# Detecting Malicious Android Game Applications on Third-Party Stores using Machine Learning

Thanaporn Sanamontre[1], Vasaka Visoottiviseth[1], and Chaiyong Ragkhitwetsagul[1]

[1]Faculty of Information and Communication Technology,
Mahidol University, Nakhon Pathom, Thailand
thanaporn.sa4@student.mahidol.edu, vasaka.vis@mahidol.edu,
chaiyong.rag@mahidol.edu

**Abstract.** Due to Android's flexibility in installing applications, it is one of the most popular mobile operating systems. Some Android users install applications from third-party stores even though they have the official application store, Google Play. These third-party stores usually have the mod version and the self-proclaimed original applications, which can be repackaged applications. Applications on these third-party stores can introduce security risks because of the non-transparent alteration and uploading processes. In this research, we inspect 492 Android applications from ten third-party stores for repackaged applications using information of APK files and a token-based code clone detection technique. We also classify repackaged applications as benign or malicious and categorize malicious applications into twelve malware categories. For the malware classification, we use machine learning techniques, including Random Forest, Decision Tree, and XGBoost, with the CCCS-CIC-AndMal-2020 Android malware dataset. Finally, we compare the results with VirusTotal, a well-known malware scanning website.

**Keywords:** Android Security, Android Malware Detection, Repackaged Applications, Machine Learning

## 1 Introduction

Android OS has the most market share in the mobile OS from the first and second quarter of 2023 [1]. Based on Statista [2], The application category Android users downloaded the most is the game category. Usually, users download game applications from Google Play, the official application store for Android devices. To protect users from malicious applications, Google Play has a feature called Google Play Protect [3], which automatically detects and protects Android devices from potentially harmful applications from both Google Play itself and from other sources. However, some people download Android applications from untrusted sources because the applications are not available on Google Play or are regionally restricted, or they need to use the other version of applications. Unfortunately, downloading games from third-party stores poses security risks due to their non-transparent alteration and uploading processes and lack of security features. Malicious actors can repackage applications, turn them into malware, and trick users to install these altered malicious applications. Installing them can lead to malicious activities, such as collecting sensitive data without user consent, stealing users' credentials, injecting hidden malware, adware, or backdoor, and compromising and harming the devices. Therefore, detecting these malicious applications from third-party stores is essential.

In this paper, we inspect ten selected third-party stores and their game applications including the self-proclaimed original and the mod versions of games. Regarding the self-proclaimed original game applications, we investigate whether they are repackaged applications by comparing hash values, a token-based code clone detection, and other attributes of the APK files. Next, we classify and categorize both repackaged and mod versions of game applications into twelve malware categories using machine learning techniques, including Random Forest, Decision Tree, and XGBoost, with the CCCS-CIC-AndMal-2020 Android malware dataset [4][5].

The contributions of this paper are as follows.

- We inspect a total of 557 selected game applications: 65 original games from Google Play and 492 games from selected third-party stores. Among 492 applications, there are 196 self-proclaimed original applications, which we investigate whether they are repackaged.
- We use machine learning techniques with a recent Android malware dataset to classify altered game applications, including the repackaged version and mod versions of games, whether they are benign or malicious applications. Malicious applications are classified into twelve categories based on malware characteristics.
- We compare the results with VirusTotal [6], a famous malware scanning website.

This research paper's outlines are as follows. **Section 2** describes the background and related works. The proposed work details are in **Section 3**. **Section 4** shows the experiments and results. Lastly, we conclude and discuss our findings in **Section 5**.

## 2 Background and Related Work

This section provides the definitions, the background, and the related work.

### 2.1 Types of Altered Android Game Applications

On third-party Android application stores, there are games stating they are the same as the original ones on Google Play and altered games. Some modifications are with malicious intentions, while some of them are not. The alteration processes of applications affect their appearance, source code, functionalities, and behaviors. There are two categories of altered applications based on owners' authorization. Authorized altered applications have the owners' approval, such as games with updates and UI improvement. However, most application alterations do not have the proper authorization from the owners and violate the copyrights and distribution rights. Games are modified for some reasons, for example, accessing the premium features without paying, using the advertisement-free applications, and cheating the games by gaining the advantage against them. Examples of altered game applications are cracked games, repackaged applications, and mod versions of applications. Below are the definitions of the repackaged applications and the mod versions of applications.

- **Repackaged Applications**

Third-party developers decompile an original APK file, do unauthorized alteration, recompile, repackage it into a Repackaged Application, and distribute it on third-party stores. These repackaged applications have significant security risks because the alteration usually contains malicious intents. Malicious actors use these applications' appearance and functionalities as baits to trick users to download and install them. On third-party stores, applications claiming they are the same as the original ones on Google Play are called "Self-Proclaimed Original Application." Therefore, users will not know if they download "Repackaged Applications."

- **Mod Version of Game Applications**

Mod versions of game applications are created by enthusiasts and third-party developers who modify the original applications and then distribute them to third-party stores. They may modify the games to enhance gameplay, add new features, or cheat games. Even though mod versions of games are usually not modified with malicious intent, using applications from untrusted sources always carries security risks. Third-party stores always tell users that the games are the "Mod Versions" in their descriptions. Thus, users always know whether they install the mod versions.

### 2.2 CCFinderSW

Semura et al. [7][8] proposed CCFinderSW, a token-based code clone detection tool. CCFinderSW uses a lexical analysis mechanism. Users can select programming languages such as Python, Java, and Ruby by selecting options for comments and reserved words of each language. CCFinderSW can detect Type-1 and Type-2 clone pairs. According to Roy and Cordy [9], Type-1 pairs are pairs with identical code fragments, while Type-2 pairs are similar to Type-1, but the variable names, values, and identifiers are changed.

### 2.3 Related Work

To prevent downloading malicious applications, some researchers proposed approaches for detecting Android malware. Chen et al. [10] proposed an Android malware detection with text-based classification and static analysis technique. After extracting static features, they converted them into word vectors for text classification using the BiLSTM processing model. They also used the DPCNN and Fasttext algorithms for performance comparison. However, the BiLiSTM has the highest accuracy score at 97.47% and the F1-score at 97.43%. Jaiswal et al. [11] provided a study of differences between benign, malicious, and clone applications based on system calls and behaviors. They also proposed a gaming malware detection system using dynamic analysis techniques. They found that some system calls, such as *stat64* and *llseek*, are more frequently called in malicious games. Moreover, some system calls that are not in the original games, such as *stat64*, *pread63*, and *brk*, but appear in the clone ones. The clone applications also ask for high-level permissions that the original ones do not require.

Several papers use the CCCS-CIC-AndMal-2020 dataset for Android malware detection. DIDroid [12] provided a detection system using 2D images with a deep learning algorithm based on CNN. They extracted static features, created vectors, selected the features, converted them into 2D gray images, and used the convolutional layers for training and testing models. DIDroid has an accuracy of 93.36%. The other provided work regarding this dataset is the EntropLyzer. The EntropLyzer [13] is an entropy-based detection system using dynamic features. Its entropy algorithm is Shannon entropy. They used Naive Bayes, Support Vector Machine, Random Forest, and Decision Tree algorithms for training the models. They analyzed and compared the entropy values of their features before and after rebooting and visualized their behavioral changes. They found that Decision Tree with the F1-score of 98.3% has the best performance for classifying malware categories.

## 3 Proposed Work

This section provides an explanation of the workflow, including the repackaged application detection and the malware detection and categorization.

### 3.1 List of Target Applications

We select top 200 free Android game applications for this study. We collected the Android games from the following ten third-party stores based on their popularity:

- APKAward
- APKCombo
- APKMody
- APKPure
- APKVision
- HappyMod
- LiteAPKs
- LuckyModAPK
- ModYolo
- PlayMods

The following are the criteria for selecting the target applications.

- Original games must have at least five third-party stores with the same versions.
- If the third-party stores have both self-proclaimed original and mod versions of applications, we will select the mod versions as our target.
- If there are many mod versions of an application, we will select the mod version with the latest updated date as our target.
- We will filter out the selected applications with the updated date and versions conflicted with the original ones on Google Play.

**Table 1.** Numbers of target applications in each category.

| | Amount | Description |
|---|---|---|
| Original Applications | 65 | Filtered games from the top 200 free game applications on Google Play |
| Self-Proclaimed Original Applications | 196 | Applications on third-party stores which claimed that they are the same as the original ones on Google Play |
| Mod Version of Applications | 296 | Modified games on third-party stores for adding features and cheating games. |
| Total | 557 | |

After the collection and filtering, we retrieve 557 games for this study. We classify target game applications into three categories: original, self-proclaimed original, and mod versions. Table 1 shows the amount of target applications in each category.

## 3.2 Analysis Workflow

We propose an approach for inspecting the self-proclaimed original games whether they are repackaged. Then, we classify and categorize mod versions and the repackaged games whether they are in one of twelve malware categories or benign. Therefore, our proposed approach consists of two main processes: (1) the repackaged application detection and (2) the malicious application detection and categorization. Fig. 1 shows the workflow of the proposed approach in detail. The analysis steps of the proposed approach are as follows.

1. Download and decompile the target applications.
2. Select a self-proclaimed original game and pair it with the original one for repackaged application detection processes.
3. Conduct the repackaged application detection on the pair.
4. Sort the result from the experiment. We will store the repackaged games for the malicious applications detection and categorization section. However, if the game is not a repackaged, record it as a "Benign Original Application."
5. If any self-proclaimed original games are remained, select a new one and repeat the process. If not, go to the next step.
6. Conduct malicious application detection and categorization on the repackaged and mod versions of games.
7. Sort the result from the experiment. If the application is benign, record it as a "Benign Application" with its type. However, if the application is malicious, classify it as a "Malicious Application" with its malware category.

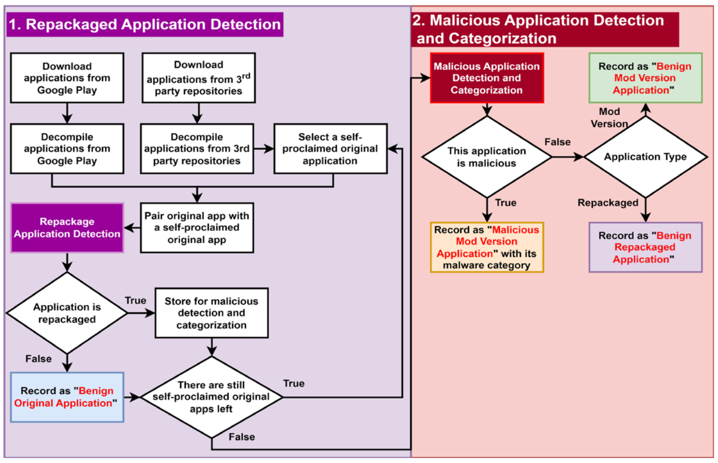
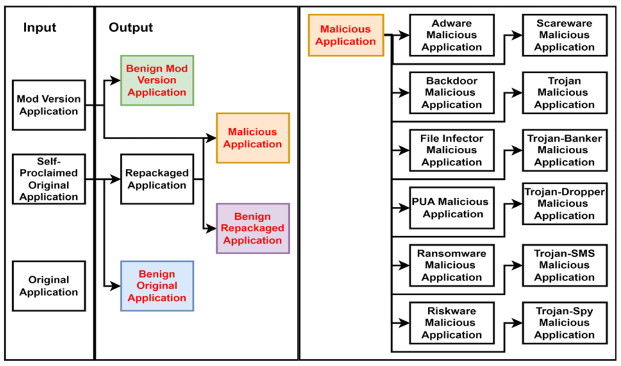

1. Repackaged Application Detection
Download applications from Google Play
Download applications from 3rd party repositories
Decompile applications from Google Play
Decompile applications from 3rd party repositories
Select a self-proclaimed original application
Pair original app with a self-proclaimed original app
Repackage Application Detection
Application is repackaged
True
Store for malicious detection and categorization
False
Record as "Benign Original Application"
There are still self-proclaimed original apps left
True
False
2. Malicious Application Detection and Categorization
Malicious Application Detection and Categorization
This application is malicious
False
Application Type
Mod Version
Record as "Benign Mod Version Application"
True
Record as "Malicious Mod Version Application" with its malware category
Repackaged
Record as "Benign Repackaged Application"

Fig. 1. Analysis workflow of the proposed approach


Input
Output
Mod Version Application
Self-Proclaimed Original Application
Original Application
Benign Mod Version Application
Repackaged Application
Benign Original Application
Malicious Application
Benign Repackaged Application
Malicious Application
Adware Malicious Application
Backdoor Malicious Application
File Infector Malicious Application
PUA Malicious Application
Ransomware Malicious Application
Riskware Malicious Application
Scareware Malicious Application
Trojan Malicious Application
Trojan-Banker Malicious Application
Trojan-Dropper Malicious Application
Trojan-SMS Malicious Application
Trojan-Spy Malicious Application

(a) (b)

Fig. 2 (a) Inputs and Outputs of the system (b) Malicious Application Output

As shown in Fig. 2 (a), there are three possible inputs and four main possible outputs. Three inputs include the original applications from Google Play, the self-proclaimed original, and the mod versions of games. On the other hand, four main possible outputs are as follows:

- Benign Original Applications (The same as the original games)
- Benign Repackaged Applications (The repackaged games but not malicious)
- Benign Mod Version Applications (The mod versions of games but not malicious)
- Malicious Applications (The malicious games with possible twelve categories)

In case of malicious applications, they are categorized into one of twelve malware categories as depicted in Fig. 2(b).

### 3.3 Repackaged Application Detection

This section explains the method used to detect repackaged applications from the self-proclaimed original games in detail. Fig. 3 illustrates the workflow of the repackaged application detection process. The following steps are how the process works.

1. Compute and compare the APK files' hash values.
2. Get and compare the number of files in both APK files.
3. Get and compare the names of all files in both APK files. We store all files with different names and additional files, except files for Android Application Binary Interface (ABI).
4. Compute and compare all file hash values in both APK files. We store all files with different hash values for the next step.
5. Conduct token-based code clone detection using CCFinderSW [7][8] on the JAVA files with different hash values.

The following are conditions for reporting that a self-proclaimed original application is a "Repackaged Application."

1. The APK files' hash values, numbers of files, and filename are not the same. Moreover, not all additional and different files are for Android ABI.
2. There are files having different hash values when comparing with the original one.

On the other hand, the conditions for approving that a self-proclaimed original game application is a "Benign Original Application" are as follows.

1. Their APK files' hashes, numbers, and names of files, and each file's hash values are the same compared to the original APK file.
2. Even though the APK files' hashes, numbers of files, and filenames are not the same, all additional and different files are for Android ABI, and all files' hash values are the same as their original ones.

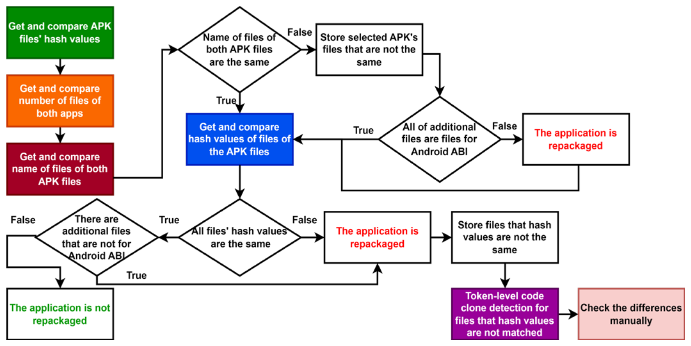

Get and compare APK files' hash values
Get and compare number of files of both apps
Get and compare name of files of both APK files
Name of files of both APK files are the same
False
Store selected APK's files that are not the same
True
Get and compare hash values of files of the APK files
True
All of additional files are files for Android ABI
False
The application is repackaged
False
There are additional files that are not for Android ABI
True
All files' hash values are the same
False
The application is repackaged
Store files that hash values are not the same
True
The application is not repackaged
Token-level code clone detection for files that hash values are not matched
Check the differences manually

Fig. 3 Workflow of the Repackaged Application Detection Process

## 3.4 Malicious Application Detection and Categorization

This section presents the method used for detecting the malicious repackaged and mod versions of games and categorizing them into twelve categories. We use codes and unique lists provided by AndroidAppLyzer [14] for feature extraction and the CCCS-CIC-AndMal-2020 Android malware dataset [4][5] for training the machine learning models. Characteristics of this dataset are as shown in Table 2. The machine learning algorithms include Random Forest, Decision Tree, and XGBoost.

**Table 2.** Number of malware samples in the CCCS-CIC-AndMal-2020 dataset.

| Malware Categories | Numbers of Families | Numbers of Samples |
|---|---|---|
| Adware | 48 | 47,210 |
| Backdoor | 11 | 1,538 |
| File Infector | 5 | 669 |
| No Category | - | 2,296 |
| PUA | 8 | 2,051 |
| Ransomware | 8 | 6,202 |
| Riskware | 21 | 97,349 |
| Scareware | 3 | 1,556 |
| Trojan | 45 | 13,559 |
| Trojan-Banker | 11 | 887 |
| Trojan-Dropper | 9 | 2,302 |
| Trojan-SMS | 11 | 3,125 |

Note that this dataset also contains 200k benign Android applications. The following steps are the workflow of the malicious application detection and categorization process.

1. Extract and create vectors of the APK file's 9,503 static features. There are 9,491 features from the provided unique lists, including permissions, actions, and categories. The rest are the additional information on the APK file as listed below.
   - Numbers of Icons
   - Numbers of Audio
   - Numbers of Videos
   - Size of the App
   - Numbers of Activities
   - Numbers of Meta-data
   - Numbers of Services
   - Numbers of Permissions
   - Numbers of Categories
   - Numbers of Actions
   - Numbers of Providers
   - Numbers of Receivers
2. Train the machine learning models with the CCCS-CIC-AndMal-2020 Android malware dataset. However as from Table 2, our number of samplings in each category of dataset is unbalanced. Therefore, for training we data-sampled each label in the dataset into 3,200 and 15,000 samples for training the model.
3. Use the models for malware detection and categorization on the target applications.

## 4 Experimental Results

### 4.1 Hardware and Software Specifications

We used Oracle VM VirtualBox version 7.0.4 to virtualize Kali Linux version 2023.2 VM with 2-core CPU and 8-GB RAM for conducting the repackaged application detection. Kali Linux has GNU bash 5.2.15, Python 3.10.9, JADX 1.4.7, and CCFinderSW 1.0 installed. For malicious application detection and categorization, we used Katana GF66 11 UG, CPU 8 cores, and RAM 16 GB with Windows 10.

### 4.2 Repackaged Application Detection

We experimented on the pair of a self-proclaimed original game and its original one on Kali Linux. We used JADX tool to decompile the APK files, SHA256 for the hash algorithm, and CCFinderSW to inspect Java files with different hash values from their original ones. The information of APK files used in the experiment are (1) The hash value of APK file, (2) numbers of files in each APK file, (3) filenames, and (4) hash values of each file. The expected outputs are the "Benign Original Application" and the "Repackaged Application." However, there is the "Exceptional Application," which are applications that has the same numbers of files, the same filenames, the same hash values of each file, but the hash values of APK files are not the same.

We conducted experiments on 196 self-proclaimed original games from third-party stores with 65 original games from Google Play. Table 3 shows results based on each third-party repository. We found 93 repackaged applications out of 196 self-proclaimed original games, which is 47.45%. We also found 19 exceptional applications. Only 84 applications are benign original applications. The store with the highest percentage of repackage applications is ModYolo which has 100%, while the store with the lowest percentage of application is LiteAPKs which has 33.33%.

Fig. 4 depicts the results of the repackaged application detection in the pie chart. In conclusion, there are 93 repackaged applications out of 492 targets from third-party stores, which is 18.90%. The benign original and the exceptional applications are 17.07% and 3.86% out of the targets, respectively.

**Table 3.** Repackaged applications on each third-party repository.

| Third-Party Stores | Self-Proclaimed Original Applications | Repackaged Applications | Benign Original Applications | Exceptional Applications | Repackaged Application Percentage |
|---|---|---|---|---|---|
| APKAward | 9 | 7 | 1 | 1 | 77.78% |
| APKCombo | 56 | 24 | 28 | 4 | 42.86% |
| APKMody | - | - | - | - | 0.00% |
| APKPure | 65 | 22 | 35 | 8 | 33.85% |
| APKVision | 11 | 6 | 2 | 3 | 54.55% |
| HappyMod | 14 | 6 | 6 | 2 | 42.86% |
| LiteAPKs | 9 | 3 | 5 | 1 | 33.33% |
| LuckyModAPK | - | - | - | - | 0.00% |
| ModYolo | 6 | 6 | 0 | 0 | 100.00% |
| PlayMods | 26 | 19 | 7 | 0 | 73.08% |
| **TOTAL** | **196** | **93** | **84** | **19** | **47.45%** |

* APKMody and LuckyModAPK do not have any self-proclaimed original games.

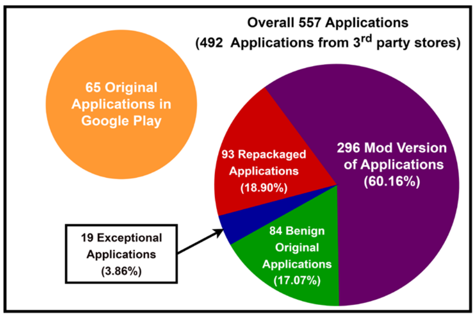


Fig. 4 Target applications after Repackaged Application Detection

## 4.3 Malicious Application Detection and Categorization

We experimented on 389 game applications, 93 of which are repackaged games and 296 of which are mod versions of games. Regarding extracting static features, we used the AndroidAppLyzer's code and unique lists, including permissions, actions, and categories lists [14]. Then, we used machine learning models including Random Forest (RF), Decision Tree (DT), and XGBoost (XGB) algorithms that are trained on the CCCS-CIC-AndMal-2020 Android malware dataset to classify them as malware or benign applications. For training the model, to balance the dataset from each category, we data-sampled each label in the dataset into 3,200 and 15,000 samples, respectively. Table 4 shows each algorithm and its performance. Criteria for performance evaluation of each algorithm are the accuracy and the F1-score. The algorithm with the highest accuracy and F1-score for both 3,200 and 15,000 samples is the Random Forest algorithm. It has accuracy and F1-score of 91.80% and 91.85% for 3,200 samples. Regarding 15,000 samples, it has accuracy and F1-score of 94.78% and 94.83%. Using 15,000 samples resulted in higher accuracy and F1-score, but all three algorithms have similar performances.

**Table 4.** Algorithms and their performances.

| ML Algorithms | 3,200 Samples | | 15,000 Samples | |
|---|---|---|---|---|
| | Accuracy | F1-Score | Accuracy | F1-Score |
| Random Forest (RF) | 91.80% | 91.85% | 94.78% | 94.83% |
| Decision Tree (DT) | 90.29% | 90.31% | 93.98% | 94.03% |
| XGBoost (XGB) | 90.99% | 91.04% | 93.12% | 93.17% |

We then tested 389 game applications with the trained models. Table 5 shows the malware classification results of each machine learning algorithm. There are 13 outputs: one benign and twelve malware categories. The decision tree algorithm detected the most numbers of malware, with 84 malwares for 3,200 samples and 39 malwares for 15,000 samples. Using the decision tree with 3,200 samples, the most numbers of malware is the SMS category. However, when using decision tree with 15,000 samples, Adware is the category with the most number of malware samples.

**Table 5.** Malware classification results of each algorithm.

| Label | 3,200 Samples | | | 15,000 Samples | | |
|---|---|---|---|---|---|---|
| | RF | DT | XGB | RF | DT | XGB |
| Adware | 0 | 14 | 0 | 0 | 30 | 0 |
| Backdoor | 0 | 0 | 0 | 0 | 0 | 0 |
| Banker | 2 | 6 | 2 | 1 | 2 | 2 |
| Benign | 383 | 305 | 381 | 387 | 350 | 384 |
| Dropper | 0 | 3 | 0 | 0 | 0 | 0 |
| File Infector | 0 | 0 | 0 | 0 | 0 | 0 |
| PUA | 0 | 8 | 0 | 0 | 0 | 3 |
| Ransomware | 0 | 3 | 0 | 0 | 0 | 0 |
| Riskware | 4 | 0 | 0 | 1 | 5 | 0 |
| SMS | 0 | 49 | 0 | 0 | 0 | 0 |
| Scareware | 0 | 1 | 2 | 0 | 1 | 0 |
| Spy | 0 | 0 | 0 | 0 | 0 | 0 |
| Trojan | 0 | 0 | 4 | 0 | 1 | 0 |
| **TOTAL (Malware)** | **6** | **84** | **8** | **2** | **39** | **5** |

We then verified our malware detection results by comparing with the APK scanning results with VirusTotal [6], a popular online malware-scanning website containing results from 62-65 security vendors. However, from 389 applications, there are 37 APK files we could not upload to VirusTotal due to their excessive size.

Therefore, we could verify only 352 applications. We counted an application as malware if one or more security vendor on VirusTotal detected the malware. Table 6 summarizes our machine learning results and VirusTotal results. Results from VirusTotal stated that 57.33% of the targets are malware. However, the highest percentage of malware from our machine learning results is only 21.59%, when using the decision tree algorithm with 3,200 samples.

**Table 6.** Results of the experiment and VirusTotal.

| Label | VirusTotal | 3,200 Samples | | | 15,000 Samples | | |
|---|---|---|---|---|---|---|---|
| | | RF | DT | XGB | RF | DT | XGB |
| Benign | 129 | 383 | 305 | 381 | 387 | 350 | 384 |
| Malware | 223 | 6 | 84 | 8 | 2 | 39 | 5 |
| Malware Percentage | 57.33% | 1.54% | 21.59% | 2.06% | 0.51% | 10.03% | 1.29% |

Table 7 compares the percentage of the detection when compares the results from our machine learning (ML) to the VirusTotal. According to Table 7, the Random Forest and the XGBoost algorithm when using 15,000 samples could detect the highest percentage of benign applications with 99.22%. On the other hand, the decision tree algorithm with 3,200 samples could detect the highest percentage of malware applications with 22.87%.

**Table 7.** Comparison and percentage of the results and VirusTotal.

| Label | 3,200 Samples | | | 15,000 Samples | | |
|---|---|---|---|---|---|---|
| | RF | DT | XGB | RF | DT | XGB |
| ML Result = Benign<br>VirusTotal = Benign | 124/129<br>(96.12%) | 101/129<br>(78.29%) | 123/129<br>(95.35%) | **128/129**<br>**(99.22%)** | 122/129<br>(94.57%) | **128/129**<br>**(99.22%)** |
| ML Result = Benign<br>VirusTotal = Malware | 222/223<br>(99.55%) | 172/223<br>(77.13%) | 221/223<br>(99.10%) | 222/223<br>(99.55%) | 202/223<br>(90.58%) | 222/223<br>(99.55%) |
| ML Result = Malware<br>VirusTotal = Benign | 5/129<br>(3.88%) | 28/129<br>(21.71%) | 6/129<br>(4.65%) | 1/129<br>(0.78%) | 7/129<br>(5.43%) | 1/129<br>(0.78%) |
| ML Result = Malware<br>VirusTotal = Malware | 1/223<br>(0.45%) | **51/223**<br>**(22.87%)** | 2/223<br>(0.90%) | 1/223<br>(0.45%) | 21/223<br>(9.42%) | 1/223<br>(0.45%) |

## 5 Conclusion and Discussions

Installing applications from third-party stores comes with security risks. Users do not know whether games on third-party Android application stores are benign, repackaged, or malicious. Therefore, we proposed a method to detect the repackaged applications from the self-proclaimed original games. We found that 93 applications are repackaged applications, which is 47.45% of the self-proclaimed original games. Then, we detected and categorized repackaged applications and mod versions of games whether they were malicious by using three machine learning techniques including Random Forest, Decision Tree, and XGBoost with the CCCS-CIC-AndMal-2020 Android malware dataset. We also used VirusTotal for the result verification. The Random Forest algorithm has the highest accuracy and F1-score at 94.78% and 94.83%, using 15,000 samples, respectively. The decision tree algorithm detected the most numbers of malware which is 84 from 389 application or 21.59%. However, VirusTotal detected 223 malwares from 389 targets, which is 57.33%. The decision tree with 3,200 samples has the highest percentage of malware detection which is 22.87% when comparing to the VirusTotal detection results. However as mentioned earlier, VirusTotal uses the scanning results from around 62-65 security vendors, but among our total of 223 applications identified as malware by VirusTotal, 138 applications or around 61.88% of them are detected as malware by only 1-2 virus security vendors. Therefore, some results from VirusTotal may be false positives and need further investigation. For the future work, to increase the accuracy and performance of this research, we should also use the dynamic analysis technique.

## References

1. Statcounter, Mobile Operating System Market Share Worldwide. https://gs.statcounter.com/os-market-share/mobile/ worldwide/ #monthly-202301-202306
2. Statcounter, Number of mobile app downloads in the Google Play Store in 1st quarter 2021 and 1st quarter 2022, by category. https://www.statista.com/statistics/1331430/google-play-app-downloads-by-category
3. Google Play, Use Google Play Protect to help keep your apps safe and your data private. https://support.google.com/ android/answer/2812853?hl=en
4. University of New Brunswick, CCCS-CIC-AndMal-2020. https://www.unb.ca/cic/datasets/andmal2020.html
5. D'hooge, L.: CCCS-CIC-AndMal-2020. https://www.kaggle.com/datasets/dhoogla/cccscicandmal2020
6. VirusTotal. https://www.virustotal.com/gui/home
7. Semura, Y., Yoshida, N., Choi, E., Inoue, K.: CCFinderSW: Clone Detection Tool with Flexible Multilingual Tokenization. In: 2017 24th Asia-Pacific Software Engineering Conference (APSEC) (2017) 654-659
8. Semura, Y.: CCFinderSW. https://github.com/YuichiSemura/CCFinderSW
9. Roy, C.K., Cordy, J.R.: A Survey on Software Clone Detection Research. In: School of Computing TR 2007-541 (2007)
10. Chen, M., Zhou, Q., Wang, K., Zeng, Z.: An Android Malware Detection Method Using Deep Learning based on Multi-features. In: 2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA) (2022) 187-190
11. Jaiswal, M., Malik, Y., Jaafar, F.: Android gaming malware detection using system call analysis. In: 2018 6th International Symposium on Digital Forensic and Security (ISDFS) (2018) 1-5
12. Rahali, A., Lashkari, A.H., Kaur, G., Taheri, F., Gagnon, F., Massicotte, F.: DIDroid: Android Malware Classification and Characterization Using Deep Image Learning. In: 2020 10th International Conference on Communication and Network Security (ICCNS '20) (2020) 70-82
13. Keyes, D.S., Li, B., Kaur, G., Lashkari, A.H., Gagnon, F., Massicotte, F.: EntropLyzer: Android Malware Classification and Characterization Using Entropy Analysis of Dynamic Characteristics. In: 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS) (2021) 1-12
14. Lashkari, A.H.: AndroidAppLyzer. https://github.com/ahlashkari/AndroidAppLyzer