An Empirical Analysis of Security and Privacy Risks in Android Cryptocurrency Wallet Apps

I Wayan Budi Sentana[1 [0000−0003−3559−5123]], Muhammad Ikram[1 [0000−0003−0336−0602]], and Mohamed Ali Kaafar[1 [0000−0003−2714−0276]]

Department of Computing, Macquarie University, 4 Research Park Drive, Sydney, NSW, Australia, 2113

Abstract. A cryptocurrency wallet app is a piece of software that manages, stores, and generates private keys of cryptocurrency accounts. With the provision of services such as easy access to transaction history, and checking account balance besides transmissions of new transactions in distributed networks such as Blockchains, cryptocurrency wallet apps gain unprecedented popularity which in turn attracts malicious actors to attack users resulting in loss of cryptocurrency assets and leakage of sensitive user data. This paper presents the first large-scale study of Android cryptocurrency wallet apps. We surveyed apps on Google Play to detect and extract meta-data and application packages of 457 cryptocurrency wallet apps. We perform several passive and active measurements designed to investigate the security and privacy features to study the behavior of cryptocurrency wallet apps. Our analysis includes investigating cryptocurrency wallet apps' third-party embedding, malware presences, and exfiltration of users' sensitive data to third-parties. Our study reveals vulnerabilities and privacy issues in cryptocurrency apps including the insecure use of HTTP to serve transactions.

Keywords: Cryptocurrency Wallet · Static Analysis · Dynamic Analysis · User-review Analysis.

1 Introduction

Cryptocurrency wallet applications (or wallet apps) for mobile devices are used to securely store cryptocurrency assets and enable users to control their assets over private keys. Like using any type of mobile application, users of wallet apps are possibly faced with security and privacy vulnerabilities rendered by developers' coding practices and privacy practices related to their business models. However, as wallet apps are involved in controlling users' financial assets, the vulnerabilities become more severe if they are leveraged by attackers. For instance, adversaries could leverage the embedded third-party libraries and request permissions of the app to steal important information such as private keys. Hence, it is important to assess wallet apps' behaviors for potential security and privacy issues to inform users as well as developers. Recently, several works have studied security and privacy issues of wallet apps [29,18,15,22]. Nevertheless, their

analysis focuses on some characteristics such as apps requested permission, and privacy leakage, and are limited to a few existing apps.

To provide a comprehensive assessment of the security and privacy issues of 457 Android wallet apps, we perform the first large-scale study of the privacy and security features of wallet apps on Google Play. We first collect wallet apps available in the Google Play store. We use static analysis and dynamic analysis to investigate the privacy and security features of these apps. In static analysis, we evaluate the source code of each app to identify the requested permissions, third-party libraries embedding, malware presence, and anti-analysis adoption. Whereas in dynamic analysis, we monitor the application during execution and capture the network traffic to observe the application interaction. Moreover, we analyse wallet apps' privacy policies for compliance and evaluate users' perceptions by investigating users' comments. To foster future research, we release our code and dataset used in this paper to the research community: `https://walletapps2021.github.io/`. The main contributions of our work are as follows:

Source code analysis. We collect a corpus of 457 wallet apps from Google Play and illuminate their evolution and popularity on Google Play (§ 2.2). We systematically analyse wallet apps' source codes and find potential security issues spanning from requesting dangerous permission, embedding third-party libraries for advertising and tracking purposes, presence of malware code, and using anti-analysis techniques, etc. (§ 4).

Our analysis identified 8 (1.7%) apps using sensitive permission such as `android.permission.DOWNLOAD_WITHOUT_NOTIFICATION` in their code which, once requested, enables the app to download any file or malware executables without user consent, 25 (5.4%) wallet apps embed malware code in their source code according to VirusTotal [39]. We also found 13 (2.8%) apps embed Cross-Library Data Harvesting (XLDH) library [40] which "illegally" extracts user information from legitimate libraries such as Facebook, Google, Twitter, and Dropbox.

Apps' network traffic analysis. We investigate the runtime and network behavior of the wallet apps by installing them into an Android phone and navigating the app's activities while running on the phone (§ 5). Our analysis detected broad evidence of potential security and privacy issues of the apps, including data leakages, using unencrypted transmission via HTTP, and requesting third-party domains containing advertisers and trackers. For instance, we detected 148 apps transmitting their traffic via unencrypted HTTP protocol. Moreover, we identified 15 apps (e.g.,`com.btcc.mobiwalletand` and `com.coinburp.mobile`) sharing user credentials and device information with third-parties via HTTP.

Privacy policy and users' reviews analysis. We analyse potential compliance issues of apps by evaluating apps' privacy practices and investigating app user reviews (§ 6). Particularly, our analysis detected 87 (19%) apps failed to provide privacy policy links for their user and 78 (17%) violated the privacy policy related to sharing users' information with third-parties.

2 Background and Data Collection Methodology

2.1 Background

Cryptocurrency is a digital asset that uses cryptography to ensure its creation security and transaction security [41]. There are over 2,500 different kinds of cryptocurrencies now, where the most well-known cryptocurrencies are "Bitcoin" and "Ethereum" which are traded at numerous cryptocurrency exchanges or marketplaces. Cryptocurrency exchanges offer trade services among cryptocurrencies. There are centralised exchanges (governed by a company), decentralized exchanges (provided an automated process for peer-to-peer trades), and hybrid exchanges [41].

The traded cryptocurrencies are normally kept at *wallet* in the form of hash values termed as *wallet addresses* [22]. Each address is corresponding to a pair of keys: public and private. The public key is used for external transactions such as sending or receiving cryptocurrencies. In order to prove the ownership of the cryptocurrency, each transaction is signed with a private key. If a user loses their private keys, they will lose their associated cryptocurrency assets. Anyone who gains the private key of a public address can authorize a transaction.

2.2 Cryptocurrency Wallet Apps Collection on Google Play

Google Play neither lists wallet applications as distinct apps' categories nor its search functionality yields `all` wallet applications. To collect and identify `all` possible wallet apps, we develop a crawling methodology. First, we query Google Play search with cryptocurrency-related keywords to collect a `seed` cryptocurrency wallet app. The keywords consist of "crypto", "cryptocurrency", "bitcoin", "coin", "Ethereum", "wallet", and others coin abbreviation such as "BTC", "ETH" and "DOGE". We then use Google Play Store Scraper API[1], a tool for scraping and parsing application data from the Google Play Store[2], to recursively crawl *similar* apps of the seed app IDs. By using this method, we found 3,629 app IDs. The API also returned the apps' meta-data such as the app's title, average rating, user reviews, descriptions, and categories to be further used for the app's selection. We then manually checked each app's description and filtered only cryptocurrency wallet apps. Finally, we obtain 457 free cryptocurrency wallet apps. We leverage `gplaycli`[3] to download the application packages (APKs) from Google Play of 457 cryptocurrency wallet apps. Given that Google Play does not report the actual release date of the apps but their last update, we use the date of their first comment as a proxy for their release date. For 63 (13.8%) apps without any user reviews as of this writing, we determine the approximate release date by their last update.

Figure 1 shows that cryptocurrency wallet apps have increased four-fold since February 2018. Figure 2 depicts the distributions of the number of installs and

[1] https://pypi.org/project/google-play-scraper/
[2] https://play.google.com/store
[3] https://github.com/matlink/gplaycli

an average rating of the analyzed cryptocurrency wallet applications on Google Play. We found that 21.8% (100) of the applications have at least 50,000 installs. We also observe that 51.3% (234) of the applications have at least 3.0 average ratings showing that the vast majority of the applications are positively rated by users. Conversely, we noted that 63 (13.8%) apps without any reviews have an average rating of 0.0 and have at most 500 installs.

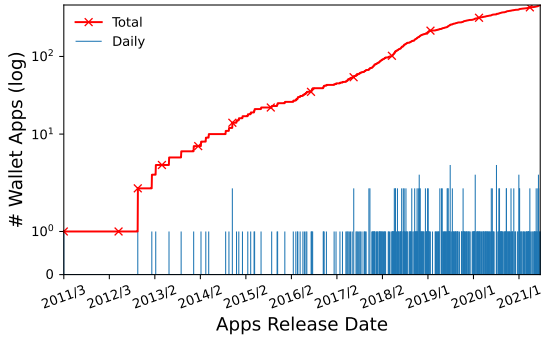

Fig. 1: Evolution of the analyzed cryptocurrency wallet apps on Google Play.

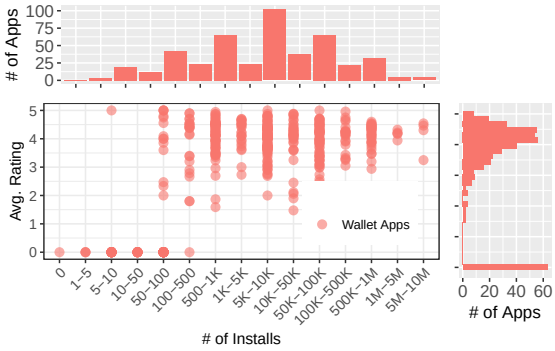

Fig. 2: Distribution of the number of installs and average app's rating of the analyzed cryptocurrency wallet apps on Google Play.

3 Assessment Methodology

We use a set of custom-built tools to assess source codes and network runtime behaviors of the apps for potential security and privacy issues. The following

sections describe our analysis categorised into static, dynamic, and compliance analysis.

3.1 Static Analysis

We assess wallet apps for requesting sensitive permission, presence of third-party tracking libraries and malware codes, use of security certificates, anti-analysis methods, compliance and user reviews analysis, and leakage of sensitive data to third-parties.

Permission Analysis. To get the resources it needs during runtime, each wallet app must declare all of its resource requirements in the `AndroidManifest.xml` file. We use `apktool` to decompress the APKs of the analysed apps and extract all permissions declared in `<uses-permission>` and `<service>`[4] tags. We analyse the requested permissions for potential vulnerabilities and wallet apps' capabilities for possible exploits.

Tracking Libraries. By inspecting the decompressed source code of each app, we examine the presence of embedded third-party libraries. To identify which libraries are associated with tracking, analytics, and advertising services, we aggregate the manually curated list of 383 tracking and advertising libraries from [32,19,20]. In particular, we inspect the APK file of an app to determine sub-directories and match them with our list of tracking and advertising libraries. If there is a match between the sub-directory and an entry of the third-party libraries, the app is deemed to be used by the corresponding third-party library. Given that apps may use obfuscation methods to hide the names of third-party libraries [8,14], we consider our results as a lower bound on the presence of third-party libraries in cryptocurrency wallet apps.

We also analyzed the existence of *suspicious* Cross-Library Data Harvesting (XLDH) libraries in cryptocurrency wallet apps. XLDH is a type of library that steals user information from legitimate libraries such as Facebook, Google, Twitter, and Dropbox. XLDH libraries actively scan the legit libraries' existence and extract important information from the library's Software Development Kit (SDK) embedded in the same apps. Extracted information including user access token, user name, advertisement ID, and user image is then sent to the XLDH libraries server. Due to the illegal operation, Facebook has taken legal action against several companies providing this XLDH library [40]. To identify the emergence of XLDH libraries, we scanned the third-party libraries embedded by the cryptocurrency wallet app and compared the results against a list published by [40].

Malware Presence. We leverage VirusTotal [39], an online solution that aggregates the scanning capabilities provided by over 70 antivirus engines (AV), to detect malware activities in the apps. We automate our malware analysis by using VirusTotal's Report API to retrieve the malware detection results. After completing the scanning process for a given app, VirusTotal generates a *positive*

[4] Note that service permissions are requested at runtime to enable specific functions such as connecting to a VPN network.

report that indicates which of the participating AV scanning tools detected any malware activity in the app and the corresponding malware signature (if any).

Out of 457 apps, 153 apps have a size more than the VirusTotal API threshold (32 MB) which cannot directly scan for by using the API. Fortunately, there are 35 apps that have been analyzed by the VirusTotal previously. Hence, we manually upload 118 apps to the VirusTotal website to scan for malware presence in the apps.

Anti-Analysis Detection. We determine whether an app uses any means of evading, obscuring, or disrupting the analysis of parties other than the application developers. In fact, malware developers rely on these techniques to evade basic analysis layers of application market stores such as Google Play [6]. For example, He et al. [16] found that 52% of their malware samples leveraged anti-analysis techniques to evade, obscure, or disrupt analysis methods. We use `APKiD`[5] tool to obtain a list of anti-analysis techniques such as "manipulator", "anti-virtual machine", "anti-debug", "anti-disassembly", "obfuscator", and "packer" (cf. Appendix A for further details).

3.2 Network Measurements

To investigate the runtime and network behavior of the cryptocurrency wallet apps, we manually install each app on a Huawei GR5 phone running Android Version 7.0 and navigate the app running on the phone. In order to capture the network traffic generated by each app, we run *mitmproxy*[6] at a WiFi access point to intercept all the traffic being transmitted between the mobile phone and the Internet. For each of the analysed apps, we manually navigate the app *activities* such as login and clicking on buttons to potentially execute app components. Our manual tests last for at least three minutes per app. We inspect the captured network traffic of each app for data leakages, communication with third-party domains, and HTTPS adoption.

3.3 Compliance and User Comments Analysis

Privacy Policy Analysis. We aim to examine the compliance of the apps' developers in two categories: *(i)* providing a privacy policy link, and *(ii)* adhering to the privacy policy in terms of sharing user information with third-parties (TP). In the first category, we mark an app as violating the privacy policy agreement if it does not provide a privacy policy link or does provide a misleading link such as a broken link or an inaccessible link. In the second category, we mark an app to be failing to adhere to the privacy policy if it declares not to share user information with the TPs in the privacy policy descriptions, but in fact, sharing them during the app's execution. We develop a tool to retrieve and extract the privacy policy description of each app using the `BeautifulSoup` Python library.

[5] https://github.com/rednaga/APKiD
[6] https://mitmproxy.org/

We then use a machine learning classification model [42] to classify if the app claims to collect and share user information with TPs in its privacy policy.

User Reviews Analysis. Unlike most financial applications such as banking apps developed and released by official financial institutions such as banks [3], cryptocurrency wallet apps can be developed and released by any party or developer. Another key feature of the wallet apps is that there is no underlying asset like the banking system. Users choose cryptocurrency wallets only based on trust. Thus, in this study, we make user review one of the important parameters in analyzing security issues and the possibility of privacy leaks from users of cryptocurrency wallet apps. We create scripts to aggregate user reviews (comments) for each app and classify them into positive (4- and 5-star), neutral (3-star), and negative (1- and 2-star) reviews according to the number of stars that the user chose for the app.

4 Static Analysis Result

4.1 Permission Analysis

Android classifies permission requests into three categories including *normal* permissions, *dangerous* permissions, and *signature* permissions [10]. We found 8,339 permission requests from 457 wallet apps. We group the permission requests according to the resource sensitivity access levels. Additionally, Android also allows third-parties to develop their own permissions with names and syntax tailored to the standard developer library. We found that 1,755, 2,802, and 570 permission requests fall into the dangerous category, normal category, and signature permissions respectively. In addition, we found that 3,212 permission requests were not covered by the permission level list listed in [10]. We then group these permission requests into the customized/third-party permission category for our further analysis.

Table 1: Top 10 dangerous permissions requested by analyzed cryptocurrency wallet apps in our dataset.

No	Permission Name	# of Request (%)
1	CAMERA	399 (87.3%)
2	WRITE_EXTERNAL_STORAGE	334 (73.1%)
3	READ_EXTERNAL_STORAGE	268 (58.6%)
4	INSTALL_PACKAGES	146 (31.9%)
5	READ_PHONE_STATE	97 (21.2%)
6	ACCESS_COARSE_LOCATION	96 (21.0%)
7	ACCESS_FINE_LOCATION	96 (21.0%)
8	RECORD_AUDIO	82 (17.9%)
9	READ_CONTACTS	68 (14.9%)
10	GET_ACCOUNTS	54 (11.8%)

Table 1 shows the 10 most dangerous permissions requested by cryptocurrency wallet apps. We detected that 399 (87%) apps request permission to access the device's camera to activate the camera during the user document verification process including taking a photo of a legitimate government-issued ID and scanning the wallet address in the form of a QR code. We also found 82 (18%) apps requesting voice recordings as shreds of evidence that users give consents for them to store user data.

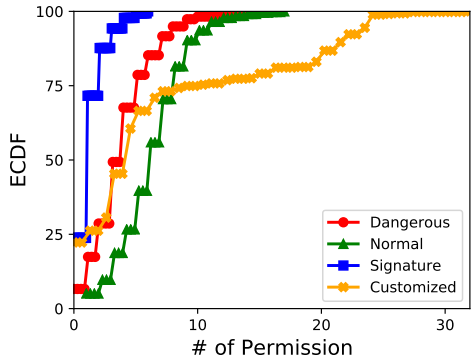

Fig. 3: Empirical cumulative distribution function (ECDF) of permission per app across various categories as per Android Official API [9].

On average, each app requests 18 permissions, which consist of 4 dangerous permissions, 7 customized permissions, 6 normal permissions, and 1 signature permission. Specifically, Fig. 3 shows that 95% of apps requesting less than 5 signature permissions, 65% of apps request 3 to 7 dangerous permissions, and 65% of apps request 5 to 10 normal permissions. Surprisingly, 35% of apps requested 5 to 20 customized permissions, and 15% of apps requested more than 20 customized permissions. This trend indicates a shift in the mobile programming paradigm from traditional Android libraries to reproducible third-party libraries.

Further analysis of customized permissions found that out of 3,212 customized permissions requested by 353 apps, there are 236 distinct permission names of this permission type. From permission names, there are 12 permissions embedded by more than 100 apps. Table 9 lists the top 20 requested third-party permissions used to support third-party libraries embedded by apps. Most of these permissions are useful in the push notification process from third-party cloud facilities to the device installing the app.

Another worth noting in apps' permission analysis is the compliance of apps' developers to the use of permission agreements and the potential security risk to certain apps. We found that 26 apps request for `android.permission.MOUNT_UNMOUNT_FILESYSTEMS` (cf. Table 2. According to [10], this permission can be used to mount and unmount external storage such as `SSD` cards. However, this permission is only for pre-installed applications or applications that were already

installed when the device was distributed. In addition, we discovered that 8 apps request `android.permission.DOWNLOAD_WITHOUT_NOTIFICATION` (cf. Table 2) which allows the app to download any file without user consent. This can be very dangerous if the app automatically downloads malicious applications into the device. `com.nexowallet` and `com.polehin.android` are two out of eight apps that requested this permission and those apps have been installed by more than 1 million devices and have review scores of 4.6 and 4.2 respectively.

Table 2: (Un)Mount file system and download without notifications permissions requested by the analysed apps.

Sensitive Permission	# of Apps (%)	Example App
MOUNT_UNMOUNT_FILESYSTEMS	26 (5.7%)	com.bityard.us2
DOWNLOAD_WITHOUT_NOTIFICATION	8 (1.8%)	com.burency.app

4.2 Third-party Libraries Penetration

We found a total of 59 distinct third-party libraries embedded in 391 apps. Due to the limited list in our dictionary and the obfuscation mechanism adopted by the apps, we were unable to detect third-party libraries in 66 apps ($\approx$ 14% of 457 wallet apps). Depending upon their intended functionalities, we group the 59 distinct libraries into four main categories: Analytics, Ads & Tracker, Payment, and Social Media libraries.

Table 3: Dominant libraries grouped by the library category. (G. : Google)

No	Analytics		No	Ads&Tracker	
	Names	Count (%)		Names	Count
1	G. Analytic	61 (3.7%)	1	G. Ads	259 (15.8%)
2	Mixpanel	20 (1.2%)	2	Appsflyer	66 (4.0%)
3	adjust	15 (0.9%)	3	Tencent	25 (1.5%)
4	Umeng	11 (0.7%)	4	AppLovin	4 (0.2%)
5	Flurry	10 (0.6%)	5	Appboy	4 (0.2%)
No	**Payment**		**No**	**Social Media**	
	Names	Count		Names	Count
1	Squareup	166 (10.1%)	1	Facebook	198 (12.1%)
2	Intuit	8 (0.5%)	2	Twitter	10 (0.6%)
3	Paypal	1 (0.1%)	3	Weibo	3 (0.2%)
4	Urbanairship	1 (0.1%)			

Table 3 shows the dominance of Google Analytics and Google Ads for the Analytics and Ads & Tracker categories which are embedded in 61 and 259 apps, respectively. As for the Payment and Social Media categories, Squareup and Facebook are the most popular and adopted in 166 and 198 apps, respectively.

According to data distribution, the adoption rate of third-party libraries by cryptocurrency wallet apps is considered to be low. Particularly, Table 4 shows that 172 (37.6%) apps did not adopt Ads & Tracker libraries whereas other 270

(59.1%) apps adopted only 1 or 2 libraries, 13 (2.8%) apps adopted from 3 to 5 libraries and 2 (0.2%) apps adopted more than 5 libraries. This adoption rate is much smaller than the adoption of Ads & Tracker libraries in other genres of apps, for example, measurement by Sentana et al., [33] found that 22.2% of Android health-related apps embed at least five different third-party libraries, or in the measurement results by [36] who found more than 43.0% of non-mobile health apps in their corpus embed more than 5 Ads & Tracker libraries. The two apps that embed the most Ads & Tracker libraries in their package are `network.xyo.coin` and `com.callsfreecalls.android`, with 10 and 9 libraries, respectively and they have been installed more than 1 million times.

Table 4: Distribution of third-party libraries embedded by the analyzed apps.

# Of Libraries	Analytics	Payment	Social Media	Ads & Trackers
1	94 (20.6%)	166 (36.3%)	194 (42.5%)	211 (46.2%)
2	12 (2.6%)	5 (1.1%)	7 (1.5%)	59 (12.9%)
3	1 (0.2%)	0 (0.0%)	1 (0.2%)	10 (2.2%)
4	0 (0.0%)	0 (0.0%)	0 (0.0%)	2 (0.4%)
5	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (0.2%)
6+	0 (0.0%)	0 (0.0%)	0 (0.0%)	2 (0.4%)

The adoption rate of the other three types of libraries is less than the adoption rate of the Ads & Tracker library. There are only 202 (44.2%) apps that adopted 1 to 3 Social Media libraries, 171 (37.4%) adopted 1 to 3 Payment libraries, and only 107 (23.4%) apps adopted 1 to 3 Analytics libraries. This phenomenon reinforces the exposure made by [25] where advertising is not the main source of income in the cryptocurrency wallet ecosystem. Most apps support their operational costs through affiliate fees. In this model, the cryptocurrency wallet apps accommodate the crypto swap process from the cryptocurrency swap service provider. Some of the profits obtained by the service provider are from the difference in currency values which are shared with apps. Another business model is to make cryptocurrency wallets become a part of cryptocurrency exchange apps and get a split fee from the exchange transaction. In addition to the business model, some apps gain funding from the Initial Coin Offering (ICO) for cryptocurrency wallets that are integrated with certain coins.

Apart from the four types of third-party libraries, we also analyzed the existence of *suspicious* Cross-Library Data Harvesting (XLDH) libraries in cryptocurrency wallet apps. We found that 13 (2.8%) apps embed XLDH libraries in their package as shown in Table 5. Particularly, `com.yandex.metrica` library, which is embedded by 4 apps, extracts the Google Advertising id, Android ID, and user MAC address and then exfiltrates the information to `https://startup.mobile.yandex.net`. Similar information was also extracted by `com.leanplum, cn.sharesdk, com.inmobi`. More detailed information is harvested by `com.umeng.socialize` which includes AccessToken and user data (ID/name/link/photo) on several social media platforms including Facebook, Twitter, Dropbox, Kakao, Yixin, Wechat, QQ, Sina, Alipay, Laiwang, Vk, Line, and Linkedin.

Table 5: Crypto Wallet embedding Cross-Library Data Harvesting (XLDH) Libraries

No	App ID	XLDH Library Name
1	app.hodlify	com.yandex.metrica
2	co.bitx.android.wallet	com.leanplum
3	com.evraon.trading	com.yandex.metrica
4	com.ixx_android	cn.sharesdk
5	com.okinc.okex.gp	cn.sharesdk
6	io.bincap.exchange	com.yandex.metrica
7	network.xyo.coin	com.inmobi
8	com.btcc.mobiwallet	cn.sharesdk
9	com.zengo.wallet	com.leanplum
10	com.sixpencer.simplework	cn.sharesdk
11	com.fox.one	com.umeng.socialize
12	com.beecrypt.beecrypthd	com.yandex.metrica
13	com.hconline.iso	cn.sharesdk

4.3 Malware Presences

Based on VirusTotal results, we found 25 apps detected containing malware. Specifically, 18 apps were detected by 1 antivirus engine, 3 apps were detected by 2 antivirus engines, and 4 apps were detected by more than 3 antivirus engines. Moreover, 9 antivirus engines detected malware on `com.top1.group.international.android`, 8 engines detected malware on `com.jex.trade`, and 7 and 4 engines detected malware on `com.legendwd.hyperpayW` and `im.token.app`, respectively.

We further cross-validate this analysis results with the anti-analysis measurement results in § 4.4. As we shall show later in § 4.4, there are four apps that embed the Jiagu packer to encrypt the .dex file. In this measurement, there are 4 antivirus engines (i.e., Ikarus, Fortinet, ESET-NOD32, and MaxSecure) that consistently detect the same malware in the four apps that embed *Jiagu* packer in their packages.

4.4 Anti-Analysis Detection

Figure 4 depicts our analysis results of anti-analysis methods employed by wallet apps. We discuss our analysis of the evasion methods in the following.

Manipulator. We found that 18 (3.9%) apps are marked as containing manipulator because the *Dalvix Executable (.dex)* files developed using *dexmerge* compiler. Note that *.dex* file, which exists in each APK, is a byte-code file converted from *Java.class*. Thus, it can be executed by the devices. Originally, *dex* file is developed using *dx* or *r8* compiler. APKiD tool will mark an app as containing a manipulator if (i) the original *.dex* files of the app are modified using a modification library such as *dexmerge* or (ii) *.dex* files are created from reverse-engineered source code using *dexlib* library, which is commonly used by decompiler tools such as apktool or smali [31,30]. APKiD tool identifies the manipulator by analyzing the change history in *Map Ordering Type* of the *.dex* files

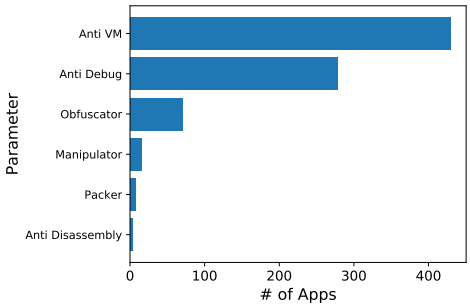

Fig. 4: Anti-Analysis mechanisms employed by the analysed wallet apps.

since the code sequence resulted from original *.dex* compiler, *dexmerge, dexlib or dex2lib* is different [13].

Anti Virtual Machine. We detected 429 (93.8%) apps adopting anti-virtual machine (Anti-VM) analysis in their package. An Anti-VM is a mechanism to detect whether the apps are executed on an emulator or real device. The goal is to impede reverse-engineering tools and techniques so the reverse engineer cannot get the source code easily. The most common mechanism to check whether the device is emulated [27] is to analyze `build.prop` file containing a list of Build API methods, including `Build.Fingerprint`, `Build.Hardward`, and `Build.Device` and other device's properties. An alternative method is to check the *Telephony manager* which contains fixed API values for Android emulators including `getNetworkType()`, `getNetworkOperator()`, `getPhoneType()` and other network-related properties.

Anti Debug. We found 278 (60.8%) apps leveraging anti-debugging techniques to disrupt the reverse engineering of their source code. The anti-debugging technique ensures that the apps are not running under a debugger or changing the app's behavior when running under the debugger mode. Android provides two levels of debugging and anti-debugging protocol [26]. The first level of debugging can be conducted in communication protocol between Java Virtual Machine and debugger using Java Debug Wire Protocol (JDWP). We can identify anti-debugging by verifying if the setup includes the *debuggable flag* in *ApplicationInfo* or by checking the *timer checks routine*. While the next level of anti-debug technique is to conduct traditional debugging by using *ptrace* in Linux *system call*. In this research, we found all of the 287 cryptocurrency wallets activating the debuggable flag of `Debug.isDebuggerConnected() check`, which is part of JDWP anti-debug level.

Obfuscator. Overall, we found 70 (15.3%) apps leveraging obfuscation tools in their package. Of that number, 3 apps obfuscated by Dexguard [14], 3 apps obfuscated by Arxan [11]. obfuscation is a process of concealing the original source code, binary code, or byte-code into an obscure set of characters, and 6 apps obfuscated by Clang [35]. Dexguard is a proprietary Android obfuscation tool that provides multi-layer protection against the static and dynamic analysis of byte-code, manifest, and all other resources included in distribution packages. While Dexguard obfuscated the byte-code level of Android Apps, Arxan and

Clang are categorized as Low-Level Virtual Machines tools that obfuscated the binary code level of Android Apps.

Anti Disassembly. This technique prevents the reverse engineer from disassembling the byte-code into higher-level code such as Java or Smali. The most popular anti-disassembly mechanism in Android is by developing part of the code segment in C or C++ using Native Development Kit (NDK) [34]. NDK provides platform libraries to manage native activities and access physical device components [7]. NDK uses *CMake* as a native library compiler that creates a different *byte-code* structure compared to the code written in Java or Kotlin. Hence, it impedes common Android tools such as Apktool or Smali to disassemble the *byte-code*. It required an advanced reverse engineer familiar with ARM processor architecture, Assembler language, Java Native Interface (JNI) convention, and Application Binary Interface (ABI) compiler to decompile the *byte-code*. More advanced techniques are used by malware developers to evade disassembly tools explained by [24]. The technique leveraging *jmp* and *call* commands in *byte-code* level to direct the instruction flow to a certain location with a constant value, or direct the flow to the same target memory location. This technique will produce a false listing of source code when it disassembles using a decompiler tool. We found 4 apps leveraging the anti-disassembly techniques. Analysis of those apps returns the value of "Illegal class name", indicating the decompiler result violates the standard structure of Java or Kotlin.

Packer. Initially, Packer was created to protect intellectual property in the form of source code on Android apps. These tools prevent third-parties from analyzing source code or doing reverse engineering. Packer works by encrypting the `.dex` files in the Android package and storing the encryption results in a secure new block architecture. Unlike the obfuscator which will be decrypted when executed on the device, the packer is remained stored in the packer block and uses unpacker when it will be executed on the device. However, commercial packers are often used by malware developers to hide malicious codes [12,5]. We found that 7 (1.5%) apps use packers, of which 4 apps embed `Jiagu` packer, while the three remaining apps use `Ijiami`, `SecNeo`, and `Yidun` packers. Note that the wallet apps using `Jiagu` packer are also detected as malware by VirusTotal (cf. § 4.3).

5 Dynamic Analysis Result

We investigate the runtime and network behavior of the cryptocurrency wallet apps. Out of 457 tested apps, we successfully captured the network traffic of 391 apps. The reasons that we failed to capture traffic of 66 apps (i.e., pcap files are empty) include the app navigation process crashing and the app installation failure.

5.1 Securing Network Requests

To secure communication from in-path attackers, apps leverage transport layer security (TLS) and HTTPS. For each wallet app, we filter TCP flows in network

traces (saved in `pcap` file), to identify whether the connection establishment between the app and the Internet is over TLS. We also identify the TLS version, cipher name, and the HTTP version used in the apps' traffic. We also filter network traffic for HTTP(S) requests. We found that 261 (66.7%) apps have established TLS for all of their traffic where they use $TLSv1.2$ and/or $TLSv1.3$ in their traffic, 148 (37.8%) apps include traffic that was not over TLS when they use $HTTP$ scheme in their traffic and 9 (2.3%) apps do not use TLS in all of their traffic. Of the 391 apps that we successfully captured network traffic, 148 (37.8%) apps transmitted some of their traffic over unencrypted HTTP protocol which could potentially result in in-path modification and traffic analysis of transactions made by wallet apps.

5.2 Sensitive Data Aggregation and Sharing

We observe whether an app collects and transmits data related to users' information and wallet information in their network traffic. Based on the captured traffic dumped into the `.har` file, we extract the HTTP request package's header and payload and find whether user information is in the app's traffic.

Data Aggregation Practices. For each app, we extract all HTTP requests with the `POST` method and inspect the *postData* field of the package's header to identify data related to the user, credential, device, location, and wallet information. If so, the app is claimed to collect personal information as well as wallet information. In order to detect if an app collects user information, we match predefined keywords with the key values in the *postData* field. For any matching keywords, we claim that the app collects user information. A similar strategy is applied to the detection of collecting device information, credential information, location information, and wallet information. To define related keywords to each information category, we first scan all unique keys in the *postData* field of all apps' traffic. We then search for related keywords for each information category. As the keys used in *postData* field to refer to the same information are not exactly matched. We carry all possible keywords for each information category to ensure that we do not miss any app that collects a piece of given information. For instance, the predefined keywords for user information are determined based on the list of all unique keywords we scanned for from all apps' traffic. We search for all keys in that list that contain the string *'name'*. After that, we manually filter out keywords that include *'name'* but do not refer to personal information. Finally, we obtain a predefined keyword list related to user information including *'first_name', 'last_name', 'surname', 'Username', 'email', 'account_name', 'username', 'name', 'partner_name', 'given_name', 'named_user_id', 'user_name', 'full_name', 'emailAddress', 'emailToken', 'email_id', 'fullname'*. Similarly, we obtain the keywords for device information, credentials, location, and wallet information. We provide details of our keywords for each user-related information category in Appendix B.

Based on the apps' traffic and our predefined keywords, we identify a number of apps that collect personal information as well as wallet information. Table 6

shows the identified number of apps (out of 391 apps) that collect a given information.

Table 6: Number of apps collecting personal and wallet information.

Collected Attributes	Apps Collecting Info	Apps Sharing with FP	Apps Sharing with TP	Sharing via HTTP
User Info	67 (17.1%)	10 (2.5%)	57 (14.6%)	4 (1.0%)
Credential Info	50 (12.8%)	9 (2.3%)	41 (10.5%)	4 (1.0%)
Device Info	83 (21.2%)	5 (1.2%)	78 (19.9%)	5 (1.3%)
Location Info	16 (4.1%)	-	16 (4.1%)	-
Wallet Info	44 (11.2%)	2 (0.5%)	42 (10.7%)	-

Sharing Sensitive Data with Third-Parties. For all apps that collect personal information and wallet information, we further inspect the requested `URL` in the *URL* field of the packet's header to identify if the collected data is shared with third parties.

We also identified apps that share the collected information with TPs via the HTTP scheme. Table 6 also shows the identified number of apps that share collected personal information and wallet information with third parties and apps that share these data using the HTTP scheme. Specifically, we found that 15 apps (e.g., `com.btcc.mobiwallet` and `com.coinburp.mobile`) share user credentials and device information with third-parties via HTTP.

5.3 Third-party Domain Request

We identify third-party domains in the app traffic from the captured `pcap` file. For each analysed app, we capture all requested domains from `har` file and classify requested domains into first-party domains and third-party domains (i.e., Domains that do not belong to apps' developers). To determine apps' communication with third-parties, we leveraged filter lists: EasyList [1] an advert block list, and EasyPrivacy [2] a supplementary block list for tracking, to filter advert and tracking related third-party domains requested by the tested apps.

113 (28.9%) apps requested more than 10 unique domains during the app execution. Among them, the app with the handle name of `io.atomicwallet` requested 66 different domains. Individually, 70 (17.9%) apps requested *one* third-party domain, 338 (86.4%) apps requested more than *two* third-party domains and 72 (18.4%) apps requested more than 10 third-party domains. The app with the handle name of `com.paymintlabs.paymint` requested 44 different third-party domains. Top 5 third-party domains `firebaseinstallations.googleapis.com`, `google.com`, `crashlytics.com`, `rqmob.com` and `facebook.com` were found to be requested by 169 (43.2%), 133 (34%), 91 (23.2%), 87 (22.2%), and 76 (19.4%) apps, respectively.

6 Privacy Policy Compliance and User Review Results

6.1 Privacy Policy Analysis

Upon the privacy policies extraction, we managed to extract 327 privacy policies and failed to extract 130 privacy policies due to several reasons as listed in Table 10. 31 (6.8%) cryptocurrency wallet apps provide an untraceable link for their privacy policy in Play Store content page information, while 17 (3.7%) cryptocurrency wallet apps did not provide any link to their privacy policy documents. The result from privacy policy extraction is then used in the following analysis:

Compliance to Provide Privacy Policy Link. As a part of our first analysis, which is related to the developer compliance to provide a privacy policy link, we used the extraction result and exclude several reasons for failures, such as `HTTP error 403` and `HTTP error 503` which is used by the server as a part of the defense against Denial of Service (DoS) attack (cf. Table 10). We also exclude several links that can be traced if it navigated manually by humans to reach privacy policy such as the `HTTP error 104` and `HTTP error 111` as well as `SSL_ERROR_BAD_CERTIFICATE_DOMAIN`. And for the last, we exclude the apps that are no longer available in Play Store. Hence, in total Overall, we found 87 (19.0%) cryptocurrency wallet apps failed to provide a privacy policy on Google Play.

App's adherence to information sharing policy. This analysis consists of two steps: classification of the app's privacy policy for collecting user information and validation of sharing the collected information with third-parties (TP). For each app, we first classify the app's privacy policy as *true* or *positive* if they declare or claim that the apps share the user information with the TPs, while the app's privacy policy is labeled as *false* or *negative* if they do not declare to share the user information to the TPs. Second, we cross-validate the apps that are labeled to be false to the third-party libraries embedded by corresponding apps and cross-validate those apps to the third-party domains accessed during the app's runtime. We assume that the cryptocurrency wallet apps violate the privacy policy in terms of sharing user information with the third-parties if they do not declare it in privacy policies while embedding tracking libraries (§ 4.2) or they send user information to the third-party domains (§ 5.3).

To automatically detect if the app claims to share user information with TPs, we rely our classification on an existing corpus of privacy policies [42] annotated by legal experts. This corpus contained 213 and 137 privacy policies labeled as positive and negative, respectively. However, the positive annotation in this corpus is only given for the policy that contains very specific information sharing and annotates the general information sharing phrases to the negative. For instance, specific phrases such as "We share your email address with third parties" would be annotated as *positive*, while general phrases such as "We share your information with third parties" would be labeled as *negative*. Hence, we relabel this corpus by removing privacy policies that contain general information sharing in the negative class. After this step, 35 privacy policies remain in the

negative subset. To avoid an imbalance of class data, we reduce the size of the positive subset and make our *new corpus* containing 68 and 35 privacy policies labeled as positive and negative, respectively. We then train a Support Vector Machine (SVM) classification model based on our new corpus.

Since our classification model is trained in English-based language, we then exclude non-English privacy policies from 327 (out of 457) cryptocurrency wallet apps from which we could extract their privacy policy text. We found that 60 (out of 327) privacy policies were written in a non-English language, including binary texts. We then classify these 267 privacy policies using our trained classification model. As a result, we found 93 privacy policies classified as false meaning that the corresponding apps did not declare or claim to share user information with TPs, and 175 privacy policies classified as true indicating that the apps declare to share user information with the TPs.

We then cross-validate 93 apps which corresponding privacy policy apps labeled as false to the apps adopting third-party libraries in Sec. 4.2. We then marked the apps that embed at least one third-party library as apps that failed to adhere to privacy policies related to sharing users' information with third parties. As a result, we found 78 (17.0%) cryptocurrency wallet apps fall into this category.

We examine 93 apps that do not claim to share user information with third parties (i.e., being labeled as false in the privacy policy classification) by investigating their network traffic during execution time (§ 5.2). We found 26 apps that did not declare to share user information with the TPs had shared user information including personal information, device information, credentials, location, and wallet information. Details of our results are shown in Table 7. Consequently, we also marked these apps as violating privacy policy in terms of sharing user information with TPs.

Table 7: Apps sharing user information with TPs w/o claiming in privacy policy

Attribute	**# Apps**	**Shared Information**
Personal Info	4 (4.3%)	'username', 'surname', 'email', 'name'
Device Info	11 (11.8%)	'deviceType', 'device uuid' 'deviceData', 'device name' 'deviceFreeSpace', 'device id' , 'device token', 'deviceModel'
Credentials Info	4 (4.3%)	'password', 'password score'
Location Info	2 (2.2%)	'locale'
Wallet Info	5 (5.3%)	'primary public key', 'branch key', 'wallet version' 'backup public key', deployment key'

6.2 User Reviews Analysis

We use users' negative comments (reviews) to capture the perceptions and concerns about the security and privacy features of wallet apps. Our reasoning to focus our analysis on negative reviews, 1- and 2-star reviews appearing on Google Play, for popular apps is that any serious concern exposed by a user should receive a negative review. Overall, we obtained 673,424 reviews corresponding to

403 (88.2%) wallet apps. We noted that 54 (12.8%) apps have no user reviews while 78.5% (359) of the analyzed apps have at least one negative comment with a total of 175,564 or 26.1% of total reviews.

For further analysis, we process negative reviews by grouping them based on the type of complaint. We detect the complaint type in each negative review based on the occurrence of keywords in the review and group it into six categories, including fraudulent, bugs, authentication, security, usability, ads, and tracker. For example, the appearance of the word "scam", "fake" or "liar" in a negative review indicates a complaint about `fraudulent` activities. Similarly, reviews containing keywords such as "bugs" or "error" into `Bugs complaint`. More details about the complaint category and the keywords mapping can be seen in Table 8.

Table 8: Number of User Complaint per Category as well as the correlated apps. Percentage in "# of Complaint" using the total negative review as the denominator.

Category	**# of Complaint (%)**	**#of Apps(%)**	**Keywords**
Fraudulent	50,628 (28.8%)	287 (62.8%)	'scam','fake','money', 'liar','purchase','payment', 'credit card','debit card', 'cash','manipulation'
Bugs	22,742 (13.0%)	256 (56.0%)	'bug','error','crash','update', 'upgrade','not responding', 'freeze','stuck'
Authentication	37,791 (21.5%)	262 (57.3%)	'verification','verify', 'verif' 'verified','account', 'notification','login','register'
Security	4,692 (2.7%)	176 (38.5%)	'security','secure','hack','bot' 'hacking','hacker','insecure'
Usability	11,599 (6.6%)	213 (46.6%)	'confuse','confusing','bad', 'rubbish','slow', 'junk','user interface'
Ads and tracker	772 (0.4%)	102 (22.3%)	'ads','video ads','tracker', 'intrusive ads','massive ads' 'advert', 'advertisement'

Table 8 also shows the number of complaints and their percentage of total negative reviews grouped by category. Most of the users complaints are related to *fraud* with 50,628 (22.5%) complaints, followed by *authentication* with 37,791 (14.9%), *bugs* with 22,742 (12.3%), *usability* with 11,599 (4.5%), *security* with 4,692 (2.6%), and *ads and tracker* with 772 (0.6%). This result is surprising as we expect ads and trackers to get a large proportion of complaints considering that the corpus of the analyzed wallets apps are *free* apps. However, after looking in more detail at the business process of wallets and considering the low adoption of third-party ads and tracker libraries discussed in § 4.2, it is acceptable that ads and trackers are not a "big player" in wallet apps. To observe the distribution of complaint categories in cryptocurrency wallet apps, we calculate the ratio

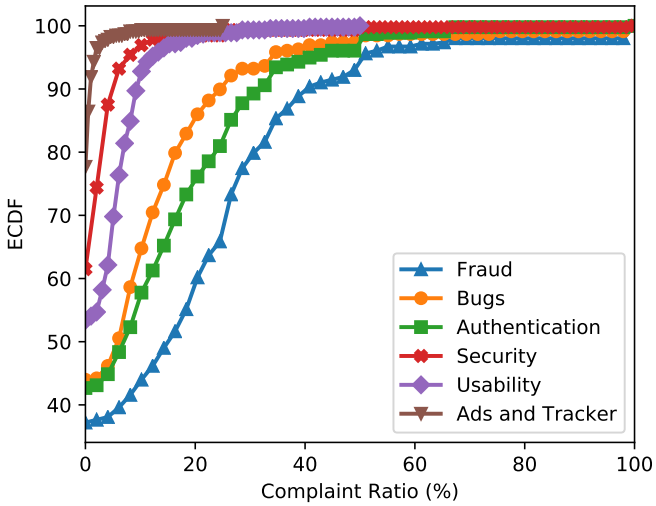

Fig. 5: ECDF of user complaints ratio per app across various categories.

of complaints occurrence per category to the total negative reviews for each app. This is to accommodate gaps between the apps with a large number of reviews and a small number of reviews. For example, `com.coinbase.android` has a fraudulent ratio of 32.8% calculated from 10,210 fraudulent complaints divided by 31,107 negative reviews owned by the app.

Based on this calculation, fraud still dominates compared to other complaint categories. There are 20 apps (4.4%) that have more than 50% fraud complaints. All of these apps have a totally negative review below 100 and 15 apps have a total review (positive, normal, and negative) under 100. Figure 5, depicts the distribution of the ratio of negative reviews to the total number of reviews across various categories.

We also cross-validated the negative review ratio of VirusTotal's detection results on malicious apps, as shown in Table 11. As a result, of the 25 apps that were detected as malicious by the VirusTotal, 8 apps have received negative ratios above 25% where 3 of them have been installed more than 100K times.

7 Related Work

Recently, few work studied the security and privacy risks of Android cryptocurrency wallet apps. For instance, He et al. [15] discovered threat vectors of cryptocurrency wallet apps and studied the security weaknesses of both the Android system and cryptocurrency wallets. However, this work provides an assessment of just two apps.

Hu et al. [18] explored the security features of the 10 most popular Bitcoin wallet applications and discovered three security vulnerabilities of Bitcoin wallet applications including privacy leakage, spamming and financial loss and demonstrated corresponding proof-of-concept attacks. Sai et al. [29] examined the se-

curity issues of 48 commonly used Android cryptocurrency wallet applications. Nevertheless, this work mainly focuses on security vulnerabilities rendered by requested permissions. Li et al. [22] studied the security risks of 8 common Android cryptocurrency wallet apps by examining requested permissions, backup files, Android clipboard, and accessibility service.

Biryukov and Tikhomirov [4] proposed a security and privacy analysis method based on clustered transactions on timing analysis for Bitcoin, Dash, Monero, and Zcash cryptocurrencies. The measurement relies on four main parameters including information Leak to external storage, XSS attacks via Javascript in WebView, Insecure connection, and information leak into logs. Uddin et al [37], proposed Horus, a security assessment framework for Android Cryptocurrency apps that includes static and dynamic analysis. Static analysis in Horus focuses on the security assessment based on the API calls while dynamic analysis was only used to profile the app's structure and to identify the key location storage during the runtime.

In contrast, our work provides a large-scale comprehensive analysis of the security and privacy features of 457 Android cryptocurrency wallet applications. The static analysis in our study consists of several security and privacy parameters that include permission, app packaging, anti-analysis adoption, the use of third-party libraries, and malware presence. While the dynamic analysis in our study conducted the identification of standard security practices and capture the network traffic to seek the potential information leakage to the third party. This study also provides analysis results from the user's point of view and potentially of developer compliance related to the privacy policy.

8 Conclusion

Unlike most free apps which use ads to monetize their apps, wallet apps use split or sharing fees to support their business process. Hence, it gives a unique characteristic to the apps' structure in terms of third-party libraries' adoption and permission requests. In addition, most wallet apps also utilize third-party cloud infrastructure, especially to support push notification facilities, which causes the adoption of third-party libraries in this regard to be large. The adoption of this third-party library is helpful in accelerating application development, but on the other hand, it also leaves security and privacy issues for users.

In this study, we found several permission requests that are very dangerous for users and also the adoption of a malicious library adopted by wallet apps. The adoption of anti-analysis techniques by wallet apps has also led to a malware-detecting issue by a number of antivirus engines. We also found several violations of the privacy policy agreement between the developer and the user by the wallet apps. Thus, we expect the results of this assessment would assist users in observing the security and privacy of wallet apps on marketplaces such as the Google Play Store. To foster future research, we release our code and dataset used in this paper to the research community: https://walletapps2021.github.io/.

References

1. Easylist (2021), https://easylist.to/easylist/easylist.txt
2. Easyprivacy. https://easylist.to/easylist/easyprivacy.txt (2021)
3. BBVA: Blockchain - what are the differences between a digital currency and a cryptocurrency? (January, 2021), https://www.bbva.com/en/what-are-the-differences-between-a-digital-currency-and-a-cryptocurrency/
4. Biryukov, A., Tikhomirov, S.: Security and privacy of mobile wallet users in bitcoin, dash, monero, and zcash. Pervasive and Mobile Computing **59**, 101030 (2019). https://doi.org/https://doi.org/10.1016/j.pmcj.2019.101030, https://www.sciencedirect.com/science/article/pii/S1574119218307181
5. Caijun, S., Hua, Z., Sujuan, Q., Nengqiang, H., Jiawei, Q., Hongwei, P.: Dexx: A double layer unpacking framework for android. IEEE Access (2018)
6. Chau, N., Jung, S.: An entropy-based solution for identifying android packers. IEEE Access (2019)
7. Developer, A.: The android ndk: toolset that lets you implement parts of your app in native code, using languages such as c and c++. (2020), https://developer.android.com/ndk
8. Developer, A.S.: Shrink, obfuscate, and optimize your app (2020), https://developer.android.com/studio/build/shrink-code
9. Developers, A.: Permissions overview: Android developers (2020), https://developer.android.com/guide/topics/permissions/overview
10. Developers, A.: Android documentation - manifest.permission (2021), https://developer.android.com/reference/android/Manifest.permission#MOUNT_UNMOUNT_FILESYSTEMS
11. Digital.ai: Arxan: App code obfuscation. https://digital.ai/glossary/app-code-obfuscation (2020), last accessed: 18/08/2021
12. Duan, Y., Zhang, M., Bhaskar, A.V., Yin, H., Pan, X., Li, T., Wang, X., Wang, X.: Things you may not know about android (un)packers: A systematic study based on whole-system emulation. In: NDSS (2018)
13. Fenton, C.: Building with and detecting android's jack compiler (2016), https://calebfenton.github.io/2016/12/01/building-with-and-detecting-jack/
14. Guardsquare-Mobile-Application-Protection: Dexguard: Android app security - protecting android applications and sdks against reverse engineering and hacking (2020), https://www.guardsquare.com/en/products/dexguard
15. He, D., Li, S., Li, C., Zhu, S., Chan, S., Min, W., Guizani, N.: Security analysis of cryptocurrency wallets in android-based applications. IEEE Network **34**(6), 114–119 (2020)
16. He, R., Wang, H., Xia, P., Wang, L., Li, Y., Wu, L., Zhou, Y., Luo, X., Guo, Y., Xu, G.: Beyond the virus: A first look at coronavirus-themed mobile malware (2020)
17. Hsieh, W.C., Engler, D.R., Back, G.: Reverse-engineering instruction encodings. In: USENIX Annual Technical Conference, General Track. pp. 133–145 (2001)
18. Hu, Y., Wang, S., Tu, G.H., Xiao, L., Xie, T., Lei, X., Li, C.Y.: Security threats from bitcoin wallet smartphone applications: Vulnerabilities, attacks, and countermeasures. In: ACM CODASPY (2021)
19. Ikram, M., Kaafar, M.A.: A first look at mobile ad-blocking apps. In: IEEE NCA (2017)
20. Ikram, M., Vallina-Rodriguez, N., Seneviratne, S., Kaafar, M.A., Paxson, V.: An analysis of the privacy and security risks of android vpn permission-enabled apps. In: ACM IMC (2016)

21. Kruegel, C., Robertson, W., Valeur, F., Vigna, G.: Static disassembly of obfuscated binaries. In: USENIX security Symposium. vol. 13, pp. 18–18 (2004)
22. Li, C., He, D., Li, S., Zhu, S., Chan, S., Cheng, Y.: Android-based cryptocurrency wallets: Attacks and countermeasures. In: 2020 IEEE International Conference on Blockchain (Blockchain). pp. 9–16. IEEE (2020)
23. Maier, D., Müller, T., Protsenko, M.: Divide-and-conquer: Why android malware cannot be stopped. In: 2014 Ninth International Conference on Availability, Reliability and Security. pp. 30–39. IEEE (2014)
24. Nair, R.: Techbliss - tutorial anti-disassembly techniques used by malware (a primer) (2015), https://www.techbliss.org/threads/anti-disassembly-techniques-used-by-malware-a-primer-by-rahul-nair.804/
25. Ohayon, O.: Hackernoon - the business model of crypto-wallets (2018), https://hackernoon.com/the-business-model-of-crypto-wallets-89aeed8322dc
26. OWASP: Testing anti-debugging detection (mstg-resilience-2) - android anti-reversing defenses (2020), https://mobile-security.gitbook.io/mobile-security-testing-guide/android-testing-guide/0x05j-testing-resiliency-against-reverse-engineering, oWASP Mobile Security Guide - Accessed: 18/01/2020
27. OWASP: Testing emulator detection (mstg-resilience-5) - android anti-reversing defenses (2020), https://mobile-security.gitbook.io/mobile-security-testing-guide/android-testing-guide/0x05j-testing-resiliency-against-reverse-engineering, oWASP Mobile Security Guide - Accessed: 18/01/2020
28. Petsas, T., Voyatzis, G., Athanasopoulos, E., Polychronakis, M., Ioannidis, S.: Rage against the virtual machine: hindering dynamic analysis of android malware. In: Proceedings of the seventh european workshop on system security. pp. 1–6 (2014)
29. Sai, A.R., Buckley, J., Le Gear, A.: Privacy and security analysis of cryptocurrency mobile applications. In: 2019 Fifth Conference on Mobile and Secure Services (MobiSecServ). pp. 1–6. IEEE (2019)
30. Security, R.: Apkid and android compiler fingerprinting (2016), https://rednaga.io/2016/07/30/apkid_and_android_compiler_fingerprinting/
31. Security, R.: Detecting pirated and malicious android apps with apkid (2016), https://rednaga.io/2016/07/31/detecting_pirated_and_malicious_android_apps_with_apkid/
32. Seneviratne, S., Kolamunna, H., Seneviratne, A.: A measurement study of tracking in paid mobile applications. In: WiSec (2015)
33. Sentana., I.W.B., Ikram., M., Kaafar., M., Berkovsky., S.: Empirical security and privacy analysis of mobile symptom checking apps on google play. In: Proceedings of the 18th International Conference on Security and Cryptography - SECRYPT, (2021)
34. SIMFORM: How to avoid reverse engineering of your android app? (2015), https://www.simform.com/how-to-avoid-reverse-engineering-of-your-android-app/
35. Source, G.: Android clang/llvm prebuilts (2020), https://android.googlesource.com/platform/prebuilts/clang/host/linux-x86/+/master/README.md, last accessed: 18/08/2021
36. Tangari, G., Ikram, M., Ijaz, K., Kaafar, M.A., Berkovsky, S.: Mobile health and privacy: cross sectional study. BMJ **373** (2021). https://doi.org/10.1136/bmj.n1248, https://www.bmj.com/content/373/bmj.n1248

37. Uddin, M.S., Mannan, M., Youssef, A.: Horus: A security assessment framework for android crypto wallets. In: Garcia-Alfaro, J., Li, S., Poovendran, R., Debar, H., Yung, M. (eds.) Security and Privacy in Communication Networks. pp. 120–139. Springer International Publishing, Cham (2021)
38. Vidas, T., Christin, N.: Evading android runtime analysis via sandbox detection. In: Proceedings of the 9th ACM symposium on Information, computer and communications security. pp. 447–458 (2014)
39. VirusTotal: Multitude anti-virus engines (2021), https://www.virustotal.com/gui/home/upload
40. Wang, J., Xiao, Y., Wang, X., Nan, Y., Xing, L., Liao, X., Dong, J., Serrano, N., Lu, H., Wang, X., Zhang, Y.: Understanding malicious cross-library data harvesting on android. In: USENIX Security Symposium) (2021)
41. Xia, P., Wang, H., Zhang, B., Ji, R., Gao, B., Wu, L., Luo, X., Xu, G.: Characterizing cryptocurrency exchange scams. Computers & Security **98**, 101993 (2020)
42. Zimmeck, S., Story, P., Smullen, D., Ravichander, A., Wang, Z., Reidenberg, J., Russell, N.C., Sadeh, N.: Maps: Scaling privacy compliance analysis to a million apps. Proceedings on Privacy Enhancing Technologies **2019**(3), 66–86 (2019)

A List of Anti-analysis Methods in Wallet Apps

We further explain the anti-analysis techniques used by the analyzed wallet apps.

- **Manipulator.** It is a tool or mechanism to modify `.dex` file and to repackage the modified `.dex` file into new apps [30,31,13].
- **Anti Virtual Machine.** A technique [38,28,23] to detect and evade analysis via sandboxes Android emulators or virtual machine.
- **Anti Debug.** It is a mechanism to prevent program analysis or debugging activities [26].
- **Obfuscator.** Obfuscation is a process of concealing the original source code, binary code, or byte-code into an obscure set of characters by encrypting or changing the original code without omitting the functionality. The goal is to impede the reverse engineering of the code by unauthorized parties. An *Obfuscator* renames libraries, variables, methods and class names [8,14].
- **Anti Disassembly.** It is a technique aiming to prevent the extraction of symbolic representations of the assembly code instructions from the `APK` of an Android app [17,21].
- **Packer.** It is a techniques aiming at evading reverse engineering by encrypting the `.dex` file [5,12].

B Keywords for Personal Data Transmission Analysis

Device information keywords: *deviceDescription, unidentified_device, deviceID, deviceToken, deviceRegKey, rooted_device, device_model, device_audio, deviceData, device_token, deviceType, device_brand, deviceId, devicetoken, kochava_device_id, device_name, deviceFreeSpace, deviceOEM, deviceOS, device, device_uuid, device_os, deviceHeight, device_api, deviceInfo, device_type, device_time, deviceWidth, actual_device_type, deviceRm, deviceKey, deviceModel, device_fingerprint_id, device_id, deviceMAC, deviceId, deviceVersion, deviceFingerPrintId, device_data*

Credential information keywords: *second_password_confirmation, password, password_score, password_confirmation, second_password, password*
Location information: *localeIdentifier, location, locale_language_code, locale_country_code, locale, h_region*
Collected wallet information: *devkey, walletId, previous_deployment_key, walletAddress, key_index, subwallets, coin, master_public_key, deployment_key, wallet_type, deployment_key, gApikey, wallet_version, bitcoin_notification_enabled, key, sort_key, previous_deployment_key, temporary_private_key, primary_public_key, branch_key, public_key, walletAddressIndex, wallet_transaction_notification_enabled, wallets, coinId, backup_public_key, wallet_id, walletAddressNm, api_key, walletID, pub_key*

Table 9: Top 20 customized and third-party permissions requested by the analyzed apps.

No	Permission Name	# of Request (%)
1	com.google.android.c2dm.permission.RECEIVE	307 (67.2%)
2	com.google.android.c2dm.permission.SEND	297 (65.0%)
3	.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE	290 (63.5%)
4	.gms.auth.api.signin.permission.REVOCATION_NOTIFICATION	113 (24.7%)
5	com.huawei.android.launcher.permission.CHANGE_BADGE	102 (22.3%)
6	com.anddoes.launcher.permission.UPDATE_COUNT	101 (22.1%)
7	com.sonyericsson.home.permission.BROADCAST_BADGE	101 (22.1%)
8	com.sec.android.provider.badge.permission.WRITE	101 (22.1%)
9	com.sec.android.provider.badge.permission.READ	101 (22.1%)
10	com.majeur.launcher.permission.UPDATE_BADGE	101 (22.1%)
11	com.htc.launcher.permission.READ_SETTINGS	101 (22.1%)
12	com.htc.launcher.permission.UPDATE_SHORTCUT	101 (22.1%)
13	com.huawei.android.launcher.permission.READ_SETTINGS	96 (21.0%)
14	com.huawei.android.launcher.permission.WRITE_SETTINGS	96 (21.0%)
15	com.sonymobile.home.permission.PROVIDER_INSERT_BADGE	96 (21.0%)
16	android.permission.READ_APP_BADGE	92 (20.1%)
17	com.oppo.launcher.permission.READ_SETTINGS	91 (19.9%)
18	com.oppo.launcher.permission.WRITE_SETTINGS	91 (19.9%)
19	me.everything.badger.permission.BADGE_COUNT_READ	90 (19.7%)
20	me.everything.badger.permission.BADGE_COUNT_WRITE	90 (19.7%)

Table 10: Cause of Privacy Policy Extraction Failured.

No.	Caused of Failured	# of Apps (%)
1	404: Not Found	31 (6.8%)
2	403: Forbidden	19 (4.2%)
3	No Privacy policy link	17 (3.7%)
4	3: Temporary failure in name resolution	14 (3.1%)
5	2: Name or service not known	10 (2.2%)
6	No Metadata Found	9 (2.0%)
7	503: Service Temporarily Unavailable	7 (1.5%)
8	110: Connection timed out	3 (0.7%)
9	SSL: CERTIFICATE_VERIFY_FAILED	3 (0.7%)
10	522: Cloudflare times out	3 (0.7%)
11	SSL_ERROR_BAD_CERT_DOMAIN	2 (0.4%)
12	502: Bad Gateway	1 (0.2%)
13	104: Connection reset by peer	1 (0.2%)
14	526: Origin SSL Certificate Error	1 (0.2%)
15	403: Ip Forbidden	1 (0.2%)
16	500: Internal Server Error	1 (0.2%)
17	400: Bad Request	1 (0.2%)
18	308: Permanent Redirect	1 (0.2%)
19	307: server returned infinite loop	1 (0.2%)
20	111: Connection refused	1 (0.2%)
21	0: Error	1 (0.2%)
22	5: No address associated with hostname	1 (0.2%)
23	UNRECOGNIZED_NAME_ALERT	1 (0.2%)
	Total	**130(28.4%)**

Table 11: List of cryptocurrency wallet apps that are considered as malicious by users in Google Play reviews and by VirusTotal (AV-positive column). For each cryptocurrency wallet app, the NR-Ratio represents the ratio of the number of negative users' comments to the total number of all users' comments.

#	App ID	NR-Ratio	Installs	Rating	AV-positives
1	com.top1.group.international.android	1.0%	10000+	4.7	9
2	com.jex.trade	43.1%	500000+	4.1	8
3	com.legendwd.hyperpayW	10.4%	50000+	4.8	7
4	im.token.app	25.2%	500000+	4.1	4
5	com.vidulumwallet.app	13.8%	1000+	4.2	2
6	com.remint.app	18.8%	10000+	4.4	2
7	com.portto.blocto	23.8%	10000+	4.3	2
8	roseon.finance	4.6%	10000+	4.8	2
9	com.fox.one	2.9%	1000+	4.8	1
10	com.studentcoin	63.5%	50000+	4.4	1
11	one.mixin.messenger	15.8%	10000+	4.5	1
12	net.ethylyte.com	2.9%	5000+	4.5	1
13	augstrain.asn	33.3%	10+	0.0	1
14	co.edgesecure.app	27.1%	100000+	4.0	1
15	com.viabtc.pool	14.8%	100000+	4.6	1
16	com.bitcoinglobal	50.0%	5000+	0.0	1
17	com.crypto.multiwallet	18.1%	100000+	4.4	1
18	com.friendst.strangr	80.0%	10000+	3.8	1
19	com.quidax.app	32.4%	100000+	3.7	1
20	com.digifinex.app	21.0%	100000+	3.9	1
21	com.lingxi.bexplus	1.8%	50000+	4.7	1
22	app.goodcrypto	15.5%	100000+	4.5	1
23	com.enjin.mobile.wallet	9.5%	500000+	4.6	1
24	com.holacoins.wallet	20.4%	50000+	4.2	1
25	africa.bundle.mobile.app	9.9%	100000+	4.5	1