2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW) | 978-1-6654-3583-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ASEW52652.2021.00023

A First Look at Security Risks of Android TV Apps

Yonghui Liu*, Li Li*, Pingfan Kong†, Xiaoyu Sun*, Tegawendé F. Bissyandé†
*Monash University, Melbourne, Australia
† University of Luxembourg, Luxembourg
{yonghui.liu, li.li, xiaoyu.sun}@monash.edu, {pingfan.kong, tegawende.bissyande}@uni.lu

***Abstract*—In this paper, we present to the community the first preliminary study on the security risks of Android TV apps. To the best of our knowledge, despite the fact that various efforts have been put into analyzing Android apps, our community has not explored TV versions of Android apps. There is hence no publicly available dataset containing Android TV apps. To this end, We start by collecting a large set of Android TV apps from the official Google Play store. We then experimentally look at those apps from four security aspects: VirusTotal scans, requested permissions, security flaws, and privacy leaks. Our experimental results reveal that, similar to that of Android smartphone apps, Android TV apps can also come with different security issues. We hence argue that our community should pay more attention to analyze Android TV apps.**

I. Introduction

Smart TV has become the default TV type nowadays. More and more users are switching from traditional TVs to Smart TVs [1], which go beyond conventional TVs by providing additional programs through internet connectivities such as allowing users to watch Netflix, YouTube directly over the TVs. Among various types of Smart TVs in the market, Android TV, with a lot of benefits extended from the popular Android ecosystem, has undoubtedly become one of the most popular ones. Similar to that of Android phones, Android TV users can also connect to the Google Play store to download and update apps and utilize Google Assistant to achieve hands-free tasks.

Unfortunately, while extending the benefits from the Android ecosystem, Android TVs also extend the potential security flaws appearing in other Android-based devices such as Smartphones or Smartwatches. For example, the video and voice features enabled by many Android TVs through respectively camera and microphone could be exploited by malicious apps to create a surveillance device for recording the users' daily activities. Besides the misuse of existing features, the customization made for Android TVs can further bring security risks to the TV users. Indeed, as recently discovered by Aafer et al. [2], through log-guided fuzzing, they found that there are 37 unique vulnerabilities available in 11 Android TV boxes that could lead to high-impact cyber threats.

Despite the continuously growing popularity of Android TVs and the public reports of the aforementioned potential security risks, our community has not yet paid enough attention to the security of Android TVs, including the security analysis of Android TV apps. Indeed, our lightweight literature search (over Google Scholar with different combinations of keywords including *Android TV*, *Smart TV*, *security*, *privacy*, and *vulnerability*, etc.) finds no paper published in this direction. To fill the gap, in this work, we conduct an exploratory study on the security risks of Android TV apps. Through this study, we aim to understand the status quo of Android TV apps and their security situations, and subsequently observe actionable insights towards achieving a better Android TV ecosystem. Unfortunately, there is no dataset that are mainly made up of *Android TV apps*. To that end, we firstly collect Android TV apps from Google Play to fulfill this research gap. Particularly, we resort to a crawling process to pinpoint Android TV apps available on the official Google Play store. Our preliminary attempt has harvested $3,163$ TV apps spanning over 17 major categories. We then download those apps and their corresponding metadata to prepare the dataset and thereby support the subsequent exploratory experiments.

Based on the carefully prepared dataset, we first examine the harvested Android TV apps from their popularity point of view. Our empirical study reveals that TV apps are mainly used for entertainment. Then, to illuminate the security behaviors of these apps, we investigate them from the presence of malware, the requested permissions, the possible security flaws, and the potential privacy leaks. We observe that different kinds of security issues do exist in the Android TV ecosystem.

To conclude, our work has made the following major contributions.

- We have collected an amount of 3,163 Android TV apps from the official Google Play store. The dataset has been made publicly available at Github [3].
- We have further collected the TV apps' metadata and conducted a preliminary investigation about the apps' category and size distribution, as well as their overall popularities.
- We have conducted a detailed security risk analysis based on the collected TV apps. Experimental results show that Android TV apps indeed suffer from various security flaws and may contain malicious behaviors, including leaking users' private information.

II. Dataset and Preliminary Study

In this section, we first discuss the process we have leveraged to collect real-world Android TV apps (cf. Section II-A). After that, we present a preliminary study to have an initial understanding of those harvested TV apps (cf. Section II-B).

978-1-6654-3583-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ASEW52652.2021.00023

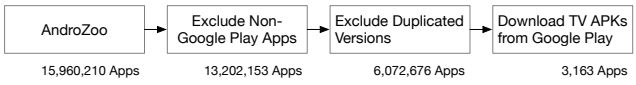

Fig. 1. Working Process to Collect Android TV apps.

A. Dataset

To help the ensuing exploratory experiments, we need to prepare a set of Android TV apps. To this end, we resort to the official Google Play store to collect such a dataset.

Figure 1 illustrates the working process of our approach for collecting Android TV apps. It unfolds in 4 steps. First, We obtain a list of Android smartphone apps from the popular AndroZoo [4] dataset. AndroZoo contains Android smartphone apps that are collected from Google Play and many other third-party app markets. Till now, the AndroZoo dataset contains around 16 million apps. Then, we exclude non Google Play Apps and only kept 13 million apks. Afterwards, we remove legacy apk versions of any same app, resulting in 6 million unique Google Play app apks in the dataset. Finally, for each of these retained apps, we resort to Python scripts to search if it has a dedicated TV counterpart version on Google Play. To this end, we utilize the app's package name (also known as app id) as the keyword to conduct the search. We use the package name since it is used by Google Play as the unique identifier of any app. We also need to prepare the search by feeding the Google Play with TV-specific device settings, e.g., Build.DEVICE, Screen.Density. In this way, the search results returned by Google Play will only contain an app's TV-oriented versions.

Our process eventually identifies $3,163$ Android TV apps, for which we further write scripts to crawl their metadata from Google Play, e.g., app category, total installation, etc..

B. Preliminary Study

We now present a preliminary study to have a first look at the $3\,163$ TV apps and their metadata. Figure 2 illustrates the word cloud generated based on the verbal descriptions of the collected TV apps. Clearly, the most recurrent words include *game*, *video*, *audio*, *music*, etc., which suggest the main service the TV apps provide. Table I groups the collected TV apps based on their categories. We further confirm that entertainment, music & audio, game, and video players & editors are among the most popular app categories on the Android TVs. More than half of the apps are collected from these 4 categories. Note that only categories containing at least 10 apps are shown in Table I, while the rest apps are grouped in "Others" category.

In Table I, the second column presents the number of apps available in each category. The third column further shows the number of apps in that category that are marked as containing advertisements on Google Play. In general, around one third of TV apps are equipped with advertisements. Specifically, apps in categories such as Music & Audio, Game, News & Magazines, and Finance are more likely to include advertisements to target the Android TV end-users. Indeed,

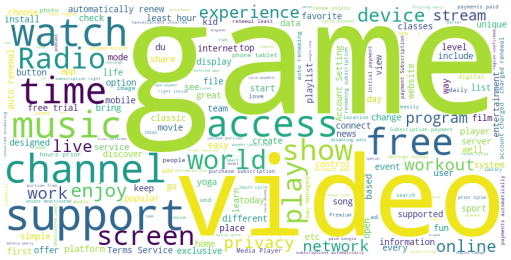

Fig. 2. Word cloud based on the descriptions of the collected TV apps.

TABLE I
STATISTICS OF THE COLLECTED ANDROID TV APPS.

Category	#. of Apps	#. of Apps With Ads	Size (MB)	#. of Installs	#. of Ratings	Rating (Star)
Entertainment	892	216	35.2	2,216,225	35,122	3.6
Music&Audio	484	408	8.7	623,327	12,793	4.0
Game	447	306	26.1	2,694,097	80,260	3.8
Video&Editors	321	63	39.5	17,806,353	48,158	3.8
Tools	223	63	12.4	105,357,326	279,415	3.8
Health&Fitness	144	8	26.9	177,917	4,225	3.6
News&Magazines	108	78	16.5	562,207	12,165	4.0
Education	100	39	25.3	74,444	662	3.8
Lifestyle	100	15	25.8	147,315	4,517	3.9
Business	69	10	18.0	1,822,982	15,078	3.3
Productivity	66	9	18.9	362,851	4,439	3.5
Communication	45	5	22.7	22,798,398	15,530	3.8
Food&Drink	36	20	15.1	871,740	19,826	3.9
Books&Reference	24	11	20.0	366,839	11,593	4.0
Personalization	22	8	9.3	452,820	3,735	3.7
House & Home	11	0	32.0	15,278	51	3.3
Finance	10	8	11.8	164,200	4,829	3.9
Others	61	24	15.3	84,193,442	2,038,839	3.9
Totals	3,163	1,291	25.2	12,382,322	88,898	3.8

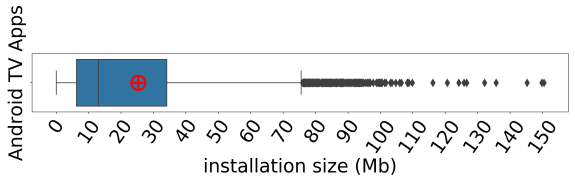

Fig. 3. Distribution of Android TV App size

around 84.3% of Music & Audio apps and 68.5% of Game apps have been embedded with advertisements. The fourth column breaks down the average size of the collected TV apps. In general, the average size of all apps is 25.2 MB. Figure 3 further illustrates the size distribution of those apps. Clearly, over half of the collected TV apps are larger than 13 MB.

We further study the popularity of the collected TV apps from two aspects: (1) Apps installation and (2) Apps User rating.

Apps Installation Analysis We now analyze the number of installations for these apps (cf. column 5 in Table I) to learn the scale of potential users. Since the TV apps are downloaded from Google Play, we only take into account the installation statistics recorded on Google Play. We calculate the accumulated installs count for each of these apps, as shown in Figure 4. The accumulated trend displays the general dissemination of application installs for all the $3,163$ applications, among which the majority of them have been installed over $5,000$ times. Specifically, 25.6% (811 apps) of them have collective install counts as high as $100,000$, and approximately 12% (384 apps) of these apps have over 1

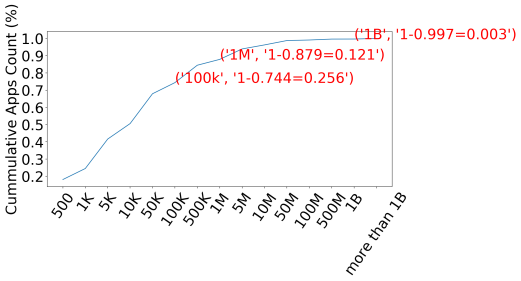

Fig. 4. Distribution of App Installs

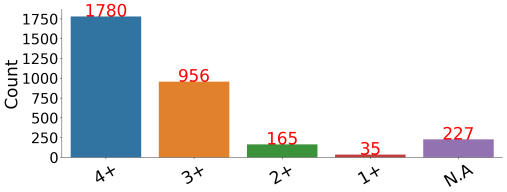

Fig. 5. Distribution of App Rating

million installs, among which 0.3% (10 apps) of them even have over 1 billion downloads. These pieces of evidence show that the harvested TV apps are unlikely to be toy apps but popular real-world apps.

User Rating Analysis We then seek to understand how users feel about these Android TV apps by collecting the average ratings for these apps from Google Play Store. We group these apps based on their average rating score (cf. last column in Table I). The distribution of average app ratings is further shown in Figure 5. We find that more than half of the total apps (56.2% of them) are rated higher than four stars. Also, 956 apps are rated *3+* stars. Since the rating is not an obligation, 227 apps do not have enough user rating to compute an average score (marked as N.A).

III. Security Analysis

We now look into the harvested Android TV apps from the security point of view. Specifically, in this section, we first send the apps for scrutinizing under VirusTotal to collect their malicious status (cf. Section III-A). Then we investigate the apps' requested permissions which the Android ecosystem manages with great caution in order to protect user privacy (cf. Section III-B). After that, we leverage state-of-the-art tools to explore the security flaws and privacy leaks in those collected TV apps, which are detailed in Section III-C and Section III-D, respectively.

A. VirusTotal Analysis

To understand whether the harvested TV apps are malicious or not, we resort to the API [5] provided by VirusTotal to upload and scan all the collected apps. VirusTotal is a website that provides public services for analyzing suspicious files (including Android APKs) to detect malware. Particularly, each file will be scanned by over 60 anti-virus products such as Kaspersky when uploading to VirusTotal. In this work, we label a TV app malicious as long as any anti-virus product considers it as such. Based on generated VirusTotal results, we find that 89 apps are labeled as malicious by at least one anti-virus vendor, showing a quite low malicious rate among all the harvested apps. However, it is worrisome that about one third (30 apps) of the flagged malicious TV apps belong to the Game category. This evidence suggests that, although generally speaking, Android TV apps have not yet been the primary target of attackers, the existence of malicious apps (especially being popular in a certain category) do suggest that there are interests and incentives for attackers to exploit this direction.

B. Requested Permission Analysis

We now investigate how Android TV apps request sensitive permissions. Among different asset documents included in an Android app, we extracted the declared permission from *AndroidManifest.xml*. It is the configuration file liable for defining app features such as permissions. We perform this task by resorting to Android Asset Packaging Tool 2 (AAPT) [6]. App permission framework is a frontline component of the Android ecosystem to keep the privacy of Android users secure. Each Android application has to declare the permissions in the *AndroidManifest.xml* file in case they want to get access to sensitive user data (such as location and audio). Android Developer Guide [7] suggests that permissions can be categorized into different levels grounded on their riskiness, i.e., *normal*, *signature* and *dangerous*.

In total, we are able to observe 1,103 unique permissions requested by the collected TV apps. Among such unique permissions, 299 (27.1%) of them contain the "tv" keyword (hereinafter referred to as TV-specific permissions). Interestingly, only 5 out of the 299 TV-specific permissions are provided by the original Android Open Source Project (cf. Table II). All the remaining permissions are customized ones provided by third-party vendors, e.g., *com.mitv.patchwall.permission.MANAGE_MEDIA* by **MI** TV, *com.amazon.tv.permission.LAUNCHER_SETTINGS* by **Amazon** TV.

TABLE II
TV-SPECIFIC PERMISSIONS

Permissions	Count
com.android.providers.tv.permission.WRITE_EPG_DATA	171
com.android.providers.tv.permission.READ_EPG_DATA	170
com.android.providers.tv.permission.ACCESS_ALL_EPG_DATA	4
android.permission.BIND_TV_INPUT	2
com.android.providers.tv.permission.ACCESS_WATCHED_PROGRAMS	1

Figure 6 illustrates the distribution of the number of permissions declared by each Android TV app. Around half of the studied TV apps have declared more than 10 permissions. The average number is 9.5, as highlighted by the ⊕ sign. Table III further summarizes the total permissions (17 permissions) requested by at least 10% of all apps. A number of 6 permissions highlighted in Table III are labeled as dangerous permissions that require the users to give the consents at runtime. These permissions are required to read from and write

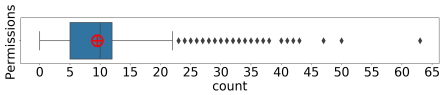

Fig. 6. Distribution of the Number of Permissions

TABLE III
POPULAR PERMISSIONS IN ANDROID TV APPS

Permissions in Android TV Apps	#Apps	Protection Level
INTERNET	3,081	Normal
ACCESS_NETWORK_STATE	2,981	Normal
WAKE_LOCK	2,476	Normal
READ_EXTERNAL_STORAGE	1,831	**Dangerous**
FOREGROUND_SERVICE	1,551	Normal
WRITE_EXTERNAL_STORAGE	1,504	**Dangerous**
ACCESS_WIFI_STATE	1,477	Normal
RECEIVE_BOOT_COMPLETED	1,449	Normal
RECEIVE	1,419	Signature
BIND_GET_INSTALL_REFERRER_SERVICE	1,331	Normal
BILLING	1,258	Normal
RECORD_AUDIO	560	**Dangerous**
SYSTEM_ALERT_WINDOW	496	Signature
VIBRATE	477	Normal
READ_PHONE_STATE	422	**Dangerous**
ACCESS_COARSE_LOCATION	377	**Dangerous**
ACCESS_FINE_LOCATION	355	**Dangerous**

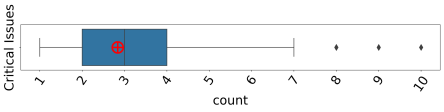

Fig. 7. AndroBugs Critical Issue for Android TV Apps

to external storage, record audio, access to phone state, and access the approximate and precise location of the device [7]. As shown in the table, 3,081 apps (97.4%) have requested the 'Internet' permission, indicating a uniquely high demand for internet access from the TV apps. Also, we found that fewer Android TV apps require the location of end-users compared with mobile Android apps. For instance, only around 11% of Android TV Apps have requested permissions "ACCESS_-COARSE_LOCATION" and "ACCESS_FINE_LOCATION", while previous studies [8] suggested that the percentages for the mobile Android apps are 25% and 26%, respectively.

Declaration of Duplicated Permissions. As revealed by Li et al. [9], [10], smartphone apps may contain duplicated permission declarations, i.e., the same permission is declared twice in the same configuration file. Such duplication is often the indicator for app repackaging, this is because the opportunists often directly append needed permissions without even checking whether they have been already declared or not. In this work, we further check if such permission duplication also happens in Android TV apps. In our exploration, we find that there are 119 apps that have declared the same permission more than once. In total, a number of 14 distinct permissions involve duplication issues. Permissions INTERNET, WAKE_LOCK, ACCESS_NETWORK_STATE, SYSTEM_ALERT_WINDOW, and READ_PHONE_STATE are the most recurrent ones. This evidence suggests that app repackaging attack is probably already happening in TV platforms and requires attention.

TABLE IV
CRITICAL VULNERABILITIES FOUND BY ANDROBUGS

Type	Detailed Vulnerability Descriptions	Count
SSL_Security	SSL Connection	2,817
	Verifying Host Name in Classes	553
	SSL Certificate Verification	228
	Verifying Host Name in Fields	51
WebView	WebView RCE Vulnerability	1,545
Implicit_Intent	Implicit Service	1,436
AndroidManifest	ContentProvider Exported	480
	"intent-filter" Settings	56
	Critical Use Permission	37
	System Use Permission	32
Command	Runtime Command	278
	Runtime Critical Command	32
Encryption	Base64 String Encryption	275
Fragment	Fragment Vulnerability	230
Permission	App Sandbox Permission	199
KeyStore	KeyStore Protection	173
Others		70
Total		8,492

C. Security Flaw Analysis

To understand the potential security vulnerabilities in these Android TV apps, we resort to AndroBugs Terminal Framework [11] to analyze these apps. The AndroBugs reports [12] list vulnerabilities found in such apps together with their severity levels, i.e., either *critical*, *warning*, *notice* or *info*.

On average, AndroBugs reports at least 50 vulnerabilities for each Android TV app. Among all the $3,163$ TV apps, $2,997$ apps were asserted to contain *critical* security bugs. Figure 7 demonstrates the number of critical vulnerabilities detected for each app. Over half of those apps contain at least 3 critical issues, and the average number of critical issues is 2.8, as highlighted by the $\oplus$ sign. This evidence shows that most Android TV apps are flawed by security vulnerabilities.

We further investigate the *critical* vulnerabilities. Table IV summarizes the types and the detailed descriptions of such *critical* vulnerabilities. Vulnerabilities with few counts are grouped into the "Others" category. The vulnerability types appear in Table IV in descending order of counts. Now we discuss below 3 most recurrent vulnerability types in Android TV apps:

- **SSL Security:** In total 3,687 (43%) vulnerabilities belong to this type. These vulnerabilities are closely related to the apps' accessing mechanism to the internet. Application might utilize SSL [13] erroneously such that malicious substances may be able to caught an app's information over the network. For instance, Android TV apps may suffer from man-in-the-middle attacks if they access the internet without solid encryption (e.g., when not using HTTPS [14]).
- **WebView:** In total, 1,545 (18%) vulnerabilities belong to this type. It is a common kind of vulnerability that can be exploited by attackers to execute Java code in the host apps, which can further allow attackers to get access to the command-line tools and pose further security threats to users. For instance [15], a remote attacker may utilize a WebView to execute dynamic HTML content (written

in JavaScript) and invoke Java *Runtime.exec()* API to run commands like *id* or *rm*.

- **Implicit_Intent:** In total 1,436 (17%) vulnerabilities belong to this type. The Android ecosystem uses the Intent mechanism to encourage functionality reuse [16]. However, attackers may exploit implicit Intents to access private information or to bring damage to the user data.

Such high number of vulnerabilities detected in the Android TV apps calls for attention and action from both the researchers and developers.

D. Privacy Leaks Analysis

To understand the potential privacy leaks existing in Android TV apps, we resort to the FlowDroid tool [17] to analyze these apps. FlowDroid reveals the data flows from sensitive *sources* to unsafe *sinks*. Since dataflow analysis are both time and memory intensive. We set a timeout of 2 hours with 32 GB memory limit. FlowDroid was successfully run on $2,448$ Android TV apps. In total $1,673$ apps are found with leaks. Figure 8 demonstrates the number of leaks detected per app. Over half of these apps (889 apps) are detected with at least 5 leaks, and over 25% of those apps contain at least 11 leaks. To our great concern, we observe that leaks in both Health & Fitness and Lifestyle category are twice as much the total average number. This observation suggests that data related to users' daily lives may be at stake.

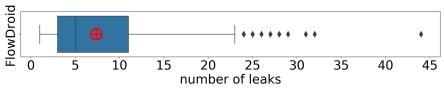

Fig. 8. Distribution of the Count of Leaks Found by Flowdroid

TABLE V
TOP 10 SOURCES FOUND BY FLOWDROID

Sources	count
android.database.Cursor.getString	22,055
android.location.Location.getLongitude	4,436
android.location.Location.getLatitude	4,435
android.content.pm.PackageManager.queryIntentServices	3,021
java.net.HttpURLConnection.getInputStream	1,391
android.content.pm.PackageManager.queryIntentActivities	1,232
android.content.pm.PackageManager.queryBroadcastReceivers	515
java.util.Locale.getCountry	308
android.app.Activity.findViewById	146
java.net.URL.openStream	135

Table V lists the top 10 recurrent sources (APIs) found by FlowDroid. We can see that the most leaked source is from the database. This is no surprise since database often contain large scale information. Based on the second and third most recurrent sources, we find that that attackers care about the geographical location of the Android TV app users. Unlike mobile devices, Android TVs' geographical locations are often fixed and reveal people's home address. Attackers may exploit this information to know much more of the users or even pose physical threat.

Table VI lists the top 10 most recurrent sinks found by FlowDroid. From the first, second and fourth sinks, we understand that sensitive private information is often documented with the built-in Logcat functionality that are normally used by developers to do debug. Users' data have certain chance leaked from there, and attackers can extract the user's sensitive data by analyzing the Log information on the corresponding device. The Bundle object is another place the sensitive information would go. It is used to carry information across activities, processes and configuration changes. Much information in Bundle objects can be exploited by attackers.

TABLE VI
TOP 10 SINKS FOUND BY FLOWDROID

Sinks	count
android.util.Log.d	2,567
android.util.Log.w	977
android.os.Bundle.putString	863
android.util.Log.v	842
java.lang.String.replace	678
android.content.Intent.setComponent	579
android.content.SharedPreferences$Editor.putString	536
android.util.Log.i	402
android.content.Context.registerReceiver	303
android.content.Context.bindService	276

TABLE VII
TOP 10 SOURCE-SINK CATEGORY PAIRS FOUND BY FLOWDROID

Source		Sink
DATABASE_INFORMATION		LOG
LOCATION_INFORMATION		LOG
DATABASE_INFORMATION		NETWORK
NETWORK_INFORMATION		LOG
UNIQUE_IDENTIFIER		LOG
DATABASE_INFORMATION	$\longrightarrow$	FILE
LOCATION_INFORMATION		NETWORK
NETWORK_INFORMATION		FILE
NETWORK_INFORMATION		NETWORK
ACCOUNT_INFORMATION		LOG

The SuSi [18] tool further puts the sources and sinks into different categories. Table VII summarized the top 10 source-sink category pairs.We are not surprised to see that the most recurrent pair has the databse information as source and the log information as sink, each corresponding to the most recurrent sources and sinks reflected in Table V and Table VI, respectively. Our concern for the leak of user's geographical location is worsened since we see that this information is also logged often. The fact that lots of sensitive information is finally leaked in the log or over network calls for more security examination for Android TV apps.

IV. RELATED WORK

Past studies have extensively explored Android apps from different aspects. A huge number of studies have been focused on analyzing the metadata of mobile app [19], [20]. Wang et al. [21] conducted an extensive study on 6 million Android apps downloaded from 17 different app markets to understand catalog similarity across app stores. Chen et al. [22] collected 223 pairs of Android Smartphone and Smartwatch app pairs

and investigate them from both non-code and code aspects to understand the relationship between them. Large-scale studies have been performed on analyzing mobile apps from security and privacy aspects [23], [24]. Researches have attempted to detect Android security vulnerabilities either through static analysis [17], [25], [26], dynamic analysis [27], [28], or resorting to machine learning models [29]–[32]. Moreover, recent researches have revealed issues (e.g. outdated, over-privileged) appeared in an extensive number of third-party library in Android apps. [9], [33]. To the best of our knowledge, the existing works are mainly targeting smartphone apps. Android TV apps have not yet been well investigated. As our future work, we plan to go beyond this work by inventing better approaches for improving the security and reliability of Android TV apps.

V. Conclusion

In this work, we present the first study towards understanding and characterizing Android TV apps. We collected 3,163 Android TV apps and analyzed their metadata. Particularly, we investigate these apps from a security perspective. Our observation has revealed the presence of malware among Android TV apps. We also found suspicious repackaged apps. Furthermore, we find that many Android TV apps suffer from security vulnerabilities. Many Android TV apps also leak users' sensitive information, including their home addresses. Such findings call on actions for our community to pay more attention to Android TV apps, especially from the security perspective.

Acknowledgements

This work was supported by the Australian Research Council (ARC) under a Discovery Early Career Researcher Award (DECRA) project DE200100016, and a Discovery project DP200100020.

References

[1] *Number of smart TV users in the United States from 2016 to 2022 (in millions)**, 2021. [Online]. Available: https://www.statista.com/statistics/718737/number-of-smart-tv-users-in-the-us/

[2] Y. Aafer, W. You, Y. Sun, Y. Shi, X. Zhang, and H. Yin, "Android smarttvs vulnerability discovery via log-guided fuzzing," in *USENIX Security*, 2021.

[3] *First Look at Security Risks of Android TV Apps*, 2021. [Online]. Available: https://github.com/DannyGooo/Android-TV-apps-found-in-Google-Play-Store

[4] L. Li, J. Gao, M. Hurier, P. Kong, T. F. Bissyandé, A. Bartel, J. Klein, and Y. Le Traon, "Androzoo++: Collecting millions of android apps and their metadata for the research community," *arXiv preprint arXiv:1709.05281*, 2017.

[5] *VirusTotal API v3 Overview*, 2021. [Online]. Available: https://developers.virustotal.com/v3.0/reference

[6] *AAPT2*, 2021. [Online]. Available: https://developer.android.com/studio/command-line/aapt2

[7] *Android Developer Guide*, 2021. [Online]. Available: https://developer.android.com/reference/android/Manifest.permission

[8] Y. Hu, H. Wang, L. Li, Y. Guo, G. Xu, and R. He, "Want to earn a few extra bucks? a first look at money-making apps," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2019, pp. 332–343.

[9] L. Li, D. Li, T. F. Bissyandé, J. Klein, Y. Le Traon, D. Lo, and L. Cavallaro, "Understanding android app piggybacking: A systematic study of malicious code grafting," *TIFS*, 2017.

[10] L. Li, T. F. Bissyandé, and J. Klein, "Rebooting research on detecting repackaged android apps: Literature review and benchmark," *IEEE Transactions on Software Engineering (TSE)*, 2019.

[11] "Androbugs, open source repository," 2021. [Online]. Available: https://github.com/AndroBugs/AndroBugs_Framework

[12] Y.-C. Lin, "Androbugs framework: An android application security vulnerability scanner," *Blackhat Europe*, vol. 2015, 2015.

[13] *Security SSL*, 2021. [Online]. Available: https://developer.android.com/training/articles/security-ssl

[14] Y. Desmedt, "Man-in-the-middle attack," in *Encyclopedia of cryptography and security*. Springer, 2011, pp. 759–759.

[15] D. R. Thomas, A. R. Beresford, T. Coudray, T. Sutcliffe, and A. Taylor, "The lifetime of android api vulnerabilities: case study on the javascript-to-java interface," in *Cambridge International Workshop on Security Protocols*. Springer, 2015, pp. 126–138.

[16] C. W. Enumeration, "Use of implicit intent for sensitive communication," 2017.

[17] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," *Acm Sigplan Notices*, vol. 49, no. 6, pp. 259–269, 2014.

[18] *SuSi*, 2021. [Online]. Available: https://github.com/secure-software-engineering/SuSi/tree/develop/SourceSinkLists/Android%204.2/SourcesSinks

[19] Y. Hu, H. Wang, R. He, L. Li, G. Tyson, I. Castro, Y. Guo, L. Wu, and G. Xu, "Mobile app squatting," in *Proceedings of The Web Conference 2020*, 2020, pp. 1727–1738.

[20] L. Li, T. Bissyandé, and J. Klein, "Moonlightbox: Mining android api histories for uncovering release-time inconsistencies," in *ISSRE*. IEEE, 2018, pp. 212–223.

[21] H. Wang, Z. Liu, J. Liang, N. Vallina-Rodriguez, Y. Guo, L. Li, J. Tapiador, J. Cao, and G. Xu, "Beyond google play: A large-scale comparative study of chinese android app markets," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 293–307.

[22] X. Chen, W. Chen, K. Liu, C. Chen, and L. Li, "A comparative study of smartphone and smartwatch apps," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 2021, pp. 1484–1493.

[23] P. Kong, L. Li, J. Gao, K. Liu, T. F. Bissyandé, and J. Klein, "Automated testing of android apps: A systematic literature review," *IEEE Transactions on Reliability*, vol. 68, no. 1, pp. 45–66, 2019.

[24] L. Li, T. F. Bissyandé, M. Papadakis, S. Rasthofer, A. Bartel, D. Octeau, J. Klein, and L. Traon, "Static analysis of android apps: A systematic literature review," *Information and Software Technology*, vol. 88, pp. 67–95, 2017.

[25] L. Li, A. Bartel, T. F. Bissyandé, J. Klein, Y. Le Traon, S. Arzt, S. Rasthofer, E. Bodden, D. Octeau, and P. McDaniel, "Iccta: Detecting inter-component privacy leaks in android apps," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, 2015, pp. 280–291.

[26] X. Sun, L. Li, T. F. Bissyandé, J. Klein, D. Octeau, and J. Grundy, "Taming reflection: An essential step toward whole-program analysis of android apps," *TOSEM*, vol. 30, no. 3, pp. 1–36, 2021.

[27] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 2, pp. 1–29, 2014.

[28] F. Dong, H. Wang, L. Li, Y. Guo, T. F. Bissyandé, T. Liu, G. Xu, and J. Klein, "Frauddroid: Automated ad fraud detection for android apps," in *ESEC/FSE*, 2018.

[29] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, and K. Ren, "Android hiv: A study of repackaging malware for evading machine-learning detection," *TIFS*, 2019.

[30] Y. Liu, C. Tantithamthavorn, L. Li, and Y. Liu, "Deep learning for android malware defenses: a systematic literature review," *arXiv preprint arXiv:2103.05292*, 2021.

[31] Y. Zhao, L. Li, H. Wang, H. Cai, T. F. Bissyandé, J. Klein, and J. Grundy, "On the impact of sample duplication in machine-learning-based android malware detection," *TOSEM*, vol. 30, no. 3, pp. 1–38, 2021.

[32] X. Yang, D. Lo, L. Li, X. Xia, T. F. Bissyandé, and J. Klein, "Characterizing malicious android apps by mining topic-specific data flow signatures," *Information and Software Technology*, 2017.

[33] L. Li, T. F. Bissyandé, J. Klein, and Y. Le Traon, "An investigation into the use of common libraries in android apps," in *SANER*, 2016.