Computer and Information Science; Vol. 15, No. 1; 2022
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

MiniWarner: An Novel and Automatic Malicious Phishing Mini-apps Detection Approach

Junhan Chen[1]

[1] College of Cyber Security, Jinan University, Guangzhou, China

Correspondence: College of Cyber Security, Jinan University, Panyu District, Guangzhou, China, 511436.

Received: December 23, 2021 Accepted: January 18, 2022 Online Published: January 21, 2022

doi:10.5539/cis.v15n1p57 URL: https://doi.org/10.5539/cis.v15n1p57

Abstract

WeChat mini-apps are "sub-applications" built within the WeChat platform. Unlike full-function native applications, they are streamlined, "light" versions of the apps, and enable users to open and use them inside WeChat without downloading and installation. Since being introduced by WeChat in 2017, 4.3 million WeChat mini-programs have been developed, and they attract around 410 million daily active users Up to 2021. However, motivated by financial gains, many malicious mini-app developers use some intended description and icon to mislead users to click and open their mini-apps. These mini-apps are full of annoying advertisements and collect users' privacy information stealthily, which can expose users to privacy risks and financial losses.

Although security personnel of WeChat has enforced various countermeasures to prevent malicious phishing mini-apps sneaking into WeChat, rampant malicious leading mini-apps still have been observed recently. In this paper, we present MiniWarner, a novel approach that leverages Natural Language Processing and a number of reverse engineering techniques to detect whether a mini-app is malicious and phishing when users open it. MiniWarner will only ask users whether to continue to open the malicious phishing mini-app, thus it can protect users against the intended misleading by attackers, and still preserve the original user experience. Besides, this approach is implemented as an Xposed module, making it practical to be quickly deployed on a large number of user devices. Our paper will introduce how we developed MiniWarner and the measurement results of MiniWarner in detail.

Keywords: Mini-apps, malware detection, natural language processing, WeChat

1. Introduction

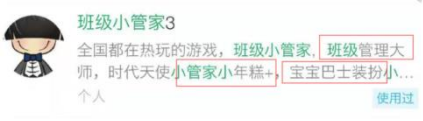
In 2017, WeChat introduced a novel program paradigm named mini-apps that enable users to open and use them directly inside WeChat without downloading and installing (C. Lee, 2017). Mini-apps are "sub-applications" built within the WeChat platform and WeChat allows 3rd party companies to develop mini-apps providing advanced features to users that can run within the APP. Normally, mini-apps are streamlined and light versions of full-function native applications, and they offer mobile users necessary functions and elevated convenience without leaving WeChat. Users can access to Mini Program via 6 ways: (1)WeChat Official Account menu, (2)Embedded into WeChat article, (3)Discover tab of WeChat, (4)Sharing card on WeChat groups, (5)Scan of Mini Program QR code, (6)Search (THOMAS GRAZIANI, 2019).

Up to 2021, there are more than 4.3 million WeChat mini-programs attracting around 410 million daily active users. Nowadays, the mini-apps are not only applied in WeChat. "Following WeChat's growth in China, companies like Facebook, Google, and Apple began allowing third-party apps and services to plug into their own messaging platforms" (L. Eadicicco, 2020).

Mini-apps are everywhere today, ranging from communication to games and media, from which users can enjoy great convenience. Nevertheless, Mini-apps may expose users to security and privacy issues, such as location information tracking and economic loss (Prateek Gupta, 2020). Although the risks may be caused by users' misuse of the apps, malicious mini-apps are the most intense concerns that expose users to such risks. Our work focuses on malicious phishing mini-apps which is one type of malicious mini-apps. This kind of mini-apps can not only gain profit from user clicks but also breach the privacy of users (e.g., stealing user location).

Specifically, a malicious phishing mini-app uses some intended description and icon to mislead users to click and open their mini-apps, but the contents inside their mini-apps are totally different from what the description and

icon show to users. As shown in Figure 1, the description shows that this mini-app is used to help the teachers to manage their classes. However, once the uses open this mini-app, they will find that this mini-app is full of annoying game advertisements and also collects users' privacy information stealthily, which can expose users to privacy risks and economic losses. Although the protection of user privacy is always focused on by application security researchers, and WeChat has applied my strategy to prevent malicious phishing mini-apps sneaking into WeChat, a large number of phishing apps still have been observed by WeChat users in these years.



(a) description

(b) contents

Figure 1. Malicious phishing mini-app

When users download an installation package of a native application, most anti-virus software will scan this package before installation if they were granted permissions (Wu, D. J. et al, 2012). And they will warn users not to install the application once they detect it as malware. Although the mini-app has been proposed for many years, it is pity that there are still no approaches to detect malicious mini-app and warn the users. The difficulty to develop an approach to detect malicious phishing mini-app is twofold.

First, unlike the malicious native application, malicious phishing mini-apps do not contain any features of virus (e.g., control flows and hash values), all their code behavior is the same as normal mini-apps. Thus, it's not workable to detect malicious phishing mini-apps with traditional ways (Black, P., & Opacki, J., 2016; Iadarola et al., 2019). Second, for security reasons, mobile systems do not allow apps to access other apps' data or execution flows. Thus, as a part of WeChat, anti-virus software is incapable to access and scan mini-apps. Besides, WeChat is not an open system in which developers can only get a small amount of information (e.g. Development Guide) and use the public available APIs.

To overcome the above two problems, we propose MiniWarner, a novel approach that leverages Natural Language Processing and some reverse-engineering techniques to detect whether a mini-app is malicious and phishing when users open it. We start from a straightforward observation that in order to confuse users, the description text of malicious phishing mini-apps is made attractive and totally different from the page contents inside the mini-apps. Based on the observation, we leverage Natural Language Processing technique to weigh the similarity between description and content inside a mini-app and identify it as a malicious phishing mini-app if the similarity is minuscule.

The next problem is, how do we access WeChat and analyze mini-apps when users open them. For this issue, we use reverse-engineer techniques to analyze WeChat and uncover the code functions and execution flows. we firstly use reverse-engineered techniques to analyze WeChat static code and found an opening interface that can

open a mini-app and a search interface that can download a mini-app and get its description text and wxapkg (the packed binary file of mini-app) when provided its ID. Then we developed an Xposed (Saariko, 2019) module to hook and change the execution flow of the access interface. The new execution flow first aggressively invokes the search interface with provided mini-app ID and gets the description text and wxapkg. Next, we use the mini-app unpacking tool to unpack the wxapkg without manual effort and get some resource files including the content text we need. The next step is to weigh the similarity between description and content inside the mini-app. Once the similarity is minuscule, MiniWarner will warn the user that he/she is opening a malicious phishing mini-app and ask for the final decision from the user, who is more capable of making the best decision for him/herself based on the warning.

Contribution. The main contributions in this paper are summarized as follows:

1. New understanding of the mini-apps security. To the best of our knowledge, we are the first to leverage Natural Language Processing and reverse engineering techniques to detect malicious and phishing mini-apps. This new understanding can further inspire follow-up research on mobile application security.

2. Novel and effective Tool. We propose MiniWarner, the first tool which can automatically detect malicious phishing mini-apps inside user devices and can be quickly deployed and protect a large number of user devices.

3. Empirical Results. We measured the effectiveness and accuracy of MiniWarner with a set of malicious phishing mini-apps and normal mini-apps we collected. Our experimental results show that MiniWarner can recognize malicious phishing mini-app and normal mini-app with 100% and 97.6% average accuracy and takes less than 5.03 seconds to analyze a mini-app.

2. Background

In this section, we explain the background knowledge about mini-app architecture and mini-app page.

2.1 Mini-app Architecture

As shown in Figure 2, a mini-app contains a front-end and an optional back-end to let the users feel like using the native app.

(i) Front-end is a layered architecture that is responsible for user interaction and system resources handling. More specifically, WeChat provides a lot of APIs to the front-end of mini-apps, these APIs is capable to handle the resources (e.g., Album, Camera, and Location information) inside the mobile system and manipulate the data provided by WeChat (API, 2021). Similar to the APK file in the Android platform, the codes and resources of the mini-app are also packed in a compressed file named wxapkg. A wxapkg typically contains (Mini Program development framework, 2021): (1) a global configuration file named app.json, which determine page file paths and window behaviors, set the network timeout, and set multiple tabs; (2) a number of resource files (e.g., videos and images) that stored in multiple folders; (3) multiple files which describe Logic Layer (App Service) and View Layer (mini-app pages) of the mini-app.

(ii) Back-end is a server that runs in the cloud to provide various services to end-users. These services are mostly related to complex computing tasks and large storage. Moreover, if mini-app developers want to communicate with the back-end server(e.g., accessing the storage), they only need to configure their back-end servers and invoke the APIs which offered by WeChat (Network, 2021).

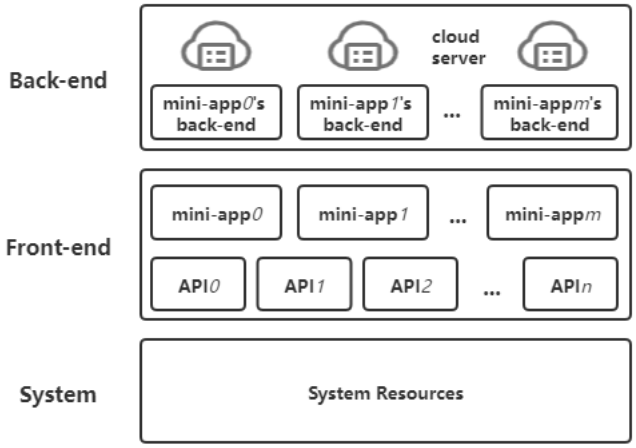


Figure 2. Mini-app Architecture

2.2 Mini-app Page

Mini-app page is an essential mini-app component that provides windows to users. Each page contains multiple GUI elements to interact with users. The GUI elements are designed by app developers to show images, text information, or accept users' input. Most mini-apps consist of multiple pages, one of which can be specified as the home page by developers. When a user launches an app, the first page shown on the screen is the home page.

The Mini-app page is composed of four files including (Directory Structure, 2021) (1) a WXML file, which is used to build the page structure and design the UI elements(e.g. buttons and text boxes)with a markup language; (2) a JavaScript file, specifying the initial data of a page, the lifecycle callback function, the event processing function, and the like of the page.; (3) a WXSS file, which describes WXML component styles, and determine how WXML components are displayed with a set of style languages defined by Tencent; and (4) finally a JSON file, which is used to configure the window behaviors of a page.

3. Challenges and Insights

Our work is aim to build a tool that can brief the user that he/she is opening a malicious phishing mini-app. To achieve this goal, we have to design an approach to intercept the mini-app opening process for obtaining the wxapkg of this mini-app and analyzing the similarity between description and content. However, it's not an easy task, firstly, WeChat is not an open system and it contains a lot of roadblocks for us to intercept the mini-app opening process. Second, developers may implement the pages of mini-apps in various ways with different styles and texts, by solely viewing the pages of the mini-apps, humans can get confused, no mentioning the machine-based detection method. We next introduce our challenges in detail when developing MiniWarner and the insights about how to overcome them.

3.1 Challenges

C1: Analyze the similarity between description and page content. Humans can easily recognize whether the page content of mini-apps is similar to the description, but it is challenging for a machine to recognize it. Prior approaches commonly focus on analyzing the codes to implement the page or UI screens (Black, P. , & Opacki, J., 2016; Wazid, M. et al. 2019), such as judging the difference between different UI screens with their layout structures and program codes. But these approaches cannot be used to analyze the similarity between description and page content. The description of mini-apps is text information, and it’ s not workable to process with code analysis

C2: Find out the mini-app opening and searching API inside the static code of WeChat. As the opening and searching APIs do not disclose by WeChat, and they can’t be invoked by other apps except WeChat, we have to use reverse-engineer techniques to analyze the app and seek out the interfaces and codes. However, the size of WeChat APK is almost 210 MB, and WeChat consists of a large amount of Java and native codes. More specifically, after decompiling the latest version of WeChat APK in the Android platform with tool Jadx (Skylot, 2015), we find that it includes more than 82,952 files which are composed of 56,867 Java classes and 94 shared object (so) files. Moreover, to prevent attackers from reverse-engineering the WeChat to do some malicious behavior, WeChat uses many techniques to heavily obfuscate its codes. How to identify the APIs we need, intercept them at the right time, invoke these APIs with correct parameters, and get the result data from the remote server of WeChat is one of the challenges we need to overcome.

3.2 Insights

We proceed to elaborate our insights with regard to our challenges in a one-by-one manner.

S1: Calculating similarity with Natural Language Processing technique. The key is texts on the description and page content of mini-apps contain specific features. Specifically, to attract users, the texts on the description tell the category of mini-apps and give a brief introduction to the functions of them. As for the texts on the page content, they contain rich semantic information to guide the users on how to use the functions in the mini-apps. Thus, we calculate the similarity between description and page content by analyzing its texts by leveraging Natural Language Processing. To start with, we extract texts from page contents with static analysis. Then, the texts of description and pages content are both processed into word arrays, and this stage consists of two phases, text splitting and stop words removal. Afterward, we use Natural Language Processing tool to calculate the similarity of these two word arrays.

S2: Reverse engineering with static and dynamic analysis. To overcome C1, we need to use reverse-engineer techniques to analyze the codes of WeChat. To begin with, we discover that if we want to open a mini-app that is shared by other users, we need to click a button to open it. Thus, WeChat must contain an interface to handle this

click event, and this interface will first construct a request and send it to the remote server of WeChat to search and get the description and wxapkg of this mini-app. Then this function will parse the wxapkg and open the mini-app. That is, there must be an opening function which we need to hook and intercept (Heuser, S. et al., 2014), and a server API to search mini-app and we just need to invoke the API with correct parameters. Thus, we first used a dynamic instrumentation toolkit Frida (Oleavr, 2016) to hook and trace the interface to handle the click event. We next decompiled the static codes of WeChat and located the handler interface with static analysis. However, the code heavily uses obfuscations, it's complicated to find out where to get the description and wxapkg. As such, we hook all the functions which are invoked after the handler function is triggered. Through printing the parameters and return values of these functions, we finally located the function to return the description and wxapkg with a mini-app ID as input. We followed similar practice to identify the function that parses the wxapkg and opens the mini-app.

4. Current Design of MiniWarner

This section begins with the assumption and scope we are concerned about, and then figure out the design of MiniWarner in detail. As shown in Figure 3, MiniWarner consists of three stages: (1) Package Download, which intercepts the handle function when users open a mini-app and gets its description and downloads its wxapkg by invoking the specific interface inside WeChat. (2) Content Text Extract, which unpacks the wxapkg of target mini-app and extracts the content text from its mini-app pages. (3) Malware Detect, which uses natural language processing to process the description and content texts and calculate the similarity of them.

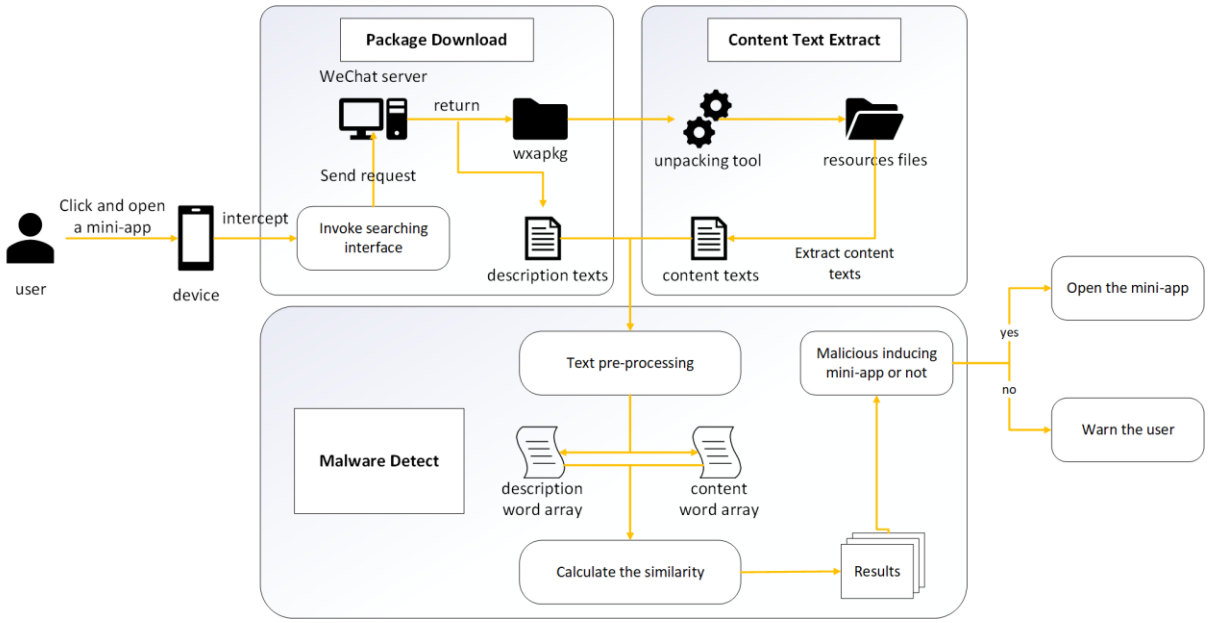


Figure 3. MiniWarner overview

4.1 Assumption and Scope

MiniWarner is an Android malicious phishing mini-app detection approach that combines natural language processing and reverse-engineering technique. Like most malware, malicious phishing mini-apps may pretend harmlessly and do not have any system privileges. They only require little permission and use attractive descriptions to confuse users without damaging the Android system and WeChat platform. We assume that malicious phishing mini-apps only can use APIs which are available to legitimate mini-apps. Besides, we do not consider those malicious phishing mini-apps which build their pages dynamically or use any other techniques to escape from our detection purposely, and thus it is out of the scope of our paper. To better implement and improve our experiment, we assume that most mini-apps in WeChat are legitimate and we only consider Chinese apps here. We implement the MiniWarner as a module for the Xposed framework, a popular code-injection framework for rooted Android devices.

4.2 Package Download

As mentioned in Assumption and Scope, we implement the MiniWarner as a module for the Xposed framework. The Xposed module will be loaded to the Android system once its targeting app (WeChat in this paper) is opened by users or the system. In order to change the execution flow of the access interface, MiniWarner hooks the opening interface, which means that once users open a mini-app, MiniWarner will intercept this interface and run the codes defined by ourselves. Our new codes first get the parameter of the opening interface. This parameter is the unique identifier of mini-app and we call it mini-app ID. MiniWarner then invokes the searching interface with this mini-app ID. This searching interface will send a request to the remote server of WeChat and get the description text and wxapkg matching the mini-app ID searched. The description text will be recorded for further use and the wxapkg will be stored in a specific location of the device.

4.3 Content Text Extract

In this stage, MiniWarner gets the wxapkg of target mini-app downloaded from the last stage and uses the mini-app unpacking tool to unpack the wxapkg without manual effort. This process results in a number of code files and resources files of mini-app pages. As we mentioned in the Background, developers need to use four kinds of files to construct the mini-app page, a WXML file , a JavaScript file, a WXSS file, and a JSON file to configure the window behaviors of a page, we only focus on the WXML files which include the texts drawn on the screen and visible to users. The text in the WXML files of all pages will be extracted for further use. Note that some developers may use images to show key texts for a better visual experience (e.g. title, prompt). In that case, MiniWarner gets all images from the folder where the resource is stored and adopts Optical Character Recognition (OCR) (Mithe, R., Indalkar, S., & Divekar, N., 2013) techniques to extract texts from these images. The final output of this stage is a set of content texts.

4.4 Malware Detect

After intercepting the opening interface in Stage 1 and processing the wxapkg of the target mini-app through Stage 2, we gain the description texts and content texts. This Stage detects whether the target mini-app is a malicious phishing mini-app based on the similarity of these two sets of texts. As different developers have different coding practices, texts in pages are written in all kinds of formats, which can extremely reduce the accuracy of similarity calculation. To improve the accuracy, texts extracted from the Stage 1 and Stage 2 need to be processed into word arrays whose elements are separate words with the unified format. This process consists of two phases, text splitting and stop words removal.

Texts splitting: In the text splitting phase, we firstly regard all non-Chinese characters as delimiters and split texts with those delimiters. For example, “字母/数字” can be split into two separate words “字母” and “数字”. Besides, texts and strings without any separated characters are also commonly used, such as ”银行账户”. For this case, we split such text into separated words by iteratively matching the maximum length word in cacl2 (Limccn, 2021) until the text can not be split anymore. Thus, “银行账户” can be split into “银行” and “账户”.

Stop words removal: Through the Texts splitting phase, description texts and content texts are processed into two word arrays. We remove stop words defined in stopwords (goto456, 2019) from the two word arrays, such as “我的”, “那些”, which provide worthless information to pages.

Finally, we use text2vec to calculate the similarity between the two word arrays processed from the Texts splitting and Stop words removal phases. If the similarity value is smaller than α (we will discuss in the Evaluation section), MiniWarner will warn the user that he/she is opening a malicious phishing mini-app and ask for the final decision from the user, who is inherently more capable of making the best decision for him/herself based on the context. Otherwise, open the mini-app as usual.

5. Evaluation

In this section, we present our evaluation results. We first discuss the parameter α. Then we evaluate the detection accuracy and the efficiency of MiniWarner.

5.1 Evaluation Setup

We firstly get 36 malicious phishing mini-app wxapkgs which are uncovered by WeChat users and reported in the WeChat open community. Then we use mini-app crawler MiniCrawler (Zhang, Y. et al, 2021) to get 500 normal mini-apps wxapkgs which have been pre-classified into 34 categories in the WeChat platform. To ensure that the mini-apps we crawl from WeChat are benign, we send them to VirusTotal (VirusTotal, 2020) and remove those flagged as positive by anti-virus engines.

All experiments are performed on three Android phones, Google Nexus 6P, a Google Pixel, and a Google Pixel XL. In particular, the system of our devices is Android 7.1 and all devices have installed WeChat version 7.1.8.

5.2 Parameter α Discussion

In this section, we explore a tunable hyperparameter for MiniWarner, α. In particular, α is an important factor that determines whether a mini-app is malicious and phishing. Once the similarity value between the description texts and content texts of a mini-app is smaller than α, we regard this mini-app as a malicious phishing app. If the parameter α is too big, the accuracy to detect normal mini-apps will be small, but If the parameter α is too small, the accuracy to detect malicious phishing mini-apps will be small. Thus, we need to set the parameter α to a proper value.

We unpacked the 36 malicious phishing mini-app wxapkgs and calculate the similarity values between the description texts and content texts with our Natural Language Processing method. As shown in Figure 4, all similarity values the 36 malicious phishing mini-app are smaller than 20. We also use the same approach to test the similarity values of 100 normal mini-apps we collected, as shown in Figure 4, the range of them is between 50-80. As it is significant to strike a balance between normal mini-apps and malicious phishing mini-apps detection accuracy, we finally choose α = 35.

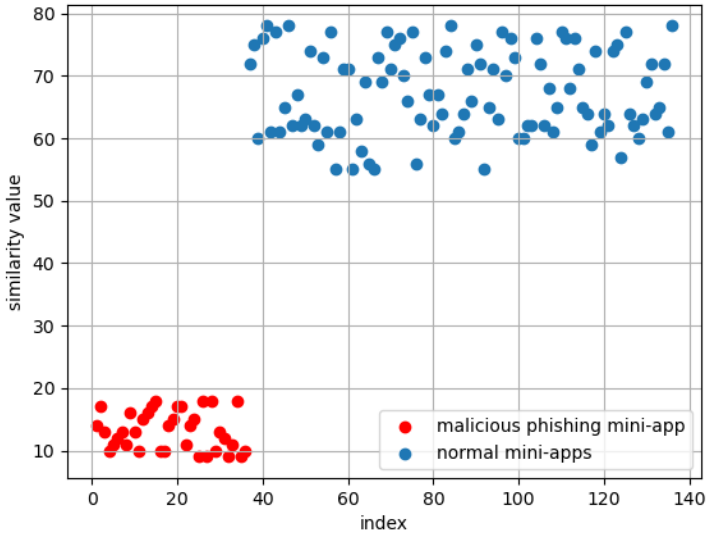


Figure 4. Similarity values between description texts and content texts

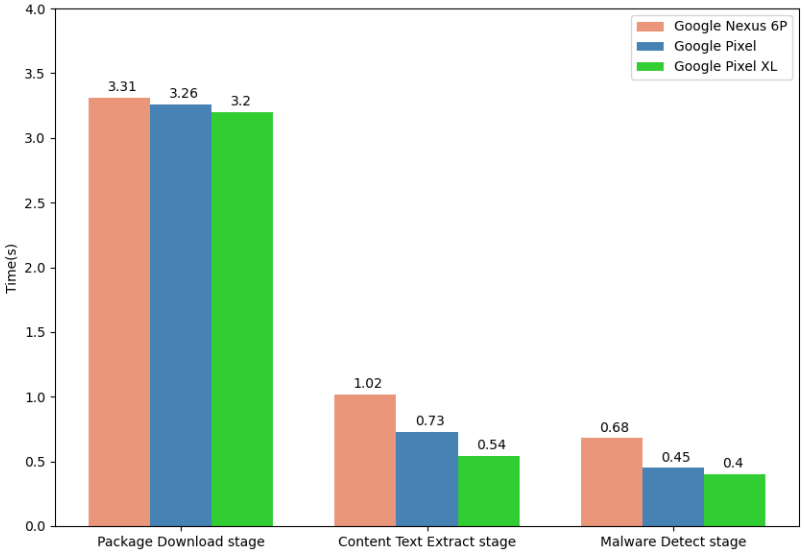


Figure 5. Computational efficiency of MiniWarner in different phones

5.3 Efficiency of MiniWarner

We discuss the efficiency of MiniWarner by running with three Android phones in this section. The datasets we use in this section include 36 malicious phishing mini-apps and 500 normal mini-apps. We run our MiniWarner with these 536 mini-apps in three Android phones, and record the time consumption in every stages. The average time costed in these 536 mini-apps is our final result. As shown in Figure 5, in the Package Download stage, for a single mini-app, all of the three phones took near the same average time to get a mini-app downloaded, about 3.3 seconds. And in the Content Text Extract stage, Google Nexus 6P, Google Pixel, and Google Pixel XL took 1.02 seconds, 0.73 seconds, and 0.54 seconds on average respectively. In the Malware Detect stage, Google Pixel XL spent the shortest time, and Similar to it, Google Pixel costs about 0.45 seconds. Google Nexus 6P perform the worst, it took about 0.68 seconds. As can be seen from the above analysis, even though Google Nexus 6P, the phone with lowest efficiency, only took 5.03 seconds in the whole process, which does not negatively affect the user experience.

5.4 Detection Accuracy of MiniWarner

We now proceed to the accuracy evaluation of MiniWarner. We manually select 36 malicious phishing mini-apps which are uncovered by WeChat users in the WeChat open community and 500 normal mini-apps that are crawled by MiniCrawler (Zhang, Y. et al, 2021). As shown in Table 1, 36 malicious phishing mini-apps in our dataset can all be detected correctly, but 12 normal mini-apps are detected as malicious phishing mini-apps by mistake. After analyzing them manually, we find that these 12 mini-apps are incomplete, which means the developers do not finish their development, and these mini-apps contain inadequate description texts and content texts or contain lots of meaningless texts. Thus, the similarity value calculated by our NLP process is smaller than α.

Table 1. The results of accuracy evaluation

Mini-apps Types	Types after detection		Accuracy
	Normal mini-apps	malicious phishing mini-apps	
Normal mini-apps	488	12	97.6%
malicious phishing mini-apps	0	36	100.0%

6. Conclusion

In this paper, we propose MiniWarner, a novel approach that leverages Natural Language Processing and a number of reverse engineering techniques to detect malicious phishing mini-apps. We implement the MiniWarner as a module for the Xposed framework. This Module will intercept the opening interface once the users open a mini-app. It first dynamically download the wxapkg and description of this mini-app from the server of WeChat and unpack the wxapkg with the unpacking tool resulting in a set of resources files. MiniWarner then uses static analysis to extract the content texts from these resources files. The description texts and content texts will be used to identify whether this mini-app is malicious and phishing or not. Our evaluation on a lot of malicious phishing mini-apps and normal mini-apps shows that MiniWarner can effectively detect banking malware with high precision and low false alarms.

In order to analyze the content text, MiniWarner need to extract the text from the WXML files statically. Thus, MiniWarner cannot deal with those malicious mini-apps which load their WXML files dynamically (e.g., loading WXML files via reflections or downloading WXML files from their server). Moreover, although the Android platform are the major damage of phishing attacks, some other kinds of system are still in danger, such as Windows and iOS, we do not discuss them in our paper. In our future work, we plan to use dynamic exploration to solve the first problem and do more research to make a more generic framework.

References

API. (2021). In *Weixin Docs*. Retrieved from https://developers.weixin.qq.com/miniprogram/en/dev/framework/app-service/api.html#API

Black, P., & Opacki, J. (2016, October). Anti-analysis trends in banking malware. *2016 11th International Conference on Malicious and Unwanted Software (MALWARE) (pp. 1-7).* https://doi.org/10.1109/MALWARE.2016.7888738

C. Lee. (2017). *WeChat launches mini-app feature.* Retrieved from https://www.zdnet.com/article/wechat-launches-mini-app-feature/, 012017

Directory Structure. (2021). In *Weixin Docs*. Retrieved from https://developers.weixin.qq.com/miniprogram/en/dev/framework/structure.html

goto456. (2019). *中文常用停用词表（哈工大停用词表、百度停用词表等）*. Retrieved from https://github.com/goto456/stopwords

Heuser, S., Nadkarni, A., Enck, W., & Sadeghi, A. R. (2014). ASM: A Programmable Interface for Extending Android Security. *USENIX Association*.

Iadarola, G., Martinelli, F., Mercaldo, F., & Santone, A. (2019, October). Formal methods for android banking malware analysis and detection. *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS) (pp. 331-336)*. https://doi.org/10.1109/IOTSMS48152.2019.8939172

L. Eadicicco. (2020). *How facebook, Apple, Google copied china's WeChat messaging app - business insider.* Retrieved from https://www.businessinsider.com/facebook-apple-google-copied-wechat-app-trump-executive-order-2020-8

Limccn. (2021). *Lexicon for Chinese lexical analyzing, 中文语言分词词库*. Retrieved from https://github.com/limccn/cacl2

Mini Program development framework. (2021). In *Weixin Docs*. Retrieved from https://developers.weixin.qq.com/miniprogram/en/dev/framework/MINA.html,.

Mithe, R., Indalkar, S., & Divekar, N. (2013). Optical character recognition. *International journal of recent technology and engineering (IJRTE), 2*(1), 72-75.

Network. (2021). In *Weixin Docs*. Retrieved from https://developers.weixin.qq.com/miniprogram/en/dev/framework/ability/network.html

Oleavr. (2016). *Clone this repo to build Frida*. Retrieved from https://github.com/frida/frida

Prateek, G. (2020). *Researchers listed the most dangerous malware, viruses Android viruses of 2020*. Retrieved from https://gizmeek.com/researchers-listed-the-most-dangerous-malwareviruses-android-viruses-of-2020

Saariko. (2019). *Introduction to android hook framework Xposed*. Retrieved from https://programmer.ink/think/introduction-to-android-hook-framework-xposed.html,

Skylot. (2015). *Dex to Java decompiler*. Retrieved from https://github.com/skylot/jadx

Thomas, G. (2019). *What are WeChat Mini-Programs? A Simple Introduction*. Retrieved from https://walkthechat.com/wechat-mini-programs-simple-introduction/

VirusTotal. (2020). *VirusTotal:A free virus, malware and URL online scanning service*. Retrieved from https://www.virustotal.com/gui/

Wazid, M., Zeadally, S., & Das, A. K. (2019). Mobile banking: evolution and threats: malware threats and security solutions. *IEEE Consumer Electronics Magazine, 8*(2), 56-60. https://doi.org/10.1109/MCE.2018.2881291

Wu, D. J., Mao, C. H., Wei, T. E., Lee, H. M., & Wu, K. P. (2012, August). Droidmat: Android malware detection through manifest and api calls tracing. *2012 Seventh Asia Joint Conference on Information Security (pp. 62-69)*. https://doi.org/10.1109/AsiaJCIS.2012.18

Zhang, Y., Turkistani, B., Yang, A. Y., Zuo, C., & Lin, Z. (2021, May). A Measurement Study of Wechat Mini-Apps. *2021 ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems (pp. 19-20)*. https://doi.org/10.1145/3410220.3460106