ_PHONE_STATE
Malicious: 278   Safe: 80       Permission: READ_HISTORY_BOOKMARKS
Malicious: 991   Safe: 908      Permission: RECEIVE_BOOT_COMPLETED
Malicious: 315   Safe: 293      Permission: SYSTEM_ALERT_WINDOW
Malicious: 138   Safe: 103      Permission: ACCESS_LOCATION_EXTRA_COMMANDS
Malicious: 596   Safe: 217      Permission: RECEIVE_SMS
Malicious: 497   Safe: 397      Permission: GET_TASKS
Malicious: 415   Safe: 354      Permission: CHANGE_WIFI_STATE
Malicious: 213   Safe: 17       Permission: INSTALL_PACKAGES
Malicious: 586   Safe: 172      Permission: READ_SMS
Malicious: 382   Safe: 95       Permission: WRITE_SMS
Malicious: 736   Safe: 187      Permission: SEND_SMS
Malicious: 259   Safe: 189      Permission: READ_LOGS
Malicious: 182   Safe: 20       Permission: WRITE_APN_SETTINGS
Malicious: 564   Safe: 274      Permission: INSTALL_SHORTCUT
Malicious: 140   Safe: 41       Permission: WRITE_HISTORY_BOOKMARKS
Malicious: 245   Safe: 119      Permission: MOUNT_UNMOUNT_FILESYSTEMS
Malicious: 7     Safe: 0        Permission: USER_PRESENT
Malicious: 47    Safe: 37       Permission: RECEIVE_MMS
Malicious: 113   Safe: 78       Permission: UNINSTALL_SHORTCUT
Malicious: 69    Safe: 34       Permission: ACCESS_LOCATION
Malicious: 307   Safe: 75       Permission: RESTART_PACKAGES
Malicious: 82    Safe: 38       Permission: ACCESS_GPS
Malicious: 39    Safe: 17       Permission: ACCESS_ASSISTED_GPS
Malicious: 56    Safe: 45       Permission: CHANGE_CONFIGURATION
Malicious: 32    Safe: 31       Permission: WRITE_SECURE_SETTINGS
Malicious: 19    Safe: 12       Permission: ACCESS_DOWNLOAD_MANAGER
Malicious: 16    Safe: 0        Permission: GLOBAL_SEARCH_CONTROL
Malicious: 43    Safe: 14       Permission: RECEIVE_WAP_PUSH
Malicious: 13    Safe: 3        Permission: USES_POLICY_FORCE_LOCK
Malicious: 22    Safe: 19       Permission: ACCESS_COARSE_UPDATES
Malicious: 76    Safe: 11       Permission: DELETE_PACKAGES
Malicious: 4     Safe: 0        Permission: PAYMENT_BROADCAST_PERMISSION
Malicious: 4     Safe: 3        Permission: SET_PREFERRED_APPLICATIONS
Malicious: 9     Safe: 1        Permission: BOOT_COMPLETED
Malicious: 14    Safe: 3        Permission: BAIDU_LOCATION_SERVICE
Malicious: 18    Safe: 10       Permission: SYSTEM_OVERLAY_WINDOW
Malicious: 16    Safe: 2        Permission: ACCESS_MTK_MMHW
Malicious: 6     Safe: 0        Permission: MOUT_UNMOUNT_FILESYSTEMS
Malicious: 12    Safe: 5        Permission: PLUGIN
Malicious: 8     Safe: 1        Permission: ADD_SYSTEM_SERVICE
Malicious: 3     Safe: 0        Permission: SET_PROCESS_FOREGROUND
Malicious: 9     Safe: 8        Permission: HARDWARE_TEST
Malicious: 9     Safe: 2        Permission: READ_SECURE_SETTINGS
Malicious: 50    Safe: 48       Permission: MODIFY_PHONE_STATE
Malicious: 6     Safe: 1        Permission: ACCESS_WIMAX_STATE
Malicious: 6     Safe: 0        Permission: CHANGE_WIMAX_STATE
Malicious: 11    Safe: 4        Permission: ACCESS_CACHE_FILESYSTEM
Malicious: 6     Safe: 3        Permission: ACCESS_DOWNLOAD_MANAGER_ADVANCED
Malicious: 7     Safe: 1        Permission: ACCESS_DRM
Malicious: 6     Safe: 2        Permission: INSTALL_DRM
Malicious: 10    Safe: 1        Permission: READ_TASKS
Malicious: 3     Safe: 0        Permission: RECEIVE_SENDTO
Malicious: 6     Safe: 1        Permission: BROADCAST_WAP_PUSH
Malicious: 3     Safe: 0        Permission: ACCESS_WAKE_LOCK
Malicious: 4     Safe: 3        Permission: BROADCAST_PACKAGE_REMOVED
Malicious: 4     Safe: 0        Permission: PERMISSION_NAME
Malicious: 4     Safe: 0        Permission: FULL_SCREEN
```

For example, READ_PHONE_STATE was referenced by 1913 Malicious APK's and 1266 Safe APK's.

Notice that many of those permissions can help the attacker compromise private information related to the user and his android device.

We added each of those 58 permissions as features, since they are correlated with the target label.

.

This is how some of the added permission features look like once in the DataFrame:

permission_READ_PHONE_STATE	permission_READ_HISTORY_BOOKMARKS	permission_RECEIVE_BOOT_COMPLETED	permission_SYSTEM_ALERT_WINDOW
0	0	0	0
1	0	0	0
0	0	0	0
0	0	0	0
3	0	0	0

Intents

Even though developers can chose to name their intents however they like, we know that certain naming conventions should be followed.

We used the same approach as with permissions and got the set of all possible intent names, in order to see which ones are the most popular among the malicious APK's.

An example of the format of an intent name is **application.MAIN**. For the same reason as with permissions, we will only look at the last part, which is **MAIN,** and we will disregard the first part.

Therefore, in the set of all intents used in our training set, we found 738 different intents.

Among these 738 names, 39 are popularly used by Malicious APK's:

```
Malicious: 181   Safe: 0       Intent: BATTERY_CHANGED_ACTION
Malicious: 225   Safe: 2       Intent: SIG_STR
Malicious: 478   Safe: 146     Intent: HOME
Malicious: 493   Safe: 350     Intent: PACKAGE_ADDED
Malicious: 445   Safe: 180     Intent: USER_PRESENT
Malicious: 82    Safe: 3       Intent: DATA_SMS_RECEIVED
Malicious: 85    Safe: 82      Intent: NEW_OUTGOING_CALL
Malicious: 207   Safe: 90      Intent: PHONE_STATE
Malicious: 51    Safe: 26      Intent: SCREEN_ON
Malicious: 116   Safe: 1       Intent: INPUT_METHOD_CHANGED
Malicious: 78    Safe: 3       Intent: UMS_CONNECTED
Malicious: 76    Safe: 2       Intent: UMS_DISCONNECTED
Malicious: 3     Safe: 0       Intent: INITIALACT
Malicious: 3     Safe: 0       Intent: ACTMIOFULL
Malicious: 28    Safe: 27      Intent: BATTERY_LOW
Malicious: 3     Safe: 2       Intent: CLOSE_SYSTEM_DIALOGS
Malicious: 3     Safe: 2       Intent: UNREGISTRATION
Malicious: 6     Safe: 0       Intent: START_AGENT
Malicious: 6     Safe: 0       Intent: HEART_CODE
Malicious: 24    Safe: 7       Intent: ACTION_EXTERNAL_APPLICATIONS_AVAILABLE
Malicious: 12    Safe: 0       Intent: MEDIA_NOFS
Malicious: 14    Safe: 0       Intent: START_SMS_SERVICE
Malicious: 10    Safe: 3       Intent: REBOOT
Malicious: 2     Safe: 1       Intent: FINDGO
Malicious: 6     Safe: 0       Intent: send
Malicious: 3     Safe: 2       Intent: SEND_MESSAGE
Malicious: 2     Safe: 0       Intent: TASKSERVICE
Malicious: 9     Safe: 0       Intent: Default
Malicious: 15    Safe: 1       Intent: ACTION_SCREEN_OFF
Malicious: 2     Safe: 1       Intent: DREAMING_STOPPED
Malicious: 32    Safe: 2       Intent: SAMPLE_CODE
Malicious: 4     Safe: 0       Intent: defult
Malicious: 2     Safe: 0       Intent: ACTION_BATTERY_CHANGED
Malicious: 2     Safe: 0       Intent: SECURITY_GUARDER_SERVICE
Malicious: 2     Safe: 0       Intent: CLOSE_SYSTEM_ALARM
Malicious: 2     Safe: 0       Intent: TIME_ZONECHANGED
Malicious: 2     Safe: 0       Intent: REDOWNLOAD
Malicious: 8     Safe: 2       Intent: default
Malicious: 2     Safe: 0       Intent: SQUARE
```

```
Malicious: 181   Safe: 0       Intent: BATTERY_CHANGED_ACTION
Malicious: 225   Safe: 2       Intent: SIG_STR
Malicious: 478   Safe: 146     Intent: HOME
Malicious: 493   Safe: 350     Intent: PACKAGE_ADDED
Malicious: 445   Safe: 180     Intent: USER_PRESENT
Malicious: 82    Safe: 3       Intent: DATA_SMS_RECEIVED
Malicious: 85    Safe: 82      Intent: NEW_OUTGOING_CALL
Malicious: 207   Safe: 90      Intent: PHONE_STATE
Malicious: 51    Safe: 26      Intent: SCREEN_ON
Malicious: 116   Safe: 1       Intent: INPUT_METHOD_CHANGED
Malicious: 78    Safe: 3       Intent: UMS_CONNECTED
Malicious: 76    Safe: 2       Intent: UMS_DISCONNECTED
Malicious: 3     Safe: 0       Intent: INITIALACT
Malicious: 3     Safe: 0       Intent: ACTMIOFULL
Malicious: 28    Safe: 27      Intent: BATTERY_LOW
Malicious: 3     Safe: 2       Intent: CLOSE_SYSTEM_DIALOGS
Malicious: 3     Safe: 2       Intent: UNREGISTRATION
Malicious: 6     Safe: 0       Intent: START_AGENT
Malicious: 6     Safe: 0       Intent: HEART_CODE
Malicious: 24    Safe: 7       Intent: ACTION_EXTERNAL_APPLICATIONS_AVAILABLE
Malicious: 12    Safe: 0       Intent: MEDIA_NOFS
Malicious: 14    Safe: 0       Intent: START_SMS_SERVICE
Malicious: 10    Safe: 3       Intent: REBOOT
Malicious: 2     Safe: 1       Intent: FINDGO
Malicious: 6     Safe: 0       Intent: send
Malicious: 3     Safe: 2       Intent: SEND_MESSAGE
Malicious: 2     Safe: 0       Intent: TASKSERVICE
Malicious: 9     Safe: 0       Intent: Default
Malicious: 15    Safe: 1       Intent: ACTION_SCREEN_OFF
Malicious: 2     Safe: 1       Intent: DREAMING_STOPPED
Malicious: 32    Safe: 2       Intent: SAMPLE_CODE
Malicious: 4     Safe: 0       Intent: defult
Malicious: 2     Safe: 0       Intent: ACTION_BATTERY_CHANGED
Malicious: 2     Safe: 0       Intent: SECURITY_GUARDER_SERVICE
Malicious: 2     Safe: 0       Intent: CLOSE_SYSTEM_ALARM
Malicious: 2     Safe: 0       Intent: TIME_ZONECHANGED
Malicious: 2     Safe: 0       Intent: REDOWNLOAD
Malicious: 8     Safe: 2       Intent: default
Malicious: 2     Safe: 0       Intent: SQUARE
```

For each of these 39 intent names, we created a one-hot feature.

If an APK has one of those intent names, we set the value to 1, otherwise set it to 0

This is how some of the intents features look like once in the DataFrame

intent_CLOSE_SYSTEM_ALARM	intent_TIME_ZONECHANGED	intent_REDOWNLOAD	intent_default	intent_SQUARE
0	0	0	1	0
0	0	0	0	0

Strings

For String arrays, we will not follow the same approach as with permissions and intents.

Logically speaking, **malicious APK's typically have less strings/overall code/application features than safe APK's**. This makes sense, as malicious applications are usually quite simple and hackers typically don't go through the hassle of building large scale applications. They rather build simple yet attractive apps to use as weapons for their malicious purposes.

Therefore, we converted the array of strings in each APK into a simple continuous feature that tells us how many strings the APK used.

We ended up finding a high correlation between the number of Strings used by an APK, and whether or not it is malicious.

This is how the String feature looks like once in the DatFrame

Strings
49675
57289
52522

API’s

As for the arrays of API's, we figured that it is unlikely to find a correlation between the calling of a specific API and whether or not an APK is malicious.

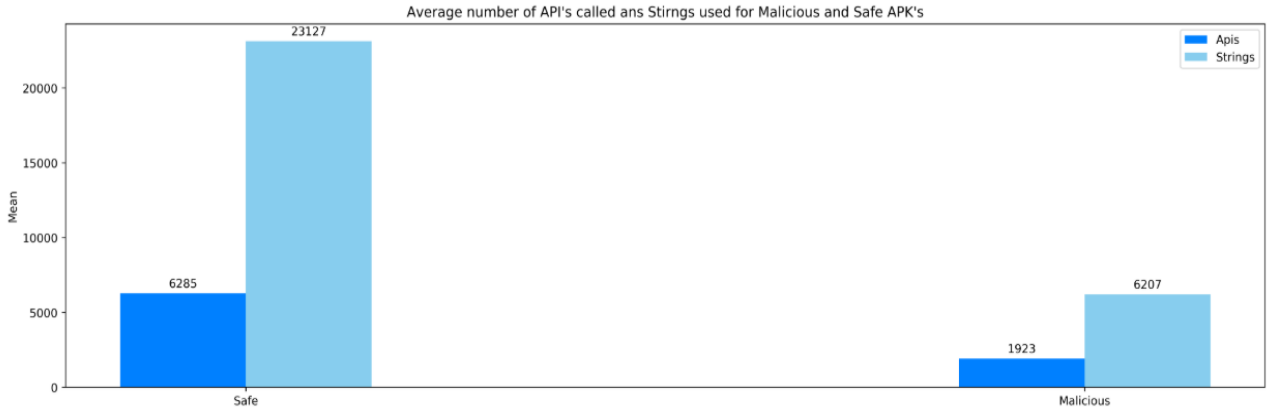
Instead, we followed the same approach as with Strings (above), and figured that a Malicious Application will call less API's since it is usually simpler/has less code.

Therefore, we converted the array of API's in each APK into a simple continuous feature that tells us how many API's the APK has called.

This is how the Apis feature looks like once in the DataFrame:

Apis
13848
15570
13493

The following bar plot offers a visualization of the average number of API’s called and the average number of Strings used by Malicious and by Safe APK’s.



Notice that for API's the average number goes is 6285 API's for safe APK's – as opposed to 1923 for malicious ones.

Moreover, for Strings, the average number of Strings used is 23,127 for Safe APK's –as opposed to 6207 for Malicious APK's.

The bar plot confirmed that Malicious APK's have less overall code than Safe APK's. It also confirmed the correlation between the two features and the target label.

We're now done with Array-Like features. Let's deal with the others (Sha1, Sha256, Md5, Package Version, Date, FileSize and Package Name)

We dropped Sha1, Sha256 and Md5 since there is no correlation between an encryption key and whether or not an APK is malicious. We also **dropped Package Version** since it was empty for all APK's.

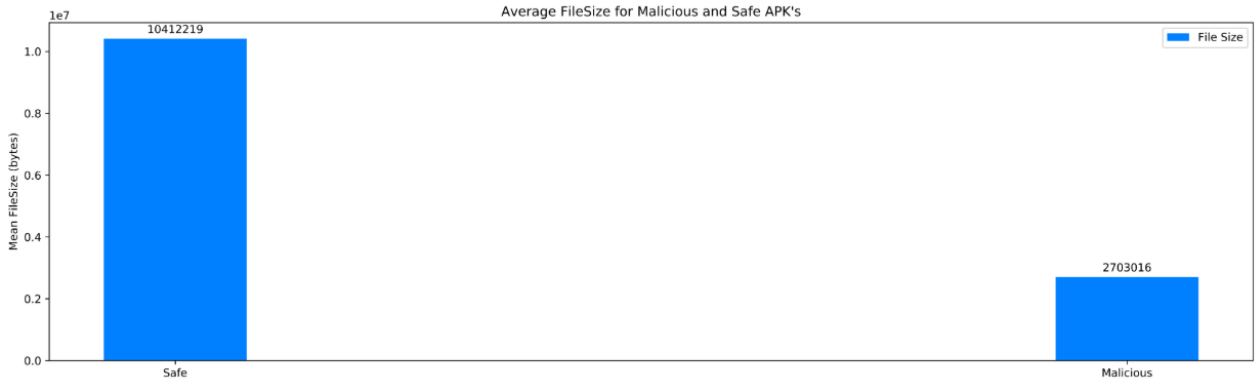
File Size

We kept File Size and **figured that since Malicious APK's have less code, their overall size would be smaller than Safe APK's.**

File Size is a continuous feature, and looks like this in the DataFrame:

File Size
31295093.0
69367204.0
5585016.0

This Bar Plot displays the average File Size for Malicious and Safe APK's



The plot confirmed that Safe APK's are typically much larger in size than Malicious APK's. It also confirmed the correlation between APK size and the target label.

Package Name

An APK package name is comprised of two parts: {extension}.{name}.

Since the name part is a variant unique to each APK, we did not make use of it.
However, there is only a limited set of extensions that a developer can use. After some inspection, we found certain extensions that are heavily used by Malicious APK's, which are shown in the image below:

```
Malicious : 17  Safe: 7 Extension :cn
Malicious : 3   Safe: 1 Extension :HamsterSuper
Malicious : 4   Safe: 0 Extension :emr
Malicious : 3   Safe: 0 Extension :JawbreakerSuper
Malicious : 39  Safe: 0 Extension :ad
Malicious : 2   Safe: 0 Extension :keis
Malicious : 2   Safe: 0 Extension :sunkay
Malicious : 4   Safe: 0 Extension :tj
Malicious : 17  Safe: 0 Extension :tk
Malicious : 3   Safe: 1 Extension :io
Malicious : 3   Safe: 0 Extension :googleanzhi
Malicious : 2   Safe: 0 Extension :miao
Malicious : 4   Safe: 0 Extension :tp5x
Malicious : 2   Safe: 0 Extension :gomez
Malicious : 3   Safe: 0 Extension :system
Malicious : 4   Safe: 0 Extension :solution
Malicious : 2   Safe: 0 Extension :content
Malicious : 4   Safe: 0 Extension :wbs
Malicious : 2   Safe: 0 Extension :android
Malicious : 2   Safe: 0 Extension :wode
Malicious : 2   Safe: 0 Extension :iec
Malicious : 2   Safe: 0 Extension :exam
Malicious : 2   Safe: 0 Extension :FiveChessSuper
Malicious : 2   Safe: 0 Extension :cmp
Malicious : 2   Safe: 0 Extension :qq
```

Quite a few malicious APK's have used extensions that are uncommon. We see an obvious correlation between those extensions and the target.

Hence, we will add a one-hot feature for each of the extensions seen on the left.

Also, we will add a one-hot feature called pkg_name_other that is set to 1 in case an APK uses none of these extensions.

Date

We found a correlation between Day and the target, so we kept the Day and dropped all other information related to Date.

Malicious

All that remains is to label encode the target using LabelEncoder. If an APK is malicious, we will convert the value of the 'Malicious' column to 1, otherwise convert it to 0.

After pre-processing and engineering the original features, we ended up with a total of 127 features. They are all listed below:

```
Apis
Strings
FileSize
Malicious
permission_READ_PHONE_STATE
permission_READ_HISTORY_BOOKMARKS
permission_RECEIVE_BOOT_COMPLETED
permission_SYSTEM_ALERT_WINDOW
permission_ACCESS_LOCATION_EXTRA_COMMANDS
permission_RECEIVE_SMS
permission_GET_TASKS
permission_CHANGE_WIFI_STATE
permission_INSTALL_PACKAGES
permission_READ_SMS
permission_WRITE_SMS
permission_SEND_SMS
permission_READ_LOGS
permission_WRITE_APN_SETTINGS
permission_INSTALL_SHORTCUT
permission_WRITE_HISTORY_BOOKMARKS
permission_MOUNT_UNMOUNT_FILESYSTEMS
permission_USER_PRESENT
permission_RECEIVE_MMS
permission_UNINSTALL_SHORTCUT
permission_ACCESS_LOCATION
permission_RESTART_PACKAGES
permission_ACCESS_GPS
permission_ACCESS_ASSISTED_GPS
permission_CHANGE_CONFIGURATION
permission_WRITE_SECURE_SETTINGS
permission_ACCESS_DOWNLOAD_MANAGER
permission_GLOBAL_SEARCH_CONTROL
permission_RECEIVE_WAP_PUSH
permission_USES_POLICY_FORCE_LOCK
permission_ACCESS_COARSE_UPDATES
permission_DELETE_PACKAGES
permission_PAYMENT_BROADCAST_PERMISSION
permission_SET_PREFERRED_APPLICATIONS
permission_BOOT_COMPLETED
permission_BAIDU_LOCATION_SERVICE
permission_SYSTEM_OVERLAY_WINDOW
permission_ACCESS_MTK_MMHW
```

```
permission_MOUT_UNMOUNT_FILESYSTEMS
permission_PLUGIN
permission_ADD_SYSTEM_SERVICE
permission_SET_PROCESS_FOREGROUND
permission_HARDWARE_TEST
permission_READ_SECURE_SETTINGS
permission_MODIFY_PHONE_STATE
permission_ACCESS_WIMAX_STATE
permission_CHANGE_WIMAX_STATE
permission_ACCESS_CACHE_FILESYSTEM
permission_ACCESS_DOWNLOAD_MANAGER_ADVANCED
permission_ACCESS_DRM
permission_INSTALL_DRM
permission_READ_TASKS
permission_RECEIVE_SENDTO
permission_BROADCAST_WAP_PUSH
permission_ACCESS_WAKE_LOCK
permission_BROADCAST_PACKAGE_REMOVED
permission_PERMISSION_NAME
permission_FULL_SCREEN
intent_BATTERY_CHANGED_ACTION
intent_SIG_STR
intent_HOME
intent_PACKAGE_ADDED
intent_USER_PRESENT
intent_DATA_SMS_RECEIVED
intent_NEW_OUTGOING_CALL
intent_PHONE_STATE
intent_SCREEN_ON
intent_INPUT_METHOD_CHANGED
intent_UMS_CONNECTED
intent_UMS_DISCONNECTED
intent_INITIALACT
intent_ACTMIOFULL
intent_BATTERY_LOW
intent_CLOSE_SYSTEM_DIALOGS
intent_UNREGISTRATION
intent_START_AGENT
intent_HEART_CODE
intent_ACTION_EXTERNAL_APPLICATIONS_AVAILABLE
intent_MEDIA_NOFS
intent_START_SMS_SERVICE
```

```
intent_REBOOT
intent_FINDGO
intent_send
intent_SEND_MESSAGE
intent_TASKSERVICE
intent_Default
intent_ACTION_SCREEN_OFF
intent_DREAMING_STOPPED
intent_SAMPLE_CODE
intent_defult
intent_ACTION_BATTERY_CHANGED
intent_SECURITY_GUARDER_SERVICE
intent_CLOSE_SYSTEM_ALARM
intent_TIME_ZONECHANGED
intent_REDOWNLOAD
intent_default
intent_SQUARE
Day
pkg_name_cn
pkg_name_HamsterSuper
pkg_name_emr
pkg_name_JawbreakerSuper
```

```
pkg_name_ad
pkg_name_keis
pkg_name_sunkay
pkg_name_tj
pkg_name_tk
pkg_name_io
pkg_name_googleanzhi
pkg_name_miao
pkg_name_tp5x
pkg_name_gomez
pkg_name_system
pkg_name_solution
pkg_name_content
pkg_name_wbs
pkg_name_android
pkg_name_wode
pkg_name_iec
pkg_name_exam
pkg_name_FiveChessSuper
pkg_name_cmp
pkg_name_qq
pkg_name_other
```

Models, Techniques & Results

KNN

We thought that for a binary classification task, using KNN's may work since we saw that most malicious APK's had highly similar patterns, and the same goes for Safe APK's, therefore we pictured that we would obtain two clusters of data points that are quite close to each other in distance.

KNN gave us a 72.8% accuracy, which is not bad, but we thought we could do better since our features were highly correlated to the target label.

SVM

We figured that the use of rbf kernels would serve us well, and it kind of did. Using SVM's with the RBF kernel gave us an accuracy of 75.52%, which is an increase from the last model.

Decision Trees

Since we have many one hot encoded features, we thought that the question-like nature of decision trees would serve us well, and it did. Decision Trees significantly increased our accuracy to 86.75%.

Adaboost

We tried the ensemble learning method Adaboost with the default classifier (decision trees) and we tried with Logistic Regression. We see the number of estimators to 100 (we tried others but 100 gave us the best result) and with Logistic Regression we got an accuracy of 97.94%.

Gradient Boosting – Best Model

Ensemble learning methods gave us good results. Gradient boosting performed quite well and gave us a 98.4% accuracy.

Naïve Bayes

Naïve Bayes gave us a 61.95% accuracy. This is most likely due to the fact that NB assumes independence among the features. In the given dataset, this does not hold. Recall we mentioned that a Malicious APK will have less code, hence less API's, Strings and a smaller File size. These three features are somewhat related to each other, and this might have affected the result.

XGBoost

Since Gradient Boosting worked so well, we thought it would be a good idea to use XGBoost, since it generalizes better, hence giving better out-of-sample results. It gave us however a lower validation accuracy of 98.1%.