What WeChat Knows: Pervasive First-Party Tracking in a Billion-User Super-App Ecosystem

Mona Wang
Princeton University
monaw@princeton.edu

Pellaeon Lin
Citizen Lab
pellaeon@citizenlab.ca

Jeffrey Knockel
Citizen Lab / Bowdoin College
jeff@citizenlab.ca

Will Greenberg
Electronic Frontier Foundation
willg@eff.org

Jonathan Mayer
Princeton University
jonathan.mayer@princeton.edu

Prateek Mittal
Princeton University
pmittal@princeton.edu

Abstract

This work studies the analytics and first-party tracking ecosystem of WeChat Mini Programs. WeChat Mini Programs have almost one billion monthly active users, comprising one of the largest application and analytics ecosystems in the world. A key challenge in investigating the privacy of WeChat's Mini Programs is WeChat's use of a proprietary network encryption protocol, MMTLS, to transmit analytics data. First, we reverse-engineer WeChat's network stack, and release tooling and specifications for investigating network requests sent to WeChat servers. Leveraging this tooling, we analyze the requests sent by 104 popular Mini Programs to perform the first characterization and analysis of WeChat's user tracking across their Mini Program ecosystem. Overall, we identified fine-grained browsing data in 76.0% of the network traces we decrypted. This tracking including browsing and search queries performed within third-party Mini Programs, some of which manage particularly sensitive data; for instance, we also identified browsing data in 89.7% of the traces we decrypted from 40 health-related Mini Programs. We ultimately find that the first-party platform, WeChat, is comprehensively tracking user activity with third-party Mini Programs, at an unprecedented scale. There is no way for users nor Mini Program developers to opt-out of this data collection.

1 Introduction

What if Google tracked every webpage you visit in Chrome, or Apple tracked every action you take in iOS apps? What if there wasn't even a way for you, or the web or app developers, to opt out of that tracking? In this work, we find that this type of pervasive and uncontrollable first-party data collection exists in the WeChat super-app ecosystem, which has about a billion users. Third-party apps ("Mini Programs") in the WeChat ecosystem provide health, financial, and government services—and WeChat monitors all of them.

The WeChat Mini Program ecosystem is one of the largest app ecosystems in the world, with 928 million monthly active users [1]. Mini Programs are third-party applications that can be downloaded and launched within the WeChat program. This study analyzes network flows from popular Mini Programs within this ecosystem, through WeChat as a platform.

Due to the ongoing mass infringements of user privacy by opaque mobile application ecosystems, researchers have been thoroughly investigating data privacy and security practices of apps, third-party advertisers, and application platform operators [2–4]. According to their privacy policy, WeChat systematically collects analytics and other logging data from Mini Program usage, but the extent of this data collection is an open question. In addition, there is currently no way to opt-out of this analytics, for users nor for developers [5]. Despite the popularity of WeChat Mini Programs, there has not been a study on exactly what data WeChat collects for this mandatory analytics program, during Mini Program usage.

The largest challenge for privacy researchers studying WeChat is that almost all network communications between the WeChat client application and WeChat servers, including when sending messages, viewing Moments (a feed akin to a Facebook timeline), and even when using WeChat Mini Programs, utilize a proprietary transport protocol called MMTLS (as named by WeChat). The closed-source nature of MMTLS has prevented researchers from conducting a thorough review of information flows from Mini Programs.

In order to analyze MMTLS and perform a comprehensive analysis of data that WeChat collects from its Mini Program ecosystem, we must first reverse-engineer WeChat's network stack. In this work, we first comprehensively reverse-engineer the MMTLS protocol, developing tooling to inspect associated network traces. We then analyze the network requests flowing from the WeChat client to the WeChat server during the operation of popular Mini Programs, leveraging our tooling. We evaluated 104 popular Mini Programs, interacted with them manually to exert browsing, search, and profile update flows when possible. Finally, we analyzed all network traces to check for the presence of sensitive browsing, search, and user data.

We found that 76.0% of the network traces we decrypted contained detailed user browsing data. WeChat collected user browsing data from 89.7% of 40 popular health-related Mini Programs. Our results demonstrate that WeChat is ultimately tracking user activity with third-party Mini Programs, which is unprecedented in the application platform space. As a demonstration of this result, we highlight an example of data sent to WeChat about a user's browsing of a popular shopping Mini Program, which includes the product the user viewed, as well as the user's search query, in Figure 1. In medical or health-based Mini Programs, this includes sensitive information about the users' doctors or other browsing.

Proceedings on Privacy Enhancing Technologies 2025(4), 896–911

https://doi.org/10.56553/popets-2025-0163

```
{ ...
  "pagepath": "pages/item/detail/detail.html",  "referpagepath": "pages/search_pg/index/index.html",
  "query": "ss_pos_type=normal&referer=https://wq.jd.com/wxapp/pages/search_pg/index/index&__navVer=1&search_tab_result=0
&csid=9527036b7fdef1f5619ce3823fa6da80_1706752267069_1_1706752267069&actid=&search_result=2&appCode=msc9ed9e31&ss_item_type=1&__pid=Pxatsdyboel0a
&cover=https://img20.360buyimg.com/n1/jfs/t1/200392/11/34403/24690/6538dd03Fe7c3bb76d48516219c63a15a.jpg!q70.dpg&pps=&ss_tab_type=1&sf=14&pos=1
&price=1675.00&name=Skechers Men's Athletic Casual Shoes - Segment The Search Classic Trendy Comfortable Low-Top Shoes, Dark Brown, Standard Size
41.5 (US 8.5)&sku=10089285173453&factory_goods=0&key=XXXXXX_SEARCH_DATA&navStart=1706752266671",
  "clickTime": 1706752266696,  "reportTime": 1706752266767,
  "anchorTargetRelatedText": "Only 2 Left: Skechers Men's Athletic Casual Shoes - Segment The Search Classic Trendy Comfortable Low-Top Shoes,
Dark Brown, Standard Size 41.5 (US 8.5); Fabric; ¥1675; Interest-Free Installments; Lightning Refund; Free Shipping; HEADED EAGLE Overseas
Specialty Store"}
```


Figure 1: Decrypted and translated JSON data sent to WeChat over MMTLS during operation of Jingxi, a popular shopping Mini Program. Highlighted: a live link to the image of the product loaded, the full name of the product, the search query used to access the product (`XXXXXX_SEARCH_DATA`, our search canary), and the timestamp at which we clicked the product image.

Beyond our primary findings on user activity tracking, we also find that WeChat's network encryption is complex and unusual, especially for a protocol that governs the network security of over one billion users. WeChat's network encryption contains two layers of encryption, one of which differs substantially from public documentation. We found security issues with one of the layers of encryption; while it didn't expose an exploitable vulnerability because of an outer layer of encryption, this highlights that WeChat previously may have been vulnerable prior to the addition of the second encryption layer. There is also the added risk that the insecure inner layer might be used without the secure outer layer in WeChat's internal infrastructure.

In summary, we make the following contributions:

- We performed the first large-scale analysis of WeChat Mini Program's tracking ecosystem across 104 different Mini Programs. We provide conclusive evidence that WeChat is collecting browsing, profile, and behavioral data about Mini Programs that it hosts; a preview of this data can be found in Figure 1. We also provide our dataset of decrypted network traffic sent from popular Mini Programs.
- In this process, we developed tooling for intercepting and decrypting MMTLS traffic and released a detailed specification for MMTLS covering its structure and cryptographic primitives, and make this tooling public at https://github.com/citizenlab/wechat-security-report.
- We found that WeChat's custom-designed encryption protocol contains two layers, and that the inner layer contained multiple severe issues. We reported these issues to Tencent, who acknowledged our findings and committed to fixing them.

We hope to raise awareness of the pervasive nature of WeChat's first-party tracking to developers, users, and privacy researchers. We also hope that our MMTLS tooling and privacy dataset will enable future researchers to study other aspects of WeChat privacy. As MMTLS governs billions of network connections per day, it should be open to scrutiny by public-interest researchers.

2 Background and prior work

In this section, we discuss relevant background and prior work on WeChat's Mini Program ecosystem and WeChat's network stack.

2.1 Mini Program background

WeChat Mini Programs are applications that WeChat users can launch from within the larger WeChat application. A significant portion of WeChat users also use Mini Programs; WeChat Mini Programs had 928 million monthly active users in September 2024 [1]. Mini Programs can also sync and link with users' WeChat account and certain associated data, like their contact information. The breadth and variety of Mini Programs is similar to other application ecosystems, covering e-commerce, health, public services, gaming, etc. Some Mini Programs deal with particularly sensitive data, such as health apps (e.g., Tencent Health), government service apps (e.g., the local contact tracing apps that were compulsory during the COVID-19 pandemic), or apps that perform financial transactions on behalf of the user (e.g., shopping apps like Pinduoduo, financial budgeting apps, or banking apps).

2.1.1 Super-app architecture. The model of embedding platform-locked web applications within another application is becoming increasingly popular, with the host applications generally dubbed as "super-apps," and the "Mini Program" equivalents as "sub-apps". Most commonly, super-apps implement this feature by loading the sub-app code into an embedded browser, or Android WebView instance. Mini Program developers, then, provide HTML-like, CSS-like, and Javascript files to load into the embedded browser.

In the case of WeChat, `WXML` is the HTML-like markup language used to define the page structure, and `WXSS` is the CSS-like language used for styling. The super-app will also inject JavaScript libraries into each page. Many of these libraries act as "bridges" to the underlying WeChat platform. This interface allows the web-based sub-app access to various Operating System-level features, like geolocation, via the JavaScript library bridge. The interface also allows the sub-app to access data associated with the user's super-app account, such as payment information or the user's phone number. This architecture is represented in Figure 2.

To give a concrete example as demonstrated in Figure 2, network requests made by the Mini Program must use the `wx.request` API. When the Mini Program makes a network request to load a 3rd-party resource via `wx.request()`, the WeChat platform performs this request via HTTPS. This way, the platform enforces HTTPS usage across all Mini Programs to avoid unencrypted traffic. However, in addition, we note that WeChat simultaneously collects data about requests made via `wx.request()` and sends networking logs back to WeChat servers via MMTLS. Interestingly enough, though WeChat forces Mini Programs to use HTTPS, WeChat itself uses MMTLS for almost all network requests, including for these requests made during Mini Program usage.

There are other reasons in addition to analytics or logging for which WeChat may make MMTLS requests to WeChat servers.

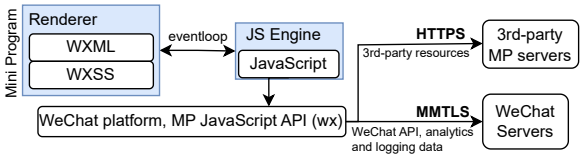

Figure 2: Overview of WeChat Mini Programs (MP) architecture and network requests. This work's examines the data sent via MMTLS to WeChat servers during regular Mini Program usage.

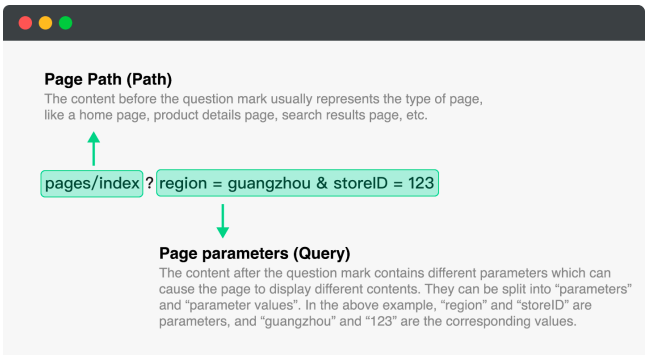

Figure 3: Translated documentation about data provided by WeChat's analytics program.

For instance, if the Mini Program wishes to authenticate the user via WeChat's SSO service, then WeChat makes this request over MMTLS.

2.1.2 WeChat API and WeData Analytics. WeData Analytics is an analytics program that Mini Programs are enrolled in by default. Tencent collects and stores all the analytics data for WeData. From WeData's documentation, this Analytics program claims to collect which "pages" (as in HTML pages) they visit, the query parameters provided to that page, and for how long they stay on each page [6]. We provide a translation of this documentation in Figure 3. Mini Program developers can also register other "analytics events" to track other types of data from user interactions. However, the extent of the default data collection (e.g. what type of data are included in "path" and "query" based on the default architecture of most Mini Programs) is unclear. Most importantly, **there is no way for developers or users to opt out of this program.** This motivates our work to understand exactly what type of behavioral information is included in this tracking program.

2.2 Prior work on Mini Program measurement

In this section, we provide an overview of the rapidly growing field of privacy and security measurements of WeChat Mini Programs and other "super-app" platforms, to situate this work in context. Prior work has focused on studying HTTP(S) data flows to Mini Program developers' or other third-party servers, rather than to first-party WeChat servers.

2.2.1 WeChat Mini Program network encryption. The Mini Program ecosystem has been subject to criticism concerning various privacy and security issues. A 2020 report from CNCERT/CC found that 60 percent of the 50 banking applications that they investigated did not encrypt any user data transmitted over the network, among a litany of other security issues [7]. Since the CNCERT report, WeChat has attempted to tighten user data controls within its Mini Program ecosystem. One such example is that web requests can no longer be directly made from the Mini Program and instead must go through WeChat's internal API. This way, WeChat can enforce which third-parties are contacted during Mini Program usage, and also enforce minimum security for network requests, in particular by ensuring that they are all encrypted using HTTPS.

Our work's focus is on network requests made by Mini Programs to WeChat, rather than network requests made by Mini Programs to servers controlled by the Mini Program developers (or other third-parties).

2.2.2 Super-app measurement studies. The foundation of recent work on WeChat Mini Programs is rooted in large-scale Mini Program crawling work, which constructed large datasets of Mini Programs for static analysis [8]. Our work draws on these established methods to identify popular Mini Programs. We also ran into much of the same restrictions, such as accounts banned for automated behavior, which we further discuss in Section 5.5.

In the past few years, researchers have discovered numerous widespread security issues in super-app ecosystems, including key leakages, various API privilege escalation vectors, CSRF-like attacks, among others [9–13]. Other measurement work performs large-scale static analyses, covering third-party library usage, dark patterns, malware analyses, permission abuse, and other security and privacy aspects [14–17].

A few related works use static taint analysis to study Mini Program data flows and privacy [18, 19]. However, the threat model of this work differs slightly from ours; while TaintMini studies the exfiltration of data to servers owned by the Mini Program developer, our work studies the exfiltration of sensitive Mini Program behavioral data to WeChat; i.e., the default behavioral analytics implemented by the first-party platform.

Another report investigated the third-party tracking ecosystem across 52 commonly used Mini Programs, finding that many shared user data with third parties and generally provided unsatisfactory notice and opt-outs to users [20].

Situated in this context, our work is the first to analyze a "first-party"-enabled mobile app tracking ecosystem where all hosted applications are automatically enrolled into a first-party large-scale analytics platform.

2.2.3 Privacy measurement of other tracking ecosystems. Our work is also situated in a long lineage of privacy measurement studies, and especially those that expose the scale of information collected by tracking libraries and analytics ecosystems. Historically, this field started with measurement of third-party tracking ecosystems across the web, including by third-party trackers, analytics libraries, and session-replay libraries [21–26]. Numerous studies have also documented the tracking ecosystems of mobile apps, including in the Android and iOS ecosystems [27, 28]. Recent work has been targeted at particular audiences, such as tracking on websites and mobile applications targeted at children [29, 30]. Prior work has also

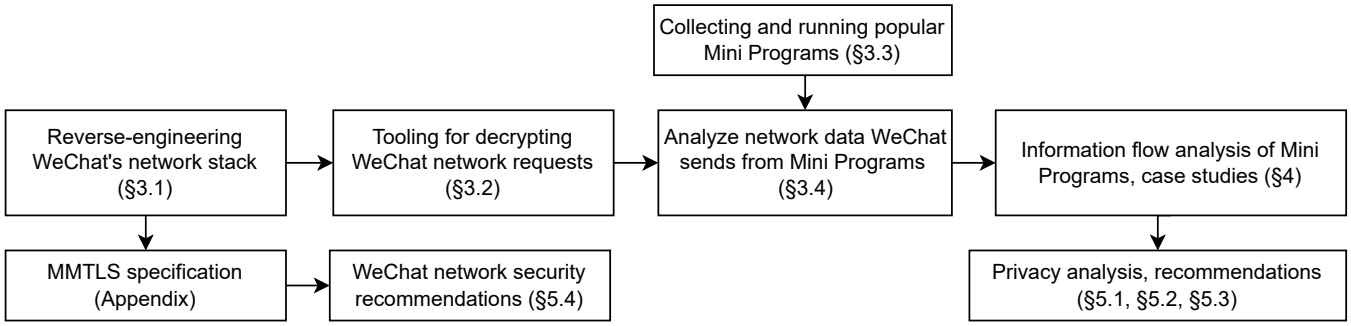

Figure 4: Our analysis framework (§3) and map of paper sections.

Table 1: Summary of the three cryptosystems observed, how they derived keys, and the symmetric encryption used. We also note which native library in which the encryption and logic for that particular cryptosystem is handled.

WeChat cryptosystem	Key derivation	Encryption	Library
MMTLS layer	DH with resumption	AES-GCM with tag	libwechatnetwork.so
Business-layer, logged-out	Static DH	AES-GCM with tag	libwechatmm.so
Business-layer, logged-in	Fixed key from server	AES-CBC with checksum	libMMProtocalJNI.so

analyzed the tracking ecosystems popular TV streaming devices and IoT devices [31–33].

2.3 WeChat's network stack and MMTLS

There is limited prior research on WeChat's proprietary network stack and MMTLS. Wan et al. conducted the most comprehensive study of WeChat transport security in 2015 using standard reverse-engineering techniques [34]. However, this was before the deployment of MMTLS, WeChat's upgraded security protocol. In 2019, Chen et al. study the login process of WeChat, and specifically study packets that are encrypted with TLS and not MMTLS [35]. Besides a blog post published by WeChat comparing MMTLS to TLS 1.3, MMTLS has largely remained a black box and has no prior academic or technical study in either English or Chinese publications [36].

3 Methodology

In this section, we present our methodology for examining the WeChat Mini Program tracking ecosystem. In order to achieve this, we first thoroughly reverse-engineer and develop tooling for decrypting WeChat network requests. Building upon this tooling, we describe the our methods for the privacy analysis of the Mini Program first-party tracking program. First, we evaluated popular Mini Programs from WeChat according to best practices. Then, we systematically interact with them and analyzed the resulting network traces to identify what kind of data WeChat was collecting about user interactions with Mini Programs. The overall analysis flow is also summarized in Figure 4.

Threat model. Our threat model is concerned with WeChat as a platform operator and the data they collect in practice about user interactions with third-party Mini Program applications. We note that our privacy work does **not** cover third-party tracking or requests that are sent to the third-party Mini Program developer. We are primarily considering network data that **WeChat**, the first-party platform operator, collects via MMTLS about third-party Mini Program operation.

In our analysis of WeChat's network security and subsequent report to Tencent, we also consider passive and active network attackers in our threat model. While we consider this output valuable, this is not the primary focus of our study.

3.1 Reverse-engineering WeChat's network stack

Since WeChat relies heavily on several proprietary encryption methods rather than TLS to secure its network traffic, standard techniques for network data analysis, like using standard dynamic instrumentation scripts to override pinned TLS certificates, do not work. Thus, understanding WeChat network requests is a prerequisite to any network data collection needed for studying WeChat network privacy. In this section, we thoroughly describe MMTLS behavior as observed on Android, and confirmed on iOS. We also perform a high-level evaluation and analysis of the encryption methods and compare it to the more standard TLS encryption stack.

We analyzed WeChat app version 8.0.23 and WeChat 8.0.49 (released April 2024) on Android. The account associated with our research was registered to a US phone number. We used both a rooted Pixel 6 device and an emulator, both running Android 14. We also confirmed that the MMTLS network format matches that used by WeChat 8.0.49 for iOS.

In order to fully reverse-engineer WeChat, we used contemporary dynamic analysis methods. For disassembly, we used Jadx, a popular Android decompiler, to decompile WeChat's Android Dex files into Java source code [37, 38]. We also used Ghidra and IDA Pro to analyze the native libraries bundled with WeChat. We used Frida, a dynamic instrumentation toolkit, to hook into the app's

functions and manipulate application memory [39]. We also captured and studied network traffic using Wireshark and `tcpdump` to study the network protocol [40, 41].

Broadly, we start by tracing low-level socket API calls (using Frida) that correspond with requests that we are interested in: e.g., MMTLS network requests observed via Wireshark. From there, we can identify relevant native libraries and functions. We also use static analysis to identify code both in Java and in native libraries which uses cryptographic constants (such as the AES S-box), and observe whether those functions are called consistently before the relevant socket calls. This gives us a starting point for identifying interesting libraries for further static reverse-engineering, as well as identifying codepaths for further analysis with Frida-assisted dynamic analysis.

We reverse-engineered both the WeChat APK's compiled Java code and several native libraries to understand how WeChat represents, processes, and encrypts network requests. Specifically, we reverse-engineered WeChat's networking stack and related utilities, as well as the three native libraries that are used heavily by WeChat for networking and encryption: *libwechatmm.so*, *libMMProtocalJni.so*, and *libwechatnetwork.so*. We describe which layer of encryption is implemented in which library in Table 1.

3.2 MMTLS protocol description and tooling

In this section, we describe our tooling and our high-level documentation of WeChat's network stack and encryption protocols, which we leveraged in order to study privacy in the WeChat Mini Program ecoystem.

3.2.1 MMTLS protocol description. Outside of exceptional cases (e.g., large file downloads), all network messages to WeChat servers are encrypted with MMTLS, including network requests during regular third-party Mini Program operation.

WeChat network requests are encrypted **twice**, with distinctly negotiated sets of keys and different cryptographic primitives. To simplify the explanation, we differentiate between these two cryptosystems as **MMTLS encryption** and **business-layer encryption**. This naming scheme is derived from WeChat's blog post, which describes MMTLS as an encryption protocol designed to replace the original "business-layer" encryption protocol. However, instead of *replacing* the "business-layer", we find that MMTLS *augments* the "business-layer": requests are encrypted first with this "business-layer" encryption, and then with MMTLS.

MMTLS's description in WeChat's blog post mostly matches our findings about **MMTLS encryption**, but the post does not describe **business-layer encryption** [36]. In addition, the type of encryption, authentication, and keys used by **business-layer encryption**, and even the network-layer transport will often change depending on the specific network request type. Various internal services within WeChat seem to exclusively support different types of **business-layer encryption**.

Finally, the keys and encryption used by **business-layer encryption** also change depending on whether the user is logged-in or logged-out. We summarize these details in Table 1, and provide a high-level graph of requests sent during the first MMTLS handshake in Figure 5. Further details of the reverse-engineered protocol can be found in Section A.

3.2.2 Tooling and cryptosystem specifications. We developed tooling to dynamically hook and retrieve keylogs from application memory, during the **MMTLS encryption**, **business-layer encryption** process, regardless of whether the user is logged-in or logged-out. We also developed tooling to decrypt network captures using these keylogs. We found the interception and decryption method more robust than simply exporting plaintexts directly via Frida. This tooling both confirmed our understanding of WeChat's cryptosystems and enabled us to perform a thorough network privacy analysis of the data WeChat collects during Mini Program usage.

Finally, by documenting our findings and open-sourcing our tooling, we hope that MMTLS will no longer be a barrier to comprehensive WeChat research. MMTLS governs billions of network connections per day and should be open to scrutiny by researchers for public benefit. This reusable toolkit will enable future researchers to study other aspects of WeChat privacy, security, and its network traffic in general. Researchers with different capabilities and perspectives on the massive WeChat ecosystem can perform additional privacy and security studies using this work.

3.3 Collecting and running popular Mini Programs

Building upon our reverse engineering efforts, we studied decrypted traffic from WeChat's network requests that were sent during Mini Program operation. We re-iterate from our threat model that our work does **not** cover third-party tracking or requests that are sent to the third-party Mini Program developer. As a concrete example, when we study the Pinduoduo shopping application, we are not studying network data that is sent to Pinduoduo-owned servers or other third-party servers. We are primarily looking at network data that the client sends to **WeChat**, the first-party platform operator, via MMTLS.

From our analyses, WeChat collects a significant amount of user interaction and behavioral data during Mini Program operation. Some data-sharing between WeChat and the third-party app, for instance, linking a WeChat account to a Mini Program, asks for user consent and permission. However, the data flows covered by our analysis, do not.

3.3.1 Collecting popular Mini Programs for analysis. We collected and attempted to analyze 170 different popular Mini Programs [42–44]. Of these, we collected 44 popular health-based Mini Programs as the data stored by those Mini Programs may be particularly sensitive.

WeChat itself does not spotlight or reveal which are the most commonly downloaded Mini Programs, and only provides a ranked search interface for discovering Mini Programs. The current state-of-the-art in Mini Program measurement is to search common Chinese keywords and download the Mini Programs from the search results [8]. We used similar methods to identify and install popular Mini Programs. We collected top Mini Programs rankings from 2021 and 2022 from a marketing firm and other blog posts [42–44]. Finally, for the purposes of investigating sensitive information flows, we collected health-related Mini Programs via the keyword search method. We searched health-related keywords, "health", "doctor", "health management", "fitness", and "period tracking", and collected

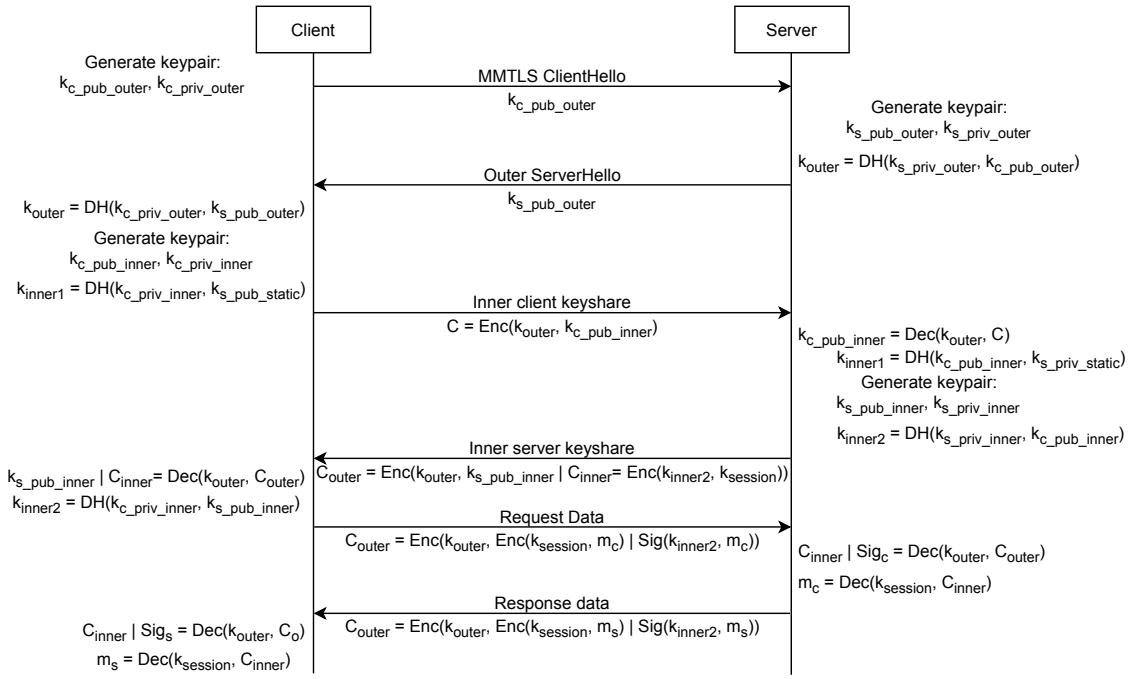

Figure 5: WeChat network encryption handshake flow when logged-in, showing interaction between MMTLS encryption (outer) and business-layer encryption (inner). The MMTLS encryption handshake occurs first, to establish the k_{outer} key, then the key agreement occurs for business-layer encryption. $k_{session}$, the encryption key for business-layer encryption is generated by the server via a process that is unknown to us. $k_{s\_pub\_static}$ is pinned in the application beforehands.

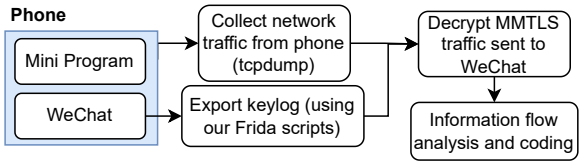

Figure 6: Overview of our Mini Program analysis method.

45 apps in total from the search results. The full list of Health Mini Programs studied is in Table 4.

These Mini Programs all had "categories" assigned from their respective datasets. We combined a few of these categories, such as "big data health" and "health", "online shopping" and "offline retail" into the broader "shopping" category. We also further broke down the large "lifestyle services" category, which we found to be vague, and comprised almost entirely of "ridesharing" (such as Didi and Didi Bikeshare) or "shipping" services (such as ZTO or SF Express).

In total, we attempted to analyze 170 Mini Programs. Unfortunately, 38.8% of them required ID verification or Chinese phone number verification. All "COVID Health Code" Mini Programs, financial service Mini Programs, and other government or public service Mini Programs had such requirements, among others. We also excluded "games" since WeChat Games are implemented differently from Mini Programs. We were able to successfully collect analytics data from 104 Mini Programs; the breakdown of these Mini Programs is summarized in Table 2.

3.3.2 Tooling setup. Our methods and setup for running dynamic analysis scripts are the same as for reverse-engineering WeChat, on a rooted Pixel 6 device running Android 14.

We ran the Frida scripts that can export a keylog (for both **MMTLS encryption** and **business-layer encryption**) used in a particular WeChat session. Finally, we also ran our script that decrypts WeChat PCAP traffic using these keylog files. We then opportunistically decompress (via zlib), expand Protobufs, decode URL-encoded strings, and resolve Unicode character sequences in the decrypted network data we collected from each Mini Program.

3.3.3 Systematically interacting with Mini Programs. Due to the diversity of the Mini Program ecosystem, we sought to develop a rigorous and consistent methodology for interacting with each application, while capturing analytics behavior as comprehensively as possible. After initially reviewing ten Mini Programs from different categories, we noted that there were generally three user flows in common that were accessible in many Mini Programs: (1) entering user profile data, (2) searching, and (3) browsing. We note that not all of the Mini Programs contained all three functionalities, and we provide the number of Mini Programs with each flow type in Table 3.

Flow 1. Updating user profile data. Generally, Mini Programs provide functionality to login by linking your WeChat account, as well as the ability to customize your "user profile" for that Mini Program. When possible, we entered a *canary value* into any of the profile update fields. In most cases, this was an e-mail or name

Table 2: Breakdown of 104 Mini Programs we analyzed, with categories extracted from marketing datasets.

# MPs	High level categories
40	Health / Big data health
24	Online shopping / Offline retail
6	Video
5	Food & drink
5	Tool
4	Ridesharing
3	Content & information
3	Social
15	Miscellaneous

section. In certain cases, such as with Health-related Mini Programs, profile fields comprised more sensitive data such as height, weight, and even menstrual cycle information. Mini Programs that did not contain this flow usually did not allow us to log in with a non-Chinese phone number, or if they did allow login, did not enable updating any profile data fields. We were able to update user profile data in 51 Mini Programs.

Flow 2. Querying the search bar. Almost all the Mini Programs we studied also contained some search functionality. We entered a canary value, `XXXXXX_SEARCH_DATA` into search bars when possible. 85 Mini Programs provided some searching functionality.

Flow 3. Fine-grained browsing. Finally, almost every Mini Program we studied had some notion of browsing, or visiting specific "pages" or pieces of content. In total, we were able to "browse" in 93 different Mini Programs. If a Mini Program did not have a "browsing" flow, it was either minimalist (e.g., containing only a landing page and profile page) or gated content browsing behind Chinese phone number validation. To give concrete examples, this would include visiting a video, product page, wiki page, or the profile of a particular doctor. This portion of the analyses varied the most between applications, and required manual analysis to determine whether the page view for specific content is captured by WeChat.

When interacting with each Mini Program, we tagged pieces of content interacted with, that was not a landing page. For example, this would be the title of a video in a video MP, a doctor's name in a health MP, a product name in a shopping MP, or a particular blog post title in an informational MP. We then aimed to identify the pieces of content viewed from just the network traces.

3.4 Analyzing the decrypted network traces

Finally, we analyzed the decrypted network logs from each Mini Program. We coded the type of data sent from each Mini Program and whether data was collected from each interaction we performed. From a manual analysis of the network traces, sampled at random, we only observed structured (JSON/Protobuf) plaintext; there was no further layer of obfuscation for these decrypted network logs.

We first search for the canary values entered into the application. Next, we manually review the network data to code it for the presence of search results, and the presence of fine-grained browsing data. Examples of these that have appeared in our data and were manually coded are product page names, search result summaries, and video thumbnail image URLs.

Table 3: Data collected by WeChat from Mini Programs (MPs), including across Health and Shopping MPs. We show # MPs that contained a particular interaction flow: updating a profile, searching, or browsing content. We then show % MPs in which those flows were sent to WeChat.

	All MPs	Health	Shopping
Total analyzed	104	40	24
Profile update flows	51	19	12
% sent to WeChat	84.3%	68.4%	83.3%
Search flows	85	30	24
% sent to WeChat	72.9%	76.7%	70.8%
Browsing flows	96	39	22
% sent to WeChat	76.0%	89.7%	77.3%

The following were valid criteria for constituting "fine-grained browsing data" under our coding scheme, if the decrypted network logs included:

- URL containing page contents.
- Image URL, where the image identified the page.
- The page title, where the title identified the page.
- The entire page's contents.

Network traces reporting visits to a `search_results` page would not be considered "fine-grained". In addition, product or page IDs in the URLs alone would not be sufficient alone since we do not know how those IDs are generated or used. We note that our criteria is a lower bound for what would constitute "fine-grained browsing data". For instance, ID strings or other opaque data may identify the particular page a user visited, but since they are not intelligible to us, we would not recognize it as "fine-grained browsing data". Even with this strict definition of browsing data, we find a surprisingly large amount of this data in our results.

4 Results

In this section, we review our high-level results and thoroughly describe three case studies: Pinduoduo, the most popular shopping application and Mini Program in China (who also runs Temu), Baidu Health, a popular health Mini Program, and Bilibili, a popular video and livestreaming site, similar to Youtube or Twitch.

4.1 Results overview

We ultimately find that WeChat regularly sends reports about user interactions with Mini Programs, back to WeChat servers using MMTLS. This is likely a part of their WeData Analytics program, which neither developers nor users can opt-out from [5], and thus is done with questionable user or developer consent. In this section, we illustrate the breadth and depth of this analytics program and the type of data it collects.

WeChat systematically collects network request data about user interactions and usage of Mini Programs. We find that WeChat systematically collects network request data about user interactions and usage of Mini Programs, including the specific interaction flows we studied. Of the 104 Mini Programs we coded, 51 had profile update flows, 85 had search flows, and 96 had browsing flows. 84.3%, 72.9%, and 76.0% of those flows were exfiltrated to WeChat, likely via these analytics endpoints. The summary of our results can be found in Table 3, and a detailed breakdown of flows found per-app can be seen in Section 4.1.

For the health Mini Programs we reviewed, WeChat collected a similar amount, if not more, data about user activity. Of the 39 Health-related Mini Programs we analyzed that contained browsing flows, 89.7% of those Mini Programs sent browsing data to WeChat. Our full set of coded data for health Mini Programs can be found in Table 4.

WeChat sends requests to three different internal APIs during regular Mini Program operation, two of which are used for tracking user behavior. We observed WeChat sending Mini Program data via MMTLS using three different internal Java objects. The first, `JsApiOperateRealtimeReport`, seems to be related to WeChat's Mini Program analytics program. It uses similar terminology that is used by the WeData Analytics platform, which tracks "user events" through their "path" and "query", as demonstrated in the documentation in Figure 3. This API is marked by the internal URI cgi-bin/wxartrappsvr/route. The body of this request contains a JSON object describing an "event", including internal Mini Program URL data such as `pagepath` and `query`. Much of the user behavior tracking came through this API.

Then, we observed `AppBrandIDKeyBatchReport`, which was used to send detailed logs of user behavior, including the timing of specific user interactions with the Mini Program, lists of resources that were loaded by the Mini Program, and other internal JS APIs that were called by the Mini Program. This API used URI cgi-bin/wxausrevent/wxaappidkeybatchreport, and contained a long Protobuf log of user events. The remainder of tracking we identified came through this API.

Finally, we also found requests handled by `JsApiOperateWXData`, which is used to complete user-initiated API requests to WeChat from the Mini Program. The contents of these requests contained serialized JSON data. As an example, if the user authorizes the Mini Program to retrieve the phone number associated with their WeChat account, this WeChat request would be completed via this API. Unlike the previous two types of requests, many of these requests are gated by an explicit permissions/consent dialog presented to the user, or seemed necessary for the functioning of the Mini Program, such as fetching operational metadata through WeChat. We did not observe tracking data sent through this API.

4.2 Case studies

Below, we highlight a few case studies to more concretely demonstrate the analysis we performed, and the type of data that is collected by WeChat. We highlight the data sent by three popular applications in very different categories: Pinduoduo, a shopping application, Baidu Health, a health application, and Bilibili, a video application.

4.2.1 Case study: Pinduoduo. Pinduoduo is the most popular shopping WeChat Mini Program in 2022 according to all the sources we compiled. It is an online shopping application, and we were able to create an account without a Chinese phone number or ID verification via WeChat SSO.

Pinduoduo was tagged with sending `PROFILE` data and `BROWSING` data. Below is a snippet of the JSON payload that contained the `PROFILE` data canary:

```
{"nickname":"XXXXXX_PROFILE_DATA",
 "request_id":"0f0e04f6dc8987"}
```

We also tagged Pinduoduo as sending fine-grained `BROWSING` data. Here is the abbreviated JSON payload containing what we coded as `BROWSING` data:

```
{... "pagepath": "package_c/goods/goods.html",
"networktype": "wifi",
"costtime":111820, "staytime":3982,
"query": "preloadImgUrl=[live image URL]
   ?imageView2/2/q/50/w/400/format/
   webp&goods_id=529197873578 ...
   &_oak_gallery=[live image URL],
   &oak_gallery=[live image URL]}
```

In this example, the `pagepath` parameter collected by WeChat would not have been sufficient to qualify as fine-grained `BROWSING` data. However, the URL query parameters specify both the ID of the product viewed, as well as live image URLs containing item preview images.

4.2.2 Case study: Baidu Health. We also selected Baidu Health as a case study. It was the first non-government result after "Tencent Health" when searching for "health" apps in WeChat's internal Mini Program search. Though we also analyzed Tencent Health (and it was tagged with sending `PROFILE`, `SEARCH`, and `BROWSING` data) we decide to show Baidu Health as a case study since the third-party developer in this case (Baidu) is a different one from WeChat's developer (Tencent). Thus, we can demonstrate that this data is being sent to a different party (Tencent) than the Mini Program's developer (Baidu).

Baidu Health was tagged with `SEARCH` and `BROWSING` data. We were unable to successfully log-in with our WeChat phone number, since it was not a Chinese phone number, so it is unknown whether they send `PROFILE` data. However, we were still able to browse doctors and perform search queries (such as for drugs or health inquiries).

Below is a string within a Protobuf structure containing the `SEARCH` canaries.

```
request,
https://jiankang.baidu.com/wzcui/uiservice/expert/
    list?channel_code=weixin-miniapp
&word=XXXXXX_SEARCH_QUERY
&locCityCode=110100  ...
&filterCityCode=&filterProvinceCode=,
null,,0,null,wifi,0,1000,null
```

Finally, we also coded this application as sending fine-grained `BROWSING` data. For instance, after selecting a doctor's page resulting from the earlier search query, the following is sent to WeChat

Table 4: Data analysis results from popular health Mini Programs. For reproducibility, we kept the original app name in parentheses when there was no official English translation and when the name could be ambiguous.

App name (translated)	PROFILE	SEARCH	BROWSE	BROWSING CODING NOTES
39 Health	-	●	●	Title of post viewed
ABC Health	○	●	●	Live links to the doctor and post viewed
AI Period Helper	●	○	●	Title of post viewed
iFuture Doctor	-	●	●	Title of post viewed
WOW Health	-	-	●	Title of post viewed
Dr Clove	●	●	●	Live link to content
Clove Home	●	●	●	Title of product viewed
3Nuo Health	●	○	●	Title of post viewed
JD Internet Hospital	○	●	●	Live links to the doctor and post viewed
JD Health	-	○	●	Title of product viewed
ZhongAn Health	●	●	●	Live image of insurance post viewed
Health Helper	○	○	●	Title of health study page visited
Health: I Help You	-	-	●	Image of doctor viewed
Easy Health (健康易点)	-	-	●	Title of product viewed
Health Intelligence Management	-	-	●	Live link of help page viewed
Cloud Health Pro	-	-	●	Title of post viewed
Banmi Health	●	●	●	Title of post viewed
Big Ginseng Health	●	●	●	Title of product viewed
Period Calendar (姨妈日历)	●	●	-	
Xiaodong Health	○	○	●	Full doctor details whose profile we viewed
Peace Health	-	●	●	Live links to product viewed
Peace Health Insurance	●	●	●	Live link of insurance page viewed
WeDoctor Health Platform	○	●	●	Live links to doctors viewed
Ask Doctors Quickly (快速问医生)	●	●	○	
Shiguo Health	●	-	●	Title of post viewed
Dr. Chunyu	-	●	●	Live link to hospital page viewed
Doctor is coming (有来医生)	-	●	●	Live link to doctor
Youcong Health	-	-	●	Title of product viewed
Dr Taikang Mini Program	-	-	○	
Haixin Health	○	●	●	Name of product viewed
Love Health (爱康)	-	○	●	Title of post viewed
Baidu Health	-	●	●	Full doctor details whose profile we viewed
Baidu Health Market	-	●	●	Title of product viewed
Every Year Big Health (美年大健康)	-	-	○	
Meiyou APP	-	-	●	Title of particular tips page visited
Tencent Health	●	●	○	
Tencent Medicine Box	-	●	●	Name of product viewed
Tencent Growth Guard	-	●	●	Title of post viewed
Mint Health	-	●	●	Title of product viewed
Gaoji Health Pro	●	○	●	Live image of product viewed

Table key for Tables 4 and 5.
Grid cells that are empty ("-") indicate that the flow was not available on that Mini Program; it was either simply not present or gated behind further I.D. or mobile phone verification. Mini Programs with all three flows unavailable were entirely gated behind mobile phone or I.D. verification.

A filled-circle ● indicates that the related canary (i.e., `XXXXXX_PROFILE_DATA` or `XXXXXX_SEARCH_DATA`) was found in the decrypted network data sent to WeChat, or in the case of `BROWSING` data, ● indicates that content that we marked as having visited was discovered manually in the decrypted network data. An empty-circle ○ indicates that the tester performed that flow, but the canary or content was not present in the decrypted network data. The "BROWSING CODING NOTES" column specifies the type of browsing data that was found.

Table 5: Data analysis results for popular non-health Mini Programs. For reproducibility, we kept the original app name in parentheses when there was no official English translation and when the name could be ambiguous.

App type	App name (translated)	PROFILE	SEARCH	BROWSE	BROWSING CODING NOTES
Banking	Zhongbang Bank	-	○	●	Title of product viewed
Content & information	WeRead	-	●	●	Title of book viewed
Content & information	Sogou Encyclopedia	-	●	●	Entire/full contents of the entry viewed
Content & information	Zhihu	-	●	●	Question viewed
Financial management	Peace Securities (平安证券)	-	●	●	Live link to content viewed
Financial management	Tencent Financial Management	●	○	●	The stock name that the user tried to buy
Food & drink	Luck In Coffee	●	-	●	Title of product viewed
Food & drink	Hey Tea GO	●	●	○	
Food & drink	Starbucks	●	●	●	Title of product viewed
Food & drink	Walmart	-	●	-	
Food & drink	KFC	-	-	○	
Misc	BOSS Recruiting	-	●	●	Title of job listing visited
Misc	Dianping Reviews	-	●	●	Full contents of post viewed.
Misc	Coupon Butler (接龙管家)	●	●	○	
Misc	Soudian	●	-	-	
Misc	Car Home (汽车之家)	-	●	○	
Misc	Maoyan Movies, shows	-	○	●	Title of post/page visited
Misc	Group Coupons (群接龙)	●	○	●	Title of post/page visited
Misc	Beike Househunting	-	●	●	Title of article visited
Misc	Survey Star	●	●	●	Title of questionnaire visited
Online Shopping	Nike	-	●	●	Title of product viewed.
Online Shopping	China Duty Free	●	●	●	Title of product viewed.
Online Shopping	Yunhuo Preferred	●	○	●	Title of product viewed.
Online Shopping	JD Pinpin	●	●	●	Title of product viewed.
Online Shopping	JD Shopping	●	●	●	Title of book viewed
Online Shopping	JD	●	●	●	Title of product viewed.
Online Shopping	Uniqlo	-	●	○	
Online Shopping	Convenience Bee	-	○	○	
Online Shopping	Xingsheng Preferred	○	○	●	Title of product viewed.
Online Shopping	Dingdong Groceries	-	●	●	Title of product viewed.
Online Shopping	VIP Shop	-	●	○	
Online Shopping	VIP Specials	-	●	○	
Online Shopping	Duoduo Groceries	-	●	●	Title of product viewed.
Online Shopping	Duoduo Shop Client	-	●	-	
Online Shopping	Kid King (孩子王)	-	●	●	Title of product viewed.
Online Shopping	Watsons	●	○	○	
Online Shopping	DeWu APP	-	●	●	Title of product viewed.
Online Shopping	Fast Group (快团团)	●	●	-	
Online Shopping	Pinduoduo	●	○	●	Live image URL viewed
Online Shopping	Pinduoduo Coupons	○	●	●	Title of product viewed.
Online Shopping	Pop Mart	●	○	●	Title of product viewed.
Online Shopping	Pop Mart Vending Machine	●	○	●	Title of product viewed.
Online Shopping	Tao Veggie	-	●	●	Title of page viewed.
Online Shopping	Suning Shopping	-	●	●	Title of appliance product viewed.
Ridesharing	Ride Code (乘车码)	●	-	-	
Ridesharing	Hello Travel	●	●	○	
Ridesharing	Didi	-	-	○	
Ridesharing	Car's Here (车来了)	-	○	-	
Shipping	ZTO Express	-	○	○	
Social	RedNote	-	-	○	
Social	Weibo	-	●	●	Title of page/post viewed.
Social	Meipian	●	○	○	
Tool	ETC Helper	●	○	○	
Tool	Guest Hospitality (来客有礼)	●	-	●	Title of product viewed.
Tool	Group Reports (群报数)	●	●	○	
Tool	Tencent Documents	●	●	-	
Travel	Tongcheng Travel	●	●	○	
Travel	CTrip Booking	-	●	●	Name of Hotel viewed.
Video	BiliBili	-	●	●	Title of video viewed
Video	Kuaishou	●	○	○	
Video	Zhufu	-	-	●	Title of video viewed
Video	PiaoQuan Vlog	●	-	●	Title of video viewed
Video	Sweet Bean Video	●	●	○	
Video	Tencent Video	●	●	●	Title of video viewed

(we have anonymized information that could identify the particular doctor):

```
pages/doctor/detail/index.html,
wifi,56,0,pages/doctor/list/index.html,
0,,1706841926282,2.24.7,0,,0,
expert_id=2254418
&doc_id=4rsca1r3qog4t36jheog,1000,2,
"Deputy Chief Physician [name], Department of
    Orthopedics, [location] Hospital, Ranked No. 5
     of [location] Hospitals, 750 consultations
    monthly, average response time within 30
    minutes, 89 yuan for multimedia messaging, 150
     for phone call" (translated)
```

The translated text above is the title of a doctor's page that the researcher tagged as having visited when interacting with Baidu Health. The text contains the doctor's name, their medical department and hospital location, as well as rates for consultation.

4.2.3 Case study: Bilibili. Finally, we selected Bilibili's Mini Program, a popular video application, as a case study. Unfortunately, Bilibili did not support SSO-like sign-in with WeChat accounts, and instead required validation with a Chinese phone number. Our non-Chinese test phone account did not receive the validation code. However, we were still able to browse the application and tagged this Mini Program with SEARCH and fine-grained BROWSING data.

The following is a portion of a string from a Protobuf that was sent over the network, containing our SEARCH data canary:

```
    createRequestTask,
    https://s.search.bilibili.com/main/suggest?
        func=suggest&suggest_type=accurate&
        sub_type=tag&main_ver=v1&highlight=&
        tag_num=10&from_source=xcx_search&term=
        XXXXXX_SEARCH_DATA
```

Finally, the researcher interacted with (i.e., opened the page for) a single video from the home page. In the network log, the following JSON request was sent:

```
{"pagepath": "pages/video/video.html",
"networktype": "wifi",
"referpagepath": "pages/index/index.html",
"query": "__preload_=17070246878623&__key_=1707024
    6878624&avid=1900223929",
"anchorTargetText": "[Title of video viewed]",
"clickTime": 1707024745901, ...}
```

This JSON blob includes the full title of the video visited when interacting with the application, as well as other metadata like the video ID, and the times at which the video was clicked and loaded.

5 Discussion

From our analysis, we find that a massive amount of network request data is sent by default to WeChat across popular applications. In this section, we discuss the privacy implications for users, Mini Program developers, and WeChat, and make recommendations for all three parties.

5.1 Third-party developer consent

One of the core issues at play is whether third-party developers are able to opt their Mini Programs out of WeData Analytics, or whether they are even aware of the baseline level of data collection. There is documentation provided by WeData Analytics, shown earlier in Figure 3. However, it is currently not possible for developers to opt-out or turn off the analytics data collection [5]. This is effectively a mandatory user tracking program.

We do not know whether Mini Program developers are aware of, or have explicitly consented to the analytics features before hosting their programs on WeChat's platform. One could surmise that the use of certain Mini Program analytics features implies consent. However, because this analytics program is mandatory, Mini Program developers do not have to have actively enabled the platform in order for the tracking to take place.

For instance, we studied the analytics data that is collected for an open-source Mini Program that does not call the internal WeChat analytics API at all [45]. The analytics data that is collected includes the HTML path and query of the pages visited, and how long users stay on each page; this level of data collection was the minimum present on all the apps we analyzed. This also corresponds with the most basic level of analytics provided to the WeData Analytics platform that is mandatory by default [6]. Effectively, developers do not have to explicitly call analytics APIs for tracking to occur. For instance, any internal "page/URL" navigation triggers tracking data to be sent to WeChat via MMTLS.

In summary, from our study, all applications are enrolled by default into this analytics program, and there is currently no opt-out [5]. This is especially concerning. To repeat an analogy made at the start of our paper, it would be comparably concerning if Google injected Google Analytics libraries into all apps available on the Play Store, without any way for developers to opt-out of the program.

5.2 Privacy policies

WeChat has a different privacy policy for its users, depending on whether the account is attached to a mainland Chinese mobile number. The WeChat privacy policy refers to accounts with mainland Chinese mobile numbers as "Weixin accounts," (Weixin is the phonetic romanization of WeChat's Chinese name) which are covered under a separate privacy policy [46, 47]. Ultimately, the WeChat (English, international) privacy policy doesn't disclose this analytics program, but the Weixin (China) privacy policy does.

There were specifically inconsistencies within the "WeChat privacy policy," which governs our (international, English-language) test account. Specifically, the English-language privacy policy implies that only third-party privacy policies govern Mini Programs, when in fact, WeChat also collects a lot of data about Mini Program usage. In the English-language policy, Mini Programs are **not** listed as subject to the Chinese (Weixin) privacy policy, and instead listed under "Weixin Open Platform," which are only governed by third-party privacy policies [48].

However, the Chinese (Weixin) privacy policy, is explicitly open about this data collection, though it does not mention the use of this data for an analytics program [47]:

> ...Weixin will collect and use logs from Mini Program login, browsing, usage, addition/removal from "My Mini Programs", floating window management (adding/removing/closing windows), and other operations.

With the academic and policy privacy consensus moving away from *notice and consent* towards *data minimization* (including a prior Mini Program privacy framework [49]), we also pose whether this data collection is necessary [50]. In fact, data minimization is one of the core principles of data processing per the EU's GDPR [51]. From the above disclosure, it is not at all clear whether the data collection is "adequate, relevant, and limited to what is necessary in relation to the purposes for which they are processed," as stated in GDPR [51]. The requests we analyzed that contained tracking data did not explicitly ask for user consent, nor did they seem necessary for the functioning of the Mini Program. Finally, though the data collection itself is disclosed by the above privacy policy, the purpose of the data collection is not transparent, as there is no mention of any analytics program.

From reviewing the privacy policies of our case study Mini Programs (i.e., Pinduoduo, Baidu Health, and Bilibili), they also make no mention that Mini Program user data and browsing data is provided to WeChat [52, 53].

5.3 Privacy recommendations

First, we recommend that WeChat remove forced enrollment of Mini Program analysis and tracking features, and change to an opt-in model. Currently, both developers and users are automatically enrolled into the analytics tracking program with little notification and no way to opt-out [5]. Next, we recommend WeChat disclose the extent of their data collection of Mini Program user behavior in the WeChat Privacy Policy (and not just in the Weixin Privacy Policy). We also recommend WeChat allow users to opt-out of analytics tracking during Mini Program usage, as they should be able to opt-out of tracking that is not necessary to the function of the app.

5.3.1 Developer recommendations. Although developers cannot wholly opt-out of this program, the default tracking is based on URL structure, which developers can control. We recommend that developers design Mini Programs to use URLs that do not encode app usage or PII in the URL or URL query parameters, which can significantly limit default data collection.

5.3.2 User recommendations. We recommend users use regular applications/websites over Mini Programs (MPs) when possible. Additionally, since services covered by the WeChat privacy policy performed less tracking than services covered by the Weixin privacy policy, we recommend avoiding services that are covered by the Weixin privacy policy, and when possible, sticking to services such as Chat and Moments which are under the stricter WeChat privacy Policy.

5.4 WeChat disclosure and network security recommendations

In their post from 2016, WeChat developers note that they wished to upgrade their encryption, but the addition of another round-trip for the TLS 1.2 handshake would significantly degrade WeChat network performance, as the application relies on many short bursts of communication. At that time, TLS 1.3 was not yet an RFC (though session resumption extensions were available for TLS 1.2), so they opted to "roll their own" and incorporate TLS 1.3's session resumption model into MMTLS.

This issue of performing an extra round-trip for a handshake has been a perennial issue for application developers. The TCP and TLS handshake each require a single round-trip, meaning each new data packet sent requires two round-trips. Today, TLS-over-QUIC combines the transport-layer and encryption-layer handshakes, requiring only a single handshake. QUIC provides the best of both worlds, both strong, forward-secret encryption, and halving the number of round-trips needed for secure communication. **Our recommendation would be for WeChat to migrate to a standard QUIC implementation.**

There is also the issue of client-side performance, in addition to network performance. Since WeChat's encryption scheme performs two layers of encryption per request, the client is performing double the work to encrypt data than if it used a single, standardized cryptosystem.

5.4.1 Responsible disclosure. Our team responsibly reported our full MMTLS security findings in 2024 to Tencent, WeChat's parent company. We provided them our recommendations to fix their security issues while reducing complexity, and maintaining performance, primarily by switching to a standard web protocol like QUIC/HTTP3.

While they did not commit to migrating to QUIC or TLS1.3, Tencent acknowledged and confirmed our findings, responded that they "conducted a careful evaluation", and that to mitigate the reported issues, they intended to keep the inner layer of encryption, but migrating away from AES-CBC and towards a full AES-GCM implementation. This may indicate that **MMTLS encryption** is removed at front servers and that within WeChat's data centers, **business-layer encryption** may be the only layer of encryption used, motivating Tencent's decision to improve it. This is also not a desirable outcome, as the **business-layer encryption** properties were markedly worse than **MMTLS encryption**.

5.4.2 Ethics of publication. Tencent did not elect to switch to TLS. As such, there is therefore some risk to publishing our work reverse-engineering MMTLS, and there may yet be further issues we did not discover. However, as MMTLS governs billions of users' network traffic, we believe it deserves the same public scrutiny as TLS, which operates on a similar order of magnitude. We make our documentation and tooling available in the interest of furthering WeChat privacy research, as users and researchers alike have the right to understand what data is being sent from their own devices.

5.5 Limitations

We encountered various ethical and practical limitations during this work. First, we considered the risk of cutting off individuals' access to the WeChat network unacceptable [54]. Unlike typical app measurement studies, we were limited by the number of WeChat accounts we could access. Accounts are aggressively monitored for automation (ours have been banned for automation before),

and creating new accounts is difficult (requiring active accounts to vouch). Other WeChat measurement studies have also noted that WeChat's spam detection policies are opaque and particularly stringent, such that reverse-engineering behavior, including running the application on an emulator or employing other dynamic analysis techniques, can lead to account suspension [8]. As such, though we is inherently adversarial and more resource-constrained than ordinary app measurement. We detail related limitations in our results below.

5.5.1 Using a non-Chinese account. It has become increasingly difficult to obtain Chinese phone numbers for research since Chinese phone number registration requires government ID verification. We purchased Canadian and American phone numbers, which resulted in various restrictions and limitations to our study. First, there is evidence that WeChat behaves differently when registered to a Chinese phone number [55], which may bias our results. For instance, it has been generally observed that international versions of some applications tend to perform less tracking than the domestic versions [56]. In addition, without a mainland Chinese account or ID verification, the types of interaction with certain features and Mini Programs were limited. For instance, we were not able to perform financial transactions.

5.5.2 Restrictions on automation. Our team faced difficulties registering non-Chinese accounts with our strict risk posture. Registering new WeChat accounts requires "attestation" from another sufficiently active WeChat account. We were able to obtain an additional account for research by attesting with an old, unused account belonging to an inactive phone number. One of the accounts was later frozen due to "suspicious activity," likely from the use of an emulator and our tests involving automating account interactions. We were unable to successfully register further accounts without attesting from an active account. As a result, we decided against further automation; thus, all data collected in this study is derived from our manual interactions with the applications. This also means our study poses little risk of abusing networked services.

5.5.3 Completeness of manual testing. We did not evaluate the completeness of our manual evaluations, and in many cases, were not able to create a valid account, or otherwise log into the application. These limitation biases results towards false negatives, and our measurements thereby reflect a baseline of WeChat analytics' data collection.

5.6 Future work

By providing tooling and documentation of WeChat's network stack, we hope it will enable researchers to perform follow-up work related to WeChat that may cover our limitations, as detailed in Section 5.5. For instance, we did not study the privacy of financial transactions, a core feature of the WeChat ecosystem. Finally, although we focus on popular Mini Programs in this work, additional long-tail measurements in this ecosystem would be valuable.

5.6.1 Other super-app ecosystems. WeChat is also far from the only super-app ecosystem. As the super-app and sub-app model becomes more popular, it is imperative that they are subject to further privacy and security investigations as well. Though WeChat's Mini Program ecosystem is the largest super-app ecosystem, as prior work has shown, other super-app ecosystems are becoming increasingly popular and influential [10, 13, 14]. We similarly suspect this type of default invasive tracking is a concerning privacy trend across multiple popular ecosystems. For instance, both Baidu's Smart Program ecosystem [57] and Alipay's Mini Program ecosystem [58] advertise similar default tracking features. Studying these tracking ecosystems would be a strong candidate for future work, and we would make similar recommendations to super-app developers and users across all of these ecosystems.

6 Conclusion

This work is the first in-depth analysis of Mini Program ecosystem data collected by WeChat, demonstrating that the platform comprehensively tracks users on third-party Mini Programs. We also develop and release tooling, documentation, and collected traffic dataset for future researchers to continue studying WeChat network traffic. The problematic information flows that we discovered reveal the need for researchers to identify and address the unique issues facing the understudied Chinese app ecosystem.

Acknowledgments

The authors would like to thank Jedidiah Crandall, Jakub Dalek, Ron Deibert, and Masashi Crete-Nishihata for their review of this work, and Sarah Scheffler for her additional review of the cryptographic constructions that we reverse-engineered. This work was supported by an Open Technology Fund (OTF) Information Controls Fellowship (ICFP). Funding for this project was also provided by the Citizen Lab at the University of Toronto's Munk School of Global Affairs & Public Policy.

References

[1] Statista, "China: MAU of major mini-program platforms in China 2023," https://www.statista.com/statistics/1298897/china-number-of-leading-mini-programs-monthly-active-users/.

[2] X. Liu, J. Liu, S. Zhu, W. Wang, and X. Zhang, "Privacy risk analysis and mitigation of analytics libraries in the android ecosystem," *IEEE Transactions on Mobile Computing*, vol. 19, no. 5, pp. 1184–1199, 2020.

[3] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill, "Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem," in *Network and Distributed System Security Symposium*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:43993007

[4] S. Egelman, "Taking responsibility for someone else's code: Studying the privacy behaviors of mobile apps at scale," in *2020 USENIX Conference on Privacy Engineering Practice and Respect (PEPR 20)*, 2020.

[5] "How do i turn off We Analytics?" https://developers.weixin.qq.com/community/develop/doc/00060ac21587802cf25210af16b800, Nov. 2024.

[6] WeChat, "WeChat developer documents / Mini Programs / Data Platform / Page Analysis," https://developers.weixin.qq.com/miniprogram/analysis/wedata/data/page.html, Apr. 2022.

[7] CNCERT/CC, "China's internet in 2020: Network security report," https://www.cert.org.cn/publish/main/upload/File/2020%20Annual%20Report.pdf, 2021.

[8] Y. Zhang, B. Turkistani, A. Y. Yang, C. Zuo, and Z. Lin, "A measurement study of Wechat Mini-Apps," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 2, jun 2021. [Online]. Available: https://doi.org/10.1145/3460081

[9] Z. Zhang, Q. Hou, L. Ying, W. Diao, Y. Gu, R. Li, S. Guo, and H. Duan, "Minicat: Understanding and detecting cross-page request forgery vulnerabilities in mini-programs," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 525–539. [Online]. Available: https://doi.org/10.1145/3658644.3670294

[10] Y. Zhang, Y. Yang, and Z. Lin, "Don't leak your keys: Understanding, measuring, and exploiting the appsecret leaks in mini-programs," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '23.

New York, NY, USA: Association for Computing Machinery, 2023, p. 2411–2425. [Online]. Available: https://doi.org/10.1145/3576915.3616591

[11] Y. Yang, Y. Zhang, and Z. Lin, "Cross miniapp request forgery: Root causes, attacks, and vulnerability detection," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 3079–3092. [Online]. Available: https://doi.org/10.1145/3548606.3560597

[12] L. Zhang, Z. Zhang, A. Liu, Y. Cao, X. Zhang, Y. Chen, Y. Zhang, G. Yang, and M. Yang, "Identity confusion in WebView-based mobile app-in-app ecosystems," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 1597–1613. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/zhang-lei

[13] Y. Yang, C. Wang, Y. Zhang, and Z. Lin, "Sok: Decoding the super app enigma: The security mechanisms, threats, and trade-offs in os-alike apps," 2023.

[14] Y. Yang, Y. Zhang, and Z. Lin, "Understanding miniapp malware: Identification, dissection, and characterization," in *Network and Distributed System Security Symposium*, feb 2025.

[15] Y. Wang, M. Fan, H. Zhou, H. Wang, W. Jin, J. Li, W. Chen, S. Li, Y. Zhang, D. Han, and T. Liu, "Minichecker: Detecting data privacy risk of abusive permission request behavior in mini-programs," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1667–1679. [Online]. Available: https://doi.org/10.1145/3691620.3695534

[16] J. Tao, J. Shi, M. Fan, Y. Wang, J. Liu, and T. Liu, "Jslibd: Reliable and heuristic detection of third-party libraries in miniapps," in *Proceedings of the 2023 ACM Workshop on Secure and Trustworthy Superapps*, ser. SaTS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 11–16. [Online]. Available: https://doi.org/10.1145/3605762.3624428

[17] M. Long, Y. Xu, J. Wu, Q. Ou, and Y. Nan, "Understanding dark ui patterns in the mobile ecosystem: A case study of apps in china," in *Proceedings of the 2023 ACM Workshop on Secure and Trustworthy Superapps*, ser. SaTS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 33–40. [Online]. Available: https://doi.org/10.1145/3605762.3624431

[18] S. Meng, L. Wang, S. Wang, K. Wang, X. Xiao, G. Bai, and H. Wang, "Wemint:tainting sensitive data leaks in WeChat Mini-Programs," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2023, pp. 1403–1415.

[19] C. Wang, R. Ko, Y. Zhang, Y. Yang, and Z. Lin, "Taintmini: Detecting flow of sensitive data in Mini-Programs with static taint analysis," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 932–944.

[20] K. David, "China's 'Mini Apps' have big privacy issues, report says," https://www.sixthtone.com/news/1006196, Sep. 2020.

[21] J. R. Mayer and J. C. Mitchell, "Third-party web tracking: Policy and technology," in *2012 IEEE Symposium on Security and Privacy*, 2012, pp. 413–427.

[22] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and defending against Third-Party tracking on the web," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX Association, Apr. 2012, pp. 155–168. [Online]. Available: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/roesner

[23] A. Lerner, A. K. Simpson, T. Kohno, and F. Roesner, "Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/lerner

[24] T. Libert, "Exposing the invisible web: An analysis of third-party http requests on 1 million websites," *International Journal of Communication*, vol. 9, no. 0, 2015. [Online]. Available: https://ijoc.org/index.php/ijoc/article/view/3646

[25] G. Acar, S. Englehardt, and A. Narayanan, "No boundaries: data exfiltration by third parties embedded on web pages," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, pp. 220–238, 10 2020.

[26] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1388–1401. [Online]. Available: https://doi.org/10.1145/2976749.2978313

[27] R. Binns, U. Lyngs, M. Van Kleek, J. Zhao, T. Libert, and N. Shadbolt, "Third party tracking in the mobile ecosystem," in *Proceedings of the 10th ACM Conference on Web Science*, ser. WebSci '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 23–31. [Online]. Available: https://doi.org/10.1145/3201064.3201089

[28] K. Kollnig, A. Shuba, R. Binns, M. Van Kleek, and N. Shadbolt, "Are iphones really better for privacy? a comparative study of ios and android apps," *Proceedings on Privacy Enhancing Technologies*, vol. 2022, pp. 6–24, 04 2022.

[29] Z. Moti, A. Senol, H. Bostani, F. Z. Borgesius, V. Moonsamy, A. Mathur, and G. Acar, "Targeted and troublesome: Tracking and advertising on children's websites," in *2024 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2024, pp. 121–121. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00118

[30] I. Reyes, P. Wijesekera, J. Reardon, A. Elazari, A. Razaghpanah, N. Vallina-Rodriguez, and S. Egelman, ""won't somebody think of the children?" examining coppa compliance at scale," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, pp. 63–83, 06 2018.

[31] H. Mohajeri Moghaddam, G. Acar, B. Burgess, A. Mathur, D. Y. Huang, N. Feamster, E. W. Felten, P. Mittal, and A. Narayanan, "Watching you watch: The tracking ecosystem of over-the-top tv streaming devices," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 131–147. [Online]. Available: https://doi.org/10.1145/3319535.3354198

[32] D. Y. Huang, N. Apthorpe, F. Li, G. Acar, and N. Feamster, "Iot inspector: Crowdsourcing labeled network traffic from smart home devices at scale," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 4, no. 2, jun 2020. [Online]. Available: https://doi.org/10.1145/3397333

[33] Z. B. Celik, L. Babun, A. K. Sikder, H. Aksu, G. Tan, P. McDaniel, and A. S. Uluagac, "Sensitive information tracking in commodity IoT," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1687–1704. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/celik

[34] Y. Wan, Y. Gu, and W. Qiu, "Research on interactive protocol and encryption mode of WeChat," *Microcomputer Applications*, vol. 31, no. 2, pp. 31–35, 2015.

[35] W. Chen, H. Lu, M. Li, and Y. Sun, "Network protocol analysis base on WeChat PC version," in *Artificial Intelligence and Security*, X. Sun, Z. Pan, and E. Bertino, Eds. Cham: Springer International Publishing, 2019, pp. 288–297.

[36] WeMobileDev, "Introducing WeChat's secure communication protocol mmtls based on TLS1.3," https://github.com/WeMobileDev/article/blob/master/%E5%9F%BA%E4%BA%8ETLS1.3%E7%9A%84%E5%BE%AE%E4%BF%A1%E5%AE%89%E5%85%A8%E9%80%9A%E4%BF%A1%E5%8D%8F%E8%AE%AEmmtls%E4%BB%8B%E7%BB%8D.md, May 2016.

[37] A. Desnos and G. Gueguen, "Android: From reversing to decompilation," *Proc. of Black Hat Abu Dhabi*, vol. 1, pp. 1–24, 2011.

[38] A. Albakri, H. Fatima, M. Mohammed, A. Ahmed, A. Ali, A. Ali, and N. M. Elzein, "Survey on reverse-engineering tools for Android mobile devices," *Mathematical Problems in Engineering*, vol. 2022, pp. 1–7, 2022.

[39] A. Druffel and K. Heid, "Davinci: Android app analysis beyond frida via dynamic system call instrumentation," in *Applied Cryptography and Network Security Workshops: ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, October 19–22, 2020, Proceedings 18*. Springer, 2020, pp. 473–489.

[40] A. Possemato and Y. Fratantonio, "Towards HTTPS everywhere on Android: We are not there yet," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 343–360. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/possemato

[41] K. Liu, M. Yang, Z. Ling, H. Yan, Y. Zhang, X. Fu, and W. Zhao, "On manually reverse engineering communication protocols of Linux-based IoT systems," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6815–6827, 2021.

[42] WeChatMiniProgramObservationNetwork, "WeChat Mini Program ranking list (2021 latest rankings)," http://www.mpgcw.com/5872.html, Nov. 2021.

[43] AladdinIndex, "Top 100 Mini Programs 2021," http://www.mpgcw.com/7603.html, Nov. 2020.

[44] ——, "Top 100 Mini Programs march-april 2022," May 2022.

[45] wechat miniprogram, "Wechat mini program example," https://github.com/wechat-miniprogram/miniprogram-demo.

[46] WeChat, "WeChat privacy policy," https://www.wechat.com/en/privacy_policy.html, Jun. 2023.

[47] ——, "Weixin privacy protection guidelines," https://weixin.qq.com/cgi-bin/readtemplate?lang=zh_CN&t=weixin_agreement&s=privacy&head=true, Nov. 2023.

[48] ——, "List of features operated by Weixin," https://www.wechat.com/tpl/oversea/new/page/weixin_features/index?lang=en, Sep. 2022.

[49] S. Wang, Y. Zhao, K. Wang, and H. Wang, "On the usage-scenario-based data minimization in mini programs," in *Proceedings of the 2023 ACM Workshop on Secure and Trustworthy Superapps*, ser. SaTS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 29–32. [Online]. Available: https://doi.org/10.1145/3605762.3624435

[50] S. Geoghegan. Data minimization: Limiting the scope of permissible data uses to protect consumers. Electronic Privacy Information Center (EPIC). [Online]. Available: https://epic.org/data-minimization-limiting-the-scope-of-permissible-data-uses-to-protect-consumers/

[51] European Parliament and Council of the European Union. Article 5 of the GDPR. [Online]. Available: https://gdpr-info.eu/art-5-gdpr/

[52] Pinduoduo, "Pinduoduo privacy policy," https://www.pinduoduo.com/pdd_privacy_policy.pdf, Aug. 2023.

[53] Baidu, "Baidu health mini program privacy policy," https://muzhi.baidu.com/dcwap/drprivacyright, Nov. 2023.

[54] Z. Yang, "The dark side of a super app like WeChat," https://www.technologyreview.com/2022/10/18/1061899/dark-side-super-app-wechat/, Oct. 2022.

[55] J. Knockel, C. Parsons, L. Ruan, R. Xiong, J. Crandall, and R. Deibert, "We Chat, They Watch: How International Users Unwittingly Build up WeChat's Chinese Censorship Apparatus," University of Toronto, Tech. Rep. Citizen Lab Research Report No. 127, May 2020. [Online]. Available: https://citizenlab.ca/2020/05/we-chat-they-watch/
[56] P. Lin, "TikTok vs Douyin: A security and privacy analysis," University of Toronto, Tech. Rep., Mar. 2021. [Online]. Available: https://citizenlab.ca/2021/03/tiktok-vs-douyin-security-privacy-analysis/
[57] Baidu Smart Program Documentation, "Baidu Smart Program Documentation: Data Analytics," https://smartprogram.baidu.com/docs/data/concept/, May 2019.
[58] AliPay Mini Program Docs, "AliPay Mini Program Docs: Analytics," https://miniprogram.alipay.com/docs-alipaymo/miniprogram_alipaymo/platform/analytics, Apr. 2021.
[59] E. Rescorla, "RFC 8446: The transport layer security (TLS) protocol version 1.3," https://datatracker.ietf.org/doc/html/rfc8446/, Aug. 2018.

A Appendix: MMTLS Protocol Description

A.1 Network transports

MMTLS requests can occur over one of two network transports, dubbed `shortlink` and `longlink` in the text of WeChat's logs. For the `longlink` transport, the client opens a long-lived TCP connection and sends MMTLS packets directly over the TCP connection. We observed the TCP port used by `longlink` to be either 443 and 8080. Certain WeChat requests prefer to be sent over `longlink`. For instance, when the user sends a WeChat message, a `longlink` transport message is sent. The TCP connection used for `longlink` messages is generally quite long-lived, opened on client startup and kept alive using periodic heartbeat messages containing no-op data. In their blogpost, WeChat developers note that `longlink` connectivity is not guaranteed as some middleboxes will drop TCP packets containing non-standard payloads [36].

The fallback, and the default for the bulk of WeChat data, is the `shortlink` transport. MMTLS data are sent in the body section of HTTP POST request/response. The HTTP path identifies the MMTLS session, since there may be more than one simultaneous MMTLS session. The `shortlink` transport connection closes after the client receives a response.

Since `shortlink` itself only supports a single round-trip of data (HTTP POST request and response), this necessitates the use of either a fixed encryption key, static Diffie-Hellman, or some form of "session resumption" for all data encrypted and sent in the body of `shortlink`, since a standard Diffie-Hellman handshake would take one additional round-trip. We observe "session resumption" used in the **MMTLS encryption** layer, static Diffie-Hellman used in **business-layer encryption** when the user is logged-out, and a per-user fixed static key when the user is logged-in.

A.2 MMTLS encryption

Generally, the **MMTLS encryption** layer is modeled heavily after TLS and uses similar structures, primitives, and terminology, such as Records and Extensions [59]. We use the same terms of art.

Handshake. The **MMTLS encryption** layer, like TLS, performs an elliptic-curve Diffie-Hellman handshake to establish shared key material with the server. The app sends a `ClientHello` containing the client's public key and `ClientRandom`. In response, the server sends a `ServerHello` containing the server's public key share and `ServerRandom`, then the server's certificate and a resumption ticket, both encrypted with a key derived from the shared secret.

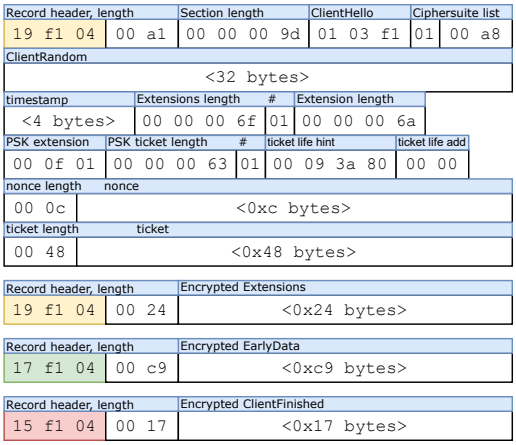

Figure 7: Annotated sample of a MMTLS encryption Session Resumption ClientHello packet, as collected by `tcpdump`. This packet was in the Request body of a POST request, so it was over the `shortlink` transport. The "Encrypted Earlydata" contains the encrypted business-layer encryption payload.

Once the shared secret is derived, further Data Records can be sent via TCP if the connection is over `longlink`. Future `shortlink` connections will use the resumption ticket, and use the same shared secret to derive future keys via HKDF, a key derivation function. We note that the initial shared secrets established by `longlink` and `shortlink` handshakes differ from each other.

When the application is opened, the client first simultaneously sends a standard `ClientHello` packet over both the `shortlink` and `longlink`. The first connection to complete the handshake is then used to perform the first WeChat request. If the user is logged in, this includes establishing key material for **business-layer encryption**.

Encryption. The **MMTLS encryption** layer encrypts data and calculates a message authentication code (MAC) using AES-GCM. Keys and IVs are derived via HKDF from the shared secret for a particular session. The IV is incremented by the number of Records previously encrypted with the same key.

A.3 Business-layer encryption

WeChat requests are generally double-encrypted. This section describes **business-layer encryption**. We note that the key material and logic used in this section is entirely separate from that used in the **MMTLS encryption** layer.

In our testing, **business-layer encryption** was consistent between all logged-in requests and between all logged-out requests, so we chose this dichotomy to present our findings.

Internal network request metadata. Internally, each WeChat network request is referred to as a `Scene`. Each `Scene` defines an internal `cgi-bin` URL and a unique `request_type` integer. For instance, the first request the app makes when logged-in is named the `AutoAuth Scene`, with internal URL /cgi-bin/micromsg-bin/autoauth and a `request_type` of `721`.

Logged-out. When logged-out, the **business-layer encryption** essentially uses a key generated from static Diffie-Hellman

to encrypt all requests. The Diffie-Hellman key is generated from a static server key and a fresh client key each time, which means there is no key management on the client-side. The app is shipped with a *static server public key* and generates a new client keypair for each new request. Then, the client generates the *shared secret* from the server public key and their newly generated private key. The shared secret is then used to extract a key and IV via HKDF. Encryption is performed via AES-GCM. The ciphertext and tag are sent alongside the generated client public key so the server can also calculate the shared secret and decrypt data.

The server responds with a newly generated public key as well as a ciphertext. The client then decrypts the server response with a new shared secret, `ecdh_secret` generated from their private key and the public key provided by the server.

Logged-in. If the user is logged-in, the cryptosystem is different after the first request. The first outgoing request is the same as in the logged-out case. It is encrypted with a key derived from static Diffie-Hellman. The server also sends a response, and a new public key. The client then generates a new `ecdh_secret` from this data. However, on decrypting the server response, the client *also* receives a `session_key` that is sent by the server. This provided `session_key` is then used for all future encryption and decryption, and there is no more Diffie-Hellman ratcheting. The aforementioned `ecdh_secret` is only used to generate a checksum for future requests (described in the next paragraph), and not used for encryption.

When logged-in, the **business-layer encryption** uses AES-CBC with PKCS7 padding. The `session_key` is used as both the key and IV. **Business-layer encryption** also generates and appends a checksum. Below, U is the user's WeChat ID, k_{ecdh} is `ecdh_secret`, p is the plaintext, and l_p is its length:

$$S = \text{adler32}(\text{md5}(U \mid k_{ecdh} \mid l_p) \mid p)$$

Within WeChat's code, this checksum is referred to as a `Signature`, calculated by a function named `genSignature`. However, the checksum is forgeable and has no cryptographic properties.