Artificial Intelligence Review (2026) 59:48
https://doi.org/10.1007/s10462-025-11452-y

Androbank: the impact of API levels on mobile malware detection

Milan Oulehla[1] · **Ladislav Dorotík**[1] · **Zuzana Komínková Oplatková**[1]

Received: 18 November 2024 / Accepted: 11 November 2025

Abstract
Android is the most widely used operating system, making it a prime target for mobile malware, leading to data breaches and financial losses (e.g., Dark Herring). To address these issues, AI-based forensic tools are crucial for investigating security incidents, but their accuracy depends on high-quality mobile malware datasets. While dynamic analysis has limitations, recent research has shifted towards static analysis and AI-based methods for malware detection. However, there are three key challenges: lack of reproducibility, low dataset quality, and bias in AI datasets. This paper focuses on an overlooked bias—the incorrect API Level distribution in malware datasets. Such bias skews AI detection results, making them appear effective in tests but less applicable in real-world scenarios. To highlight the importance of dataset quality, three case studies on API Level Analysis were conducted, showing how biased datasets can distort detection results. To address this, the paper introduces methods and terms like Delayed Interception, Dataset of guaranteed quality, API Milestones, AndroBank, and Sample Unification, which aim to enhance dataset reliability and improve AI-based mobile malware detection.

Keywords Android OS · APK · Decompilation · Detection · Malware · Static analysis

1 Introduction

Smartphones have rapidly become one of the most widespread technological devices in the history of humankind. The number of mobile devices has even surpassed the number of computers. This fact can be demonstrated in the statistics from 2022. The global shipment of computers in 2022 was only around 310 million units (Gartner 2021), while the number of smartphone shipments in the same year reached a much higher total of 1,205.5 million units (IDC 2023). This significant difference in the number of smartphones shipped highlights their growing importance.

✉ Ladislav Dorotík
l_dorotik@utb.cz

[1] Faculty of Applied Informatics, Tomas Bata University in Zlín, Nám. T. G. Masaryka 5555, 760 01 Zlín, Czech Republic

Published online: 02 December 2025

However, the popularity of the Android operating system is double-edged. On the one hand, there is a large user base and many useful applications. On the other hand, devices running on the Android OS have become a primary target for mobile malware creators. One of the most extensive mobile malware campaigns was Dark Herring. Over a 21 month long period, the cybercriminals responsible for Dark Herring created and distributed almost 470 applications on the Google Play Store. The earliest incident dates to March 2020, and the most recent to November 2021. The download statistics indicate that over 105 million Android devices worldwide were infected with this scamware, making them vulnerable to the campaign and potentially resulting in significant financial losses for the victims. The perpetrators of this cybercrime campaign have established a steady stream of illicit revenue, earning millions of dollars in recurring income each month. The total amount of stolen funds could exceed hundreds of millions of dollars (YASWANT 2022). Security incidents like this are classified as criminal offenses due to the significant financial losses caused, thus necessitating investigation by law enforcement agencies.

The increasing complexity of digital devices and the software that runs on them has led to a significant rise in the volume of digital data available for forensic analysis. However, this increase in the quantity of data has made it impractical to perform static, or dynamic analysis manually. To address this challenge, automated forensic tools have become essential in malware mitigation process (Gonçalves et al. 2022). Research practice implies that the best investigative results are achieved by tools based on artificial intelligence methods using static analysis. In the case of dynamic analysis, there are several limitations that make it quite challenging to use dynamic analysis methods for automated mobile malware investigation, but they can provide valuable results. The following list brings the formulation of major limitations:

1. An emulation of the responses: Malware often tries to connect to the attacker's server while running. It does not perform malicious activity if it does not get the correct response.
2. Execution of malware: Executing suspicious applications can cause damages such as DDoS attack (distributed denial-of-service), remote code execution, or other criminal activity.
3. Anti-Analysis and Anti-Debugging: These techniques are employed by mobile malware and can make them resistant to dynamic analysis (e.g., runtime detection in a mobile phone emulator).
4. The emulation of human interaction: It is difficult to emulate natural user behavior, such as the correct call order of onClick methods, etc.

To develop effective detection tools using artificial intelligence (AI), it is crucial to have high quality datasets that include both mobile malware samples and samples of uninfected legitimate applications. Available malware datasets are often reused; they also frequently contain obsolete or poor-quality samples used in research, as described by RASHED (2021). Therefore, using these "raw" datasets without further sample processing is problematic as it may distort the detection results.

The current offer of datasets usable for mobile malware detection can be divided into three categories according to their accessibility:

1. The publicly available datasets that are the most common and have the highest probability of finding various defects.
2. Datasets created by academic researchers that may have restricted access. Academic datasets are essential for research that does not involve the commercial sphere. Generally, these datasets are often better quality but rarely up-to-date.
3. Professional datasets which are exclusive to antimalware companies and other commercial research entities. Such datasets contain the most recent samples. However, the access to them is strictly limited.

This paper addresses serious detection problems that can occur when unprocessed datasets are used without a proper methodology to create a Dataset of guaranteed quality. To offer a solution, we have developed a methodology to create guaranteed quality datasets which is fully described in the Sect. 3.5. The proposed methodology will ensure that the AI methods that use these datasets created via static analyses will not have distorted detection results due to poor processing. The proposed methodology is also part of the presented system, AndroBank, which is fully elaborated and described in Sect. 4.

It is important to note that proper preprocessing and static analysis guarantee the feasibility of experiments. However, they do not ensure the correct composition of the dataset regarding bias, which is crucial for real-world applicability and is demonstrated in Sect. 5.3.2. The proposed and presented AndroBank system (Sect. 4) was used in the paper to deal with these issues.

2 Related work

The problem of biased datasets used for mobile malware detection via AI is relatively new. For this reason, there are not many articles that deal with this problem. However, quality papers identifying this phenomenon exist. From the point of view of our research team, probably the most important articles are Miranda et al. (2022), Lin et al. (2022), and Kan et al. (2024).

One of the important issues related to datasets is to have balanced data that are not biased. From what we have been able to ascertain, the dataset bias is also covered in the paper (Miranda et al. 2022). While this paper makes several contributions to the field, our paper goes deeper into the issue of "Time of Incoherence of samples". One of the main contributions was the experiment with an imbalanced dataset regarding Time Incoherence, which confirms the hypothesis about the possible distortion of results. Our paper also introduces important API milestones based on several years of security experience and experimentation with a contrasting dataset.

Other important research related to the issue of dataset bias is presented in Lin et al. (2022), which asks three important research questions:

- To what extent does VirusTotal threshold in malware labelling affect the malware evaluation results?
- How does the malware family distribution in the dataset affect the detection result?
- How does the sample partition methods in evaluation affect the detection results?

Although the issues described in Lin et al. (2022) are beneficial, they only deal with biased malware datasets: “The only variable in each group of our experiment is how we select and construct the 1,000 malicious apps” (Lin et al. 2022). Our approach to this problem is different. Our research team aims to design an AI system that can handle even low-level features such as vectorized API calls. To detect differences between malware and legitimate applications that are not biased by the presence of different versions of support libraries and structures, it is necessary to establish specific relationships between malware and legitimate apps datasets. Therefore, the authors of this paper introduce methods such as API milestones and Delayed Interception. In short, it is necessary to detect malicious behavior, not differences in the internal architecture of the tested samples, such as different target APIs, etc.

An excellent paper (Kan et al. 2024) addresses two crucial aspects that can cause bias in malware classification using machine learning. First is a spatial bias, which occurs when there are unrealistic assumptions about the ratio of goodware to malware in the dataset (Kan et al. 2024). To prevent this undesirable phenomenon, it is necessary to keep the ratio of goodware to malware as close to reality as possible. Second is temporal bias caused by incorrect time splits of data, leading to unrealistic configurations. The research in Kan et al. (2024) focuses on temporal bias from a training and testing perspective: To prevent this type of bias, all the objects in the training must be strictly temporally precedent to the testing ones (Kan et al. 2024). Our approach is different since it focuses on the temporal relationship between malware samples and clean, legitimate applications. In paper (Kan et al. 2024), timestamps are used, which are created from dex_date, the app compilation date (Allix et al. 2016). Preliminary analyses of our research have shown that dex_date can be unreliable because it often contains a fixed/default value (e.g., 01-01-1981 01:01). In addition, even today, it is possible to develop applications for older Android operating system versions that contain multiple vulnerabilities. Moreover, even today, developing a malicious application for older Android operating system versions with weaker security mechanisms is possible. The application architecture will be older, but dex_date will be new. For the reason mentioned above, we used minSdkVersion and targetSdkVersion in our analysis, which are much more accurate for our purposes. The temporal aspect is also addressed in our research through Delayed Interception.

3 Fundamental terminology

Some key terms are explained in this section for better understanding. At the same time, new concepts such as “Delayed Interception” and “Dataset of guaranteed quality” in the mobile malware detection field are defined.

3.1 APK

APK files can be considered from two perspectives. From an Android point of view, it is an installation and distribution archive file containing application logic, essential files such as AndroidManifest.xml, and additional files such as various kinds of assets (resources, images, etc.) (Google 2023). From a cybersecurity and AI perspective, these files are the primary source of information and are subject to analysis (see Sect. 4).

3.2 SHA-256

SHA-256 is the function that generates a fixed-length, unique digital fingerprint of data. This fingerprint is called a digest and is usually used to verify the integrity of the data without disclosing its content (SHA-256 2023). Since each file has a unique digest, it can also be used for mobile malware research as:

- an identifier for mobile malware samples,
- a tool to detect duplicate malware samples (two or more identical samples obtained from different sources may have different names, but the SHA-256 will always remain the same).

3.3 Delayed interception

New legitimate applications are available for static and dynamic analysis as soon as they are released on a distribution platform. However, malware samples are intercepted with a delay, resulting in malware samples having older API levels than legitimate applications. An example of this phenomenon is discussed in Case study 2 in detail (Sect. 5.2), and is shown in Fig. 5.

3.4 Contrasting dataset

A Contrasting dataset comprises old malware and modern legitimate samples, resulting in a very high detection ratio but is unusable for real-world deployment. This sample-gathering method is the most basic; thus, it is quite common in Android malware detection. Unfortunately, as presented in the Experiment Sect. 5.3, this can lead to distortion of detection results. The Contrasting dataset is a phenomenon closely related to Delayed Interception. The Contrasting dataset could be generally related to datasets with an imbalance in API level distribution.

3.5 Dataset of guaranteed quality

The quality of the dataset is the most critical factor for creating AI tools. However, the quality of publicly available mobile malware datasets is highly variable (see subsect. 5 - "API Level Analysis of available datasets" for more details). Moreover, these are incompatible regarding sample names, dataset structure, sample requirements, etc. The Dataset of guaranteed quality definition is necessary for both the development of analytical software and the reproducibility of results performed on these datasets.

There is currently no standard for sample processing and mobile malware dataset creation. Therefore, authors, based on their experiences, suggest the following criteria that should be considered:

- The dataset should only contain genuine APK packages. All false APK files (i.e., executables for other platforms) with the ".apk" extension must be identified and excluded from the dataset.
- APK samples must contain critical parts such as the DEX file(s), AndroidManifest.xml,

etc. The size of the critical files must be larger than 0B.

- Packages whose decompilation failed or was only partially successful must be excluded from the dataset. Decompilation of all APK packages must be entirely successful. Successful decompilation ensures the feasibility of static analysis on all samples in the Dataset of guaranteed quality.
- All samples must be checked for malignancy based on which samples are labeled malicious or legitimate.
- The dataset must not contain duplicate samples.
- Each sample has an individual directory, which:
  - is located in the root directory of the dataset,
  - has a name that is the SHA-256 value of its APK file.
- A sample directory contains:
  - APK file (named original.apk),
  - the directory containing the decompiled APK file and its resources (named DECOMPILED_APK).

3.6 Android API levels

The Android operating system is evolving very dynamically. Since 2008, 34 versions of this system have been created. The security mechanisms and resources vary significantly from version to version of this operating system. For example, an improved permissions model was introduced in 2015 (Android 6.0 Marshmallow) (Google 2023), and Android Protected Confirmation was added in 2018 (Android 9.0 Pie) (Google 2023). As a result, malware strategies vary depending on which operating system versions they were developed for. In the context of mobile malware static analysis, the API Level is crucial because it can be used to identify the target operating system version of an examined sample. The relation between versions of Android OS and API Levels is shown in Table 1. The API Levels of the examined applications can be retrieved using the sdkVersion (which is minSdkVersion) and targetSdkVersion attribute values; those can be obtained using the AAPT2 tool (Android Developers 2022), a part of the Android SDK Command line tools.

Specifically, the experiments described below used AAPT2 with a dump and badging parameters. According to the Android documentation, targetSdkVersion is an integer rep-

Table 1 Android versions, API levels, and years of release

Android version	Android API level	Year of release
Android 5	21–22	2014–2015
Android 6	23	2015
Android 7	24–25	2016
Android 8	26–27	2017
Android 9	28	2018
Android 10	29	2019
Android 11	30	2020
Android 12	31–32	2021
Android 13	33	2022
Android 14	34	2023

resenting the API Level that the application targets. If not set, the default value equals that given to minSdkVersion (Google 2023). The targetSdkVersion value specifies the API Level, which can be interpreted as the primary operating system version for the examined sample. It is a version of the operating system that the application is tested against and optimized for. The sdkVersion value can be interpreted as an API Level identifier specifying a compatible operating system's oldest version (minSdkVersion).

The actual percentage distribution of Android operating system versions varies by the source of information. A comparison of three sources respected by the scientific community can be seen in Fig. 1. The following sources were used: Android reach and devices (Google Play Console 2023), GlobalStats (Mobile & Tablet Android Version Market Share Worldwide 2023), and Statista (StatCounter 2022).

3.7 API milestones

The research described in this paper includes an analysis of some significant changes in the security mechanisms of the Android operating system. Introducing powerful new security mechanisms into Android OS has been termed a milestone.

These milestones are shown in Fig. 2, where the vertical lines present the milestones across the different versions of the Android operating system.

A list of the milestones that have been defined:

- Android 5.0 (API Level 21): This version introduced ART runtime, which replaced the older Dalvik runtime. That was a critical change regarding performance and securi-

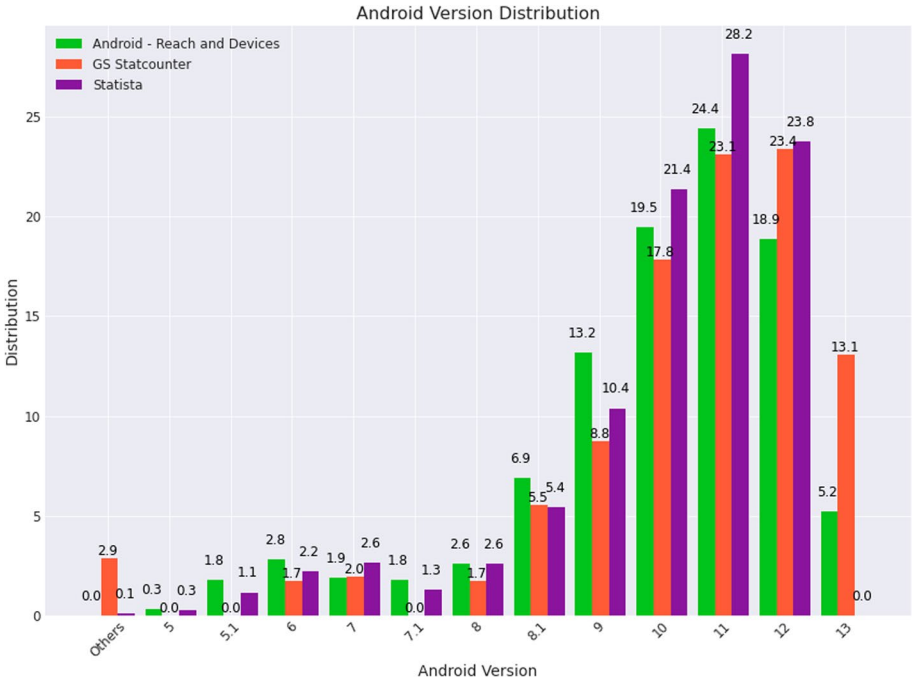

Fig. 1 Android versions distribution—comparison

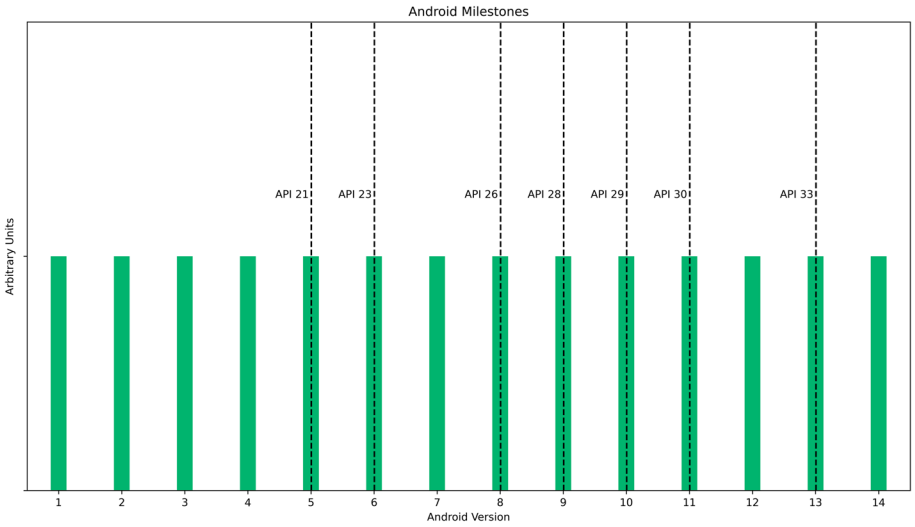

Fig. 2 Android milestones

ty. Also, this version of Android OS introduced several functionalities that may affect how malware potentially uses the API. These affect connectivity (Bluetooth or multi-networking), display capturing, video capturing, WebView changes, and JobScheduler API, which allows apps to schedule background tasks to run at specified intervals (Google 2023).

- Android 6.0 (API Level 23): Android 6 was released with an improved runtime permission model. This was a significant change, as users can now grant permissions individually, and they can revoke granted permissions at any time. The old permission model only allowed acceptance of all permissions as a whole. In addition, the user could only accept the permission once by completing the installation process. After finishing the installation, permissions could not be removed. Another change was App linking, which allowed developers to link their applications with their domain, potentially allowing malware to redirect users to harmful websites. Also, new battery-saving features were released, which could change how malware works with API (Google 2023).
- Android 8.0 (API Level 26): Android released its version number 8.0 in 2017, and for malware creators, it meant that new strategies would be needed, as "Background Executions Limits" and "Background Location Limits" were introduced. Also, the permission model has changed. "Before Android 8.0 (API level 26), if an app requested permission at runtime and the permission was granted, the system also incorrectly granted the app the rest of the permissions that belonged to the same permission group and were registered in the manifest." (Google 2023). However, from API 26, each permission from the group can be granted individually.
- Android 9.0 (API Level 28): Also, Android version 9.0 affected how malware behaves on a mobile device. New background restrictions and "Android Protected Confirmation" were introduced, which changed how users provided sensitive information to applications (Google 2023).
- Android 10 (API Level 29): Scoped Storage strictly partitions external storage so that

apps can no longer read or write outside their own sandbox without additional permission flags. GOOGLE (2025) A dedicated `ACCESS_BACKGROUND_LOCATION` permission limits silent location tracking. Also, the access to hardware identifiers (IMEI, serial) was limited. These restrictions reduce the attack surface, resulting in significant architectural changes in malicious and benign applications. GOOGLE (2025)

- Android 11 (API Level 30): introduced key changes into the permission model. Android 11 presents one-time permissions, which means the user can grant permission for just a once and then the process repeats. In addition, granted permissions newly resets after longer time without application being used by the user. GOOGLE (2025) Which has direct impact on stealth malicious applications.
- Android 12/12 L (API Level 31/32): The Android 12 tightened the attack surface by adding hardware privacy indicators when the camera or microphone is used. This significantly impacts malicious applications that silently abuse this functionality after granting permissions. With newly introduced Bluetooth permissions, the applications will change their structure once again. Besides that, only applications with known signatures may invoke AlwaysOnHotwordDetector. Also, a new feature aims to mitigate overlay attacks commonly used in Banking malware. Overall, Android 12 changes constrain surveillance functions and overlay-based attacks. GOOGLE (2025)
- Android 13 (API Level 33): In previous versions of the Android operating system, there has been a massive abuse of Accessibility services, which have become a powerful tool for threat actors to enable whole interaction with a mobile device. For this reason, the use of Accessibility services has been restricted in Android 13. Although we were unable to locate this security feature in the Android 13 release notes, many clues are pointing out that Android 13 introduced this change, such as tutorials on how to bypass the new security measures, even experts addressing this issue on their web pages, such as ESET company, which explicitly states that APK has to be downloaded from the official Google Play Store to be able to use the Accessibility service (ESET 2024). Besides this, Android 13 also introduced several API changes that modify the approach to creating mobile applications, such as notification permission, granular media picker, and more. GOOGLE (2025)
- Android 14 (API Level 34): While a media picker was introduced in the previous version, Android 14 made another improvement in privacy by allowing users to grant permission to access specified photos and videos. Android 14 introduced several important changes, such as additional restrictions on starting activities from the background. GOOGLE (2025)

4 AndroBank

The AndroBank system was created to automate the processing and analyses of infected and legitimate applications running on the Android operating system. This system originated from our research needs, such as lack of reproducibility or any standard approach to obtain the Dataset of guaranteed quality. The system processes APK samples in three stages, as shown in Fig. 3.

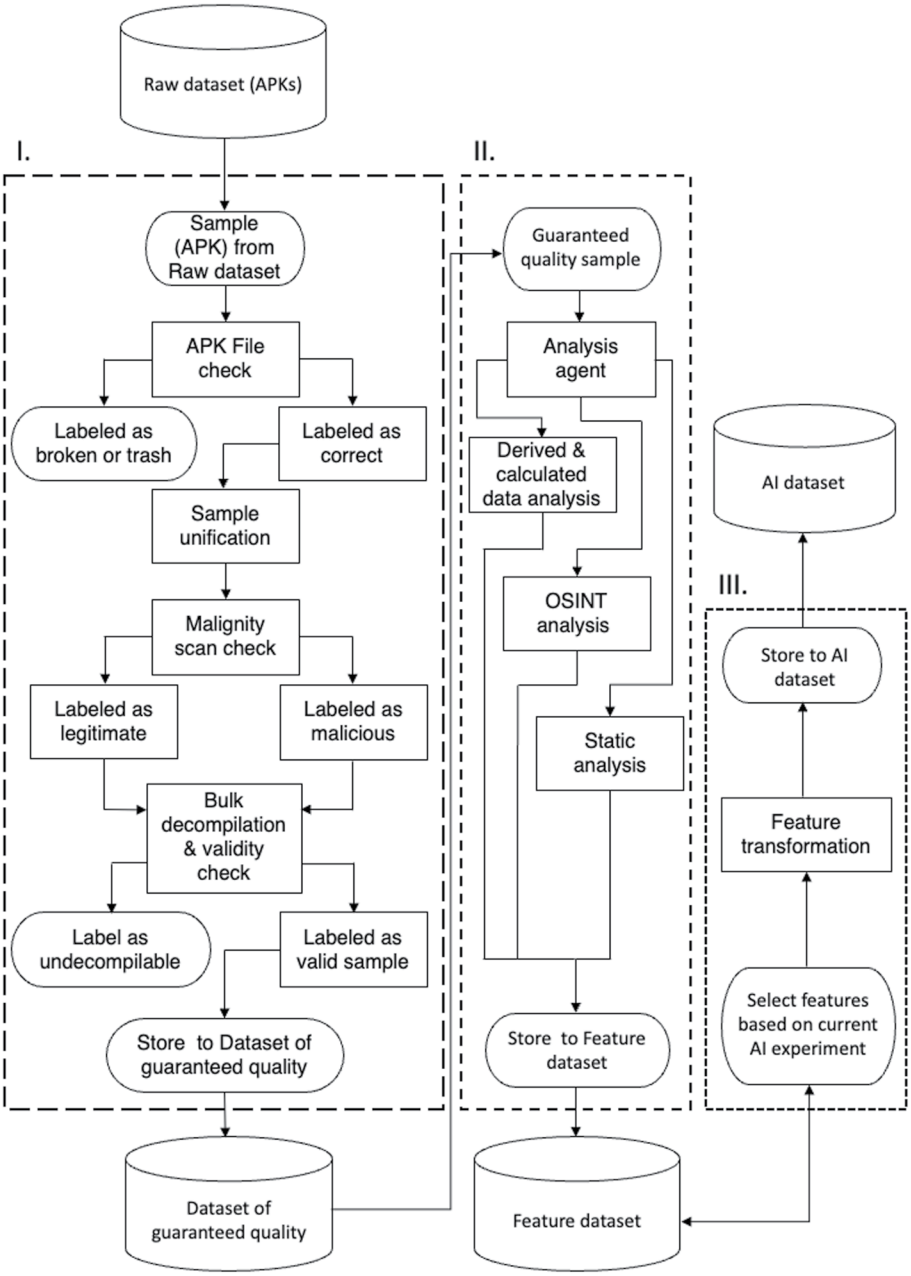

Fig. 3 AndroBank: overview diagram

4.1 The first stage (I.)

The input to the AndroBank system is a *Raw dataset*. The Raw dataset should contain samples in APK file format since it is the distribution standard for the Android platform (Google 2023). Such APK samples can be gathered from multiple sources, including file share servers, Torrents (via BitTorrent Protocol), OSINT, AndroZoo (Allix et al. 2016), etc. Since the APK can be obtained from any source on internet, their quality vary greatly (the quality is not guaranteed). Thus, samples need to be checked and further processed.

APK File check is performed on each sample from the Raw dataset to ensure it is an APK file. The sample must contain the essential files, such as AndroidManifest.xml, DEX file(s), etc. Samples that successfully pass this check are labeled as correct. If the APK File check fails, the sample is labeled as broken or trash.

Since Raw dataset samples may have different naming conventions or file extensions, it was decided to add the *Sample unification* feature to AndroBank. Sample Unification is an essential feature since it allows results to be reproduced by researchers worldwide. Two or more identical samples obtained from different sources may have different names, but the SHA-256 value will always remain the same. For this reason, each sample is placed in a directory named by its SHA-256 value, as described in Sect. 3.5.

A *Malignity scan check* was performed using the VirusTotal (VirusTotal 2023) service. After completing this check, the sample is labeled malicious or legitimate based on the result. The malignity scan check serves to validate the proclaimed sample malignity. Since samples in Raw datasets are often labeled as malignant or benign, this labeling may have some inaccuracies, so it cannot be relied upon, and additional checking is necessary. This step is required to properly evaluate the experiments conducted as part of our research.

Next is *Bulk decompilation & validity check*, designed to process multiple APK samples simultaneously. Decompilation of all samples must be entirely successful. Samples of which decompilation failed or was only partially successful were labeled undecompilable and excluded from further processing. The output of each APK sample is a directory called DECOMPILED_APK containing all the decompiled contents.

Among other tasks, the check performs the following verification in the DECOMPILED_APK directory:

- AndroidManifest.xml has to contain an activity element with an intent-filter having android:name="android.intent.action.MAIN" and android:name="android.intent.category.LAUNCHER",
- the directory must contain at least one XML layout used by an instance of the Activity class for GUI creation,
- the directory must include one or more smali files,
- etc.

As presented in Fig. 3, only samples from the Raw dataset that successfully pass all procedures of the first stage (I.) are stored in a *Dataset of guaranteed quality* with the following structure where each sample of this dataset is represented by a SHA-256 directory:

```
Dataset_of_guaranteed_quality
|
|---0bf...b46
|   |---original.apk
|   |---DECOMPILED_APK
|
|---1d3...d8e
|   |---original.apk
|   |---DECOMPILED_APK
|
...
...
...
|
|---ff9...b75
    |---original.apk
    |---DECOMPILED_APK
```

4.2 The second stage (II.)

The input to the second stage (II. in Fig. 3) is the Dataset of guaranteed quality. The *Analysis agent* sequentially processes guaranteed samples from the dataset and performs three analyses on them in parallel:

1. Derived & calculated data analysis,
2. OSINT (Open Source Intelligence) analysis,
3. Static analysis.

Derived & calculated data analysis is essential because it creates new data that is not part of the analyzed sample. For example, the component hierarchy of the sample was derived by parsing the XML elements such as activities, broadcast receivers, content providers, services, and intent-filters in the AndroidManifest.xml file. Another example can be the total number of virtual methods found in all smali files of a tested sample.

The OSINT analysis searches all publicly available information about the currently analyzed sample. This can include online scanners such as Jotti's malware scan (Jotti 2023), APKPure Signature Verification Online (APKPure 2023), etc. Although it is not a critical analysis, OSINT often provides data that can significantly support other types of analysis. For this reason, the OSINT analysis module must process each sample from the Dataset of guaranteed quality.

The static analysis module is a critical component of AndroBank since it produces the primary data source for AI methods. The module provides, for example, analysis of API calls, creation of call graphs, etc. The results of all three analysis modules are stored in an SQL database called the Feature dataset. Samples from this dataset are extracted via SQL queries where SHA-256 calculated from the corresponding APK file is an identifier.

4.3 The third stage (III.)

The very last stage is aimed at the creation of ad hoc *AI datasets* that are suitable for the application of AI methods. The input to the third stage is the *Feature dataset* obtained by extensive second stage analyses. It is crucial to highlight that the AI dataset is created ad hoc based on current research needs.

As shown in Fig. 3, the third stage (III.) is divided into three steps. The initial step is the *feature selection* of specific samples. The second step is *feature transformation* tailored to particular machine learning algorithms because some algorithms can perform better with different transformation approaches. When features are appropriately selected and transformation is done, the final step is *storing the AI dataset* in various formats, such as CSV, JSON, and XML.

These three steps are designed to create highly scalable AI datasets. It means that when a dataset is made for a complex AI detection system, a large number of features are selected. On the other hand, if a narrowly focused use case is being solved, then only a small group of particular features is selected.

4.4 AndroBank positioning

This section provides a theoretical and practical grounding of AndroBank's role in relation to relevant actors and instruments, namely AndroZoo (Allix et al. 2016) and TESSERACT (Kan et al. 2024). AndroZoo is an extensive APK collection which has currently more than 26,000,000 samples and its main purpose is to provide research community with application samples. On the other hand, TESSERACT offers rigorous evaluation framework through AUT metrics and performance analysis. Meanwhile, AndroBank addresses the critical intermediate stage of dataset preparation and quality assurance. As detailed in Table 2, these three systems serve complementary roles in the malware detection research pipeline. AndroZoo for data acquisition; AndroBank for dataset processing and bias mitigation through API Level analysis and Contrasting dataset exclusion; and TESSERACT for time-aware performance validation. AndroBank methodology specifically targets dataset quality issues through proposed preprocessing, sample unification and Contrasting dataset detection – challenges that arise before classifier evaluation begins. The systematic comparison demonstrates how each system contributes distinct capabilities, establishing AndroBank as the

Table 2 Feature-based comparison of AndroBank with existing tools in mobile malware research

Comparison of tool features			
Criteria	AndroBank	TESSERACT	AndroZoo
Purpose	Automated processing of APKs and AI dataset generation	Evaluation framework for eliminating spatio-temporal bias	APK collection and distribution
Input	Raw APK dataset	Pre-extracted features, labels, timestamps	SHA256 fingerprint
Output	AI dataset; dataset of guaranteed quality	AUT metrics, performance analysis, training ratios	APK file
Debias awareness feature	Exclusion of contrasting dataset usage	Complex debiasing methodology	Not available
AI-ready	Yes	Yes	No

essential bridge between raw data collection and rigorous performance assessment through AI-ready dataset.

5 Case studies - API level analysis of available datasets

The AndroBank system was used to analyze several datasets that are publicly or academically available. Some of those were chosen to point out that, currently, there is no standard for mobile malware dataset creation and evaluation. Even though there is no standard available, the authors would like to mention that the available databases represent a valuable source of research information, and it is also important to note that their creation was time-consuming.

Obtaining the latest mobile malware samples is limited. Thus, academic research teams often have to create their custom datasets. This process is usually done using available public and academic datasets from which suitable samples are selected. Unfortunately, this approach results in the dataset consisting of older malware samples and newer samples of legitimate applications. This is closely related to another unwanted phenomenon, the reuse of old, well-known malware datasets, as also described by Rashed et al. in their research (RASHED 2021).

These samples are then used to create datasets for AI-based methods, where it is typically necessary to have both malicious and legitimate samples. On the other hand, when collecting legitimate applications, the latest versions are typically the most available, resulting in a high contrast between malware and legitimate app classes.

The AndroBank was used to process the dataset AndroidMalware published in 2018 (SK3PTRE 2019) and CICMalDroid 2020 (Mahdavifar et al. 2020); AndroBank was also used to prepare AI dataset for an experiment. Three case studies were carried out. The first two show the strengths of the AndroBank tool for dataset analysis and the production of a Dataset of guaranteed quality. The third experiment demonstrates the applicability of the AndroBank approach in creating AI datasets and highlights the dangers of creating datasets with imbalanced API Levels between malware and legitimate applications.

5.1 Case study 1 - AndroidMalware - 2018

The first processed dataset by the AndroBank tool was the AndroidMalware dataset published in 2018 (SK3PTRE 2019). The output of AndroBank with essential information is shown in Table 3.

Although the dataset is less extensive in a count of APK files, it contained several very recent samples at the time of release. On the other hand, this dataset has various non-valid files and other defects. The Table 3 implies the importance of preprocessing the dataset, which guaranteed that only valid samples were subject to decompilation. This results in an increase in the success rate of the decompilation process. At the same time, it should be noted that malware for other platforms may also appear in datasets containing mobile malware, especially in public ones. This fact can be illustrated by preprocessing and subsequent analysis using VirusTotal, indicating the presence of 5 invalid files such as PE32 (see Tab. 3) in the dataset. The paper aims not to undermine the authors of the datasets or diminish the

Table 3 Android Malware 2018—dataset description

AndroidMalware - 2018 (SK3PTRE 2019)			
SHA-256: 9d4c1246d71a62aac518b568b7d9dec47 2fe16f0756a088deaf03d0a11d5e606			
Real number of APK files		108	
Decompilation results		Preprocessing Results	
Successfull	108	Duplicates	0
Partially Successfull	0	Incomplete APK	0
Failed	0	Invalid files	5
API levels minima and maxima			
Minimum: sdkVersion		8	
Maximum: sdkVersion		26	
Minimum: targetSdkVersion		14	
Maximum: targetSdkVersion		27	

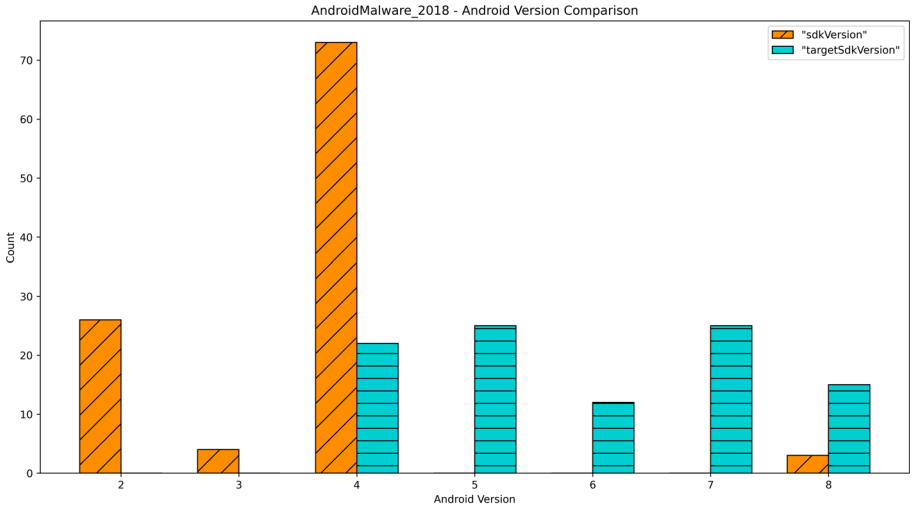

Fig. 4 Android malware 2018—targeted android versions

research significance of these datasets but to shed light on such a dangerous phenomenon, which malware for other platforms is.

Fig. 4 shows the targeted Android version comparison across the dataset from the perspective of API Level analysis. The X-axis shows the version of the Android operating system for which the application was intended. This information resulted from the sdkVersion and targetSdkVersion, which were obtained by automated analysis using the AAPT2 tool.

Fig. 4 shows that despite the occurrence of very recent applications due to the dataset's time coverage period, there are also significantly older samples in the dataset, which must be taken into account when creating the final AI dataset.

The Android Malware 2018 dataset's composition can have positive or negative effects, depending on the purpose of the final AI dataset:

- the positive effect: older samples can be used to analyze malware threats to older devices and create a suitable AI dataset for them.

- the negative effect: our experiment showing the detection result distortion by addressing differences between old and new samples. The results of the experiment implies that too much diversity in regard of substantial API level differences can have a negative effect for the detection tools.

5.2 Case study 2 - CICMalDroid 2020

CICMalDroid (Mahdavifar et al. 2020) is another processed dataset by the AndroBank tool. This dataset was chosen because it is academically accessible and because of its size, making it possible to demonstrate the true capabilities of AndroBank. CICMalDroid authors state that the dataset can be divided into five sections: Adware, Benign, Banking, Riskware, and SMS. For the purpose of experiments with AndroBank, only two sections were used: Malware (which contains all samples except Benign) and Benign (which is disjunctive to the malware set). The CICMalDroid authors have succesfully analyzed 11,598 samples, whereas the AndroBank found and succesfully analyzed 14,781 APK files. The observed discrepancy in efficiency might be caused by the differences between dynamic and static approaches in the CICMalDroid and the AndroBank systems.

The benign part is described in Table 4, which shows that the number of samples successfully analyzed by AndroBank with the static approach is 2,094, while the authors of CICMalDroid dataset reported that 1,795 samples were analyzed successfully using the dynamic approach.

With regard to the malware part of the dataset, the CICMalDroid authors analyzed 9,803 samples using a dynamic approach, while the AndroBank system successfully analyzed 12,687 samples. The result implies the advantage of static analysis and the importance of preprocessing and decompilability checks. Table 5 shows a more detailed dataset description.

Figure 5 compares the target API version of the mobile malware and legitimate apps in the CICMalDroid dataset. The figure is critical because it shows an important phenomenon that can significantly impact the quality of mobile malware detection using static analysis and AI methods. The newer the Android operating system version, the smaller the number of malware samples is. The phenomenon is called “Delayed Interception” and has been defined in Sect. 3.3. Figure 5 also shows that legitimate applications have the opposite trend: the older the Android operating system version, the smaller the number of legitimate samples is.

Table 4 CICMalDroid 2020: Benign—dataset description

CICMalDroid 2020: Benign (Mahdavifar et al. 2020)			
SHA-256: d93f26375a73c6c4a9f954742bdb9274c0f-85cb2786a473e219da346504b15fc			
Real number of APK files		2,094	
Decompilation results		Preprocessing Results	
Successfull	2,094	Duplicates	3
Partially Successfull	0	Incomplete APK	1
Failed	0	Invalid files	2
API levels minima and maxima			
Minimum: sdkVersion		3	
Maximum: sdkVersion		16	
Minimum: targetSdkVersion		7	
Maximum: targetSdkVersion		25	

Table 5 CICMalDroid 2020: Malware—dataset description

CICMalDroid 2020: Malware (Mahdavifar et al. 2020)			
SHA-256 - Adware: 7433c32214fb347f8bed62905f472cc4bf-4017b17eada7bb56dd4004747d23ac			
SHA-256 - Banking: 353c9d800ad73b0048fe6d7f8649008c30b-6d776c68137041c0be9fc1258ffa7			
SHA-256 - Riskware: 9b72f4f54dcf3ce3d505d828acca0fd453a0bbf795a6245c713e4f7d065e9f64			
SHA-256 - SMS: 765fa6e2abc0f84bf3b939f881854c21d16bb-8b54afbf22dfc08ce5ada4f294f			
Real number of APK files		12,687	
Decompilation results		Preprocessing Results	
Successfull	12,687	Duplicates	54
Partially Successfull	0	Incomplete APK	8
Failed	0	Invalid files	455
API levels minima and maxima			
Minimum: sdkVersion		1	
Maximum: sdkVersion		22	
Minimum: targetSdkVersion		4	
Maximum: targetSdkVersion		27	

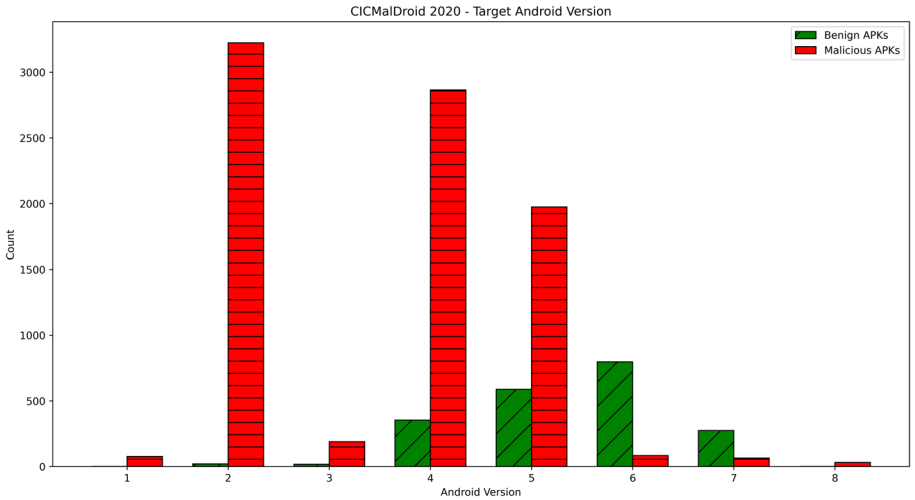

Fig. 5 CICMalDroid 2020—target API level comparison

It should be noted that some samples from the CICMalDroid dataset span across the defined API milestones. Most of the samples from this dataset target Android versions released before the defined milestones (Sect. 3.7); the consequence is that these samples could be irrelevant for modern AI detection because they can negatively affect the detection.

5.3 Case study 3 - API level detection experiment

The experiment addresses the problem of creating AI datasets (output of 3rd stage of AndroBank system), emphasizing the API Level of samples. The fundamental concept is that gathering current benign applications is easier, than searching for older versions. The situation in collecting mobile malware samples is precisely the opposite (explained as Delayed Interception, see Sect. 3.3). Especially for academic research teams, it is much more accessible to obtain older malware samples. Since the latest malware samples usually appear in professional datasets of anti-malware companies, it represents a significant challenge for academic researchers without contact with the commercial sector as mentioned earlier. It can lead to the unwanted creation of a Contrasting AI dataset, defined in Sect. 3.4.

5.3.1 Description of used data

While the current AndroBank system consists of more than 20 datasets, the AI dataset for the experiment was created as a Contrasting dataset (see Sect. 3.4 Fundamental Terminology) and generated by the AndroBank system from the following sources: SK3PTRE (2019), Mahdavifar et al. (2020), SK3PTRE (2022), SK3PTRE (2020), SK3PTRE (2021), SK3PTRE (2022), mstfknn (2022) and Torrents (via BitTorrent Protocol). The Contrasting dataset for this experiment was created to confirm the detection result distortion if the distribution of API level in the samples will be as depicted in Fig. 6, i.e., malicious APKs will be older than benign APKs. The included malware has sdkVersion on interval $\langle 7, 9 \rangle$, whereas the interval for legitimate samples equals $\langle 14, 16 \rangle$. The testing subset of the AI dataset consisted only of 104 samples of modern malware with identical versions of the Android operating system as the legitimate app samples from the test subset of the AI dataset, which

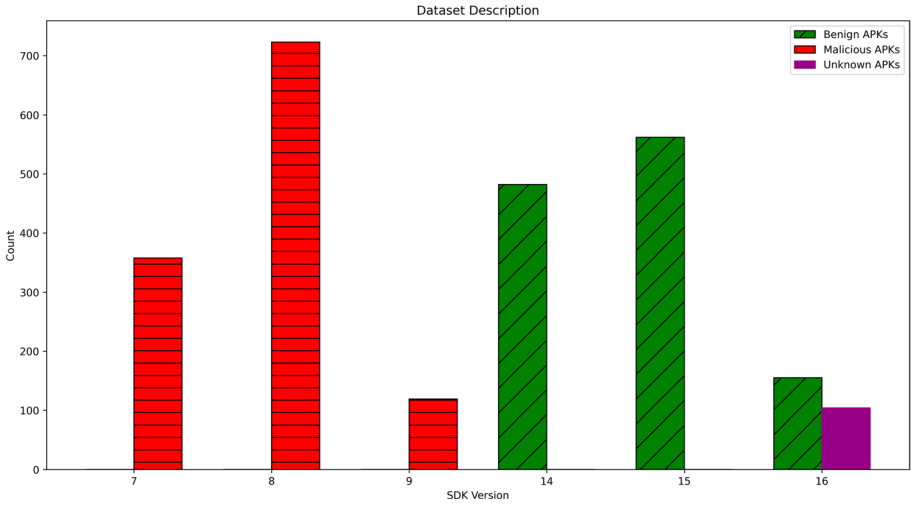

Fig. 6 Android version distribution of chosen data

Table 6 Average of sdkVersion and targetSdkVersion differences

Average SDK differences	
Benign	8.79
Malware	6.59

Table 7 Experiment: contrasting dataset—description

Experiment—contrasting dataset			
SHA-256: 4d754df809b428d12856d-4231b8d0667df353eb28594d0db06f3ea1c63a56e59			
Real number of APK files		2,503	
Dataset composition		Android Version Coverage Era	
Modern Benign	1,199	Min: 4	Est. target: 5-7
Old Malware	1,200	Min: 2	Est. target: 3-4
Unknown Modern Malware	104	Min: 4	Est. target: 7

are labeled as "Unknown APKs" in Fig. 6. The complete list of samples used for the experiment is available in a CSV format on our laboratory website.[1]

Since finding information about targetSdkVersion for some samples was impossible, an analysis of average differences between sdkVersion and targetSdkVersion was conducted. More than 16,000 APK samples were analyzed, resulting in the estimation of the targetSdkVersion value, as presented in Table 6. Based on the average differences analysis, it was possible to approximate targetSdkVersion for applications without this information as follow:

- Benign approximated API: 22-24 (Android 4-7),
- Malware approximated API: 13-15 (Android 2-4).

That means malware samples are mostly defined before API milestones, whereas legitimate samples are mainly spanned across defined API milestones.

As already stated, the primary purpose of this experiment was to create a Contrasting dataset to demonstrate the negative impact of using such datasets for AI mobile malware detection. The detailed description of the created Contrasting dataset is presented in Table 7.

5.3.2 Experiment results

The AndroBank was also used to prepare the Contrasting dataset, which included features such as permissions, VirusTotal tags, counts of file types, etc. Since the aim of the experiment is to highlight the impact of the dataset for training and testing AI classification methods, the Sci-kit (Pedregosa et al. 2011) library for Python was used to carry out the computation. The Sci-kit contains built-in machine learning models for classification such

[1] Dataset description: https://ptlab.fai.utb.cz/androbank-the-impact-of-api-levels-on-mobile-malware-detection/AndroBank: The Impact of API Levels on Mobile Malware Detection (2023)

Table 8 Model evaluation on unknown data

AI experiment results					
Algorithm	Model training			Test data (Unknown APKs)	
	F1-Score	Precision	Recall	F1-Score	Recall
Random Forest	0.92	0.97	0.88	0.68	0.51
SVM	0.89	0.89	0.91	0.41	0.26
KNN	0.93	0.95	0.92	0.45	0.29
Decision tree	0.84	0.89	0.79	0.58	0.73
AdaBoost	0.98	1	0.97	0.68	0.51
Log. Regression	0.93	0.94	0.92	0.71	0.55

Table 9 Confidence intervals

Mean F1 and 95 % CI over 1 000 random splits						
Algorithm	Model training			Test data (Unknown APKs)		
	F1 Mean	F1 Min	F1 Max	F1 Mean	F1 Min	F1 Max
Random forest	0.925	0.886	0.957	0.683	0.658	0.708
SVM	0.889	0.863	0.917	0.399	0.375	0.412
KNN	0.936	0.915	0.956	0.479	0.375	0.545
Decision Tree	0.882	0.780	0.938	0.697	0.412	0.844
AdaBoost	0.985	0.973	0.994	0.664	0.623	0.700
Log. Regression	0.975	0.961	0.988	0.678	0.658	0.700

as Random Forest, Support Vector Machines (SVM), K-Nereast Neigbor (KNN), Decision Tree, AdaBoost or Linear Regression.

The experimental results fully support our hypothesis about the Contrasting dataset's composition, as seen in Table 8. The predicted unwanted behavior has been observed in all models. When the newer legitimate samples (latest API Levels) and only old malware samples (old API Levels) were used to train AI models, the results of all them were satisfactory, as shown in column "F1-Score" in Tab. 8 (Model Training). However, the problem arises when these models try to detect modern malware samples (latest API Levels) that were not part of the learning process (Unknown APKs). The results are of poor quality in all cases, as seen in column "F1-Score" (Test Data). The commercial sphere would consider such a system as completely unusable for anti-malware software offered to public.

To confirm the empirical stability of our findings on the whole test set, we repeated the entire pipeline 1,000 times. We derived 95 % confidence intervals (CIs) for the resulting F1-scores. Table 9 reports the mean F1 and CI for the resampled validation splits and the fixed set of all unknown samples.

The reported confidence intervals fully support our claims. Moreover, it shows that the average Deterioration on the Unknown samples is 0,33 %, while the most significant average difference can be seen at SVM, which dropped by 0.49 on average.

The paired bootstrap analysis reveals substantial and statistically significant performance degradation across all evaluated machine learning algorithms when models trained on Contrasting datasets encounter unknown modern malware samples. As detailed in Table 10, the magnitude performance loss ranges from 0.185 (Decision Tree) to 0.491 (SVM) in F1-score units, representing relative decreases of 18.5% to 49.1% respectively.

Table 10 Bootstrap significance test of the F1-score difference on unknown samples

Statistical significance of F1-score differences			
Algorithm	ΔF1	95% CI	p-value
Random forest	+0.242	[0.241, 0.243]	< 0.00001
SVM	+0.491	[0.490, 0.492]	< 0.00001
KNN	+0.457	[0.454, 0.460]	< 0.00001
Decision tree	+0.185	[0.179, 0.191]	< 0.00001
AdaBoost	+0.321	[0.320, 0.322]	< 0.00001
Log. Regression	+0.297	[0.297, 0.298]	< 0.00001

These findings validate our hypothesis that models trained on Contrasting datasets achieve artificially inflated performance but fail when confronted with modern malware that shares similar API levels to the training benign samples. The consistent degradation pattern across diverse algorithmic approaches underscores the fundamental methodological flaw - using temporally imbalanced datasets for malware detection training, supporting the necessity of AndroBank's API Level analysis and dataset quality assurance mechanisms.

6 Conclusion

The paper addresses serious detection problems that can occur when unprocessed datasets are used (without a proper methodology).

Three-staged methodology and a tool called AndroBank were proposed. This tool is capable of processing any raw dataset regardless of its size or structure. It can also handle format, naming, and other sample defects. The outputs of AndroBank are Dataset of guaranteed quality, Feature dataset and AI dataset.

There are three fundamental problems in the contemporary mobile malware detection field: missing reproducibility, insufficient quality of available samples, and dataset bias.

The first mentioned problem is missing reproducibility and is closely related to the lack of standardization. The authors have proposed several factors that should be considered for standardization. One factor is naming unification (SHA-256 fingerprint), which can be easily done and serves as a unique identifier. Nowadays, samples are named differently across datasets, which often prevents reproducibility. The unification of samples is closely related to the Dataset of guaranteed quality, which is the crucial term defined with all its essentials. The Dataset of guaranteed quality ensures that all samples in the AI dataset can be used for follow-up AI applications.

The second problem is the insufficient quality of available samples, caused by a lack of samples and gathering issues. The samples in the raw dataset often need to be preprocessed to exclude defects, such as malware for other platforms and others. This could be time-consuming and needs to be verified. It is also one of the reasons why the AndroBank system was proposed.

The third issue addressed is the bias in the dataset, more specifically, problems related to a Contrasting dataset (with imbalanced API Levels). This is tied to gathering issues, including the newly defined term: Interception Delay. The Interception Delay explains the appearance of older malware in AI datasets. The importance of Interception Delay was highlighted by the introduction of significant API Milestones. The significant API milestones highlighted the need for modern malware in academic research, as systems are naturally

changing how legitimate applications/malware interact with API. The Interception Delay phenomenon was discussed also in Sect. 5.2. This phenomenon allows the creation of a Contrasting dataset which is often undesirable for real-world applications.

Three case studies were carried out in this paper. The first two case studies demonstrated AndroBank's strengths in processing and analyzing datasets. The raw datasets were succesfully processed into Datasets of guaranteed quality. Moreover, methods applied by Androbank showed better performance in comparison to the dynamic analysis approach. The AndroBank tool successfully processed and analyzed more samples than the technique based on dynamic analysis. Based on API Level analysis in the second case study, it was possible to discover and visualize the Interception Delay phenomenon.

While the first two case studies were showcased as a reaction to the mentioned problems, namely the reproducibility of experiments and insufficient quality of samples. The third case study was aimed at the bias in the dataset. The intentionally created Contrasting dataset with only old API level malware and new API level benign applications had a negative impact on detection of unknown malware of "higher" API level (mainly after introduced API level milestones in Fig. 2) which fully supports our hypothesis of result distortion in such cases.

7 Limitations and future work

As researchers know, the research never ends. We are aware of some limitations in provided experiments and will work on them in future to extend and generalize our achievements.

- In the case studies, 104 unknown malware samples were used, which were carefully selected based on their malignancy and API level. However, it would be beneficial to conduct similar experiments involving a more significant number of samples in the future.
- In the experiments that were conducted, permissions were used as features (ad hoc AI dataset). Nevertheless, for more robust detection, it is necessary to use more different features obtained, for example, by analyzing API calls or resources. In our laboratory, such analytical tools are currently developed.
- Although the possibilities of both sample unification and dataset standardization have been suggested in this paper, it would be beneficial to introduce a large-scale, systematic standardization for better reproducibility. The creation of a helpful standardization methodology requires a large number of experiments and subsequent analyses. Our team is currently working on this problem; however, for such a large project, it would be beneficial insight from other teams too. Thus, we would like to invite other teams to collaborate on the research.
- The present experiment intentionally excludes the most recent APKs. A short time gap simplifies the construction of a contrasting dataset in which benign apps target higher API levels. In comparison, malware samples remain at older levels, an imbalance that amplifies the bias, which has been studied in this case study. Moreover, older samples are more likely to be correctly detected by respected AV software. In real-world scenarios, the AI dataset should always include the latest samples so that detection models remain current.
- In the subsequent research, we will also focus on obfuscated samples and their effects on detection. While obfuscation surely complicates manual static analysis, in automated

analysis, obfuscation can be seen as a feature. Moreover, while using system API calls, which is unavoidable, the system API calls retain their origin name even when the sample is obfuscated.

Acknowledgements This work was supported by the Internal Grant Agency of Tomas Bata University in Zlin, under grant No.: IGA/CebiaTech/2023/004, and further by the resources of PT Lab (ptlab.fai.utb.cz) and A.I.Lab (ailab.fai.utb.cz) at the Faculty of Applied Informatics, Tomas Bata University in Zlin, and by a supportive environment conducive to the exploration of ideas in cybersecurity research of Mobile Threat Research s.r.o. (mt-research.eu).

Author Contributions M.O. designed process with L.D., then through the research and writing process authors collaborated equally and all authors reviewed the manuscript.

Funding Open access publishing supported by the institutions participating in the CzechELib Transformative Agreement.

Data Availability Data is provided within the manuscript or supplementary information files.

Declarations

Conflict of interest The authors, Milan Oulehla and Ladislav Dorotik, are founders of Mobile Threat Research s.r.o., which is mentioned in the Acknowledgments for providing a supportive environment conducive to exploring ideas. However, this research and its conclusions were conducted independently of the organization. The organization had no role in the study's design, data analysis, or decision to publish. The author, Zuzana Kominkova Oplatkova, has no relevant financial or non-financial interests to disclose.

References

Allix K, Bissyandé TF, Klein J, Le Traon Y (2016) AndroZoo: Collecting Millions of Android Apps for the Research Community, Proceedings of the 13th International Conference on Mining Software Repositories (MSR '16), ACM, New York, NY, USA, pp. 468–471. https://doi.org/10.1145/2901739.2903508

AndroBank: The Impact of API Levels on Mobile Malware Detection [online] (2023) Zlín: Penetration Testing Laboratory, c2023 Retrieved December 18, 2023, from: https://ptlab.fai.utb.cz/androbank-the-impact-of-api-levels-on-mobile-malware-detection/

Android Developers (2022) AAPT2 - Android Asset Packaging Tool. Retrieved March 3, 2023, from https://developer.android.com/studio/command-line/aapt2

APKPure (2023) APK Signature Verification, Retrieved September 23, 2023, from https://apkpure.com/apk-signature-verification

ESET SPOL. S R.O. (2024) [KB8366] Accessibility restriction on Android 13 for ESET mobile apps installed from .APK file. Support ESET. Retrieved July 7, 2025, from https://support.eset.com/en/kb8366-accessibility-restriction-on-android-13-for-apps-installed-from-apk-file

Gartner (2022) Total unit shipments of personal computers (PCs) worldwide from 2006 to 2022 (in million units). Statista. Statista Inc. Accessed: March 06, 2023. https://www.statista.com/statistics/273495/global-shipments-of-personal-computers-since-2006/

Gonçalves P, Dološ K, Stebner M, Attenberger A, Baier H (2022) Revisiting the dataset gap problem - On availability, assessment and perspective of mobile forensic corpora. Forensic Sci Int Digit Investig 43:301439. https://doi.org/10.1016/j.fsidi.2022.301439. (**Retrieved 3 March 2023**)

Google (2023) Android 5.0 Behavior Changes. Android Developers. Retrieved March 3, 2023, from https://developer.android.com/about/versions/lollipop/android-5.0-changes

Google (2023) Android 6.0 changes. *Android Developers*. Retrieved March 3, 2023, from https://developer.android.com/about/versions/marshmallow/android-6.0-changes

Google (2023) Android 8.0 Behavior Changes. Android Developers. Retrieved March 3, 2023, from https://developer.android.com/about/versions/oreo/android-8.0-changes

Google (2023) Android 9 features and APIs. Android Developers. Retrieved March 4, 2023, from https://developer.android.com/about/versions/pie/android-9.0

Google (2023) Android Basics in Kotlin. Android Developers. Retrieved November 23, 2023, from https://developer.android.com/courses/android-basics-kotlin/android-basics-kotlin-vocab#apk_android_application_package

Google (2023) Application Fundamentals. *Android Developers*. Retrieved March 3, 2023, from https://developer.android.com/guide/components/fundamentals

Google (2023) uses-sdk. Android Developers. Retrieved March 5, 2023, from https://developer.android.com/guide/topics/manifest/uses-sdk-element#target

GOOGLE (2025) Android 10 release notes. *Android Open Source Project*. Retrieved July 7, 2025, from https://source.android.com/docs/whatsnew/android-10-release#background_location_access_reminder

GOOGLE (2025) Behavior changes: Apps targeting Android 13 or higher. *Android Developers*. Retrieved July 7, 2025, from https://developer.android.com/about/versions/13/behavior-changes-13

GOOGLE (2025) Behavior changes: Apps targeting Android 14 or higher. Android Developers. Retrieved July 7, 2025, from https://developer.android.com/about/versions/14/behavior-changes-14#background-activity-restrictions

GOOGLE (2025) Permissions updates in Android 11. Android Developers. Retrieved July 7, 2025, from https://developer.android.com/about/versions/11/privacy/permissions

GOOGLE (2025) Permissions updates in Android 11. *Android Open Source Project*. Retrieved July 7, 2025, from https://source.android.com/docs/whatsnew/android-12-release

GOOGLE (2025) Scoped storage. *Android Open Source Project*. Retrieved July 7, 2025, from https://source.android.com/docs/core/storage/scoped

Google Play Console (2023) Reach and devices. Retrieved March 3, 2023, from https://play.google.com/console/about/reachanddevices/

IDC (2023) Global smartphone shipments from 2009 to 2022 (in million units). Statista. Statista Inc. Accessed: March 25, 2023. https://www.statista.com/statistics/271491/worldwide-shipments-of-smartphones-since-2009/

Jotti (2023) Scan File - Jotti's malware scan, Retrieved September 23, 2023, from https://virusscan.jotti.org/en-US/scan-file

Kan Z, McFadden S, Arp D, Pendlebury F, Jordaney R, Kinder J, Pierazzi F, Cavallaro L (2024) Tesseract: Eliminating Experimental Bias in Malware Classification across Space and Time (Extended Version)

Lin Y, Liu T, Liu W, Wang Z, Li L, Xu G, Wang H (2022) Dataset bias in android malware detection. arXiv preprint arXiv:2205.15532

Mahdavifar S, Abdul Kadir AF, Fatemi R, Alhadidi D, Ghorbani AA (2020) Dynamic Android Malware Category Classification using Semi-Supervised Deep Learning, In: The 18th IEEE International Conference on Dependable, Autonomic, and Secure Computing (DASC), Aug. 17-24

Miranda TC, Gimenez P-F, Lalande J-F, Tong VVT, Wilke P (2022) Debiasing Android Malware Datasets: How Can I Trust Your Results If Your Dataset Is Biased? IEEE Trans Inf Forensics Secur 17:2182–2197. https://doi.org/10.1109/TIFS.2022.3180184

Mobile & Tablet Android Version Market Share Worldwide (2023) StatCounter: GlobalStats [online]. StatCounter, c1999-2023, 6 Feb 2023 [cit. 2023-03-31]. Retrieved March 3, 2023, from https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide/#daily-20230206-20230206-bar

mstfknn (2022) *android-malware-sample-library* [online]. GitHub, c2023, last modified 28 Apr 2022. Retrieved December 8, 2023, from https://github.com/mstfknn/android-malware-sample-library

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine Learning in Python. J Mach Learn Res 12:2825–2830

RASHED (2021) Mohammed a Guillermo SUAREZ-TANGIL. An Analysis of Android Malware Classification Services. *Sensors* [online]. MDPI, **21**(16), 1-14 Retrieved March 3, 2023. ISSN 14248220. https://doi.org/10.3390/s21165671

SHA-256 (2023) Glossary CSRC [online]. NIST Retrieved March 3, 2023, from https://csrc.nist.gov/glossary/term/sha_256

SK3PTRE (2019) *AndroidMalware_2018* [online]. GitHub, c2023, Retrieved March 3, 2023, from https://github.com/sk3ptre/AndroidMalware_2018
SK3PTRE (2020) AndroidMalware_2019 [online]. GitHub, c2023, last modified 1 Jan 2020. Retrieved December 8, 2023, from https://github.com/sk3ptre/AndroidMalware_2019
SK3PTRE (2021) AndroidMalware_2020 [online]. GitHub, c2023, last modified 6 Jan 2021. Retrieved December 8, 2023, from https://github.com/sk3ptre/AndroidMalware_2020
SK3PTRE (2022) *AndroidMalware_2021* [online]. GitHub, c2023, last modified 9 Mar 2022. Retrieved December 8, 2023, from https://github.com/sk3ptre/AndroidMalware_2021
SK3PTRE (2022) AndroidMalware_2022 [online]. GitHub, c2023, Retrieved March 3, 2023, from https://github.com/sk3ptre/AndroidMalware_2022
StatCounter (2022) Mobile Android version market share worldwide 2018-2022. Statista. Statista Inc. Accessed: March 24, 2023. https://www.statista.com/statistics/921152/mobile-android-version-share-worldwide/
TUMBLESON (2022) Connor a Ryszard WIŚNIEWSKI. *Apktool: 2.6.1* [software]. [cit. 2022-04-26]. URL: https://ibotpeaches.github.io/Apktool/install/
VirusTotal (2023) *VirusTotal*. Retrieved March 3, 2023, from https://www.virustotal.com/
YASWANT Aazim (2022) Financially Motivated Mobile Scamware Exceeds 100M Installations. *Zimperium* [online]. Zimperium, c2010-2023, January 26, 2022. Retrieved March 3, 2023, from https://www.zimperium.com/blog/dark-herring-android-scamware-exceeds-100m-installations/