# The Devil is in the Details: Unwrapping the Cryptojacking Malware Ecosystem on Android

Boladji Vinny Adjibi*, Fatou Ndiaye Mbodji*, Tegawendé F. Bissyandé*, Kevin Allix*, Jacques Klein*
*SnT – University of Luxembourg
Luxembourg, Luxembourg
Email: {vinny.adjibi, fatou.mbodji, tegawende.bissyande, kevin.allix, jacques.klein}@uni.lu

***Abstract*—This paper investigates the various technical and non-technical tools and techniques that software developers use to build and disseminate crypto mining apps on Android devices. Our study of 346 potential Android mining apps, collected between April 2019 and May 2022, has revealed the presence of more than ten mining apps on the Google Play Store, with at least half of those still available at the time of writing this (June 2022). We observed that many of those mining apps do not conceal their usage of the device's resource for mining which is considered a violation of the store's policies for developers. We estimate that more than ten thousand users have run mining apps downloaded directly from the Google Play Store, which puts the supposedly "stringent" vetting process into question. Furthermore, we prove that covert mining apps tend to be embedded into supposedly free versions of premium apps or pose as utility apps that provide valuable features to users. Finally, we empirically demonstrate that cryptojacking apps' resource consumption and malicious behavior could be insignificant. We presume that typical users, even though they might be running a mobile antivirus solution, could execute a mining app for an extended period without being alerted. We expect our results to inform the various actors involved in the security of Android devices against the lingering threat of cryptojacking and help them better assess the problem.**

***Index Terms*—android, cryptojacking, malware, manual analysis, google play store**

## I. Introduction

The increased demand for cryptocurrencies following 2017's boom drove a burgeoning interest in mining [1]. The induced competition strengthened the need for miners to join mining pools to expect to make a profit. The introduction of services such as Coinhive, which made it possible to spread mining code across the web with minimal effort, further increased the attractiveness of the activity. It soon led to malicious entities covertly participating in mining pools using abused devices. This activity, known as cryptojacking, affects every type of device, from personal computers to smaller devices such as Android-powered devices. Despite their relatively low computing power, Android devices are an attractive target to hackers who can target more than four billion devices at once [2], [3].

Previous research has, for instance, identified that developers produced more than seven hundred Android mining apps between 2017 and 2019 [4]. The findings from this investigation were confirmed by another study highlighting the active use of the Google Play Store to distribute those apps [5]. The results from those studies attest that understanding the cryptojacking phenomenon on Android devices is paramount. Unfortunately, there exist no other studies of mining apps on Android besides those mentioned above. Moreover, the datasets used in those studies do not contain any mining app beyond 2019. Considering that two majors events happened around that time, namely (1) the Google Play Store's policy to ban apps that mine using the device's resources starting from July 2018 [6], [7] and (2) the discontinuation of Coinhive, which alone contributed to more than half of the mining apps in the studied period [8], the lack of research on the subject appears to us as a counter-intuitive outcome. In comparison, many recent studies of web-based [9], [10], [11], and general-purpose miners [12], [13], [14] have been conducted. The prevalence of the cryptojacking problem on Android devices and its expected damaging effects make it essential to understand the various mechanisms currently used to mine cryptocurrencies on Android devices [15]. More specifically, we are interested in answering the following research questions:

- **RQ1:** What are the methods and techniques used by developers to insert crypto mining code into Android apps, and how effective are existing detection techniques against those threats?
- **RQ2:** What processes are put in place by developers to ensure that their mining apps are downloaded and executed by as many users as they want?
- **RQ3**: To what extent does mining activity affect the behavior of an Android device, and how likely is an average user to suspect such action on their device?

*Novelty:* We investigate those questions with the intent of inspiring the research community into building a more effective defense mechanism against mining apps. Our attempt at addressing those questions consisted of an empirical study of 10 430 681 Android apps collected from various marketplaces between April 2019 and May 2022. To the best of our knowledge, this is the first study that focuses on Android mining apps in this timeframe. It allowed us to unveil an evasion technique based on assigning an audio filename to a gzipped archive containing a mining binary, which the program extracts at runtime. Those results, along with the decreased proportion of Javascript-based miners and the appearance of new crypto coins (Veruscoin, RPCoin, uPlexa), supplement prior findings [4] to provide a complete mapping of the mining

phenomenon on Android. To the best of our knowledge, this is the first study that aims to explain the marketing structure that sustains the mining ecosystem on Android. Our investigation resembles the study by Kotzias et al. [16] which discussed the topic of the sources from which users typically download malware. Unlike this study, our focus is on understanding what could get users off their guard to let apps stealthily mine coins on their devices. Earlier research works suggest the prevalence of gaming, streaming, application download, and adult content platforms with illicit mining [17], [18] on the web, but there is little scholarly work documenting illegal mining on the Android platform. We intend to fill this gap through the present study. Finally, our evaluation of the impact of mining activity, whether covert or overt, provides an up-to-date complement to the work conducted by Clay et al. [15] which estimated a five-fold increase in power consumption due to the execution of web-based miners. Besides the novelty factor, our study extends to binary-based miners and further considers the potential concealment of the mining activity. Our evaluation captures more diversity in the mining apps on Android.

*Contributions:* Overall, through our study of the suspected mining apps from a technical and non-technical standpoint, we make the following contributions:

1) We identified and duly notified the Google Play Store of the presence of at least five mining apps on their marketplace, which is a blatant violation of the applicable developer policies.
2) We discuss the case of some so-called wallet apps that hide malicious code into gzip archives renamed to have misleading extensions. We postulate that this is a technique likely to be used by evil entities.
3) We demonstrate the prevalence of hacks and cracks of premium Android apps as placeholders for mining code in cryptojacking apps.
4) We establish through various tests that, despite mining apps consuming more resources than regular apps, a social media app such as Facebook is twice as greedy as an evasive miner.

The remaining of this paper is structured as follows. In Section II, we present our methodology used in the study. Section III then discusses the findings from the manual analysis of the source code and static artifacts of the suspected mining apps. Next, our results from analyzing the non-technical aspects of the mining apps are described in Section IV. We then present our findings from profiling the resource consumption of mining apps in Section V before a brief discussion on the possible shortcomings of our study in Section VI.

## II. Study Design

### A. Data collection

The ban imposed by Google on mining apps on the Google Play Store in July 2018, along with the suspension of Coinhive late in March 2019 are both critical events that previous research has noted [4]. This study that leveraged apps collected from various companies' security advisories is arguably the only one to be aiming at understanding the crypto mining phenomenon on Android devices; a data collection process used in recent Android malware research [19]. Besides this data collection strategy, researchers have also relied on Android marketplaces such as Google Play Store [20], [21], [22] or a combination of both [4]. The dissemination of the compiled apps has contributed to the availability of a wide range of Android malware data sets usable for numerous studies. Unfortunately, those datasets are limited due to their age or specificity to a research problem. The lack of diversity in the datasets renders them inappropriate for our work which we wanted to represent the current state of the ecosystem of crypto mining on Android. To this effect, we turned our focus on AndroZoo, arguably the most extensive dataset of Android apps for research [23]. The versatility and up-to-date nature of the dataset, which contains more than 19 million (in May 2022) Android apps collected from various sources, make it a good source of information to feed our research effort. Moreover, the apps available in the dataset are already scanned on VT (Virus Total), providing researchers with an initial set of information to spearhead their works.

Consequently, we rely our study on information collected from AndroZoo. More specifically, we combined the list of hashes with VT API's results to identify all the apps that 1) had not been submitted to VT before April 2019 and 2) have been assigned a label related to mining by at least two antivirus solutions. Our approach prevented an intersection of our dataset with that of Dashevskyi et al. [4] and enabled us to discard most of the non-relevant apps. The resultant initial dataset that we built with this strategy contains 346 SHA-256 corresponding to one or more versions of 92 different apps. In this work, an app refers to a unique pair of a package name and the marketplace of origin. For each app, we only consider the latest version in our data set for the study.

### B. Manual analysis of apps

Through our manual, systematic and rigorous manual analysis of the apps in our dataset, we want to answer the following questions to establish their connection with the mining activity.

1) **Does the app mine cryptocurrencies?** To answer the question, we look for pieces of evidence that validate that the app embeds and uses some mining procedures. Based on the findings reported in previous research, we focus our search on the `app/src/main/assets` and `app/src/main/res/raw` folders where the developers usually put their mining artifacts [4]. Upon localizing the said file, we analyze the source code to confirm that the program exploits the identified mining procedure. This step allows us to identify some false positives, some miners, and a few apps for which we could not confirm beyond the point of doubt that the app is mining.
2) **How is the mining implemented into the app?** At this stage, we try to explain the specific technique the developers leveraged to insert the mining logic into the app. Besides distinguishing between web and binary-based

miners, we look at the programming languages and other relevant elements that made the implementation possible.

3) **Is the device's user aware of the mining activity?** At this stage, we look at various information, such as the listing of the marketplace app when available, the source code, and layouts, to verify whether the user is aware that mining activity is occurring on their device.
4) **Who benefits from the mining activity?** This is to check who is the final beneficiary of the mining. The beneficiary is either the user or the developer. It could also be both when a donation wallet is specified. This question is answered by looking for wallet addresses, configuration files, and similar elements that allow making an informed decision.
5) **What triggers the mining activity?** This question is to explain under which conditions the mining activity is started. The trigger could vary from a system event to an action from the user.
6) **How is the mining activity concealed?** This step is to identify strategies that developers use to evade static and dynamic detection of the mining activity.

Considering the high reliance of our approach on having access to the apps' source code, we downloaded the APKs using the AndroZoo API and further decompiled them using the JADX tool[1], one of the many available tool to extract the Java source code and the static resources bundled into apps. We also leveraged the detailed VT reports to clear up doubts when faced with uncertainty.

### C. Understanding how developers distribute the mining apps

Our objective in this regard is to understand the various processes that the developers put in place to attract users and get them to install the mining apps. More specifically, we are interested in answering the following questions:

1) **In what type of apps do developer often insert their mining code?** Previous research has highlighted the use of piggybacked and repackaged apps to distribute Android malware in the wild [24], [25]. Based on those insights, we try to evaluate the prevalence of a particular type of app in our corpus of mining apps. We leveraged the Androguard tool[2] to retrieve each app's package name and label. We searched for those names on Google to find similar words to identify links with apps from other marketplaces.
2) **Are ratings and comments of the apps manipulated to increase adoption by the users?** This question stems from the findings of Harris et al. suggesting a strong link between an app's popularity and its attractiveness to users [26]. To assess that, we analyzed the ratings and comments of the mining apps in our dataset that present specific characteristics and suggest some manipulation. We leveraged the Google Play Scraper [3] tool's review API to collect the said information for the apps still available in the store while doing our work. We used the default settings of the package (i.e., country set to the USA and the language to English)

We present the results of those investigations in Section IV.

### D. Evaluation of the impact on users

Thus far, the literature only assumed that mining applications are greedy in terms of resources. Even though there have been some estimates that clearly showed that those apps consume an enormous amount of energy[4], those reports are mainly related to the period when Coinhive still existed and was used with the intent of abusing resources. Still, the research community lacks a systematic measure of those values in the context of Android apps. In this work, we benchmark the resource consumption of 3 mining apps against some commonly used legitimate apps. The results presented in Section V suggest that mining applications may not be as greedy as is often assumed.

In the subsequent sections, we describe the findings of our investigations for answering the outlined research questions.

## III. RQ1: Understanding the implementation of the mining logic into Android apps

Our manual analysis of the potential mining apps consisted in looking into the external dependencies or Java packages used in the project, the `assets` directory that often contains HTML resources used by web-based miners on Android, the `res/raw` folder that is often used for binary files, and the `lib` directory where shared libraries are put to be loaded with the `System.loadLibrary()` method. Our analysis of those artifacts and how they are used allowed us to identify many apps that could be wrongly believed to perform mining and some other mining apps about which we provided more explanation on how they perform.

### A. False positives and possibly unconfirmed miners

From our investigations, there were a number of apps for which our manual analysis did not allow us to conclude that they were mining. We report on those apps in the following sections.

*1) The apps wrongly identified as miners:* Upon analyzing the suspected mining apps from our data set, we identified that most were mining apps. Those false positives amount to 76% (70/92) of the entire data set. In this entire corpus, three main types of apps can be observed:

*a) **The com.kaching.kingforaday based packages**:* Those represent half of the entire false positives in our data set (35/70) and were all downloaded from VirusShare into AndroZoo. The sparkling similarity in those apps starts with their naming convention which is done by appending the string .hack to the name of some apps with paid subscriptions (e.g., com.imangi.templerun.hack). Furthermore, the analysis of the source code shows the presence of a package named `com.kaching.kingofaday` which declares a service as seen in Listing 1 which is in charge of starting the mining activity. For this, it depends on a package named

[1]https://github.com/skylot/jadx
[2]https://github.com/androguard/androguard
[3]https://pypi.org/project/google-play-scraper/
[4]https://www.kaspersky.com/blog/loapi-trojan/20510/

`com.coinhiveminer.CoinHive` which uses an HTML file that instantiates the Coinhive miner.

```
1  public class StartAdsReceiver extends
       BroadcastReceiver {
2      @Override // android.content.BroadcastReceiver
3      public void onReceive(Context context, Intent
           intent) {
4          context.startService(new KingForADay(context
               ).MinerIntent());
5      }
6  }
```

Listing 1: Mining receiver code embedded into the com.kaching.kingforaday package.

However, though this logic is well implemented, it appears to not be instantiated anywhere in the code. We further ran a full string search in the folder to verify whether it was called anywhere but we could not get evidence for this at all. Moreover, the fact that the service was not declared in the Manifest means that it could not be called anyway. Thus, we hypothesize that the app had been an active miner during the rise of Coinhive but then, since the service was later shutdown, the developers just removed the initialization code from it. Our attempt to validate that hypothesis by finding earlier versions of any of the concerned app was not successful unfortunately.

*b)* ***The apps with the unused mining files****:* Very similar to the previous apps, the apps in this category embed either directly or through a third-party service, the mining code as a Javascript dependency or through the use of a binary. Among the 10 such apps, half of them have the mining code in them because of the inclusion of the Hextrix game[5] which is a game that clearly states its intention of mining. Despite that, the mining code was commented out in each of the relevant files. This is similar to the other half of the apps which used various mining services including Coinhive and CryptoLoot. In Listing 2, we show how the mining code was commented out in those apps. Here again, a full string search was performed in the directory to find and analyze all the code loading the webview, searching for instances where the comments might have been removed prior to loading the files but no activity of such kind was noticed.

```
1  <!-- <script src="<LINK_TO_SCRIPT>"></script> -->
2  <!-- <script> -->
3  <!-- var miner = new CoinHive.Anonymous(<SITE_KEY>,
       { -->
4          <!--         throttle:0.7 -->
5          <!--     }); -->
6          <!--     miner.start(); -->
7  <!-- </script> -->
```

Listing 2: Initialization code of Coinhive mining that is commented out in application

*c)* ***The wallets****:* Our data set is comprised of 17 crypto wallets that were considered as miners by at least two antivirus solutions. Our analysis of the binaries embedded in those apps shows that they indeed contain the mining logic, as evidenced by the parameters to set the type of algorithm, donation address, etc. Moreover, some of those apps download entire copies of the blockchain in order to track the transactions related to the users' wallet. One notable case among the wallets however is that of the app `io.scalaproject.vault` still available on the Play Store, which presents a mining interface through which the user is redirected to download the actual mining app from a Github repository[6]. By analyzing the source code and various parameters used to call the suspicious binary, we confirmed that the apps are not exploiting the mining capabilities of the software.

*d)* ***The outliers****:* 8 apps fall into this category which from our investigation, have no clear link to mining activity other than the appearance of strings such as *miner* or *mining* in the assets. One special case in this category has a list of links that it uses to serve as a firewall, preventing the user from accessing those links. And among those links, some look like mining pools address (e.g. 0.0.0.0rpd.cryptopool.eu) in a subfolder of the corresponding assets folder. Some of the other apps have no suspicious binary or JavaScript file that leads to thinking about a mining activity.

The high proportion of false positives and that count up to 25 detection by antivirus solutions can be explained by the fact that most of those tools are signature based and do not actually check whether the code is being run or not. This in itself is not a good practice but we suggest a better attempt at finding whether the code is being run or not.

*2) The ambiguous cases:* Besides the apps that we were able to confirm were not doing any mining activity, for 7 of the remaining apps, we could not ascertain the presence of the mining code. The information collected about those apps is summarized below:

1) 2 apps presented as movie downloader and viewer based on the use of torrents. In one of those apps, there is the presence of the xmrig binary that can be used for mining. Surprisingly, the binary is not called anywhere in the code as confirmed by our search in all the relevant files. However, we noticed that those two apps at a point during the bootstrap try to download and install an app from the link[7]. We presume that the actual miner is installed from that link and further exploits the binaries already embedded in the current package. Unfortunately, at the time of running our analysis, the link was no longer available making it impossible for us to validate our hypothesis. A third app with a different purpose redirects the user to dynamically queried links which we could not verify because the link were no longer accessible.
2) 3 other apps, all retrieved from the *anzhi* marketplace include the `libjiagu` binary which serves as a packer meant to dissimulate source code and prevent their investigation by manual analysis. Our attempt at unpacking the binaries were unsuccessful so we could not justify whether those apps are mining or not.
3) A further duo of apps from the same developer which are presented as wallets and expose a great similarity in the

[5]https://hextris.github.io

[6]https://github.com/scala-network/MobileMiner/releases/

[7]http://b3.ge.tt/gett/9JtX8Kp2/com.opera.mini.native.pdf?index=0&pdf

source code. The most interesting bit of those apps is the fact that they hide a compressed archive under a file name `private.mp3` and placed under the `assets` folder. The file which is uncompressed at runtime contains Python files that allow to run a node of the Electrum coin. Adding to this deceptive and misleading approach to installing the binaries on the device, we identified from the source code the use of a `config.json` file that determines the mode in which the node is ran. Unfortunately, we could not find any information about the said file which hampered our ability to identify whether the mining activity was occurring or not.

As a summary, our investigation have pointed out a number of apps (74/87) for which we could not find any evidence of the execution of a mining activity. Thus, for the remaining of this paper, only the remaining 13 unique apps amounting to 27 different APKs will be put in focus for us to try to understand how the miners work and explain the processes that are engaged into their creation and mass distribution to users.

### *B. Inspection of the mining apps*

As a logical conclusion to our investigation, the 13 remaining apps are considered and verified as mining one or multiple crypto currencies. Among those apps, there is a preponderance of apps originating from Google Play Store with such apps amounting to 11 out of the 13 apps, the remainder coming from VirusShare. This impressive ratio of mining apps downloaded from Google Play Store in our timeframe betrays our initial intuition to have more miners from alternative markets. Our surprise is further reinforced by the fact that 6 of such apps were still present on the store at the time of writing. We postulate that some developers manage to bypass the controls made to prevent those apps from being flagged during the verification process. Understanding the process through which they successfully get those apps on the store are out of scope of this work. Since the store's regulations ban mining apps, all the 13 apps in our data are considered *illicit*. Nonetheless, we will distinguish between the apps according to whether the mining activity is known to the user or not.

Above any consideration to the awareness of the user on the mining activity, a general view to the mining apps shows a consistency with the results from previous works that identified the web and binary-based techniques as the means of inserting mining code in Android apps [4]. In our case only a small proportion of the apps are web-based (4/13). This rapid change when compared to the period before April 2019 where web-based miners amounted to almost 75% of all apps can be explained by the shutdown of the Coinhive service which was the main driver of the mobile mining ecosystem on Android. As far as the alternatives to Coinhive are concerned, we have identified the use of CryptoLoot and other proprietary scripts[8].

Another point of variation in the implementations of the mining apps lie in the use of various programming languages

[8]https://www.craftyourserv.net/mineur

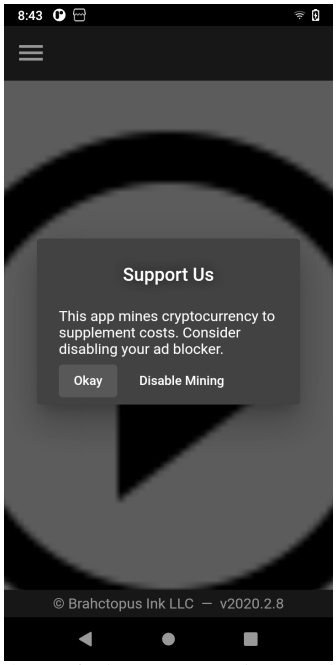



Fig. 1: Screenshot of the *com.sunderapps.brahtopus* app's interface through which the user grants permission to mine cryptocurrency.

to support the code. As such, we have identified apps using JavaScript-based hybrid mobile developments toolkit such as Ionic, but also those developed using Kotlin and obviously the Java language which powers most of those. This evaluation was performed by looking at the source code of the assets for instance (presence of the ¡ion-app¿ tags in Ionic) and the included packages which can inform on the tool used to build the app. This diversity in techniques and procedures suggests that the development of mining apps for Android devices is possible for virtually every programming language supported on Android.

Our analysis of the scan reports collected from VT has shown that the number of tools that have detected a certain miner varies from 2 to 25 suggesting that the threshold of ten used in related works is not appropriate [27]. Our findings suggest that for categorical malware such as mining apps, a threshold of 2 seems sufficient.

In order to discriminate the between legitimate and illicit miners, we used information such as the package name, description of the app where available (collected from the store), the layouts in the decompiled file and also the source code. This has allowed to identify 6 legitimate miners among which one specifically sets all the rewards to the developers while the others get only a portion of the gains, with the other part being left to the user's address. The former's screen through which the user's approval is granted is visible in Fig. 1 and declines its intention to monetize their website visits by leveraging the users' CPU for mining crypto currencies.

The next section of this paper describes the underlying functioning and evasion techniques pertaining to the illicit miners in our data set.

TABLE I: Top 10 permissions requested by the cryptojacking apps.

| Permission | # Apps (%) |
|---|---|
| android.permission.INTERNET | 7 (100.0%) |
| android.permission.ACCESS_NETWORK_STATE | 5 (71.43%) |
| android.permission.WRITE_EXTERNAL_STORAGE | 5 (71.43%) |
| android.permission.READ_EXTERNAL_STORAGE | 4 (57.14%) |
| android.permission.READ_PHONE_STATE | 4 (57.14%) |
| android.permission.VIBRATE | 4 (57.14%) |
| android.permission.WAKE_LOCK | 4 (57.14%) |
| android.permission.ACCESS_COARSE_LOCATION | 3 (42.86%) |
| android.permission.ACCESS_FINE_LOCATION | 3 (42.86%) |
| android.permission.ACCESS_WIFI_STATE | 3 (42.86%) |

### *C. Functioning and evasion techniques of the crypto jacking apps*

The balance in the type of implementation used in the list of crypto jacking apps (3 web vs 4 binary based), the mined coins vary in nature with Monero, uPlexa and RP-Coin all being represented in our data set. This describes a diversity in the available tools to embed illicit mining activity into apps. We further analyzed the permissions requested by the mining apps which, as presented in Table I, are quite similar to those identified by Dashevskyi et al [4]. When compared with legitimate apps, three permissions appear to be peculiar to the mining apps namely those related to the device's location (ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION), and the one related to the phone state (READ_PHONE_STATE) [28]. Those three permissions have been identified as being used in up to 68% Android apps according to a study of dangerous Android permissions [29]. As far as the system events are concerned, the mining apps in our dataset subscribe to a few events that do not appear in the list of events identified in Cao et al.'s study of Android malware [19]. Those events are related to the charging status of the device and to the presence or not of the user by the telephone. They are used in mining apps to find the best moment to mine without raising any suspicions from the user.

In the later sections of the paper, we will leverage those information as a means to talk about the triggers for the mining activity, the various sophistication put in place and also the evasion techniques that we identified in the various apps. The apps can therefore be declined into the following categories:

*1) The watchers:* Those apps amounting to 3 exploit the possibility of running JavaScript on the web view interfaces of the apps. In a general sense, the script is executed immediately, as soon as the user navigates into the page which is loaded either through a web view or using a language such as Ionic. Then, the mining keeps running as long as the activity is alive. We were able to confirm the malicious intent of the apps through the presence of developer (or site keys) that are used internally by the mining service providers to distribute the gains back to the developers. In order to avoid detection, we have observed two main techniques: 1) the obfuscation of the JavaScript file which is also loaded from random links, and 2) the inclusion of the mining initialization code into larger files combined with the use of variables to dissimulate the mining activity. An example of initialization code can be seen in Listing 3 two different site keys are used in the same app based on the domain name from which the user is visiting.

```
function D(){
    var e="<SITE KEY 1>"
    if(-1!=location.href.indexOf("<LINK>")||-1!=
        location.href.indexOf("<LINK>"))
        return void(e="<SITE KEY 2>");
    var n=new CoinHive.Anonymous(e);
    n.setThrottle(.5),n.start()
}
```

Listing 3: Initialization code of a web based miner

The use of those techniques ensures that a static detection tool which is for instance based on the length of the site key would be fooled and not able to detect the app as doing mining.

*2) The zombie miners:* Those ones are a group of 2 apps seemingly from the same developer that use the Monero miner file in order to launch the mining activity. After loading the system libraries, the apps connect to a mining pool and instantiate a service that keeps restarting each time there is a failure. This allows the mining process to continue running even if the user is not interacting with the app thus making up for a zombie like functioning. In order to not letting the user being aware of such activity, the apps listen to events related to the battery and stop the mining activity as soon as a threshold is reached. The apps download the configuration files for the mining from a link[9] which was no longer available during our analysis. We were also unable to run the app which kept crashing on the test device.

*3) The fetchers:* We denote by this, a set of apps that only download the mining binary at run time, or receive the task through another file. For the two apps in this category, the mining activity is started automatically when the device is powered on. Besides this first layer of obfuscation, we have observed in one of those apps that the SharedPreferences were used to store the commands and parameters to be executed to run the mining. This is then later used to launch the mining. And since the two apps in those categories offer services that are not supposed to consume too much memory, they make sure that the mining activity is running only when the user is away by subscribing to the *USER_PRESENT* system's event. This complex scheme has further been complicated by one of the apps that inserted the mining code under a Google's package name (com.google.ads) in an attempt to fool detection tools.

## IV. RQ2: How are users attracted to downloading and installing those applications?

In the Android ecosystem of mining apps, the development of the apps is just one part of the equation. Upon producing the software artifacts, the developer needs to find the appropriate ways in which they can distribute those apps to as many users as possible. In the current section, we look at two factors that we presume to positively impact the adoption of mining apps by Android users. We refer specifically to (1) the kind of app used and (2) its popularity. We identify the kind of app by

[9]http://flowcount.eidon.top:10085

TABLE II: Top 10 system events subscribed to by the cryptojacking apps.

| Event | # Apps (%) |
|---|---|
| BOOT_COMPLETED | 3 (42.86%) |
| MEDIA_MOUNTED | 3 (42.86%) |
| USER_PRESENT | 3 (42.86%) |
| net.conn.CONNECTIVITY_CHANGE | 3 (42.86%) |
| WEB_SEARCH | 2 (28.57%) |
| ACTION_POWER_CONNECTED | 2 (28.57%) |
| ACTION_POWER_DISCONNECTED | 2 (28.57%) |
| com.android.vending.INSTALL_REFERRER | 2 (28.57%) |
| TIME_SET | 1 (14.29%) |
| service.notification.NotificationListenerService | 1 (14.29%) |

TABLE III: Categories of the mining apps and their source.

| Package name | Category | Source |
|---|---|---|
| com.pangzlab.verus_box | Blockchain related apps | play.google.com |
| de.lubddetis.monerominer | | play.google.com |
| dev.waterhole | | play.google.com |
| free.bitcoin.mining.crypto | | play.google.com |
| io.waterhole | | play.google.com |
| info.bitcoinunlimited.voting | | play.google.com |
| com.karameesh.app | Game and entertainment apps | play.google.com |
| com.sunderapps.brahctopus | | play.google.com |
| com.games.tecdroid.freddynightmario | | play.google.com |
| com.sunderapps.recaptureorganics | | play.google.com |
| net.craftyourserv.www | | play.google.com |
| mikado.bizcalpro | The cracks | VirusShare |
| com.dust.clear.ola | Utility apps | VirusShare |

grouping the apps based on the features that they claim to propose. As far as popularity is concerned, we investigate the manipulation of reviews to increase an app's attractiveness.

### A. Type of apps containing mining logic

Our analysis allowed us to group the mining apps in our dataset into the following categories that are summarized in Table III.

*1) Blockchain-related apps:* The most represented category in our dataset with six unique apps, this category contains three apps that state their intent of mining, two wallet apps, and one other app that leverages the blockchain to facilitate the organization of polls (info.bitcoinunlimited.voting). In their official listing on the marketplace, all those apps appear to be associated with the *Finance* category. This denotes the attractiveness of such apps to users who can fall prey to malicious developers who abuse their devices for mining as for two apps in this corpus (i.e., *dev.waterhole* and *io.waterhole*), published on Google Play Store by the same developer.

*2) Game and entertainment apps:* Five apps in our dataset fall under this category, including games, music, and advice content. Among all those apps, only two (*com.sunderapps.brahctopus*, *net.craftyourserv.www*) inform the user of the mining intent on the first usage of the app, as seen in Fig. 1 and Fig. 2a respectively. The relatively high proportion of those apps (5/13) denotes the interest of developers in maintaining users in the app for a long time to generate the most profit, as previously demonstrated by Tekiner et al. [30]. Furthermore, all the miners in this category use the web-based strategy that works better when the user is present.

*3) The cracks:* In an attempt to generate revenues from their production, developers often require users to pay before downloading or enabling the premium features of an app. Often unable to use the apps because of those limitations, users tend to resort to cracked apps help them bypass the verification mechanisms put in place by the developers. Despite posing as free software, the cracked apps have been proven to often contain malware [25]. Our sample of mining apps contains one cracked app carrying illicit mining codes. One of those apps posed as a free version to a paid calendar app available in the Google Play Store[10]. As soon as the user installs the app, it proceeds with its stealthy mining activity.

We can further extend our list of cracks with the false positives that we discussed in Section III-A1a. The naming convention of those apps suggests that they all pretend to help users bypass the subscription controls of dozens of games and entertainment apps. We can therefore presume that they were indeed cracks shared through alternative Android marketplaces with the intent of distributing illicit mining codes.

*4) Utility apps:* Only one app (i.e. *com.dust.clear.ola*) falls into this category. From the search results available in Google and translated from Chinese to English[11], the app promises the user to clean their device's memory but also remove water droplets. The user willingly grants it many abusive permissions used by the app to conceal its mining activity. We expect that many similar apps carry mining code or related malware.

*5) Interpretation:* Despite the presence of two dominant types of apps that mining apps can be associated with the high diversity in such a small sample shows that mining apps can be found in any category of app. The small size of our dataset does not allow us to determine any specific category that developers prefer to distribute mining code.

### B. Manipulation of comments and ratings to attract users

Our review of the ratings and comments associated with the apps still available on the Google Play Store during our study was based on the Google's policy for user reviews[12] that defines a helpful review as one that provides clear information to the reader, and present both the strong and weak points of their experience with the app. As such, we focus our analysis on the various reviews of the apps looking for elements that pinpoint to a manipulation.

In total, we collected 67 reviews in English and French for eight different apps presented in Table IV. We performed a manual analysis of the metadata with two authors, both fluent in English and French actively involved in the process. The results were presented to other researchers in our group who agreed with our conclusions.

From our analysis, we made the following observations.

*1) Useless and uninformative comments:* Around five apps in our dataset have at most three reviews from users. In those specific cases, we hardly noticed any informative comments, with some reviews being as simple as "I like this style and feel

[10]https://play.google.com/store/apps/details?id=mikado.bizcalpro

[11]https://app.mi.com/details?id=com.dust.clear.ola

[12]https://play.google.com/intl/en_us/about/comment-posting-policy/

TABLE IV: Summary of review information collected on Google Play Store about the mining apps.

| Package Name | Average rating | Min. Num. of Installs | # of reviews |
|---|---|---|---|
| net.craftyourserv.www | 3.7 | 1000 | 32 |
| com.pangzlab.verus_box | 4.3 | 5000 | 14 |
| io.waterhole | - | - | 13 |
| de.ludddetis.monerominer | - | - | 3 |
| dev.waterhole | 4.2 | 100 | 2 |
| info.bitcoinunlimited.voting | 5.0 | 10 | 1 |
| com.games.tecdroid.freddynightmario | - | - | 1 |
| com.karameesh.app | - | - | 1 |

safe". Moreover, the comments in this style are all submitted by users who gave five stars to the app. This goes against the fact that a user satisfied with an app that provides clear benefits would find many things to say to motivate others to download the app.

*2) Untrustworthy comments:* This specific case applies to the app *io.waterhole* for which a five-star rating was attributed after being available for less than five days. Even though we could not prove this practically, we believe that five days remains a tiny amount of time to provide objective feedback on an app without any prior bias. Moreover, the same comment only talks about the user feeling safe with the app, which does not have any link with the app's features.

*3) Interpretation:* Even though we could not prove our hypothesis beyond the point of doubt, we have observed in our case study of eight mining apps available on Google Play Store between April 2019 and May 2022 that the comments tend to be uninformative and intentionally submitted with high ratings to increase the popularity of the apps. We believe that this has contributed to the popularity of some of those apps.

## V. RQ3: How noticeable is the execution of a cryptojacking app to the device's user

Due to the extensive computations needed to solve the proof-of-work algorithms, mining apps are expected to deplete devices' battery and resources. In this section, we quantify the values for various mining apps to estimate their impact on users. More specifically, we are interested in those apps' battery, CPU, and RAM consumption. .

To provide such information, we decided to run two sets of experiments on one Android device powered with Android 11 with a 1.8 GB RAM powered with a Mediatek MT8766B[13] CPU running with four cores and sustained with a Lithium Polymer Battery 7.7V/3500mAh. On those devices, we installed and gathered metrics about various Android apps' battery, CPU, and RAM usage. We more specifically attempt to quantify those values for three main categories of apps: 1) benign apps that perform background tasks, 2) legit mining apps that users install in complete awareness, and 3) the apps that mine on the back of the users. Despite our initial intention of profiling licit and illicit miners from the web and binary-based families in our study, all the illegal web-based miners in our dataset were no longer functioning. Similarly, we could not successfully install any binary-based miners on our devices. Fortunately for that one, we were able to find an open-source mining app for Android that we used for those tests. Below is a list of the apps that we used for our analysis.

1) *com.facebook.katana* (139e0831...42ad3ba5), the official app for the Facebook social media users on Android was used as the benign app. We made this choice because of 1) the popularity of the app which was downloaded at least five billion users across the world [31]; and 2) the fact that it is pre-installed on many Android devices with sometimes limited possibility of users to uninstall them [32], [33]. Those facts suggest that this is one of the most common apps that Android devices owners ran on a daily basis. The ability of the app to execute background tasks in order to keep users atop of their community news renders day-to-day usage possible outside of the main screens of the app. Consequently, we measured the values for this app while it was executing in the background.
2) *net.craftyourserv.www* (5464fa82...b531c76e) as the licit web-based miner. This app provides an interface (Fig. 2a) where users can simultaneously visualize ads and mine the RUBY cryptocurrency. Therefore, we measured the average impact of the advertisement feature, which we subtracted from the values collected while the mining process was on. The resultant values are considered the impact of mining.
3) *mikado.bizcalpro* (b6001aa8...ece1ed41) as the binary-based cryptojacking app for which we collected the values while the app was executing in the background. This app being the only illicit miner that we studied, we consider its performance representative of illicit mining apps.
4) *com.uplexa.androidminer* (2b4cee1d8...1f86f43c)[14], the fully-fledged binary-based miner. After launching the mining process, the interface presents the user with stats about the mining activity, as seen in Fig. 2b. It then allows the user to leave the app while the mining continues in the background. We collected the various metrics during a period when the app was running in the background.

For each of the four applications, we further ran two sets of experiments described in the following sections.

### A. Estimation of the CPU and RAM usage

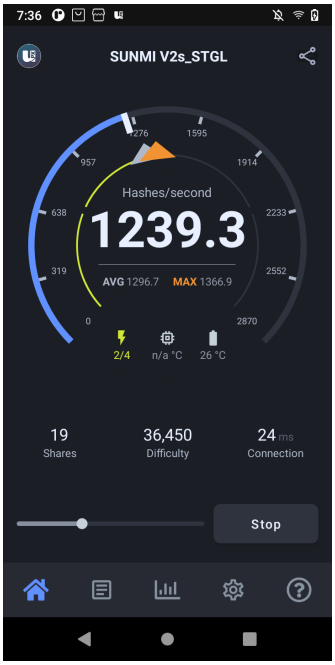
For an uninterrupted duration of 1500 seconds (25 minutes), we ran each of the apps and collected data about the CPU and RAM usage from the Snapdragon Profiler tool[15] .The tool leverages the Android Debug Bridge (ADB) interface to provide real-time measurement of various metrics for an Android app. After collecting and processing the collected values for all the apps, it appears that only the *com.uplexa.androidminer* app uses more than twice the amount of CPU used by any other app. The Facebook app comes right behind and consumes slightly more than any other miner in our test set. Moreover, those results summarized in Fig. 3 show that the illicit miner does consume the least amount of CPU.

[13]https://www.mediatek.com/products/tablets/mt8766b

[14]https://github.com/uPlexa/upx-android-miner/releases/download/v0.4.1/upx-android-miner-v4.1.apk

[15]https://developer.qualcomm.com/software/snapdragon-profiler


(a) Screenshot of the *net.craftyourserv.www* app with the red area serving the ads and the green being used for mining.




(b) Screenshot of the *com.uplexa.androidminer* app showing the ongoing mining process.

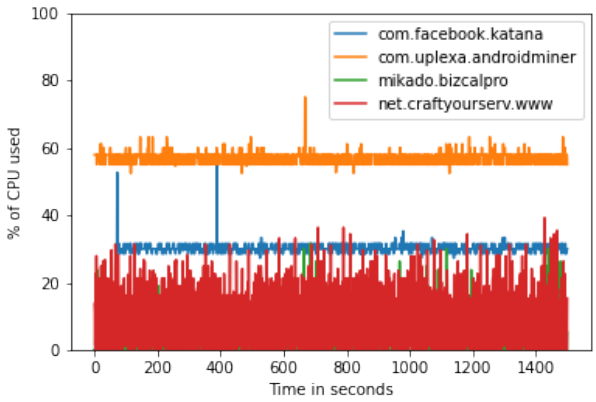



Fig. 3: Comparison of the CPU usage of three mining apps compared to the Facebook app.

We observed the same pattern in the case of RAM usage. Here, the *com.uplexa.androidminer* app alone uses up to 75% of the available memory, which is almost 300x the closest value observed on Facebook (0.26%). Due to that significant disparity, we decided not to report this app's consumption in Fig. 4. Nevertheless, the graph shows that Facebook consumes at least 2.5x more RAM than all the other mining apps, with the web-based mining app being the less greedy. On the other side, the illicit miner does seem to use quite a constant amount of RAM throughout the experiment. Because it uses a proof-of-work algorithm that does not require too much information, one can assume that the CPU is constantly needed leading to the observed irregular pattern.

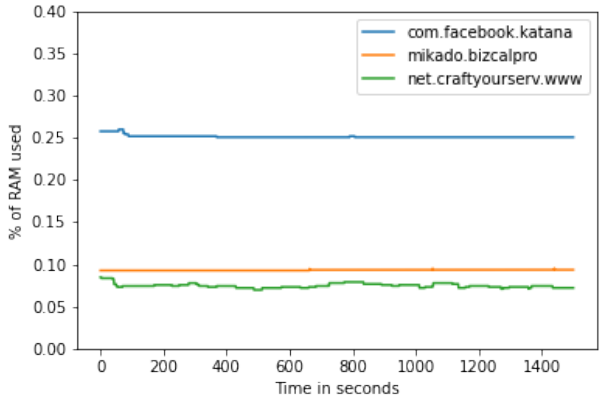



Fig. 4: Comparison of the RAM usage of two mining apps and the Facebook app.

### B. Estimation of the battery usage

For the battery usage of the apps, we leverage the features of ADB to generate and download bug reports and battery statistics summaries. Then, the files were analyzed using the Battery Historian tool[16]. The values presented in Table V are collected based on five series of five minutes execution of each app. We have reset the battery stats between each run to prevent data overlapping.

TABLE V: Battery consumption of the various apps in a five-minute time frame (in percentage).

| Package | Mean | Est. discharge time |
|---|---|---|
| com.uplexa.androidminer | $0.196 \pm 0.028$ | 1 day, 10 hours |
| net.craftyourserv.www | $0.096 \pm 0.009$ | 3 days, 14 hours |
| com.facebook.katana | $0.060 \pm 0.007$ | 5 days, 18 hours |
| **mikado.bizcalpro** | $0.032 \pm 0.004$ | 10 days, 20 hours |

As evidenced in the results, the cryptojacking app surprisingly consumes less battery than all the other apps, including Facebook. From our estimate, it would take roughly ten days of execution for that app to deplete the battery. On the other hand, we also notice that legit mining apps use more battery than any other app. We explain this significant difference in resource consumption because illicit miners attempt to go stealthy by purposefully limiting their usage of the available resources.

### C. Interpretation

The results obtained from the profiling of the sample apps lead to the conclusion that while mining apps can deplete users' batteries daily, illicit miners seem to be diligent with their consumption. Moreover, compared with some popular apps such as Facebook, the impact of illegal miners seems negligible. Such a fact suggests that the average user is likely not to observe severe deviance from normal behavior when an app covertly mines on their device. Therefore, we can assume that many users might be running mining apps in total ignorance.

Even though the user could not notice the malware execution on their own, it is customary to assume that they could

[16]https://github.com/google/battery-historian

TABLE VI: Summary of the detection performed by some free antivirus solutions after scanning the device with the mining apps installed.

| Antivirus | # of malwares | # of miners |
|---|---|---|
| com.sophos.smsec | 5 | 0 |
| com.bitdefender.antivirus | 3 | 1 |
| com.drweb | 1 | 1 |

potentially install an antivirus that would let them know that a mining app is running on their device. Therefore, we decided to execute a set of free antivirus solutions alongside the mining apps to ascertain this. We justify our decision to only test free antiviruses by the fact that there are more accessible to users than the paid alternatives. We executed our tests using three antiviruses identified on Google Play Store after we searched for "antivirus." We summarized the results of those tests in Table VI.

From the corresponding results, it appears that only one app was simultaneously detected by all the antivirus at once (mikado.bizcalpro), with one not associating the actual label to it to let the user know that the app is mining. At most, five mining apps were detecting, accounting for less than half of the ground truth. The probability that an antivirus reports a mining app to the user remains critically low.

## VI. Limitations and Discussion

Given our reliance on AndroZoo as our only data collection point, it is evident that we could not provide an accurate measurement of the mining ecosystem. The first reason is that AndroZoo does not collect any apps besides those running on smartphones leaving out many other less capable devices such as wearables, TVs, and smart appliances. The incredible three-fold growth in the population of those devices in 2022 [3] makes us believe that it is an attractive target to malicious developers as much as the other IoT devices [34]. The exclusion of those devices means we are missing out on some possibly intriguing trends related to them. Similarly, the mere possibility of some apps not being included in AndroZoo for one reason or another could have impacted the scope of our data. However, the lack of any mining family in Cao et al.'s dataset collected using security advisories from respected security companies [19] suggest that a search on security blogs was unlikely to extend our dataset any further. Consequently, we argue that our results provide the best possible understanding of the mining phenomenon on Android.

The same applies to the use of VirusTotal on which we relied to identify the set of apps to manually analyze. Despite the presence of several tools in VirusTotal, it is obvious that the tool is prone to errors which would lead to false and missed detection. Even though our manual analysis nullifies the likelihood of false detection in our final dataset, we are however likely to miss an important number of apps due to our strategy. Unfortunately, we are not able to provide an estimate to the number of miners that we potentially missed through this exercise. Nonetheless, VirusTotal remains the best existing tool for crowd-sourced malware analysis and has been used in research that is very similar to ours [19], [4].

Another potential threat to the validity of our work lies in the manual analysis performed on the malware samples. Though there might be some errors, we adopted the same technique as Dashevskyi et al. [4] to identify the mining logic. Moreover, we searched every project file in case of doubt regarding the mining activity. Our systematic approach to the problem has allowed us to identify some mining apps and present those for which we could not ascertain the presence of the mining code due to a lack of evidence. The absence of a programmable logic that can be used to reproduce our research constitutes on its own, a valid point on which scientist can doubt our results. However, proposing such a technique would have led to building a new detection mechanism which is out of scope of this paper. Furthermore, we used a public dataset which can be exploited to verify the veracity of our conclusions. We believe that our efforts could help researchers in building effective detection mechanisms against mining apps on Android.

## VII. Conclusion

Our study of 346 Android apps suspected of mining cryptocurrencies on the devices provides a first-of-its-kind review of the advances in the cryptojacking phenomenon on Android since April 2019. Though our study confirms the results from studies covering an earlier period, it sheds light on a few problems: (1) the high-rate of false positives detection from many antivirus solutions, (2) the popularity of hacks and cracks as means of carrying the illicit mining logic, (3) the relatively low-resource consumption of cryptojacking apps on Android, and (4) the availability of mining apps in the Google Play Store. Those findings reveal a misfit prevention and fight system that leads to hundreds of thousands of users downloading mining apps that the developers successfully uploaded to the Google Play Store a breach of the developers' policy in place. The upfront notice from some developers of their intent to mine on the device leads us to question the verification process put in place by Google to prevent such apps from being present. The mere fact that more than 70% of users put great trust in the store [35] explains the high number of installs for each of those apps. We believe that Google should put in place better controls to increase the safety of its platform from malicious apps, including mining ones.

We resort to the research community to investigate the problem of how developers get through the verification process and further lure users into downloading malicious and unwanted apps on their devices. It would also be interesting to understand what happens to users who had previously installed those apps after Google removes them from the store. The underlying results would help make the process more reliable and would significantly increase the confidence and security of Android users against various sorts of abuse.

## References

[1] N. Tovanich, N. Soulié, N. Heulot, and P. Isenberg, "The evolution of mining pools and miners' behaviors in the Bitcoin blockchain," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 3, pp. 1–12, Mar. 2022, early access.

[2] Google, "Google keynote (Google I/O '21) - American sign language," May 2021, accessed June 2022. [Online]. Available: https://www.youtube.com/watch?v=Mlk888FiI8A

[3] ——, "Google keynote (Google I/O '22)," May 2022, accessed June 2022. [Online]. Available: https://www.youtube.com/watch?v=nP-nMZpLM1A

[4] S. Dashevskyi, Y. Zhauniarovich, O. Gadyatskaya, A. Pilgun, and H. Ouhssain, "Dissecting Android cryptocurrency miners," in *Proc. 10th ACM Conf. Data Appl. Secur. Privacy*, ser. CODASPY '20, SIGSAC. New York, NY, USA: Association for Computing Machinery, Mar. 2020, pp. 191–202.

[5] Z. Li, W. Liu, H. Chen, X. Wang, X. Liao, L. Xing, M. Zha, H. Jin, and D. Zou, "Robbery on DevOps: Understanding and mitigating illicit cryptomining on continuous integration service platforms," in *IEEE Symp. Secur. Privacy*, ser. SP '22. Los Alamitos, CA, USA: IEEE Computer Society, May 2022, pp. 363–378.

[6] Google, "Financial services: Play console help," 2022, accessed June 2022. [Online]. Available: https://support.google.com/googleplay/android-developer/answer/9876821?hl=en

[7] BBC News, "Google bans crypto-mining apps from Play Store," Jul. 2018, accessed in June 2022. [Online]. Available: https://www.bbc.com/news/technology-44980936

[8] S. Varlioglu, B. Gonen, M. Ozer, and M. Bastug, "Is cryptojacking dead after coinhive shutdown?" in *3rd Int. Conf. Inf. Comput. Techn.*, ser. ICICT '20, San Jose, CA, USA, Mar. 2020, pp. 385–389.

[9] F. Tommasi, C. Catalano, U. Corvaglia, and I. Taurino, "MinerAlert: An hybrid approach for web mining detection," *J. Comput. Virol. Hack. Techn.*, pp. 1–14, Mar. 2022.

[10] F. Naseem, A. Aris, L. Babun, E. Tekiner, and A. S. Uluagac, "MINOS: A lightweight real-time cryptojacking detection system," in *Netw. Distrib. Syst. Secur. Symp.*, ser. NDSS '21, no. 28. Virtual: Internet Society, Feb. 2021, pp. 244–259.

[11] M. Caprolu, S. Raponi, G. Oligeri, and R. Di Pietro, "Cryptomining makes noise: Detecting cryptojacking via machine learning," *Comput. Commun.*, vol. 171, pp. 126–139, Feb. 2021.

[12] S. Varlioglu, N. Elsayed, Z. ElSayed, and M. Ozer, "The dangerous combo: Fileless malware and cryptojacking," in *SoutheastCon 2022*. IEEE, Mar. 2022, pp. 125–132.

[13] H. Badih and Y. Alagrash, "Crypto-jacking threat detection based on blockchain framework and deception techniques," *Amer. J. Sci. Eng.*, vol. 2, no. 1, pp. 1–10, Jul. 2021.

[14] N. Lachtar, A. A. Elkhail, A. Bacha, and H. Malik, "An application agnostic defense against the dark arts of cryptojacking," in *51st Annu. Int. Conf. Dependable Syst. Netw.*, ser. DSN '21, IEEE/IFIP. Taipei, Taiwan: IEEE, Jun. 2021, pp. 314–325.

[15] J. Clay, A. Hargrave, and R. Sridhar, "A power analysis of cryptocurrency mining: A mobile device perspective," in *16th Annu. Conf. Privacy, Secur. Trust*, ser. PST '18. Belfast, Ireland: IEEE, Aug. 2018, pp. 1–5.

[16] P. Kotzias, J. Caballero, and L. Bilge, "How did that get in my phone? unwanted app distribution on Android devices," in *IEEE Symp. Secur. Privacy*, ser. SP '21. San Francisco, CA, USA: IEEE, May 2021, pp. 53–69.

[17] A. Kharraz, Z. Ma, P. Murley, C. Lever, J. Mason, A. Miller, N. Borisov, M. Antonakakis, and M. Bailey, "Outguard: Detecting in-browser covert cryptocurrency mining in the wild," in *World Wide Web Conf.*, ser. WWW '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 840–852.

[18] M. Russo, N. Šrndić, and P. Laskov, "Detection of illicit cryptomining using network metadata," *EURASIP J. Inf. Secur.*, vol. 1, no. 11, pp. 1–20, Dec. 2021.

[19] M. Cao, K. Ahmed, and J. Rubin, "Rotten apples spoil the bunch: An anatomy of Google Play malware," in *Proc. 44th Int. Conf. Softw. Eng.*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, May 2022, pp. 1919–1931.

[20] Y. Zhou and X. Jiang, "Dissecting Android malware: Characterization and evolution," in *Symp. Secur. Privacy*, ser. SP '12. San Francisco, CA, USA: IEEE, May 2012, pp. 95–109.

[21] H. Wang, J. Si, H. Li, and Y. Guo, "RmvDroid: Towards a reliable Android malware dataset with app metadata," in *IEEE/ACM 16th Int. Conf. Mining Softw. Repositories*, ser. MSR '19. Montreal, QC, Canada: IEEE, May 2019, pp. 404–408.

[22] D. Arp, Michael, Spreitzenbarth, M. Huebner, and H. G. K. Rieck, "DREBIN: effective and explainable detection of Android malware in your pocket," in *21th Annu. Netw. Distrib. Syst. Secur. Symp.*, ser. NDSS '14, San Diego, Carlifornia, USA, Feb. 2014. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2017/09/11_3_1.pdf

[23] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "AndroZoo: Collecting millions of Android apps for the research community," in *Proc. 13th Int. Conf. Mining Softw. Repositories*, ser. MSR '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 468–471.

[24] 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," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 6, pp. 1269–1284, Jan. 2017.

[25] K. Khanmohammadi, N. Ebrahimi, A. Hamou-Lhadj, and R. Khoury, "Empirical study of android repackaged applications," *Empirical Softw. Eng.*, vol. 24, no. 6, pp. 3587–3629, Dec. 2019.

[26] M. A. Harris, R. Brookshire, and A. G. Chin, "Identifying factors influencing consumers' intent to install mobile applications," *Int. J. Inf. Manage.*, vol. 36, no. 3, pp. 441–450, Jun. 2016.

[27] S. Pastrana and G. Suarez-Tangil, "A first look at the crypto-mining malware ecosystem: A decade of unrestricted wealth," in *Proc. Internet Meas. Conf.*, ser. IMC '19. New York, NY, USA: Association for Computing Machinery, Oct. 2019, pp. 73–86.

[28] A. Saif, H. AL-KILANI, M. Qasaimeh, and A. Al-Refai, "Analysis of Android applications permissions," in *Int. Conf. Data Sci., E-Learn. Inf. Syst.*, ser. DATA '21. New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 243–249.

[29] A. Alshehri, P. Marcinek, A. Alzahrani, H. Alshahrani, and H. Fu, "Puredroid: Permission usage and risk estimation for android applications," in *Proc. 3rd Int. Conf. Inf. Syst. Data Mining*, ser. ICISDM '19. Houston, TX, USA: Association for Computing Machinery, Apr. 2019, pp. 179–184.

[30] E. Tekiner, A. Acar, A. S. Uluagac, E. Kirda, and A. A. Selcuk, "SoK: Cryptojacking malware," in *Eur. Symp. Secur. Privacy*, ser. EuroS P '21. Vienna, Austria: IEEE, Sep. 2021, pp. 120–139.

[31] Meta Platforms, Inc., "Facebook - Apps on Google Play," Aug. 2022, accessed on 2nd August 2022. [Online]. Available: https://play.google.com/store/apps/details?id=com.facebook.katana

[32] H. Liu, P. Patras, and D. J. Leith, "Android mobile OS snooping by Samsung, Xiaomi, Huawei and Realme handsets," techreport, Oct. 2021. [Online]. Available: https://www.scss.tcd.ie/doug.leith/Android_privacy_report.pdf

[33] S. Frier, "Samsung phone users perturbed to find they can't delete Facebook," Jan. 2019, accessed on 2nd August 2022. [Online]. Available: https://www.bloomberg.com/news/articles/2019-01-08/samsung-phone-users-get-a-shock-they-can-t-delete-facebook#xj4y7vzkg

[34] E. Tekiner, A. Acar, and A. S. Uluagac, "A lightweight IoT cryptojacking detection mechanism in heterogeneous smart home networks," in *Netw. Distrib. Syst. Secur. Symp.*, ser. NDSS '22, no. 29. San Diego, CA, USA: Internet Society, Apr. 2022, pp. 208–223.

[35] A. Mylonas, A. Kastania, and D. Gritzalis, "Delegate the smartphone user? security awareness in smartphone platforms," *Comput, Secur.*, vol. 34, pp. 47–66, May 2013.