

IERG4999 IJ01    Final Year Project II

# Android Game Hacking

BY

CHEN ZHEN  
1155157194

A FINAL YEAR PROJECT REPORT  
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF BACHELOR OF INFORMATION ENGINEERING  
DEPARTMENT OF INFORMATION ENGINEERING  
THE CHINESE UNIVERSITY OF HONG KONG

Apr, 2024

# Android Game Hacking

Chen Zhen 1155157194

---

## Abstract

The emergence of new third-party platform-based games, such as Wechat mini-games, presents higher risks of being hacked compared to traditional App-based games. In order to address this issue, I conducted extensive research on hacking methods used in these games, in order to identify common vulnerabilities and propose effective protective measures. During the testing phase, I discovered that Wechat employs various advanced data encryption techniques during data transfer, offering a significant advantage over traditional Android games.

---

## 1. Introduction

Mobile games have become increasingly popular in recent years, offering more complexity and entertainment than ever before. The market for mobile games has surpassed 75 billion dollars [1], surpassing the popularity of PC games. In light of this trend, major companies like Tencent have developed third-party platforms, such as Wechat-minigame, to allow mini companies and individuals to develop their own games.

The continuous advancement of science and technology has led to a significant increase in the number of WeChat users [6]. In 2023, the number of WeChat platform-based mobile games is projected to exceed 300,000, compared to 100,000 in 2022 [2]. WeChat mini programs not only serve as open platforms for quality services but also serve as starting points for various travel enterprises through their own WeChat mini programs [9].

The advantages of WeChat mini-games include low cost, cloud development, a powerful WeChat function library, easy publicity and monetization, and over 700 APIs [4] that support full back-end monitoring services. These games have created a new mobile game ecosystem throughout mainland China, utilizing an encapsulated language called wxscript, with rendering syntax (e.g., WXML, SWAN) similar to HTML and logic implemented using JavaScript (JS) [7].

However, while these games offer convenience and enjoyment, it is important to recognize the potential devastating consequences of hacking. As these games are interconnected with the data of millions of users on the platform, the research on third-party platform-based mobile games becomes crucial in ensuring their security and protecting user information.

### 1.1. The Structure of Wechat Mini-program Communication

Wechat employs a robust and secure structure to protect data during the transfer process. This structure includes several key components. Firstly, a login token is generated when a user logs into the mini-program. This token serves as a temporary identifier. Secondly, each mini-program is assigned a unique Application ID, ensuring the integrity of the program's identity. Additionally, a Master key, also known as the App secret, is

assigned to each mini-program along with the corresponding AppID. These keys are used for encrypting and decrypting data during communication with the server.

When data is being transferred, the mini-program utilizes the login token, AppID, and Master key to request an encryption key from the Wechat server. It's important to note that these keys are not passed through the user's computer, making it impossible for us to obtain them. Once the encryption key is generated, Wechat sends the encrypted data to the mini-program, which then uses the encryption key for decryption (as illustrated in fig 1).

It is crucial to maintain the confidentiality of these keys, as any leakage of the Master key can have severe consequences. These consequences may include user account hijacking, promotion abuse, and service theft [5]. Therefore, it is essential to prioritize the security of these keys to ensure the overall integrity of the system.

### 1.2. The Customs of The Vulnerability in Third-party Game Ecology

In WeChat mini-games, there is an intriguing aspect that sets them apart from traditional games. These games operate on the WeChat super operating system and are exclusively available online, without any charges for downloading. As a result, companies generate revenue primarily through advertising views and the sale of in-game items. It is worth noting that most advertisements are published by popular games, which leads to these game companies relying solely on revenue from the sale of in-game items. This unique phenomenon creates a distinct ecosystem for these games.

Within this ecosystem, games with a low player count, limited exposure to advertisements, and the absence of paid game items are particularly vulnerable to hacking. Conversely, games with a large player base, the inclusion of paid game items, and strategically placed advertisements in other games or WeChat Moments exhibit the least vulnerability to hacking. These companies understand the importance of addressing vulnerabilities and are willing to invest additional resources to enhance the popularity and security of their games.

### 1.3. The Research Direction

In this research, I propose to investigate various methods of hacking third-party platform-based games, specifically focusing on the unique characteristics of WeChat as a super operating system. I aim to explore the interactions between the WeChat server, WeChat client, mini-program server, and mini-program client, and how these interactions can potentially be exploited.

Additionally, I will draw upon previously identified methods of hacking Android games as a reference for understanding potential vulnerabilities in these third-party platform-based games. By combining knowledge of WeChat's operating system and existing hacking techniques, I hope to gain insights into the possible security loopholes that may exist within these games.

My research will involve conducting experiments and simulations to test the effectiveness of different hacking methods. I will analyze the results to identify common patterns and potential weaknesses in the system. The ultimate goal is to provide valuable insights into the security of these third-party platform-based games and propose effective measures to protect them from potential hacks.

## 2. Description of Problem

### 2.1. The Neglect of Game Protection

The WeChat Mini Program aims to lower the entry barrier for developers, resulting in reduced learning costs, production costs, promotion costs, and operating costs [8]. While this low threshold fosters a flourishing game ecosystem on third-party platform-based mobile games, it also presents a significant challenge in terms of game security. Many games are developed by individual developers whose primary source of income comes from advertisements rather than selling in-game items. Advertisers compensate developers based on the number of ad views. As a result, many developers tend to overlook the importance of protecting their games against hackers, leading to a lack of resilience against hacking attempts. A notable example is the game "Yang Le Ge Yang," which attracts millions of players and generates millions of Chinese yuan in revenue daily. Unfortunately, it was easily hacked through local file modification, underscoring the need for stronger security measures.

Furthermore, many developers overly rely on the client-end, which poses inherent risks. Hackers can easily decompile the front-end code if proper protection methods are not implemented, such as compiling the code into a native language. Storing sensitive data, particularly the master key, and making data revisions on the client-end can pose significant risks to the overall security of the game. For instance, a well-known game called "Kill the Virus," developed by an IT company and played by millions of people, made the mistake of placing encryption logic and the master key on the client-end. Consequently, hackers could manipulate the game's server by sending fake data packages after understanding the underlying logic.

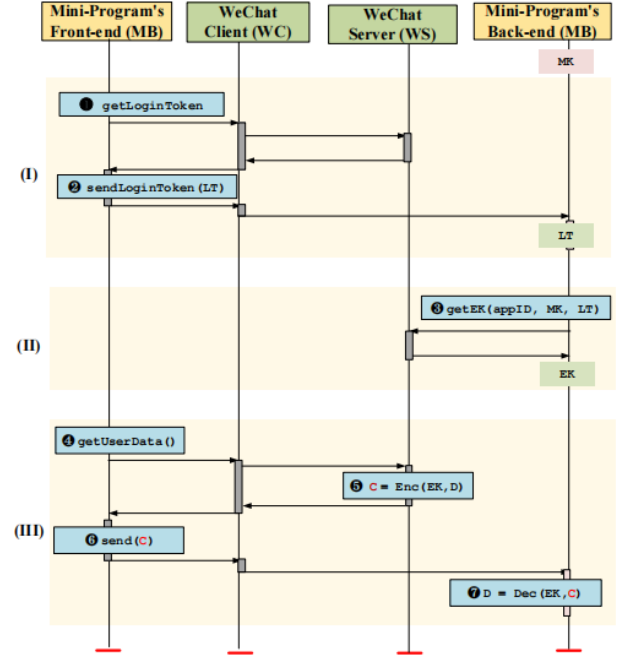


Figure 1: The structure of wechat communication from paper 'Don't Leak Our Keys' [5].

### 2.2. The Absence of Relevant Research

The realm of third-party platform-based games remains relatively unexplored in academic research, particularly in comparison to the extensive studies conducted on Android games in previous decades. Despite their profound impact on the lives of individuals, especially within mainland China, where they have transformed the way people engage with digital entertainment, there is a noticeable dearth of scholarly articles and dedicated research websites on these games. This is particularly evident in the case of WeChat Mini Program games, which boast an impressive user base of over 500 million monthly active users as of the first half of 2023. Even though these games have become a lucrative business, with an estimated income of 30 billion in 2023 and numerous companies relying on them as their primary revenue source, the academic community has yet to fully recognize the research potential and practical implications of studying these games. Conducting research on third-party platform-based games is essential not only to understand their influence on user behavior but also to provide valuable insights for developers to enhance the quality and user experience of their games. Therefore, it is imperative to acknowledge the significance of research in this domain and explore the untapped opportunities it offers for academic investigation and industry advancement.

### 2.3. Approaches by Other People

Since third-party platform-based games are relatively new in the gaming ecosystem, there is a scarcity of research specifically focused on these games. However, some relevant studies conducted in the traditional Android mobile game domain can provide valuable insights. For instance, a study in 2016 on the top 100 mobile games on Google Play [3] identified five main

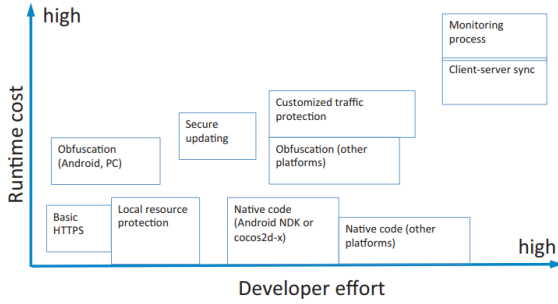


Figure 2: The level of protection as well as their time needed to implement [3].

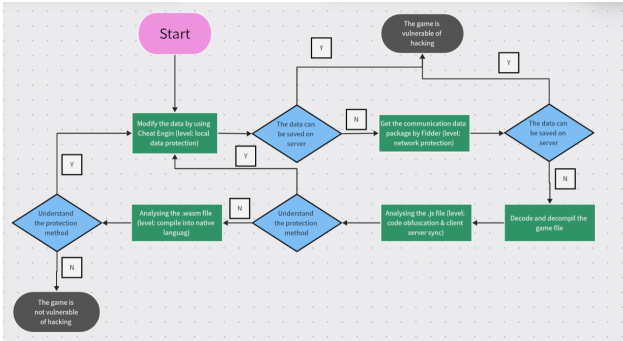


Figure 3: The flow chart shows the whole process of hacking.

areas of game hacking and defense: local-resource protection, network protection, code obfuscation, native code compilation, and server-client synchronization. The researchers utilized various hacking tools such as ProxyDroid for network analysis, GameKiller and CheatDroid for package modification, among others. They also decompiled games to understand their logic and utilized debugging tools to recover code compiled into native languages.

Based on these findings, it is reasonable to assume that similar approaches can be implemented in third-party platform-based games. Conducting research in this area can unveil additional insights and findings specific to these games. It is worth exploring how these protection methods perform in the context of third-party platform-based games to enhance game security. For a visual representation of their research findings, refer to Figure 2, which presents a conclusive summary of the performance of these protection methods.

### 3. Previous Work

Based on my personal experiences and expertise in hacking various games on WeChat, including Lie Flat and Live, Balls Jumping, Puzzle of Number, Find Ways in The Million World, and King's Intention, I have gained valuable insights into their underlying structures and have made significant breakthroughs in terms of protection methods. My findings have revealed new approaches and advancements compared to traditional methods used in hacking Android games.

To visually represent my research, I have created a flowchart (Figure 3) that illustrates the comprehensive process of hack-

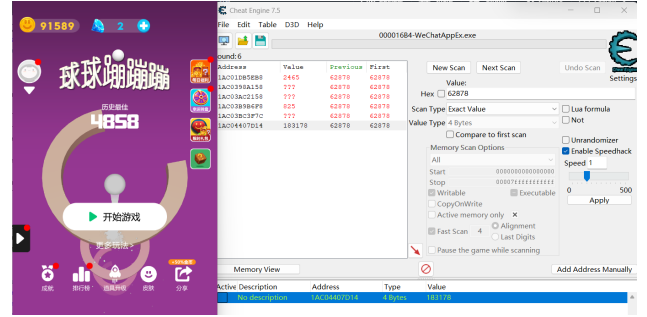


Figure 4: Full hacked and modify the data of game Balls Jumping by memory detection.

ing these games. Additionally, Figure 4 displays the successful hacking process of Balls Jumping, a highly popular game that utilizes wasm native code and extensive code obfuscation but lacks client-server synchronization and legality detection. Through memory scans, I was able to identify and modify sensitive data.

These figures serve as visual aids to enhance the understanding of my research findings. They demonstrate the vulnerabilities present in third-party platform-based games and emphasize the necessity of implementing robust protection measures to ensure the security and integrity of user data.

#### 3.1. Decode of Wechat Executable File

Indeed, the .wxapkg file format created by WeChat is an executable file that runs within the WeChat environment. This file type can be utilized on both Windows and Android platforms. In the case of Android, the file is typically located in the root data file, which requires root access to be accessed. For Windows, the file is encoded in a manner that prevents successful decompilation by conventional decompilers. However, as the encoding algorithm used by WeChat is uniform across all games, decoding methods can be developed and updated accordingly. It is important to note that these decoding tools should be regularly updated to accommodate any changes implemented by WeChat in subsequent updates.

The latest encoding method employed by WeChat involves the following steps:

1. Generate an AES key using pbkdf2 with the WeChat Mini Program ID as the password. The "saltiest" is used as the salt, and the iteration process is performed 1000 times to obtain a 32-byte key.
2. Encrypt the first 1023 bytes of the wxapkg package using AES with the generated key and IV (16 bytes).
3. XOR the remaining data bytes after the first 1023 bytes with the second-to-last character of the WeChat Mini Program ID. In cases where the ID consists of fewer than two characters, XOR is performed with 0x66.
4. Write the encrypted AES data (1024 bytes) and the XOR-ed data together into a file, starting with the identifier "VIMMWX" [11].

Although the encoding process is complex, given the vast number of WeChat users, tools can be obtained to decode these files. It is worth noting that the availability and accessibility of

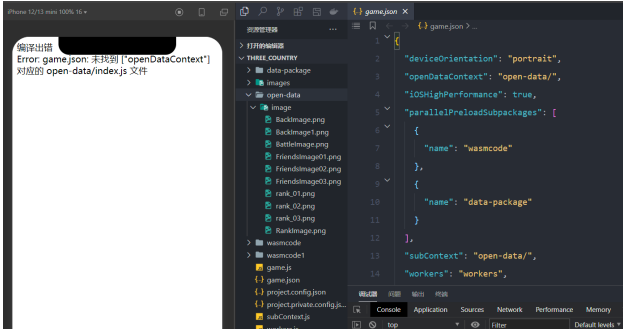


Figure 5: In the game Defence Three Country, index.js package will be downloaded when loading to prevent hackers from running the game on tools.

such decoding tools may vary, and staying up-to-date with the latest developments in decoding methods is essential for successful decoding.

### 3.2. Decompile the Executable File

When it comes to file decompilation, one major challenge arises in the case of certain small and less popular games. These games often adopt a straightforward development approach, utilizing a traditional structure that includes all resources within a single project. Unfortunately, such a structure makes it relatively easy for decompilers to reverse-engineer the game.

On the other hand, some games employ a more complex structure by dividing the game into multiple sub-packages. This approach aims to enhance the overall structure and make it more challenging for decompilers. For instance, the game 'Defence of Three Countries' takes it a step further by injecting data packages during the loading process. As a result, decompiled packages fail to run properly on the WeChat developer tool (refer to Fig 5).

By analyzing the different approaches taken by developers in structuring their games, it becomes evident that the level of complexity impacts the effectiveness of file decompilation. While simpler structures are more susceptible to decompiling efforts, complex structures, especially those involving data package injection, prove to be considerably more challenging to reverse-engineer.

### 3.3. Run the Decompiled Project

Once the files are decompiled, I utilize the WeChat developer tool to run them, as depicted in Figure 6. This allows me to gain a deeper understanding of the underlying data transmission and coding logic employed by the games. It is worth noting that games running on WeChat utilize a language called Wx-script, developed by the WeChat group and based on JavaScript. However, astute developers often incorporate native language packs within the logic section, which requires a more comprehensive effort to comprehend.

In the past, the sub-package aspect, which involves structure obfuscation, posed a challenge in research endeavors. Fortunately, with the utilization of more powerful decompiling tools, this obstacle has been overcome, enabling researchers to address this specific concern effectively.

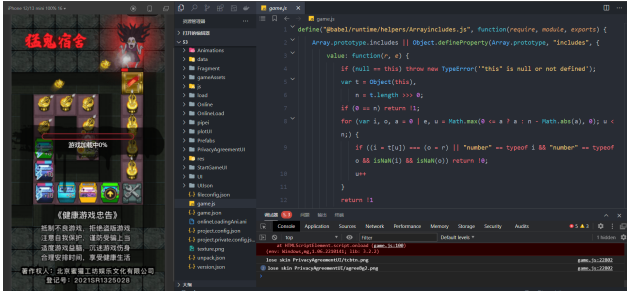


Figure 6: Succeeded in running the decompiled game Lie Flat and Live on Wechat developer tool.

### 3.4. Modify the Local Data

In the next phase, I employ Cheat Engine to modify the game data. Firstly, I use the speed hack feature to locate the target process. This is crucial because the speed of the target game will change accordingly if the process is correctly identified. Once the process is located, I proceed to search for variables by utilizing number-based searches, which involve some element of guesswork. For instance, if I observe that the in-game coin value is 100, I would guess that the variable representing the coin is also 100. I then retrieve the address of this variable and test its modification by changing the coin value in the game. If the modification behaves as anticipated, it indicates that the variable has been correctly located. However, if the modification does not yield the desired outcome, it suggests that the developers may have implemented encoding methods to protect the variables.

Certain vulnerable games, such as "Melon Cut Together," do not safeguard sensitive parameters, making it relatively easy for me to locate and modify them. However, games that employ server-client synchronization, like "Majong Online," render my modifications ineffective and result in failure. Similarly, in the game "King of Salted Fish," although I can modify the attack rate of my character, the damage calculation occurs on the server side. Consequently, even if I successfully defeat an enemy, the server will still register it as a failure. Additionally, some games obfuscate the parameters, making them challenging to locate. For instance, in the game "Minor God," the displayed amounts of money are represented as A, B, C, and D, where A signifies thousands and B represents millions.

Thankfully, Cheat Engine offers two significant advantages: memory detection and speed hack functionality. Unlike traditional data modification methods that require handling code obfuscation, Cheat Engine can directly scan the memory and bypass package defenses once the logic of how parameters are encoded and stored in fragments is understood. The speed hack feature is an advanced tool capable of modifying the client's time, which is difficult to prevent unless the game implements full client-server synchronization.

### 3.5. Modify the Data Package

In the final phase, I focused on modifying the data packages. Inexperienced developers often prioritize protecting local files by employing obfuscation techniques and compiling them into



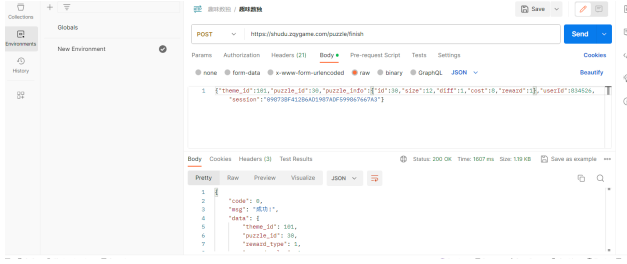


Figure 7: In the game Puzzle of Number, a fake data package is sent to server and pretend that I have passed the game.

native languages, but they overlook the security of network traffic. To begin, I utilized Fiddler, a specific tool for HTTPS, to efficiently intercept and capture communication data packages. I then proceeded to analyze the content of these packages. If the data is not encoded or if I can decipher the encoding method, I can modify the data as needed. By replicating the headers and APIs, I can send fake data back to the server.

I successfully hacked a popular game called "Puzzle of Number," which implemented code obfuscation, WebAssembly (wasm) code protection, and client-server synchronization to detect the integrity of data packages. However, I discovered a bug in the game's design. It failed to verify whether a player genuinely won the game or simply sent a legitimate data package to trick the server into believing they won. Exploiting this vulnerability, I crafted a fake legal data package using Postman, bypassing the need to actually win the game while still being recognized as a legitimate winner (refer to Fig 7).

## 4. Methods to Defend in Different Levels of Hacking

This is how I discovered the implementation of five main hacking techniques on traditional Android games, applied to WeChat mini-games. Additionally, I identified a protection measure commonly found in these third-party platform-based games.

### 4.1. Overall Techniques

In general, the primary techniques for modifying data involve manipulating local data and sending fake data packages to the server. Once the fake data package is successfully accepted by the server, the hack is considered complete. Developers should implement protective measures to prevent hackers from generating legitimate fake data packages and sending them to the server. The following protection levels are commonly employed by developers, along with a comparison of their effectiveness when compared to games on the Android platform.

### 4.2. Local Resources Protection

This is a basic protection technique that focuses on securing local files. Developers may employ methods such as setting passwords or utilizing Android permissions to lock the files. More experienced developers might incorporate internal logic to safeguard their sensitive parameters from hacking attempts. However, this level of protection is only effective when both

local files and memory are secured. During this stage, I utilize general tools like CheatDroid to modify sensitive parameters stored in local memory. For games with stronger protection, I utilize decompiling tools like dex2jar to analyze the function logic.

In this regard, WeChat performs less effectively compared to traditional games, as it is relatively easy to decode. Even when approaching game files on Android, they often lack sufficient protection. Once Cheat Engine successfully locates the parameters, the only remaining protection relies on client-server synchronization. The encoding method used for sensitive parameters in WeChat games is often "timing 2" for unknown reasons. It's possible that WeChat has addressed this issue, but it remains relatively simple to locate.

An efficient defense at this level involves using encoding techniques to conceal sensitive parameters. Since local data can be discovered through memory scanning, obfuscation techniques tend to perform poorly. Developers can implement custom encoding methods, such as generating random numbers or utilizing other simple algorithms, to hide the true values from memory scanning. Implementing such protection measures forces hackers to decompile the game package and analyze the logic, thereby increasing the difficulty and creating more obstacles for them.

### 4.3. Network Protection

In many games that can be played offline and update their data as needed, protecting the traffic becomes crucial. One popular and relatively easy method for protecting traffic is using an HTTPS proxy. However, it is also susceptible to modification through the use of fake proxies. In the case of WeChat mini-games, developers are required to use HTTPS, making it efficient for hackers to focus on HTTPS traffic analysis at this layer. HTTPS data packages that lack encryption and obfuscation are straightforward to analyze. Some games employ signature-based filtering to identify and block illegal data, but they can still be vulnerable to attacks.

The most powerful defense against hackers at this level involves using encoded messages or even transmitting data in native language. For example, in a popular game called "God Battle," a variety of encoding methods are employed to prevent clear text transmission (refer to Fig 8). In another highly popular game, "King's Intention," the data is divided into hundreds of data packages and transmitted using the game's native language. This approach serves as a robust defense mechanism against hacking attempts.

### 4.4. Code Obfuscation

Experienced developers may employ class and variable name obfuscation techniques to make it more difficult to track the game's logic. While the Google SDK provides easy access to this in traditional Android games, for WeChat mini-games, further research is required to understand the specific protection methods used and where developers hide their code. Code obfuscation is well-implemented in some particular games. Additionally, certain games have parameters that constantly change, making it challenging to locate their addresses.

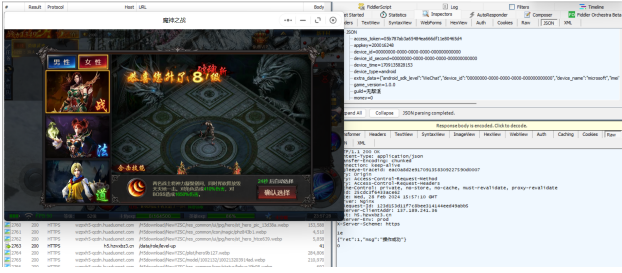


Figure 8: In the game God Battle the data are encoded to prevent clear text transmission.

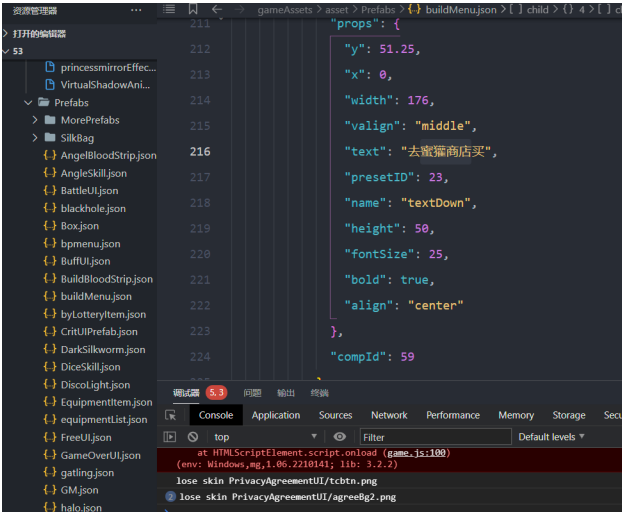


Figure 9: The descriptive message is divided which will be referred many times to make an obfuscation.

In addition to obfuscating parameters and classes, data can also be obfuscated during transfer. Using unconventional names, strange data formats, and unusual APIs can make package analysis extremely difficult after capturing them.

Once a hacker decompiles the game package and analyzes the game logic, they will encounter how developers encode local sensitive parameters, sign data packages, and encode the data itself. One approach is to obfuscate the parameters to make it challenging for hackers to discern the underlying logic. Another method involves dividing descriptive statements into separate files, such as JSON files, as many locations are often identified through searching these descriptive statements. These messages are then imported into other files and referenced multiple times, making it harder for hackers to locate the actual parameters. The game "Lie Flat and Live" serves as a good example of this obfuscation technique (refer to Fig 9).

#### 4.5. Compile into Native Code

Using WebAssembly (wasm) as a native language is an efficient defense method against hackers, although it is slightly less effective than client-server synchronization. Wasm is the only native language I observed among the many games I hacked, primarily due to WeChat regulations that restrict the use of other assembly languages like x86. By using wasm code, game

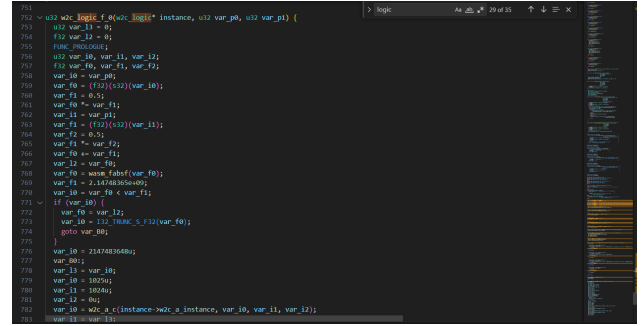


Figure 10: The logic.wasm file of YANG LE GE YANG is debugged but the result is hard to analyze.

logic can be effectively hidden as many details, such as parameter names and classes, are lost when the code is compiled into wasm. Debugging tools are also less effective as the recovered high-level programming language is automatically generated.

To further complicate the analysis process, I utilized the wasm2c tool to convert the .wasm file to .wat and then to C language. However, the resulting C code is challenging to read, and short sections of memory processes are transformed into lengthy C code, making analysis more difficult (refer to Fig 10).

As a result, one efficient approach is to transform the logic code into wasm code using Emscripten. This not only conceals the logic but also improves execution speed since wasm can perform calculations more quickly than JavaScript. Additionally, compressing the wasm code into wasm.br using Brotli compression, as demonstrated in the game "Defence Three Country," can further enhance game performance.

#### 4.6. Client-Server Sync

Implementing a robust client-server synchronization mechanism is the most effective protection against hacking. When implemented well, it significantly reduces the vulnerability of a game. Full client-server synchronization ensures that the game state and important data are validated and controlled by the server, making it extremely difficult for hackers to manipulate or exploit.

In some cases, partial client-server synchronization can still be vulnerable if there are bugs or loopholes in the implementation. For example, in the case of the Puzzle of Number game you mentioned, the server may struggle to distinguish between a legitimate win and a manipulated data sent by the client. However, for games with full client-server synchronization, it becomes nearly impossible to hack the game except by attacking the server itself.

Implementing client-server synchronization requires significant time and resources from developers. However, it is a highly efficient defense mechanism against various hacking techniques. Games like Fat Rabbit Legends, which employ client-server synchronization and ensure that the client's time is synchronized with the server, successfully defend against speed hacks and similar exploits.

#### 4.7. Structure Obfuscation

Protecting sub-packages and obfuscating them is indeed an important defense technique for WeChat mini-games. In some cases, the main package of a game cannot run without the presence of the required sub-packages. By obfuscating these sub-packages, developers can make it more challenging for hackers to understand and manipulate the game's logic.

An example you provided is the game 'Majong Online,' which only had three visible sub-packages despite requiring 18 sub-packages to function properly. It's possible that the additional sub-packages are hidden within compressed files, and the main package may include an unzip tool to extract them during runtime. This technique adds an extra layer of complexity for hackers attempting to analyze and understand the game's logic.

In the game 'Minor God,' sub-packages are imported upon logging into the game and then deleted when the player leaves. This further complicates the decompilation and understanding of the game's logic, making it more challenging for hackers to reverse engineer the game.

For developers, another effective defense technique is to download the source files during the loading process. By doing this, they prevent hackers from running the game on their own computers and gaining unauthorized access to the game's code.

### 5. AI Use in Hacking

Nowadays AI is powerful and it also brings a breakthrough in the process of hacking, or even an influence in the whole android ecosystem.

#### 5.1. AI Use in Native Code

The debugging of native languages using traditional tools like wasm2c can be challenging due to the loss of many messages, as mentioned earlier. However, AI can significantly improve this process. AI has the capability to comprehend how data flows within memory and can rewrite high-level programming languages in a more readable manner. For instance, when AI debugs a logic.wasm file, it can reduce the code to just a few lines, allowing for a perfect recovery of the main logic (see fig 11, 12). AI can also recover functions, recursion, and other elements, thereby reducing the level of protection provided by compiling into native languages. This trend of "AI hacking" may shape the future of hacking methodologies.

Another reason for the influence of AI in the Android ecosystem, particularly in WeChat mini-games, is the limitation imposed on running non-high-level language files such as .so (native language) or .lua. WeChat mini-games primarily allow the execution of WebAssembly, and AI's understanding of WebAssembly proves to be highly effective in this context.

#### 5.2. Restriction of AI Use in Native Code

AI has made significant strides in file analysis, offering a breakthrough in the ecosystem. It can greatly reduce the effort required for debugging native language files. However, current limitations include poor performance with large files (crashing

```
#include <stdio.h>

int a_a();
int a_b(int, int);
int a_c(int, int, int);

int main() {
    int e() {
        // Function e, no parameters and no return value
        return;
    }

    int f(int p0, int p1) {
        // Function f, parameters p0 and p1, returns an integer
        float l2;
        int l3;

        l3 = -2147483648;

        if (((fabs((float)(p0 * 0.5) + (float)(p1 * 0.5))) < 2147483648.0) {
            l2 = (float)(p0 * 0.5) + (float)(p1 * 0.5);
            l3 = (int)l2;
        }

        a_a(a_c(1025, 1024, 0) + l3);
        p0 = a_b(p0, p1);

        return p0;
    }

    int (*g[1])();

    char d[128 * 65536];

    // Exported functions
    g[0] = e;

    // Call function f with parameters
    int result = f(0, 0);
    printf("%d\n", result);

    return 0;
}
```

Figure 11: The AI performs much better than traditional debugging tools.

```
31021 M(T = "logic.wasm") || (A = T, T = n.locateFile ? n.locateFile(A, d) : d + A);
31022 var B = {
31023     1025: function() {
31024         return 42;
31025     }
31026 };
```

Figure 12: Sensitive parameter '1025' can be obtained in native language file and located in game.js (uses code obfuscation here to make parameter hard to locate but still great progress).



```

int main() {
    // Allocate memory
    int32_t* memory = (int32_t*)malloc(10 * sizeof(int32_t));

    // Initialize memory data
    int32_t data[] = {0xFF, 0xFF, 0xFF, 0x1F, 0x19, 0xC6, 0x00, 0x10, 0xA8, 0x0C, 0x00, 0x00,
0x01, 0x20, 0x01, 0x10, 0xE2, 0xC8, 0x00, 0x10, 0xDF, 0xC8, 0x00, 0x10, 0xE0, 0xC8, 0x00, 0x1
0, 0x2F, 0x0C, 0x00, 0x10, 0xCA, 0x45, 0x01, 0x10, 0x6B, 0xCC, 0x00, 0x00, 0xF8, 0x0F, 0x00,
0x00, 0x6A, 0xCC, 0x00, 0x00, 0x5E, 0x7C, 0x00, 0x00, 0x7A, 0xFE, 0x00, 0x00, 0x63, 0x7C, 0x0
0, 0x10, 0x61, 0x7C, 0x00, 0x10, 0x83, 0xFE, 0x00, 0x00, 0x5F, 0x7C, 0x00, 0x10, 0x84, 0xFE,
0x00, 0x10, 0xEC, 0x46, 0x00, 0x10, 0x7D, 0xFE, 0x00, 0x10, 0x80, 0xFE, 0x00, 0x10, 0x7E, 0xF
E, 0x00, 0x10, 0xE8, 0x71, 0x01, 0x10, 0x7E, 0xFE, 0x00, 0x00, 0x76, 0x2D, 0x00, 0x00, 0xAA,
0x05, 0x00, 0x10, 0x12, 0x05, 0x00, 0x10, 0x12, 0x05, 0x00, 0x10, 0x7B, 0xFE, 0x00, 0x00, 0x8
1, 0xFE, 0x00, 0x10, 0x82, 0xFE, 0x00, 0x00, 0x7C, 0xFE, 0x00, 0x00, 0x45, 0x05, 0x00, 0x10
};

```

Figure 13: Hard code the memory address that make AI and hacker hard to understand.

with files over 1MB) and difficulty understanding certain coding methods, like hard coding memory addresses in games such as Defence Three Country (fig 13).

Further development is needed to address these limitations and improve AI’s ability to handle larger files and comprehend complex coding techniques. Ongoing research aims to enhance AI’s file analysis capabilities.

### 5.3. Restriction AI Use in Code Obfuscation

Code obfuscation can indeed pose challenges in understanding the code. The traditional approach of analyzing data in HTTPS transmission and correlating it with the game.js file can be time-consuming and difficult. While AI can be utilized to rewrite the code, it still faces limitations in this aspect. The performance of AI can be hindered by lengthy and intricate code contexts, as game.js files often contain thousands of lines of code with numerous interrelated files.

AI may struggle to effectively analyze and comprehend such complex and unconventional contexts. Some text may require additional information or unusual numbers that can only be understood when sent to the server end (as depicted in fig 14). Even if AI manages to grasp the meaning, there is a risk of losing certain information during the rewriting process, leading to incorrect results and making analysis more challenging.

Despite these limitations, AI holds potential for game hacking when provided with sufficient data and specific focus in this area. Continued advancements in AI research, coupled with access to comprehensive and relevant data, can contribute to overcoming these restrictions and improving AI’s performance in code obfuscation scenarios.

## 6. Data of Game Protection

Due to the protection measures implemented by WeChat, it is challenging to obtain concrete player data and accurately measure the popularity of games. Additionally, since there is no readily available list of mini-games, it becomes difficult to determine the existence and popularity of specific games.

To evaluate the effectiveness of protection methods used in these games, we can consider a range of factors. These main protection methods, labeled as 1 to 5, can serve as indicators for assessing the level of protection employed.

```

function i(n, e) {
    var t, i, a, d, h;
    n[e >> 5] |= 128 << e % 32, n[14 + (e + 64 >>> 9 << 4)] = e;
    var l = 1732584193,
        v = -271733879,
        g = -1732584194,
        m = 271733878;
    for (t = 0; t < n.length; t += 16) i = l, a = v, d = g, h = m, l = o(l, v, g,
m, n[t], 7, -680876936), m = o(m, l, v, g, n[t + 1], 12, -389564586), g = o
(g, m, l, v, n[t + 2], 17, 606105819), v = o(v, g, m, l, n[t + 3], 22,
-1844525330), l = o(l, v, g, m, n[t + 4], 7, -176418897), m = o(m, l, v, g, n
[t + 5], 12, 1200080426), g = o(g, m, l, v, n[t + 6], 17, -1473231341), v = o
(v, g, m, l, n[t + 7], 22, -45705983), l = o(l, v, g, m, n[t + 8], 7,
1770035416), m = o(m, l, v, g, n[t + 9], 12, -1958414417), g = o(g, m, l, v, n

```

Figure 14: In this function, name is hidden and uses meaningless alphabet or numbers to obfuscate this function.

Game	Main Protecting Method
Melon Cut Together	1
Chess Piece Together	1
I Want to Catch Fish	1
Kill the virus	2, 3
Lie Flat and Live	2, 3
Yang Le Ge Yang	3, 4
Wealth of Mr Wang	3, 4
The Sword of Brave	4
Cut on the melon	4
Defend 3 Country	3, 4
Balls Jumping	3, 4
God Battle	2, 5
Puzzle of Number	2, 5
Minor God	3, 4, 5
Majong Online	3, 4, 5
King of Salted Fish	3, 4, 5
Strongest Tower	4, 5
Way on big world	3, 4, 5
Wandering Supermarket	3, 4, 5
Fat Rabbit Legend	3, 5
King’s Intention	3, 4, 5

Table 1: The table show the main methods each game applies to protect. 1=nothing applied or only local file protection, 2=network protection, 3=code obfuscation, 4=compile into native code, 5=client-server sync

## 7. Conclusions and Discussions

### 7.1. Technical Challenges and Limitations

In the whole process of hacking, there are several technical challenges and limitations of current protecting method on third-party platform-based games.

#### 7.1.1. Too Many Codes to Analysis

If a game cannot be hacked solely by modifying local data and communication data packages, analyzing its code becomes necessary. However, this task can be quite time-consuming and challenging due to various factors.

In many games, there are numerous files, often numbering in the hundreds, and each logic file may contain over 10,000 lines of code. Additionally, game developers employ different methods of code obfuscation or have distinct coding habits, further complicating the analysis process.

To successfully hack a game with higher levels of protection, a deeper understanding of the code is required. This presents a significant challenge, particularly for individuals who are not experienced or knowledgeable in the field. Amateurs may find it particularly challenging to grasp the intricacies of complex code and overcome the barriers imposed by advanced protection measures.

It is essential to acknowledge the complexity and time investment involved in analyzing game code, especially when dealing with obfuscated or heavily protected systems.

#### 7.1.2. Environment Protection of Wechat

Indeed, WeChat developer tools provide the capability to run game files locally. However, certain functions may be restricted and require the developer's privileges or a master key to access. This protection mechanism serves as an effective barrier to prevent sensitive parameters from being easily accessed on a hacker's computer.

In many cases, running the game file locally may only allow access to certain pages or limited functionalities, while the core game logic remains inaccessible. If developers have effectively hidden the master key, hackers are left with the option of analyzing the game code meticulously, line by line. They may need to create their own testing environment to evaluate and test decoding functions that can only be executed within the game's library.

This level of protection adds an additional layer of complexity for hackers, as they must invest significant time and effort to decipher the code and replicate the necessary environment to test specific functions.

#### 7.1.3. Lack of Local Resource Protection

If developers wish to protect their local variables, implementing client-server synchronization is crucial. This is because cheat engines have the ability to monitor and modify parameter changes within the game's process on the Windows platform. Unlike traditional Android games, protection methods solely relying on local files may not be effective.

To ensure the integrity of game data, developers often employ client-server synchronization. However, implementing

this synchronization requires significant effort on the part of the developers. They may need to employ techniques such as code obfuscation or compiling into native code to obscure the synchronization process. This adds a layer of complexity and makes it more challenging for hackers to identify and exploit vulnerabilities.

Hacking in such scenarios becomes a difficult and arduous process for the hacker, while developers invest considerable effort in protecting their game against exploits and cheating.

#### 7.1.4. No Perfect Client-Server Sync

Third-party platform-based games, like WeChat, often face certain restrictions. For instance, WeChat imposes a maximum file package size of 2MB, which can limit the implementation of full client-server synchronization, especially within lightweight frameworks. Large resources such as photos typically need to be sent by the server, but they can be vulnerable to capture by hackers since they cannot be effectively protected.

Furthermore, WeChat policies may restrict developers' access to certain information, which can limit their ability to detect and address hacking attempts. This limitation on available information further complicates the detection of hackers.

These challenges highlight the constraints faced by developers when operating within third-party platforms. Striking a balance between resource limitations, data protection, and hacker detection becomes a complex task.

### 7.2. The financial Influence

The financial influence of hacking is very huge since the development of these kinds of Third-party platform based games' market raised dramatically these years. In September 2022, Yang Le Ge Yang suddenly became popular and appeared on Weibo hot searches continuously. The server was crowded three times. The MAU (monthly active users) peaked at 213 million, even surpassing JD.com, Meituan and other lifestyle companies[12]. Which is only one game on Wechat.

#### 7.2.1. The Money lost for companies

Game hacking results in significant financial losses as players prefer cheaper hacked items over legitimate purchases. This undermines game fairness, devalues in-game items, and diminishes developer efforts. These financial implications can discourage high-quality game development and disrupt the gaming ecosystem. Implementing anti-cheat measures is crucial to protect the integrity of the game and restore player trust.

#### 7.2.2. Effect of the Earning Mode

Game hacking can also impact the revenue models of games that rely on advertising. If hacking becomes popular, developers may prioritize incorporating more advertisements into their games while reducing the focus on in-game items. However, this shift can have negative consequences. The increased emphasis on advertising may lead to a decrease in the availability and quality of in-game items, ultimately affecting the overall gaming experience. Additionally, excessive advertising can disrupt gameplay and reduce player satisfaction.

This change in the earning model can have a detrimental effect on the vitality of high-quality game development. Developers may face challenges in striking a balance between generating revenue through advertising and maintaining a positive player experience. Finding sustainable ways to monetize games while preserving the integrity and enjoyment of gameplay is crucial for the long-term success of the gaming industry.

### 7.2.3. The Release of Users' Information

Game hacking can lead to the unauthorized release of user information. Hackers can exploit this data for trading or other malicious purposes, causing financial implications not only within the gaming industry but also potentially affecting other markets. Implementing robust security measures is essential to protect user information and minimize the risks associated with data breaches.

## 8. The Future Trend

WeChat, primarily used in China, offers valuable insights into the process of infrastructuralization [10]. The unique structure of these games requires specific hacking methods that may not be as effective for traditional Android games. Due to their reliance on third-party platforms, their reliability and flexibility may be compromised compared to Android and Apple systems.

In addition to technical aspects, these games represent a future trend in the gaming market. The advancement of AI presents both opportunities and challenges for developers and hackers alike. The market for third-party platform-based games is expected to grow, leading to the emergence of more sophisticated hacking and defense mechanisms tailored to these games. Consequently, future research in this area will be essential.

## References

[1] "Global Mobile Game Market Outlook 2023" - Global mobile game revenue will fall for the first time to 78.8 billion in 2022 The U.S. dollar, still above pre-pandemic levels, is expected to return to its upward path in 2023, <https://sensortower-china.com/zh-CN/blog/mobile-games-market-outlook-2023-report-CN> (accessed Oct. 9, 2023).

[2] "WeChat Mini Games' five-year report card: more than 100 capabilities have been opened in total, and the scale of the developer ecosystem has exceeded 30 Wan," Game View — GameLook.com.cn, <http://www.gamelook.com.cn/2023/06/521343> (accessed Oct. 10, 2023).

[3] Y. Tian, E. Chen, X. Ma, S. Chen, X. Wang, and P. Tague, 'Swords and shields: a study of mobile game hacks and existing defenses', in Proceedings of the 32nd Annual Conference on Computer Security Applications, 2016, pp. 386–397.

[4] Wechat developer information. <https://developers.weixin.qq.com/community/d>

velop/doc/0002ae37438a800cb88a73ef151400. (Accessed on 11/30/2023).

[5] Y. Zhang, Y. Yang, and Z. Lin, 'Don't Leak Your Keys: Understanding, Measuring, and Exploiting the AppSecret Leaks in Mini-Programs', arXiv preprint arXiv:2306.08151, 2023.

[6] Y. Xinhong, S. Hailong, B. Jidong, and L. Fei, 'Construction of large-scale equipment sharing cloud service platform based on Wechat Mini Program: Construction of large-scale equipment, etc', in 2022 The 5th International Conference on Software Engineering and Information Management (ICSIM), 2022, pp. 68–75.

[7] W. Li et al., 'MiniTracker: Large-Scale Sensitive Information Tracking in Mini Apps', IEEE Transactions on Dependable and Secure Computing, pp. 1–17, 2023.

[8] L. Hao, F. Wan, N. Ma, and Y. Wang, 'Analysis of the development of WeChat mini program', in Journal of Physics: Conference Series, 2018, vol. 1087, p. 062040.

[9] A. Cheng, G. Ren, T. Hong, K. Nam, and C. Koo, 'An exploratory analysis of travel-related WeChat mini program usage: affordance theory perspective', in Information and Communication Technologies in Tourism 2019: Proceedings of the International Conference in Nicosia, Cyprus, January 30–February 1, 2019, 2019, pp. 333–343.

[10] R. Zhou and B. DiSalvo, 'User's role in platform infrastructuralization: WeChat as an exemplar', in Proceedings of the 2020 CHI conference on human factors in computing systems, 2020, pp. 1–13.

[11] Faker-Bert (no date) Faker-Bert/pc\_wechat\_miniapp\_des: WeChat PC applet package decryption, GitHub. Available at: [https://github.com/faker-bert/pc\\_wechat\\_miniapp\\_des](https://github.com/faker-bert/pc_wechat_miniapp_des) (Accessed: 13 March 2024).

[12] "1-year Increase of 15 Billion Yuan, Mini Program Games Are Winning." 36Kr, Dec. 26, 2023. [Online]. Available: [36kr.com/p/2577007442503040](https://36kr.com/p/2577007442503040).

[Consolidated Logbook for FYP Thesis II]

Name: CHEN ZHEN

Student ID: 1155157194

Date: Apr 20, 2024

Date: 29/1/2024

Group member(s): Chen Zhen

## Android (game/app) Hacking weekly log I

In the past week, I continue to handle with more games on Wechat for their vulnerable of hacking. Some games such as King's intention which is almost the most popular games recently is very hard to hack. In this semester I will focus on the field of compiling into native language as well as their encryption, since it is one of the most popular and defensive way of protection.

Moreover, I will focus on the client-server sync as well and find more ways to hack. The problem still exists, the client-server sync is very hard to hack once it is implemented well, and no more detail message can be found on internet.



Date: 5/2/2024

Group member(s): Chen Zhen

## Android (game/app) Hacking weekly log II

In the past week, I successfully in getting some natural language logic package and debug them into C language. But the language is weird which is generated by computer but not human. So it will take me a long period of time and great work to find out the logic with such a big file.

In the next week, I will continue to work for the native language logic package and find how the games works for more games and deeper understanding. I believe it will be a great process on the hacking road.

Date: 14/2/2024

Group member(s): Chen Zhen

## Android (game/app) Hacking weekly log III

Past week is the lunar new year where I have continuously making effort on the native language analysis. I have compiled several games logic file and start the analysis form last week. I can successfully understand some of the logic but how the data are processed is a more important topic that I need to know in the hacking period.

Next week I will continue the analysis on native language logic package to gain some more important findings and analysis some data process. Then I will work on hand to process the hacking using the logic that I found in native language pack.

Date: 26/2/2024

Group member(s): Chen Zhen

## Android (game/app) Hacking weekly log III

Past week I have tried to make more effort on hacking more games based on the compiling into native language level. I have tried to compile the logic package into C language and have better understanding. One game that I focus is one of the most famous games "Find the path on the thousand world" which is played by millions of people and put the advertisement on every social media that I used.

Next week I will continue to make more effort and continue to hack more game.

Date: 4/3/2024

Group member(s): Chen Zhen

## Android (game/app) Hacking weekly log V

Compiling into native language is too hard and lose so many information when compiling. Existing tools in the market is hard to debug them well. After debugging I still cannot conclude some efficient way for compiling into native language level. This step I tried to hack more popular game and find some bugs internal of the game logic and make better conclusion. One popular game is called puzzle together which uses a wasm native language but I find some bugs when transfer and successfully hacked it. I will continue to hack more games and make better conclusion.

Date: 11/3/2024

Group member(s): Chen Zhen

## Android (game/app) Hacking weekly log VI

In the past week I tried to hack more games including some top popular game such as 'lie flat and live' and 'fight between the gods'. Some of them uses strong protection which is very hard to hack but some of them is vulnerable to hack once I found the weakness of any one of data transform and local file protection.

In the next step, I will continue to hack more games and study the financial effect of the hacking.



Date: 18/3/2024

Group member(s): Chen Zhen

## Android (game/app) Hacking weekly log VII

In the past week I have made a breakthrough in the process of native language. Although all the debugging tools nowadays perform bad in debugging including wasm2c, AI can perform much better for which it can understand these codes and rewrite them. After hacking more games I found that AI performs bad(not as good as wasm2c) when the file is too large.

Next step I will focus on this new area and discover how AI can help us in hacking.

Date: 25/3/2024

Group member(s): Chen Zhen

## Android (game/app) Hacking weekly log VIII

In the past week I have tried to hack more game and try to conclude more on the financial effect of Android hacking. One popular game area that I should focus is the games which have the paid game items since these games' income may mainly comes from that items so that they should pay more to prevent hacking. Another games are those online battle game since these game should encourage fairness and avoid being hacked.

Next step I plan to make more effort on finding more game and make better conclusion.

Date: 8/4/2024

Group member(s): Chen Zhen

## Android (game/app) Hacking weekly log IX

In the past two week I have tried to make some research on tiktok minigame as well as alipay minigame. These games are implemented using similar method as those in wechat, which is downloading a file and use the programme as a super operating system. So making some research on them can make more progress on the Android ecosystem.

In the next step before the report, I will focus on some game on the alipay and tiktok especially those famous game which implemented on wechat, alipay and tiktok for more exposure. Then the financial effect on the hacking.

**The Chinese University of Hong Kong**  
**Academic Honesty Declaration Statement**

Submission Details

Student Name	CHEN Zhen (s1155157194)		
Year and Term	2023-2024 Term 2		
Course	IERG-4999-IJ01 Final Year Project II		
Assignment Marker	Professor CHOW Sze Ming		
Submitted File Name	final_report.pdf		
Submission Type	Individual		
Assignment Number	2	Due Date (provided by student)	2024-04-20
Submission Reference Number	1000044721	Submission Time	2024-04-20 22:19:38

Agreement and Declaration on Student's Work Submitted to VeriGuide

VeriGuide is intended to help the University to assure that works submitted by students as part of course requirement are original, and that students receive the proper recognition and grades for doing so. The student, in submitting his/her work ("this Work") to VeriGuide, warrants that he/she is the lawful owner of the copyright of this Work. The student hereby grants a worldwide irrevocable non-exclusive perpetual licence in respect of the copyright in this Work to the University. The University will use this Work for the following purposes.

(a) Checking that this Work is original

The University needs to establish with reasonable confidence that this Work is original, before this Work can be marked or graded. For this purpose, VeriGuide will produce comparison reports showing any apparent similarities between this Work and other works, in order to provide data for teachers to decide, in the context of the particular subjects, course and assignment. In addition, the Work may be investigated by AI content detection software to determine originality. However, any such reports that show the author's identity will only be made available to teachers, administrators and relevant committees in the University with a legitimate responsibility for marking, grading, examining, degree and other awards, quality assurance, and where necessary, for student discipline.

(b) Anonymous archive for reference in checking that future works submitted by other students of the University are original

The University will store this Work anonymously in an archive, to serve as one of the bases for comparison with future works submitted by other students of the University, in order to establish that the latter are original. For this purpose, every effort will be made to ensure this Work will be stored in a manner that would not reveal the author's identity, and that in exhibiting any comparison with other work, only relevant sentences/ parts of this Work with apparent similarities will be cited. In order to help the University to achieve anonymity, this Work submitted should not contain any reference to the student's name or identity except in designated places on the front page of this Work (which will allow this information to be removed before archival).

(c) Research and statistical reports

The University will also use the material for research on the methodology of textual comparisons and evaluations, on teaching and learning, and for the compilation of statistical reports. For this purpose, only the anonymously archived material will be used, so that student identity is not revealed.

I confirm that the above submission details are correct. I am submitting the assignment for:


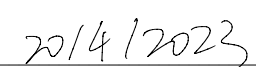
☒ [ X ] an individual project.

I have read the above and in submitting this Work fully agree to all the terms. I declare that: (i) the assignment here submitted is original except for source material explicitly acknowledged; (ii) the piece of work, or a part of the piece of work has not been submitted for more than one purpose (e.g. to satisfy the requirements in two different courses) without declaration; and (iii) the submitted soft copy with details listed in the <Submission Details> is identical to the hard copy(ies), if any, which has(have) been / is(are) going to be submitted. I also acknowledge that I am aware of the University's policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the University website <http://www.cuhk.edu.hk/policy/academichonesty/>.

I declare that I have not distributed/ shared/ copied any teaching materials without the consent of the course teacher(s) to gain unfair academic advantage in the assignment/ course.

I declare that I have read and understood the University's policy on the use of AI for academic work. I confirm that I have complied with the instructions given by my teacher regarding the use of AI tools for this assignment and consent to the use of AI content detection software to review my submission.

I also understand that assignments without a properly signed declaration by the student concerned will not be graded by the teacher(s)

	
Signature (CHEN Zhen, s1155157194)	Date

Instruction for Submitting Hard Copy / Soft Copy of the Assignment

This signed declaration statement should be attached to the hard copy assignment or submission to the course teacher, according to the instructions as stipulated by the course teacher. If you are required to submit your assignment in soft copy only, please print out a copy of this signed declaration statement and hand it in separately to your course teacher.