National Test Institute for Cybersecurity

Technical Security Analysis
Mobile App «Temu»

Security Risk Assessment from a Swiss Perspective

Version	1.0
Date	05 December 2024
Classification	Public
Authors	Tobias Castagna, Patrik Fabian, Andreas Leisibach, Dilip Many, Fabio Zuber
Responsible	Tobias Castagna

Table of contents

1 Management Summary 3
1.1 Initial Situation and Background 3
1.2 Assessment Summary 4

2 Scope and Limitations of the Security Analysis 6
2.1 Overview of the Scope of the Analysis 6
2.2 Scope of the Analysis in Detail 7

3 List of Findings 9
3.1 High Risk 9
3.2 Medium Risk 9
3.3 Low Risk 9
3.4 Relativisations 10

4 Findings in Detail 11
4.1 High Risk 11
4.1.1 Creation of Java classes through a proprietary JavaScript interpreter 11
4.1.2 Encrypted data on the system is being transmitted 16
4.1.3 Exact location can be requested 21
4.2 Medium Risk 23
4.2.1 Custom DNS implementation 23
4.2.2 Location tracking via WebView 25
4.2.3 Transmission of encrypted data with device details 27
4.3 Low Risk 30
4.3.1 Query of running processes 30
4.3.2 Multi-factor authentication is not enforced 33
4.3.3 Encrypted configuration files 35
4.3.4 Detection of elevated privileges and debugging 38
4.4 Relativisations 40
4.4.1 Read out and transmit logs of other apps 40
4.4.2 Read out and transmit system files 42
4.4.3 Screenshots of other apps 44
4.4.4 Storage of MAC addresses 46
4.4.5 Re-compiling of programme packages 47
4.4.6 Android app permissions 48

5 Test cases 49
5.1 Network communication 49
5.2 Privacy and Data Protection 50
5.3 Security of mobile apps 52

References 53

1 Management Summary

1.1 Initial Situation and Background

E-Commerce has gained immense popularity among Swiss residents, with an increasing number of consumers transitioning significant portions of their purchases to online platforms. Traditionally, purchases were predominantly made from suppliers within Switzerland or nearby countries. However, online retailers from the Far East, such as Temu, AliExpress, and Shein, are rapidly rising in prominence. Notably, Temu – a subsidiary of Chinese conglomerate PDD Holdings Inc. – has cemented its position in the Swiss e-commerce market. In October 2024 its mobile app ranked among the top downloads in both the Google Play and Apple App Stores.

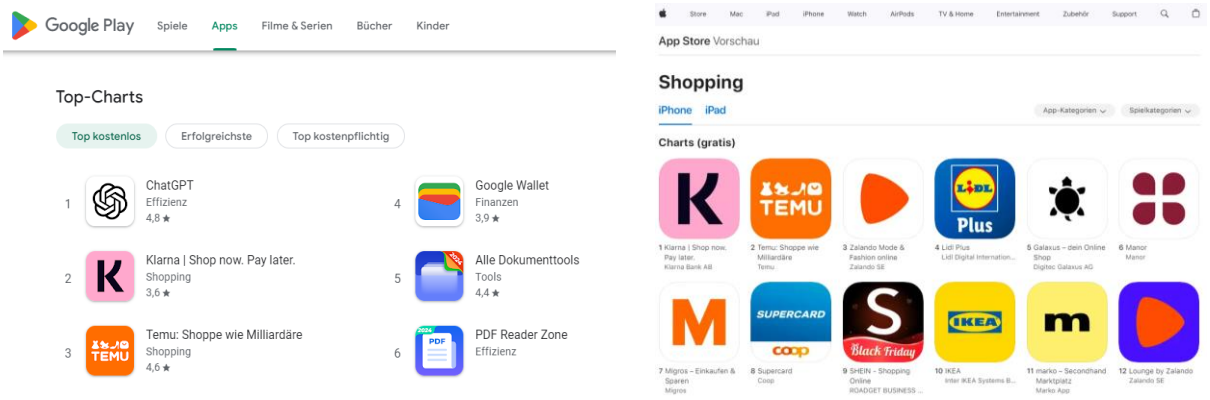

Figure 1: Google Play and Apple Store, as of 31 October 2024

Not only the number of downloads, but also the number of deliveries show that the Temu app is being actively used in Switzerland. Proof of this is the flood of parcels arriving at Zurich airport from the Far East. Hundreds of thousands of packages pass through the airport every day, most of them from vendors such as Temu and Shein [1].

Recently, cybersecurity concerns have been raised about Temu in particular: Several organisations and authorities around the world have made allegations that the Temu app spies on its users or exhibits malware-like behaviour [2]. However, these allegations are often unproven or largely based on a widely cited publication by Grizzly Research LLC [3]. The US-based investment firm is known for its short selling-reports. It has an economic interest in falling share prices and is therefore not neutral [4]. Such reports are often based on speculative information or information that is difficult to verify.

Simultaneously, certificates touting the app's high level of security are being circulated online. However, some of these certificates were commissioned by Temu itself, look rather superficial and without much technical substance [5].

Recognizing problematic behavior in mobile applications is challenging for most users, as it demands specific technical expertise and skills. This poses a significant concern, particularly since nearly everyone in Switzerland owns an internet-enabled mobile phone, which they carry with them almost constantly. These devices are equipped with sensors capable of enabling precise and continuous monitoring. Malicious applications could exploit these features, for example by recording location data to create detailed movement profiles [6]. The microphone and camera can be used to record confidential conversations or collect information about the environment.

An objective discussion about the risks associated with mobile applications requires a robust and independent technical security analysis of these apps.

The aim of this report is to verify or refute the alleged risks of the Temu app and to uncover previously unknown risks. Based on the findings, recommendations for improved security while using the the Temu app are developed. Various analysis methods such as dynamic behaviour analysis, network traffic analysis and reverse engineering are used.

This analysis has been carried out by the NTC on its own initiative, using its own resources and therefore without external influence. It focuses exclusively on aspects related to cyber security and user surveillance. It does not cover other aspects often criticised in the context of Temu. These include the quality and safety of products and certain problematic business practices. Examples are the avoidance of VAT [7] or the lack of contribution to disposal, and recycling costs through the advance recycling contribution [8]. These issues are outside the competence of the NTC.

1.2 Assessment Summary

The analysis did not identify any clearly critical vulnerabilities or evidence of user surveillance in the Temu app. In general, the behaviour observed, and the permissions requested, are in line with what can be expected from a typical e-commerce application. Compared to other popular applications from competitors such as Amazon [9], AliExpress [10], or Shein [11], the Temu app [12] asks for fewer and less problematic permissions.

An examination of the app's access to device sensors that are particularly useful for the surveillance of user activities, such as the microphone, camera and GPS, revealed that the app has no permissions for most of them. Access to them is prevented by the operating system. For sensors where permissions were granted, such as location data, the app accessed the data only in response to clear and understandable user interactions. No background processes were identified accessing these sensors without the user's knowledge.

Despite the lack of direct evidence of malicious activity, certain unusual behaviors were observed. These 'red flags' should be viewed critically. They have not been mentioned in any of the previous reports.

The application has the ability to dynamically load JavaScript-like code. This allows the behaviour of the application to be changed at runtime. This gives developers the flexibility to change features and content without having to update the original application package through the App Store. It allows the application to adapt its behaviour to specific conditions or environments. Therefore, it is difficult to predict or test for every possible behaviour. The ability to dynamically reload code is not in itself a particularly unusual feature – it is used by other applications as well. What is notable, however, is the use of an unknown, proprietary JavaScript runtime environment that has not been observed in any other application.

Also critical is the use of additional layers of encryption in several places. These supplement the usual encryption, such as TLS for communication with backend servers. Such measures can be legitimate and help increase the protection of user data from unauthorised access. However, there is a risk that encryption could be used to hide undesired data transmissions. During the analysis, it was not always possible to break this encryption and analyse the data transmitted. Therefore, it is not possible to conclusively determine what data is processed by the application and sent to Temu's servers.

Considering the fact that only the presence of vulnerabilities can be proven, but not their absence, no blanket declaration of harmlessness can be given. For example, the surveillance of users by the app would be technically feasible. The application could already contain hidden surveillance features that are only triggered under certain conditions (e.g. at certain locations or at certain times). In addition, due to frequent updates, hidden functions could be added almost unnoticed. However, this is also true of many other apps. In the case of Temu, no specific indications of such behaviours could be observed.

It should also be noted that Temu and its parent company PDD are subject to Chinese law. From a European perspective, this does not guarantee adequate data protection [13]. The standards in China are very different from those in the EU and Switzerland, where the rights of users are protected by laws such as the GDPR. Government agencies in China have easier access to personal data, and companies are often required to disclose data to government agencies.

Overall, the results of the analysis show that the risks described in the much-cited Grizzly report [3] are often exaggerated, placed in the wrong context or technically incorrect (see relativization in *Chapter 4.4.* of this document). It paints a much more negative picture than is objectively justified from NTC's point of view.

In summary, it is recommended to critically question the use of the Temu app. This is especially true in a business or government context or for particularly exposed individuals. In principle, this recommendation also applies to other applications whose benefit in a business or government context is limited. If such applications are nevertheless used, technical and organisational measures should be taken. These include granting only those permissions that are absolutely necessary [14], using an up-to-date operating system, and restricting the use of the application to the minimum necessary. Alternatively, the service could be accessed through the browser of the mobile device. In this case, the attack surface is smaller and there are fewer opportunities for permanent surveillance of the users.

2 Scope and Limitations

This section describes the scope of the security analysis performed. The self-imposed, technical, and resource-related restrictions are also discussed. This is followed by an overview of the most important points and a detailed explanation.

2.1 Overview of the Scope of the Analysis

The review focused on risks to individual privacy, surveillance, and espionage. More detailed analyses, such as long-term behavioural tracking, have not been carried out.

The test cases considered in the analysis are listed in *Chapter 5.* The following list describes the scope of the analysis:

- Communication between mobile apps and the Temu backend
- Requested app permissions and access to sensors such as camera, microphone and GPS
- Decompiled source code of the mobile apps (reverse engineering)

The following areas and aspects were deliberately **NOT** analysed in this report:

- Temu's public website and other unlisted platforms
- The effectiveness of the protection features offered by the operating systems, in particular the compliance with the permissions of an application
- Possible disclosure of (personal) data collected by PDD to third parties, including the Chinese state
- The safety and quality of the products that can be purchased through Temu
- Psychological factors, dark patterns, etc., designed to induce users to order products from Temu

The following diagram shows a schematic overview of all the components that are part of this security analysis.

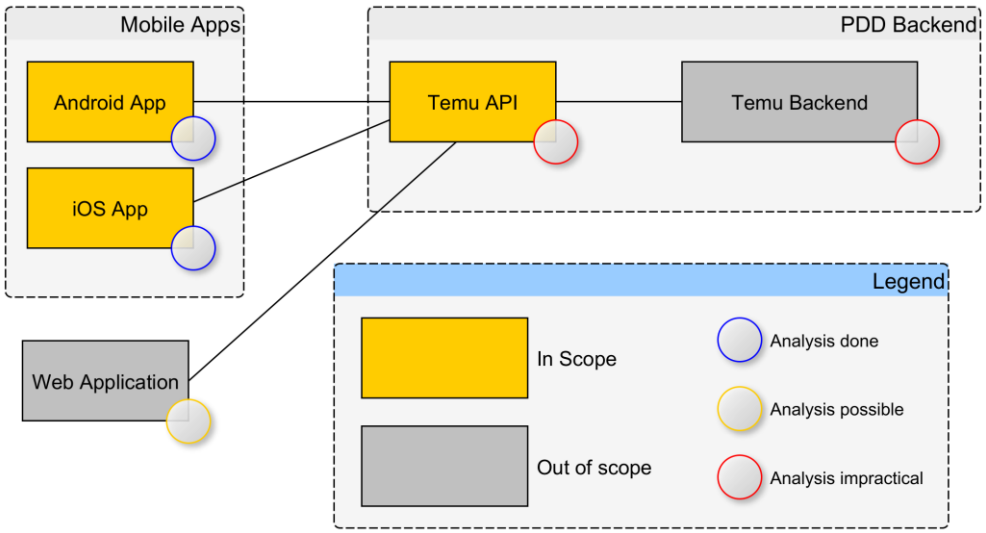

Figure 2: Overview of the components of Temu

2.2 Scope of the Analysis in Detail

The security analysis was carried out at the initiative of the NTC. The NTC provided the resources and defined the objectives, scope and context of the analysis. There is no external entity that commissioned or paid for the analysis. Temu and the parent company PDD had no knowledge of the analysis and were therefore unable to influence it.

The analysis took place between September and November 2024 and was mainly carried out by a core team of three test experts from the NTC. In total, around 60 working days were spent on research, analysis and documentation. During the analysis, care was taken to ensure that the test conditions were as realistic as possible without special protective measures, such as those provided by restrictively configured mobile device management (MDM) solution or hardened end devices.

The available time and resources were focused on analyzing critical and practical risks. Further observations of long-term behaviour over weeks, months or years were not carried out. Analyses under special conditions or at sensitive locations, such as the Swiss Parliament building (Bundeshaus Bern) or military sites, were also not carried out. Such further investigations could indicate possible advanced monitoring techniques, if these are available.

In general, it is not known what happens to the transmitted data at PDD. Without the full cooperation of the provider, a review is not possible. Therefore, no statement can be made about how the data and metadata are further processed after transmission to PDD. The physical locations of the backend servers were not analysed in detail either, as this is not a decisive criterion for making a statement about who the data is ultimately sent to or who has access to the data. However, it is relevant that the data processor, PDD, is a company subject to Chinese law.

The effectiveness of the protection features provided by the operating systems, particularly the enforcement of an application's permissions, was not tested. The evaluation assumes that these are fully effective as documented by the respective manufacturers.

The analysis focused on risks to individual privacy as well as surveillance and espionage when using the Temu app on Android and iOS mobile devices. The Temu website and other platforms were not taken into account.

In order to allow a complete analysis of network traffic, all network traffic was transferred over WLAN. The devices were not equipped with a SIM card and could not communicate with a cellular network.

Rooted or 'jailbroken' Android and iOS mobile devices were used to provide an in-depth insight into how the apps work. The table shows which devices were used for the security analysis:

Model	Operating System Version
Samsung Galaxy A13 EU	Android 13
Google Pixel 7a	Android 14
Google Pixel 8a	Android 14
Apple iPhone 8	iOS 16.6
Apple iPhone 8	iOS 16.7.10

The analysis was carried out on the following versions of the Temu app:

- Android versions: `2.95.0, 2.97.0, 2.99.0`
- iOS version: `2.97.0`

The apps were installed in Switzerland via the official app store of the respective operating system:

- Android: *https://play.google.com/store/apps/details?id=com.einnovation.temu*
- iOS: *https://apps.apple.com/ch/app/temu-shoppe-wie-milliard%C3%A4re/id1641486558*

It is important to emphasize that this analysis represents a snapshot in time. Any changes made to the app before or after the analysis are not reflected. The same applies to any versions of the app used in other countries or language regions, or obtained from other sources.

3 List of Findings

In the following sections, all findings are listed and grouped into one of four categories: High Risk, Medium Risk, Low Risk and Relativisations. All findings are discussed in detail in *Chapter 4*.

3.1 High Risk

Findings in this category pose a high risk for the user, depending on the user and the circumstances. It is recommended to analyse the individual risk as quickly as possible and, if necessary, take appropriate measures to reduce the risk to an acceptable level.

Creation of Java classes through a proprietary JavaScript interpreter 11
Encrypted data on the system is being transmitted 16
Exact location can be requested 21

3.2 Medium Risk

Findings in this category pose a medium risk for the user, depending on the user and the circumstances. It is recommended to analyse the individual risk and, if necessary, to take appropriate measures to reduce the risk to an acceptable level.

Custom DNS implementation *23*
Location tracking via WebView *25*
Transmission of encrypted data with device details *27*

3.3 Low Risk

Findings in this category pose a low risk to users depending on the user and circumstances. It is recommended to assess the individual risk and, if necessary, to take appropriate measures to reduce the risk to an acceptable level. However, the risks of the other categories should be considered first.

Query of running processes *30*
Multi-factor authentication is not enforced *33*
Encrypted configuration files *35*
Detection of elevated privileges and debugging *38*

3.4 Relativisations

The findings listed in this category relativise findings from other reports on Temu. In the view of the NTC, they present either little or no risk to users.

Read out and transmit logs of other apps 40
Read out and transmit system files 42
Screenshots of other apps 44
Storage of MAC addresses 46
Re-compiling of programme packages 47
Android app permissions 48

4 Findings in Detail

4.1 High Risk

Findings in this category pose a high risk for the user, depending on the user and the circumstances. It is recommended to analyse the individual risk as quickly as possible and, if necessary, to take appropriate measures to reduce the risk to an acceptable level.

4.1.1 Creation of Java classes through a proprietary JavaScript interpreter

The Temu app incorporates a proprietary interpreter that enables the interpretation and execution of JavaScript code at runtime. This allows the developer to deliver custom code packages to the app via its servers at any time, enabling the dynamic customization and expansion of the app's functionality. This bypasses potential checks of app stores of the respective platforms.

4.1.1.1 Background

The decompiled source code of the app contains a runtime environment that receives, interprets and executes JavaScript code from a backend. The runtime environment is also able to generate classes in other programming languages (Java for Android, Swift for iOS) from the code.

This allows the developer to change the behaviour dynamically. Code can therefore potentially be transmitted selectively to certain end users or in certain environments.

A possible motivating factor for this feature is that it allows Temu to quickly deploy changes or patches without a complex and time-consuming update of the app through app stores. However, code that is loaded during runtime is more difficult to test, as one cannot predict what code the app will execute for other users at any given time. This makes it more difficult to identify bugs or vulnerabilities in advance.

Temu's self-developed JavaScript runtime environment provides developers with greater flexibility in tailoring supported features. This proprietary runtime can reduce the range of features to the essentials compared to existing JavaScript interpreters, which optimises performance. At the same time, interactions between JavaScript and underlying system interfaces, such as access to sensors, can be made more efficient. However, the rationale behind the decision to forgo a standard runtime environment remains unclear based on this analysis.

4.1.1.2 Evidence

The Temu mobile app is loading `.lego` files, that contain JavaScript or JavaScript-like source code. For example, it loads a file from `https://static.kwcdn.com/fs-lg/lg/CameraPreNotifyPopup--a5f6b6f1c879d09bcb4c531b4882eea7.lego` that uses the React JavaScript library. Figure 3 shows a part of the file for illustration.

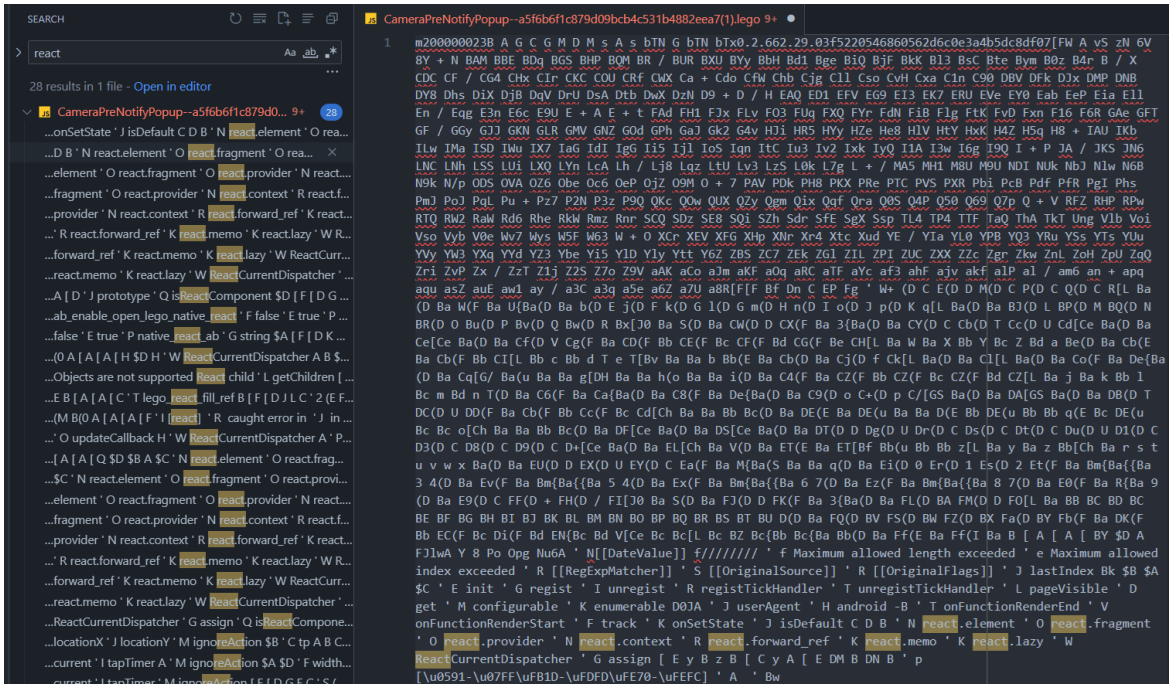

Figure 3: Example of a .lego file

On Android, the content of the `.lego` file is processed in the class `com.einnovation.whaleco.lego.m2.impl.m2.core.M2FunctionManager` and, among others, passed to the `createCustomElement` function:

```
      case M2FunctionNumber.Op_CREATECUSTOMELEMENT /* 956 */:
                  M2Lib.createCustomElement(dVar, legoContext);
                  break;
```

Within the `createCustomElement` function a class object is returned, by calling `class.ForName(String className)`[1]. The name of the class that is returned originates from the Lego file content. This allows the app to create any objects of Java classes and call static methods of the classes via reflection.

```
public static void createCustomElement(o01.d dVar, LegoContext legoContext) {
    String classtoBeCreated = M2FunctionManager.lego_object(0, dVar).getString();
    TValue lego_object = M2FunctionManager.lego_object(1, dVar);
    k b13 = j.b(dVar.C(), true, legoContext);
    if (lego_object != null) {
        b13.h(legoContext, lego_object);
    }
    Node node = new Node(classtoBeCreated, b13, legoContext);
```

[1] Oracle reference for Class.forName: https://docs.oracle.com/javase/6/docs/api/java/lang/Class.html#forName%28java.lang.String%29

```
    if (legoContext.getCustomComponent(classtoBeCreated) == null) {
        try {
            Class<?> cls = Class.forName(classtoBeCreated);
            legoContext.registerCustomComponent(string, (BaseCustomComponent.a)
cls.getDeclaredMethod("createComponentBuilder", new Class[0]).invoke(cls, new
Object[0]));
            z01.c.c(TAG, "createCustomElement %s, register ok", string);
        } catch (Exception e13) {
            z01.c.i(TAG, "createCustomElement %s, register fail, please check the class
exist and proguard-keeped", string);
            legoContext.getLegoErrorTracker().track(legoContext,
legoContext.getContext(), 1002, vx1.d.b(Locale.US, "createCustomElement %s, register
fail, please check the class exist and proguard-keeped", string), e13);
        }
    }
    M2FunctionManager.lego_return(node, dVar);
}
```

Whether the classes are also instantiated at a later point in time could not be determined in the time available. However, the resulting risk remains unchanged; code can be received and interpreted dynamically at runtime.

The strings of the Lego code that are processed by the Android app can be shown with the following Frida script. The strings can help to get an overview of the functionality of the transmitted code.

```
Java.perform(function () {
    var TValue = Java.use('com.einnovation.whaleco.lego.m2.impl.m2.core.TValue');

    var lastToStringValue = "";
    var lastGetStringValue = "";

    TValue.toString.overload().implementation = function () {
        var returnValue = TValue.toString.call(this);
        if (lastToStringValue != returnValue && lastGetStringValue != returnValue) {
            console.log("TValue.toString() = " + returnValue);
            lastToStringValue = returnValue;
        }
        return returnValue;
    }

    TValue.getString.overload().implementation = function () {
        var returnValue = TValue.getString.call(this);
        if (lastGetStringValue != returnValue) {
            console.log("TValue.getString() = " + returnValue);
            lastGetStringValue = returnValue;
        }
        return returnValue;
    }

});
```

The result of this script:

```
└─$ frida -U -l lego-string-analysis.js -f com.einnovation.temu

TValue.getString() = https://commimg-us.kwcdn.com/upload_commimg/security/b14ca8fc-c8f4-
4907-9563-567660590b55.png.slim.png
TValue.getString() = Shop safely together
TValue.getString() = #15740E
TValue.getString() = https://commimg-us.kwcdn.com/upload_commimg/security/b14ca8fc-c8f4-
4907-9563-567660590b55.png.slim.png
```

```
TValue.getString() = Shop safely together
TValue.getString() = #FFFFFF
TValue.getString() = Shop safely together
TValue.getString() = #FFFFFF
TValue.getString() = 
TValue.getString() = sub_channel_list
TValue.getString() = tel_code
TValue.getString() = inform_popup_type
[…]
TValue.getString() = need_front_approval
TValue.getString() = without_gambling_element
TValue.getString() = algo#push_new_notification_wd
TValue.getString() = dark_privacy_background_color
TValue.getString() = android Mozilla/5.0 (Linux; Android 14; SM-N935F
Build/AP2A.240805.005; wv) temu_android_version/3.8.0 temu_android_build/1730006400915
temu_android_channel/google pversion/0
TValue.getString() = android
TValue.getString() = ab_enable_open_lego_native_react
TValue.getString() = false
TValue.getString() = native_react_ab
TValue.getString() = os (\d+)_(\d+)_?(\d+)?
TValue.getString() = i
TValue.getString() = os (\d+)_(\d+)_?(\d+)?
TValue.getString() = i
TValue.getString() = Android (\d+).?(\d+)?.?(\d+)?
TValue.getString() = i
TValue.getString() = Android (\d+).?(\d+)?.?(\d+)?
TValue.getString() = i
TValue.getString() = Android
TValue.getString() = i
TValue.getString() = Android
TValue.getString() = i
TValue.getString() = iphone|ipad|ipod
TValue.getString() = i
TValue.getString() = iphone|ipad|ipod
TValue.getString() = i
TValue.getString() = ?otter_minversion=1.0.0&otter_ssr_api=%2Fapi%2Fmobile-bg-
saturn%2Fget_config%2Fsaturn_unipop_layer_160&otter_type=v1&pageName=saturn_unipop_layer
_160&rp=0&apm_app_id=100720&apm_biz_side=consumer-platform-
fe&apm_module_id=100100&apm_custom_module_id=90967&apm_custom_group_id=101050&otter_url_
config=market_message&_popup_pmm_tag_name=push_new_notification_wd&_popup_pmm_authorized
_type=10237&_popup_pmm_route_type=sms
TValue.getString() = otter_minversion=1.0.0&otter_ssr_api=%2Fapi%2Fmobile-bg-
saturn%2Fget_config%2Fsaturn_unipop_layer_160&otter_type=v1&pageName=saturn_unipop_layer
_160&rp=0&apm_app_id=100720&apm_biz_side=consumer-platform-
fe&apm_module_id=100100&apm_custom_module_id=90967&apm_custom_group_id=101050&otter_url_
config=market_message&_popup_pmm_tag_name=push_new_notification_wd&_popup_pmm_authorized
_type=10237&_popup_pmm_route_type=sms
```

The iOS app also utilizes Lego scripts. However, due to the limited time available for reverse engineering, detailed insights into the capabilities of the iOS app could not be obtained.

4.1.1.3 Impact and Assessment

A multitude of apps nowadays use JavaScript to dynamically customise and extend the integrated app functionality. For example, the Android app from Shein uses the QuickJS[2]

[2] https://github.com/taoweiji/quickjs-android/tree/master

runtime environment to interpret code.

Code loaded at runtime presents a challenge for security analysis. It is unpredictable which code is executed at any given time for different users. This dynamic makes it difficult to identify vulnerabilities or unexpected behaviour, since not all possible code paths can be predicted and fully tested.

4.1.1.4 Recommendations

At the time of the analysis, users have no straightforward way to block dynamic content. While the iOS Lockdown Mode disables JavaScript JIT for Safari, this has no effect on third-party apps [15]. On Android, there is also no way to block the execution of dynamic content. As an alternative, the app can be uninstalled preventively and, if necessary, ordered via the website.

It is recommended that further analyses and testing of the results are carried out by other security researchers to obtain a more complete picture of the potential security risks.

4.1.2 Encrypted data on the system is being transmitted

The mobile app sends encrypted content to a backend that contains details about the device on which the app is running.

4.1.2.1 Background

The Temu app sends encrypted JSON data to a backend system. This data can be used, for example, for telemetry or to collect user statistics. However, it also allows to uniquely identify users without a Temu account and to log whether the device in use can potentially be used to analyse the Temu app.

4.1.2.2 Evidence

The mobile apps for Android and iOS send HTTP POST requests to an API endpoint called `phantom`.

```
POST /api/phantom/fbdbpuedv/iurdxkfyb HTTP/2
Host: eu.temu.com
Cookie: install_token=76086C29-35A5-4DC8-9FDB-FF605222320B;
api_uid=CnBcYWbYAyk1RgBHi8s8Ag==
Content-Type: application/json
Etag: VqOGyZwwcbhs
Accept: */*
X-User-Info: rgn=192; lang=en; ccy=CHF; tz=Europe/Zurich
Accesstoken: 7P3DH6732OKAY2Q5HD4LHYTE5Y6WPMVPW2DPTJU6VWHY5RCEVEQQ0110c0dd2bd5
[…]

{"key":"AUkbErVNHCsuWu\/xaErJiJ0cNGS9QmrN5EtFaLubacLjThRW\/6bBFtD5rfGDc8QJCoUWvG3lCaHFey
z2iz59Ydx+vaib3\/V1xYwqQ6aa6axhsH3APWK\/Fiatt9WV+zK0ukxS4AsKjW0AoBT4WMH1u0d6xFGdZ1ttsx2t
+3b5XYI=","data":"vuFS2FT0yuFJaa3ZatSZGwxzEzV0hHWY\/9tIls6THso+YKLwZ\/88ApqPaVA\/Ife+NNs
Qda0ZzNZtdRlUdN\/tRadYTqprDmcucRZze3rO\/Xxa1ZR5dmopS+LqWZxXQoWRrjsMKuqTIYoWRqwhZIQiaFKX9
jHNT4FmvfsbqrKSRzPIsC8apJtzq3ULLd8o2qoI6y8osCb+AdVkPLbmb+gksPRqHZaR4MBrZxB\/+4MCjCLtHI30
rLFxFunasAM2JcIZ48t5Kk+hOOZiwbGTChVCm9tymhvpkuLeiF5iAUvgJAkTnr\/DuSSMvTsWK2HqRYXT…",
"name":"bgc"}
```

To intercept the data before encryption, the following Frida script was used with the Android app:

```
Java.perform(function () {
        var SecureNative = Java.use('xmg.mobilebase.secure.SecureNative');
        var MetaInfoInitTask = Java.use('com.baogong.secure_logic.MetaInfoInitTask');

        SecureNative.k.implementation = function (context, str, str2, str3, str4,
maybeData, l1) {
                console.log("SecureNative.k called");
                console.log("SecureNative.k(...) on data = " + maybeData);
                return SecureNative.k(context, str, str2, str3, str4, maybeData, l1);
        }
});
```

The following data exemplifies the contents of a request. The lines marked in red potentially offer the possibility to uniquely identify users and end devices even without a Temu account.

```
sc=bDjLkiI9gitesKrA5Q==
install_token=a3b7c20a-2549-493b-a020-7520b51e23b9
local_language=en
installer=com.android.vending
app_version=2.97.0
local_timezone=Europe/Zurich
google_adid=94eb0e55-042a-4e96-bdda-0b1b17144e91
scene=1
target_version=34
version=33
uuid=50a57fdb-1369-434d-8908-fcc1d6e3d980
input_mathod=com.samsung.android.honeyboard
ringtone=Galaxy+Bells
alarm=Homecoming
notification=Spaceline
instrumentation=android.app.Instrumentation
kernelVersion=
brightness=128
simState=0
totalmemory=3870367744
availablememory=1736736768
totalcapacity=54163128320
availablecapacity=42525515776
net_type=WIFI
ip_list=fe80::f05b:baff:fe44:cd8b%dummy0;fe80::b482:8aff:fe84:eca8%wlan0;10.96.226.16;
fk_result={"vInfo":{"exits":0,"id":""},"antInfo":{"exits":0}}
machine_arch=ARM
development_enabled=1
ae=1
process_id=4326
mediaDrm=0:58bf55eeb3e74b51cefcbc79124368b005d1aab84e68cc083f1da3b73053ddd6|Google|16.1.
0|Widevine CDM;
cid_inner=
cid=
input_device=3|sec_touchscreen|0fa9d3d0d40f7f5b316320876742e00ad5eb1ba9|4355;
foreground=true
secure_lock=0
user_env2=0wGrZhSFA8PUkfvsWeQyr+a5yvIW9/Tar6Ymra+iHrUn4yn+PKqXrSzNW7sJX4/HmlB8LhBmHbiIa2
BqcPPPlLMCutpvm/qOQagBqSRwGunIz4u/[…]
currentTime=1725534871937
```

The entries marked in green - `development_enabled`, `ae` (ADB-Debugging enabled) and `secure_lock` – may serve as potential indicators for detecting whether the app is running in a test or development environment. The collection of this data has also been observed in other apps and may have legitimate reasons to detect and prevent misuse or manipulation of data.

The `user_env2` data, which was highlighted in purple above, is additionally encrypted. The exact content of this data could not be determined in the time available. When reverse engineering the Android app, it was found that the content comes from the native library `libUserEnv.so`. The interaction between the native library and the Java code was intercepted with the following Frida script. This made it possible to reveal the content of the memory (RAM) of the relevant function.

```
Java.perform(function () {
  function toBase64(arrayBuffer) {
    var base64Chars =
      "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
    var bytes = new Uint8Array(arrayBuffer);
    var len = bytes.byteLength;
```

```
  var base64 = "";

  for (var i = 0; i < len; i += 3) {
    var byte1 = bytes[i];
    var byte2 = i + 1 < len ? bytes[i + 1] : 0;
    var byte3 = i + 2 < len ? bytes[i + 2] : 0;

    var enc1 = byte1 >> 2;
    var enc2 = ((byte1 & 3) << 4) | (byte2 >> 4);
    var enc3 = ((byte2 & 15) << 2) | (byte3 >> 6);
    var enc4 = byte3 & 63;

    if (i + 1 >= len) {
      enc3 = enc4 = 64;
    } else if (i + 2 >= len) {
      enc4 = 64;
    }

    base64 +=
      base64Chars.charAt(enc1) +
      base64Chars.charAt(enc2) +
      base64Chars.charAt(enc3) +
      base64Chars.charAt(enc4);
  }

  return base64;
}

var targetClass = Java.use("xmg.mobilebase.secure.SE");

targetClass.ue.overload("long").implementation = function (j13) {
  console.log("[*] Hooked ue(long j13)");
  console.log("[*] j13 value: " + j13);

  var result = this.ue(j14);
  console.log("[*] Result from native method: " + result);
  Java.perform(RevealNativeMethods);

  return result;
};

var pSize = Process.pointerSize;
var env = Java.vm.getEnv();
var RegisterNatives = 215,
  FindClassIndex = 6;
var jclassAddress2NameMap = {};

function dumpMemoryAsBase64(address, size) {
  try {
    var memoryDump = Memory.readByteArray(ptr(address), size);
    var base64Dump = toBase64(memoryDump);
    console.log("[*] Memory dump at address", address, ":", base64Dump);
  } catch (error) {
    console.log("[-] Failed to dump memory at address", address, ":", error);
  }
}

function getNativeAddress(idx) {
  return env.handle
    .readPointer()
    .add(idx * pSize)
    .readPointer();
}
```

```
  function RevealNativeMethods() {
    Interceptor.attach(getNativeAddress(FindClassIndex), {
      onEnter: function (args) {
        jclassAddress2NameMap[args[0]] = args[1].readCString();
      },
    });

    Interceptor.attach(getNativeAddress(RegisterNatives), {
      onEnter: function (args) {
        var jClassName = jclassAddress2NameMap[args[0]] || "";
        var jClass = jClassName.split("/");

        for (var i = 0, nMethods = parseInt(args[3]); i < nMethods; i++) {
          var structSize = pSize * 3; // sizeof(JNINativeMethod)
          var methodsPtr = ptr(args[2]);
          var methodName = methodsPtr
            .add(i * structSize)
            .readPointer()
            .readCString();
          var signature = methodsPtr
            .add(i * structSize + pSize)
            .readPointer()
            .readCString();
          var fnPtr = methodsPtr.add(i * structSize + pSize * 2).readPointer(); // void*
fnPtr

          var className = jClass[jClass.length - 1]; // Last part of the class name

          console.log(
            "\x1b[3" + "6;01" + "m",
            JSON.stringify({
              module: DebugSymbol.fromAddress(fnPtr)["moduleName"],
              package: jClass.slice(0, -1).join("."),
              class: className,
              method: methodName,
              signature: signature,
              address: fnPtr,
              baseAddress: Module.findBaseAddress(
                DebugSymbol.fromAddress(fnPtr)["moduleName"],
              ),
            }),
            "\x1b[39;49;00m",
          );

          if (className === "DeviceNative" || className === "SE") {
            console.log(
              "[*] Triggering memory dump for class: " +
                className +
                ", method: " +
                methodName,
            );
            dumpMemoryAsBase64(fnPtr, 1000); // Dump 1000 bytes of the function
pointer's memory
          }
        }
      },
    });
  }

  RevealNativeMethods();
});
```

Example output of the script:

```
{"module":"libUserEnv.so","package":"xmg.mobilebase.secure","class":"SE","method":"ue","
signature":"(J)Ljava/lang/String;","address":"0xb0c18fac","baseAddress":"0xb0c0b000"}
[*] Triggering memory dump for class: SE, method: ue
[*] Memory dump at address 0xb0c18fac :
8Est6RiwjeIw0E3iH9DD55SBn+UycgrjAxCg4Wwxn+V8QZ/lAJCg4QiAn+flf0LjXFGf5Vxhn+UAAJjlDHCNA[…]
```

In the time available, the encryption of the data and the contents of the memory could not be analysed in detail.

4.1.2.3 Impact

The details that are transmitted provide information about the environment in which the app is running. The data does not contain any information about the person or the account using the app. However, the content of the `user_env2` data, which is additionally encrypted, could not be decrypted in the time available.

Furthermore, the transmitted `install_token`, `uuid` and `google_adid` allow a unique identification of an installation.

4.1.2.4 Recommendations

It is recommended that further analysis is carried out on the `user_env2`, to determine the exact content and type of data stored in it. This analysis could provide valuable insights into whether the field contains sensitive or personal information and the extent of such data.

To reduce tracking via advertising ID, both the `google_adid` (Google Advertising ID) and the `IDFA` (Identifier for Advertisers) on Apple devices can be reset. However, in theory, Temu is also able to assign multiple advertising IDs to a device if the other environmental parameters have not changed.

4.1.3 Exact location can be requested

The app can request the exact location of the smartphone.

4.1.3.1 Background

Android and iOS allow you to determine only the approximate location instead of the exact location. However, the Grizzly Report suggests that Temu requests the exact location without having a legitimate interest in this data [3]. The Chinese government is said to be interested in location data, particularly that of government employees, police officers, expats, and members of oppressed minorities. Temu is supposed to provide the data to the government, as well as resell it and spy on users in general.

The Grizzly report also claims that the app uses a manipulative design with regard to location permissions [3]. The app would prompt the user to grant access to the device location when the camera search is started. The idea is that at this point, users might expect to see an authorisation request (e.g. because the camera needs the location for geotagging), which they would grant readily without reading it carefully. However, this behaviour could not be confirmed in the analysis.

4.1.3.2 Evidence

Both the Android and iOS versions of the app have the option to request permission for precise location queries. This can be seen in the Android version in the `AndroidManifest.xml`:

```
[…]
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
[…]
```

In iOS, this can be seen in the `Info.plist`:

```
[…]
        <key>NSLocationWhenInUseUsageDescription</key>
        <string>Only programs and features that you've allowed to access the location
can use it. For example: to help you add a new address to your Temu address book
faster.</string>
[…]
```

Before the app can access this data, permission must be requested from the users. However, this request never occurred during the test. Nor was the permission asked for when using the camera-based product search.

As can be seen in Figure 4, the app indicates that this authorisation is not used in Switzerland. However, it may be queried for users from the Middle East, for example, to help them enter the exact shipping address.

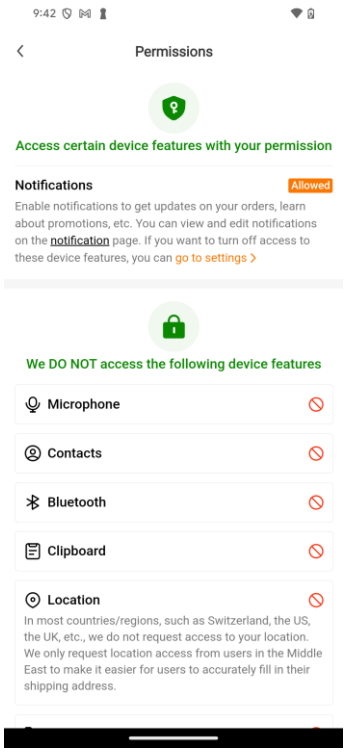

Figure 4: Location query authorisation notice

4.1.3.3 Impact

No location data requests were made during the tests. However, it is possible to request permission for the exact location. It is therefore up to the user to be aware of and control any requests for location data. Other Android and iOS apps, such as Shein [11], AliExpress [10] and Amazon [9], also ask for permission to geolocate.

How the location data would be used, if at all, could not be determined in the context of this analysis, as no precise location data was requested.

4.1.3.4 Recommendations

It is recommended to pay attention to requests for permissions. Permissions should only be granted if it is clear what they are needed for and if that is desired. It is also a recommended practice to regularly check the operating system's privacy report to ensure that the application has not made use of rights without being noticed.

Members of the Swiss Armed Forces at their bases, journalists and other exposed persons should follow these recommendations.

In general, it is also recommended to use the latest version of the operating system.

4.2 Medium Risk

Findings in this category carry a medium risk for the user, depending on the user and the circumstances. It is recommended to analyse the individual risk and, if necessary, to take appropriate measures to reduce the risk to an acceptable level.

4.2.1 Custom DNS implementation

The mobile app resolves certain domain names via its own service.

4.2.1.1 Background

To resolve certain domain names, the Android and iOS apps use their own service, which is addressed via a fixed IP address using HTTPS.

This type of name resolution might be used to prevent adverts from being blocked by ad blockers.

4.2.1.2 Evidence

Temu's own DNS service is contacted by the mobile apps using HTTPS-GET requests to the IP address `20.15.0.9`.

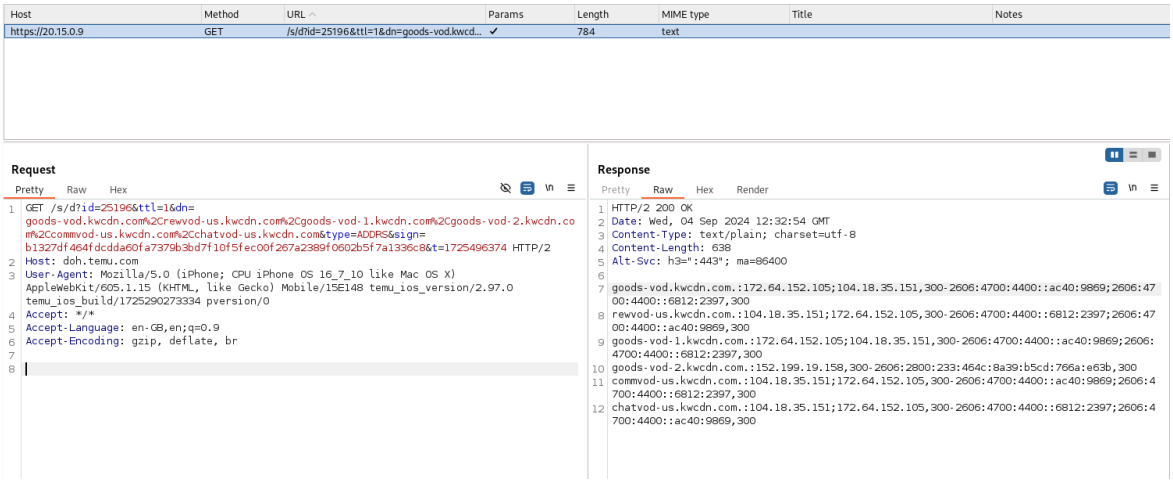

Figure 5: Domain name resolution request

The following domains were resolved during the investigation by the custom service:

```
eu.temu.com
goods-vod.kwcdn.com
rewvod-us.kwcdn.com
goods-vod-1.kwcdn.com
goods-vod-2.kwcdn.com
commvod-us.kwcdn.com
chatvod-us.kwcdn.com
img-2.kwcdn.com
```

These host names were resolved to the following IP addresses:

```
eu.temu.com: 20.157.217.118, 20.157.119.2, 20.157.217.65, 20.47.117.32, 152.199.19.158
goods-vod.kwcdn.com: 172.64.152.105, 104.18.35.151,
2606:4700:4400::ac40:9869;2606:4700:4400::6812:2397
rewvod-us.kwcdn.com: 104.18.35.151, 172.64.152.105, 2606:4700:4400::6812:2397,
2606:4700:4400::ac40:9869
goods-vod-1.kwcdn.com: 172.64.152.105, 104.18.35.151, 2606:4700:4400::ac40:9869,
2606:4700:4400::6812:2397
goods-vod-2.kwcdn.com: 152.199.19.158, 2606:2800:233:464c:8a39:b5cd:766a:e63b
commvod-us.kwcdn.com: 104.18.35.151, 172.64.152.105,
2606:4700:4400::ac40:9869;2606:4700:4400::6812:2397
chatvod-us.kwcdn.com: 104.18.35.151, 172.64.152.105, 2606:4700:4400::6812:2397,
2606:4700:4400::ac40:9869
img-2.kwcdn.com: 84.17.53.42
```

4.2.1.3 Impact

It is possible for Temu to change the resolved IP addresses dynamically. This approach can circumvent DNS- and IP-based network blockades, such as those implemented by the Swiss government to restrict access to illegal gambling sites. The dynamic resolution mechanism also makes it possible to bypass ad blockers that would recognise and block advertising and tracking based on domain names or IP addresses.

This procedure has also been observed occasionally with IoT devices but is not considered common practice by the test experts.

4.2.1.4 Recommendations

No action is required for this finding.

Users have no practical way of preventing this behaviour of the Temu app.

4.2.2 Location tracking via WebView

With the help of a bridge class, the location of the app's users can be determined by JavaScript code in the WebView.

4.2.2.1 Background

The Android app uses a WebView to display certain content. This WebView is extended by a bridge class that provides access to sensor data and other Java features. This allows, for example, the location of users to be queried.

4.2.2.2 Evidence

The Java bridge interface `com.einnovation.whaleco.web.meepo.extension.jsapi.BridgeImpl` allows Java methods to be called from WebViews. The bridge class makes it possible for JavaScript code loaded from Temu servers to access sensor data and other Java interfaces.

```
@JavascriptInterface
public String callNative(String str, String str2, String str3, long j13) {
    BridgeImpl.access$000().i("JsInterfaceImplAnnotation#callNative", new Runnable() {
// from class: com.einnovation.whaleco.web.meepo.extension.jsapi.d
        /* renamed from: b */
        public final /* synthetic */ int f21643b;
        /* renamed from: c */
        public final /* synthetic */ String f21644c;
        /* renamed from: d */
        public final /* synthetic */ String f21645d;
        /* renamed from: h */
        public final /* synthetic */ String f21646h;
        /* renamed from: t */
        public final /* synthetic */ long f21647t;
        public /* synthetic */ d(int i13, String str4, String str22, String str32, long
j132) {
            i13 = i13;
            str = str4;
            str2 = str22;
            str3 = str32;
            j13 = j132;
        }
        @Override // java.lang.Runnable
        public final void run() {
            BridgeImpl.JsInterfaceImplAnnotation.this.lambda$callNative$0(i13, str,
str2, str3, j13);
        }
    });
    return null;
}
```

In the decompiled code of the `c()` method, around 190 methods are listed, which are thus made available for WebViews. Among them is `AMLocation.get`, which enables location requests and the transmission thereof.

```
public static void c() {
        […]
        a("AMLocation.get", AMLocation.class);
        a("AMLocation.check", AMLocation.class);
        […]
}
```

The `AMLocation.get()` method is located in `com.whaleco.temu.address_map.jsapi` in the decompiled Temu source code. It calls a service to determine the location:

```
public final void b(ew.a aVar) {
    ((ILocationService) m.b("ILocationService").b(ILocationService.class)).l1(new
d.a().u("address").w(WebAssetDatabase.WEB_ASSET_DATA_BASE_LOCK_TIMEOUT).v(1).t(1).q(200.
0d).p(true).s(new a(aVar)).r());
}
```

The iOS app also uses WebViews to display dynamic content. In the limited time available, it was not possible to clarify whether the WebView in the iOS app can also access sensor data and other interfaces.

4.2.2.3 Impact

Potentially, the location of users can be queried via JavaScript code loaded from a backend. Since the JavaScript bridge interface in the WebView has access to device-specific features, it would be possible in principle to query location data and transmit it to a backend. However, no such location queries were observed in the analysis conducted: neither in the operating systems' privacy reports nor in the data traffic to the backend.

More information on location queries is provided in *Chapter 4.1.3*.

4.2.2.4 Recommendation

It is recommended not to grant the app access to the device location. Since, according to the Temu data protection guideline, these permissions are not needed for users in Switzerland, they can be denied without any restrictions to user-friendliness.

Furthermore, it is recommended to use a current version of Android and iOS, which allows the detection of app access to sensitive sensor data.

4.2.3 Transmission of encrypted data with device details

The app sends encrypted data containing details of the smartphone used to the backend.

4.2.3.1 Background

The Android and iOS applications encrypt certain data using their own encryption logic and send it to the API endpoint `eu.pftk.temu.com/pmm/api/pmm/api`. By reverse engineering the Android app, it was possible to determine that AES-GCM is used for encryption.

4.2.3.2 Evidence

The encrypted data can be decrypted on Android using Frida by hooking into the encryption logic.

The following Frida script was used to analyse the encrypted data. **Note:** Since the class names are obfuscated in the app, they vary from version to version.

```
/*
     frida -U -f com.einnovation.temu -l temu.js
*/
Java.perform(function () {
    var l92_a_class = Java.use('l92.a');
    var wu_c_class = Java.use('wu.c');

    l92_a_class.a.overload('java.util.Map', '[B').implementation = function (map,
byteArray) {
        var result = "";
        try{
        console.log('Hooked method a(Map, byte[]):');

        result = this.a(map,byteArray);
        var resultHex = [];
        for (var i = 0; i < byteArray.length; i++) {
            resultHex.push(('0' + (byteArray[i] & 0xFF).toString(16)).slice(-2));
        }
        console.log('Result byte array (hex): [' + resultHex.join('') + ']');

        var entrySet = map.entrySet();
        var iterator = entrySet.iterator();
        while (iterator.hasNext()) {
            var entry = iterator.next();
            console.log('Map key: ' + entry.getKey() + ', Map value: ' +
entry.getValue());
        }

        console.log('Result byte array: [' + result + ']');
        } catch (err) {console.log(err)}

        return result;
    };

    wu_c_class.a.overload('java.lang.String', '[B').implementation = function (map,
byteArray) {
        var result = "";
        try{
        console.log('Hooked method BG.c_USER_CYPTOUTIL.a(Map, byte[]):');
```

```
        result = this.a(map,byteArray);
        var resultHex = [];
        for (var i = 0; i < byteArray.length; i++) {
            resultHex.push(('0' + (byteArray[i] & 0xFF).toString(16)).slice(-2));
        }
        console.log('BG.c_USER_CYPTOUTIL.a: Result byte array (hex): [' +
resultHex.join('') + ']');

        var entrySet = map.entrySet();
        var iterator = entrySet.iterator();
        while (iterator.hasNext()) {
            var entry = iterator.next();
            console.log('Map key: ' + entry.getKey() + ', Map value: ' +
entry.getValue());
        }

        console.log('BG.c_USER_CYPTOUTIL.a: Result byte array: [' + result + ']');
        } catch (err) {console.log(err)}

        return result;
    };

});
```

The data is output in a hex-encoded format and can be decoded using Cyberchef, for example: *https://gchq.github.io/CyberChef/#recipe=From_Hex('Auto')*

The following is an excerpt from the decoded data:

```
cpu_arch armeabi-v7a:
runningAppId   234:/
uid(BBHEZ5PCN3D2PCQAUDTKFJHG575KOUHILJ5QCF7C:
osV66:
d6:

internalNo
1725958174277:

is64bitfalse:
piddZkwfELllbo1:
mSM-A137F:
dided023a9970fab4be:
reportStrategycount_limit|15B90728 ▒▒▒_2*

network1*
regionCH*&
custom_processcom.einnovation.temu*
custom_channelgoogle2
deviceLevel22

isdebugApkfalse2
captive_portalfalse2(
biz_svr_time_format2024-9-16 9:18:582
version_name2.99.02
timezone
Europe/Zurich2-
logId$947125d0-9256-4291-841e-1f1e9b9ba19a
[…]
```

No personal data was found in the transmitted data.

4.2.3.3 Impact

This encryption offers the possibility of hiding personal information in data traffic. However, no sensitive data was found in the encrypted data traffic during the analysis.

4.2.3.4 Recommendations

In practice, users have no way of preventing this behaviour of the mobile app. It is recommended to ensure that no sensitive information is transmitted to the backend, also for future versions.

4.3 Low Risk

Findings in this category pose a low risk to users depending on the user and circumstances. It is recommended to assess the individual risk and, if necessary, to take appropriate measures to reduce the risk to an acceptable level. However, the risks of the other categories should be considered first.

4.3.1 Query of running processes

The mobile app creates a list of running processes.

4.3.1.1 Background

The Temu app checks during execution whether certain apps are installed on the mobile device. The observed behavior may be related to the app displaying more or fewer buttons (e.g., for sharing content) depending on which other apps are installed. In the worst-case scenario, this information could potentially reveal sensitive details about the user (e.g. ethnicity or religion).

At this time, however, it is unclear whether this information is transmitted to the backend servers and analyzed by Temu. For this reason, this finding is classified as a low-risk issue in this document.

4.3.1.2 Evidence

The following extract from the `Info.plist` file of the Temu app shows the list of third-party apps that can be identified by the Temu app at the time of the analysis:

```
<key>LSApplicationQueriesSchemes</key>
<array>
        <string>satispay</string>
        <string>aupay</string>
        <string>swish</string>
        <string>number26</string>
        <string>naversearchapp</string>
        <string>vipps</string>
        <string>vippsMT</string>
        <string>mobilepay</string>
        <string>supertoss</string>
        <string>tngdwallet</string>
        <string>ms-outlook</string>
        <string>fbapi</string>
        <string>fb-messenger-share-api</string>
        <string>whatsapp</string>
        <string>twitterauth</string>
        <string>twitter</string>
        <string>instagram</string>
        <string>snapchat</string>
        <string>googlegmail</string>
        <string>instagram-stories</string>
        <string>paypal</string>
        <string>com.afterpay.afterpay-consumer</string>
        <string>klarna</string>
        <string>squarecash</string>
        <string>snapchat</string>
        <string>tg</string>
        <string>line</string>
        <string>kakaotalk</string>
```

```
        <string>gcash</string>
        <string>com.venmo.touch.v2</string>
        <string>paypay</string>
        <string>ascendmoney</string>
        <string>barcelona</string>
        <string>sgnl</string>
</array>
```

In fact, while using the Temu app on the device of the NTC test expert, the presence of the following third-party apps was detected:

```
canOpenURL: kakaotalk://
canOpenURL: squarecash://
canOpenURL: paypal://
canOpenURL: klarna://
canOpenURL: com.afterpay.afterpay-consumer://
canOpenURL: com.venmo.touch.v2://
canOpenURL: supertoss://
canOpenURL: vipps://
canOpenURL: mobilepay://
canOpenURL: naversearchapp://
canOpenURL: number26://
canOpenURL: paypay://
canOpenURL: ascendmoney://
canOpenURL: swish://
canOpenURL: whatsapp://
canOpenURL: aupay://
canOpenURL: satispay://
```

The following Frida script was used to dynamically analyse the behaviour of the Temu app:

```
/*
$ frida -U -f com.einnovation.temu -l temu.js
*/
Interceptor.attach(ObjC.classes.UIApplication["- canOpenURL:"].implementation, {
onEnter: function (args) {
console.log('canOpenURL:', ObjC.Object(args[2]).toString());
},
onLeave: function (retval) {
}
});
```

On Android, the following queries are made according to `AndroidManifest.xml`:

```
<queries>
       <package android:name="com.facebook.katana"/>
       <package android:name="com.facebook.orca"/>
       <package android:name="com.instagram.android"/>
       <package android:name="com.whatsapp"/>
       <package android:name="com.snapchat.android"/>
       <package android:name="com.twitter.android"/>
       <package android:name="org.telegram.messenger"/>
       <package android:name="jp.naver.line.android"/>
       <package android:name="com.reddit.frontpage"/>
       <package android:name="com.discord"/>
       <package android:name="com.kakao.talk"/>
       <package android:name="com.instagram.barcelona"/>
       <package android:name="org.thoughtcrime.securesms"/>
       <provider android:authorities="com.facebook.katana.provider.PlatformProvider"/>
       <package android:name="com.facebook.wakizashi"/>
       <package android:name="com.afterpaymobile.us"/>
       <package android:name="com.afterpaymobile.uk"/>
       <package android:name="com.afterpaymobile"/>
```

```
        <package android:name="com.myklarnamobile"/>
        <package android:name="com.squareup.cash"/>
        <package android:name="com.paypal.android.p2pmobile"/>
        <package android:name="com.globe.gcash.android"/>
        <package android:name="viva.republica.toss"/>
        <package android:name="dk.danskebank.mobilepay"/>
        <package android:name="fi.danskebank.mobilepay"/>
        <package android:name="no.dnb.vipps"/>
        <package android:name="com.nhn.android.search"/>
        <package android:name="de.number26.android"/>
        <package android:name="jp.ne.paypay.android.app"/>
        <package android:name="th.co.truemoney.wallet"/>
        <package android:name="se.bankgirot.swish"/>
        <package android:name="jp.auone.wallet"/>
        <package android:name="com.satispay.customer"/>
        <package android:name="com.venmo"/>
        <package android:name="com.android.vending"/>
        <package android:name="com.google.android.engage.verifyapp"/>
        <package android:name="com.google.android.apps.maps"/>
        <package android:name="com.sec.android.app.samsungapps"/>
</queries>
```

On Android, the method `getRunningAppProcesses` is called. This method can be used to read which processes are currently running. The Android development documentation mentions that this is a debug method and should not be used in production environments [16]. Due to time constraints, this was not further investigated on Android.

As part of this analysis, no request could be identified that transmits a list of running processes to a backend.

4.3.1.3 Impact

Temu creates the prerequisites for the ability to query third-party apps with the entries in the `Info.plist` file. This file is reviewed as part of the Apple App Store review process. Apparently, the app store operator has no concerns that the permissions granted to Temu could be misused.

No transmission of information about installed third-party apps to a Temu backend was detected during the analysis. However, it cannot be ruled out that this information is contained in the coded or encrypted content of HTTP requests.

If this information is only used locally on the mobile device, the behaviour of the Temu app is unproblematic. However, this would not be the case if the information is transmitted to a Temu backend. This could not be ruled out during this analysis.

Other apps such as Shein, Aliexpress or Amazon require comparable permissions to detect third-party apps.

4.3.1.4 Recommendations

Users have no practical means of preventing this behavior of the Temu app.

4.3.2 Multi-factor authentication is not enforced

When registering an account, Temu only requires an e-mail address and a password or a telephone number. Multi-factor authentication is optional.

4.3.2.1 Background

Temu aims to make registering and logging in to Temu accounts as convenient as possible. For this reason, it is assumed that Temu does not perform in-depth identity verification and also forgoes more complex MFA factors by default. For attackers, it could potentially be possible to gain access to Temu user accounts through SIM swapping by taking over the user's phone number.

Enabling the optionally offered "2-Step Verification" (as Temu refers to it) provides protection against this type of attack.

4.3.2.2 Evidence

As shown in Figure 6, Temu only requires an email address and password or a phone number to register an account. Registration via other social media platforms is also possible but was not considered in this analysis.

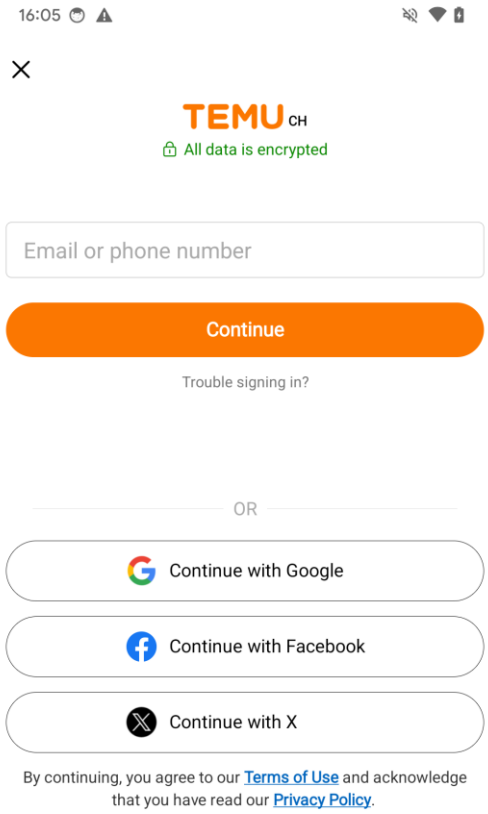

Figure 6: Temu Registration

Figure 7 shows the option of setting up an additional factor for login.

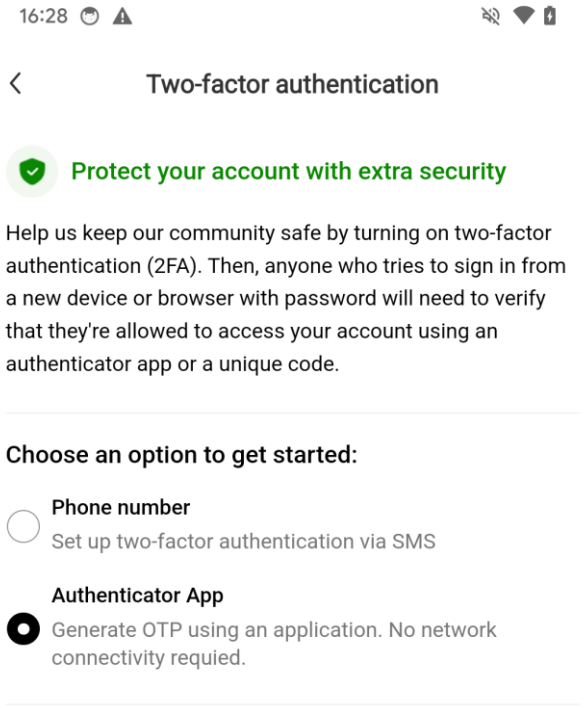

Figure 7: Temu MFA Option

4.3.2.3 Impact

If 2-Step Verification is not explicitly enabled, there is an increased risk of account takeover through phishing, reuse of known passwords, and other attacks targeting access credentials. This risk applies not only to Temu but to all apps that rely solely on receiving an SMS text message or email for authentication.

4.3.2.4 Recommendations

All users are advised to enable the 2-Step Verification option via an authenticator app in their account settings.

Additionally, it is generally recommended to use strong and unique passwords.

4.3.3 Encrypted configuration files

Temu uses encrypted configuration files.

4.3.3.1 Background

The Android and iOS apps use encrypted configuration files, presumably to provide enhanced protection for sensitive data such as API tokens. Furthermore, the configuration offers insights into the business logic, which is also better protected through encryption.

Although encryption provides protection, the configuration can still be read. However, this requires reverse engineering and makes access significantly more complex.

4.3.3.2 Evidence

In the Android app, encrypted configuration files can be found under `/assets/abc/data` and `/assets/config/data`. The encryption method used is AES-CBC. The parameters required for decryption can be uncovered through reverse engineering.

The key is constructed using the following code:

```
    public static Key d() {
        return e((d62.d.w() + y52.e.x() + f62.d.c() +
a.h()).getBytes(StandardCharsets.UTF_8));
    }
```

The following code constructs the initial vector.

```
public static IvParameterSpec c() {
        return new IvParameterSpec((a.g() + p.h() +
b62.a.e()).getBytes(StandardCharsets.UTF_8));
    }
```

The function below decrypts the configuration file using the initial vector and key.

```
public static byte[] b(byte[] bArr, Key key, IvParameterSpec ivParameterSpec, String
str) {
        try {
            Cipher cipher = Cipher.getInstance(str);
            cipher.init(2, key, ivParameterSpec);
            return cipher.doFinal(bArr);
        } catch (Exception e12) {
            kb2.b.e("ABC.MUtils", "ad4 fail. " + i.r(e12));
            return null;
        }
    }
```

The encryption uses the following parameters:

- **Algorithm:** `AES CBC`
- **Key:** `_amm_config_key_`
- **Initial vector (IV):** `213344114255776201593163165249 75`

This decryption can also be carried out using this Cyberchef 'recipe': *https://gchq.github.io/CyberChef/#recipe=AES_Decrypt(%7B'option':'UTF8','string':'_amm_config_key_'%7D,%7B'option':'Hex','string':'213344114255776201593163165249 75'%7D,'CBC','Raw','Raw',%7B'option':'Hex','string':''%7D,%7B'option':'Hex','string':''%7D)*

The following is an excerpt from the decrypted file content, which contains approximately 180 lines on Android:

```
{
  "configs": {
    "Network.dns_config_00001": {
      "v": "[2.32.0:+â^)",
      "d":
"{\"scheme\":\"http\",\"path\":\"/d\",\"hosts\":[\"20.15.0.9\",\"20.15.0.95\",\"20.15.0.
158\"],\"encryKey\":\"<redacted>\"
\",\"params\":{\"ttl\":\"1\",\"id\":\"25196\"},\"persistent_host_list\":[\"us.temu.com\"
,\"ca.temu.com\",\"au.temu.com\",\"nz.temu.com\",\"eu.temu.com\",\"jp.temu.com\",\"kr.te
mu.com\",\"br.temu.com\",\"sg.temu.com\",\"qa.temu.com\"],\"preload_host_list\":[\"us.te
mu.com\",\"ca.temu.com\",\"au.temu.com\",\"nz.temu.com\",\"eu.temu.com\",\"jp.temu.com\"
,\"kr.temu.com\",\"br.temu.com\",\"sg.temu.com\",\"qa.temu.com\"],\"pattern_str\":\"(.*)
\\\\.(temu|kwcdn)\\\\.com\",\"dns_ttl_max\":60,\"dns_bg_ttl_min_mobile\":1800,\"dns_bg_t
tl_min\":300,\"doh_params\":{\"ttl\":\"1\",\"id\":\"25196\"},\"origin_hosts\":\"doh.temu
.com\",\"doh_scheme\":\"https\",\"doh_path\":\"/s/d\",\"sign_key\":\"<redacted>
\",\"sign_timeout\":3600000,\"black_pattern_str\":\"rewimg-us-1.kwcdn.com\"}"
    },
    "Network.dns_dual_config_001": {
      "v": "[2.60.5:+â^)",
      "d": "{\"ip_stack_white_list\":{\"api\":{\"/api/bg-
aquarius/popup/homepage\":{\"ip_stack\":\"ip_stack_ipv6_first\"},\"/api/bg-
aquarius/popup/global\":{\"ip_stack\":\"ip_stack_ipv6_first\"},\"/api/bg-
aquarius/popup/default\":{\"ip_stack\":\"ip_stack_ipv6_first\"},\"/api/server/_stm\":{\"
ip_stack\":\"ip_stack_ipv6_first\"}},\"matracker\":{\"/ut\":{\"ip_stack\":\"ip_stack_ipv
6_first\"}}}}"
    },
[…]
```

The iOS app contains significantly more configuration entries (approx. 1300 lines), some of which even include JavaScript code. In certain cases, it can be useful to create a configuration file with embedded programming code to allow dynamic values or logical decisions based on specific conditions.

```
[…]
"prefetch.prefetch_common_config": {
      "v": "[1.1.0:+∞)",
      "d":
"{\"prefetch_support_types\":[\"script\",\"style\",\"image\"],\"prefetch_each_json_load_
src_max_num\":300,\"preload_js\":\"(function() {var link =
document.createElement('link');link.as = '%@';link.href = '%@';link.rel =
'preload';document.head.appendChild(link);} ());\"}"
    },
[…]
"web.weblog_script": {
      "v": "(-∞:+∞)",
```

```
      "d": "(function(){if(window&&window.console){var
methodList=['log','info','warn','error'];var newCons=(function(oldCons){var
nativeLog=function(msg){try{if(msg){window.webkit.messageHandlers.webLogMessageHandler.p
ostMessage(msg)}}catch(error){}};var getArgumentsStr=function(args){var logs=[];for(var
i=0;i<args.length;i++){try{if(Object.prototype.toString.call(args[i])=='[object
Function]'){logs.push(args[i].toString())}else if(typeof
args[i]=='undefined'){logs.push('undefined')}else{logs.push(JSON.stringify(args[i]))}}ca
tch(error){logs.push('[stringify error]')}}return logs.join(', ')};var myCons={};var
oldConsMethods={};methodList.forEach(function(method){oldConsMethods[method]=oldCons[met
hod];myCons[method]=function(){if(typeof
oldConsMethods[method]=='function')oldConsMethods[method].apply(oldCons,arguments);nativ
eLog({logType:method,text:getArgumentsStr(arguments)})}});return
myCons}(window.console));methodList.forEach(function(method){window.console[method]=newC
ons[method]})}}());"
    },
[…]
```


The `/assets/config/data` configuration file has the same content, but uses different parameters for encryption:

- **Algorithm:** `AES CBC`
- **Key (HEX value):** `42132b322f063b161d4a7303745a596e`
- **Initial vector (IV, HEX value):** `22505c58440b433c09520446547b7e12`

The decryption can also be carried out using this Cyberchef 'recipe':

https://gchq.github.io/CyberChef/#recipe=AES_Decrypt(%7B'option':'Hex','string':'42132b322f063b161d4a7303745a596e'%7D,%7B'option':'Hex','string':'22505c58440b433c09520446547b7e12'%7D,'CBC','Raw','Raw',%7B'option':'Hex','string':''%7D,%7B'option':'Hex','string':''%7D)

4.3.3.3 Impact and Assessment

In the time available, it has not been possible to fully determine where these configuration entries are used. Based on the names and values of the configuration entries, it is assumed that the configuration is used to control WebView content.

Encrypting the configuration file makes it more difficult to analyse.

4.3.3.4 Recommendations

Users have no practical means to prevent this behavior of the Temu app. A more in-depth analysis by additional security researchers is recommended to determine where these configuration values are being used.

4.3.4 Detection of elevated privileges and debugging

The mobile app checks whether a phone has debugging options enabled and is rooted or jailbroken.

4.3.4.1 Background

The Temu application collects information about the operating environment. This can be used to determine whether the application is running on a test device. Such methods can be used by manufacturers to adjust the program flow to that environment, e.g. by suppressing behaviour that needs to be kept secret.

4.3.4.2 Evidence

The decompiled iOS app contains strings that indicate a jailbreak detection. Figure 8 shows a section of the data structure in which the results are stored.

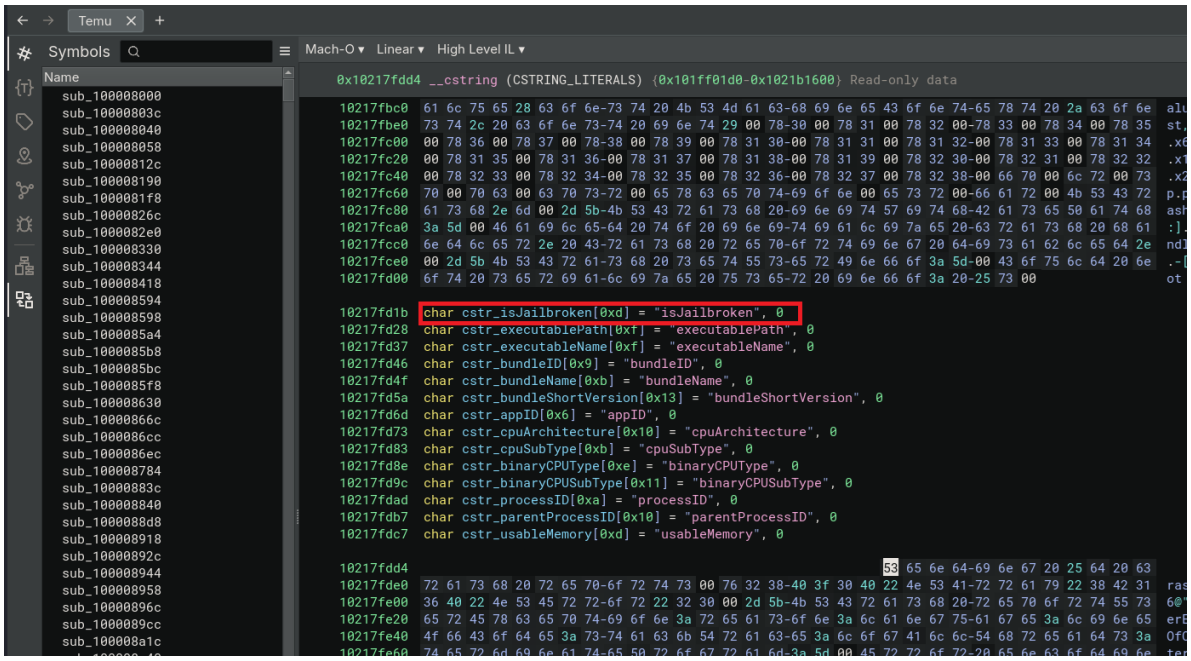

Figure 8: Jailbreak detection on iOS

As can be seen in Figure 9 , strings were found in the `libUserEnv.so` Android library that indicate root detection.

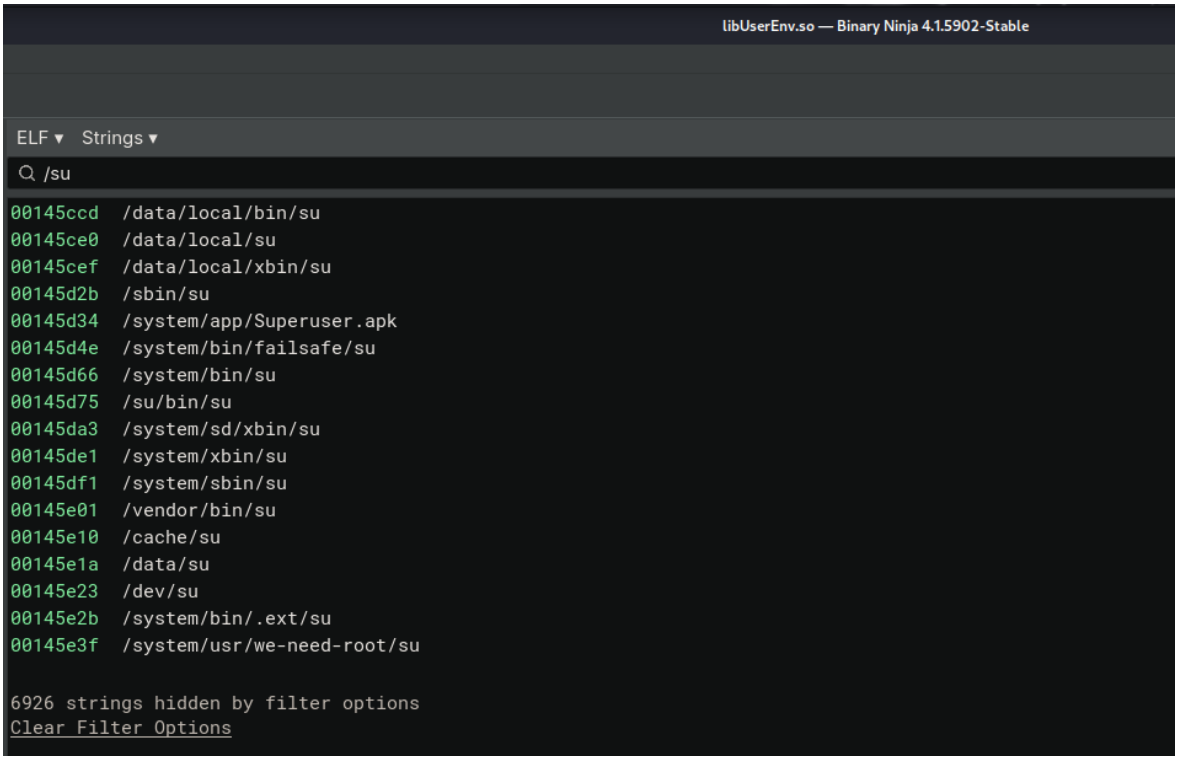

Figure 9: Strings for determining rooted devices

As part of this analysis, no message to a backend could be found to pass the result of the root/jailbreak detection to a backend.

The app also determines whether debugging options are activated on the device. These details are also transmitted to a backend. This is described in more detail in *Chapter 4.1.2*.

4.3.4.3 Impact

It is unclear what the above information from Temu is used for. Therefore, it cannot be ruled out that this is a test bench detection. If this is the case, such a test bench mode could hide certain behaviour of the Temu app.

Other mobile apps such as AliExpress or Shein have also implemented root/jailbreak detection.

4.3.4.4 Recommendations

It is recommended that more in-depth investigations are carried out to rule out any abnormal behaviour of the Temu app in test bench situations.

4.4 Relativisations

The findings listed in this category relativise findings from other reports on Temu. In the view of the NTC, they present either little or no risk to users.

4.4.1 Read out and transmit logs of other apps

For the tested version, no evidence could be found that the app can read event logs of other apps.

4.4.1.1 Background

On Android, it is possible to log events using Logcat. This can be used to log errors or other events that developers consider important. Developers are instructed to avoid logging sensitive data such as login credentials. Adherence to these best practices is the responsibility of the application developer.

The Grizzly report claims that the Temu app reads the log data of other apps in order to spy on user behaviour [3]. However, since Android 4.1, which has been on the market for 12 years, it is generally no longer possible to access log data from third-party apps. Only specially privileged and OEM apps are excluded [17]. Therefore, this should not normally be possible for the Temu app.

4.4.1.2 Evidence

The Temu app was found to be sending own log data to backend servers, but not third-party log data. Since Android 4.1, access to third-party log data is generally not possible. The only exceptions are, for example, if apps are pre-installed and have permission to read system logs. Apps published via the Play Store are not allowed to use this permission.

Figure 10 shows an example of the app's data transfer.

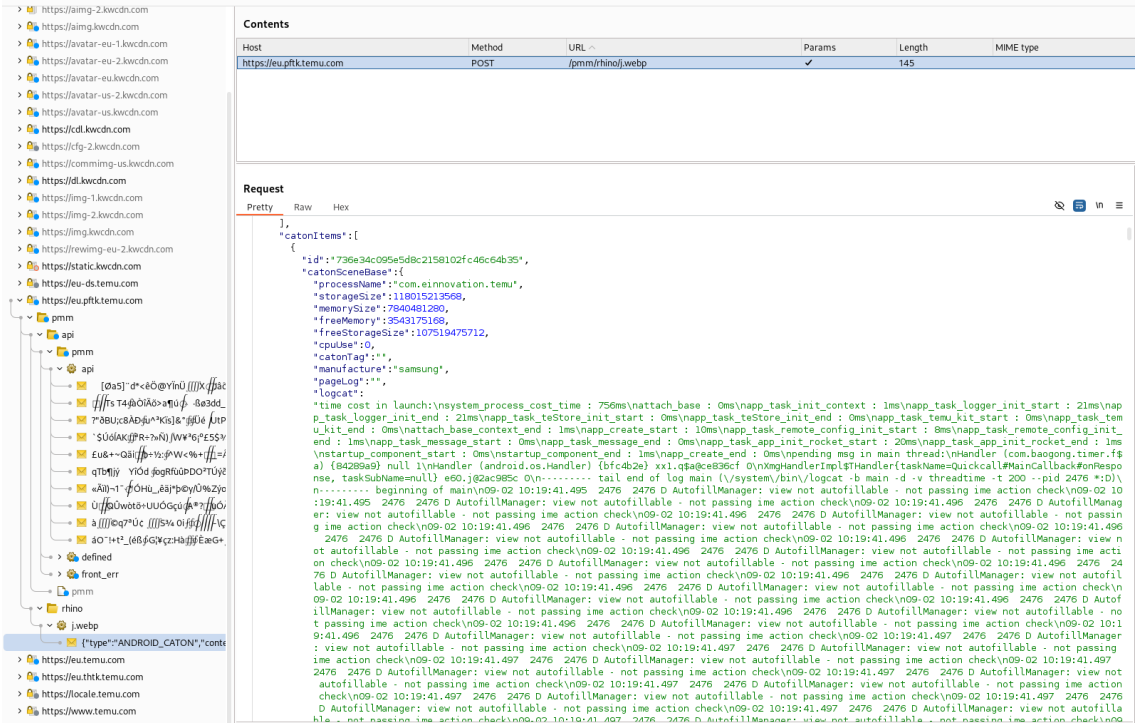

Figure 10: Log data is transmitted

The transferred data is collected in a native library called `libcrash.so`. Figure 11 shows the command used to query the event logs.

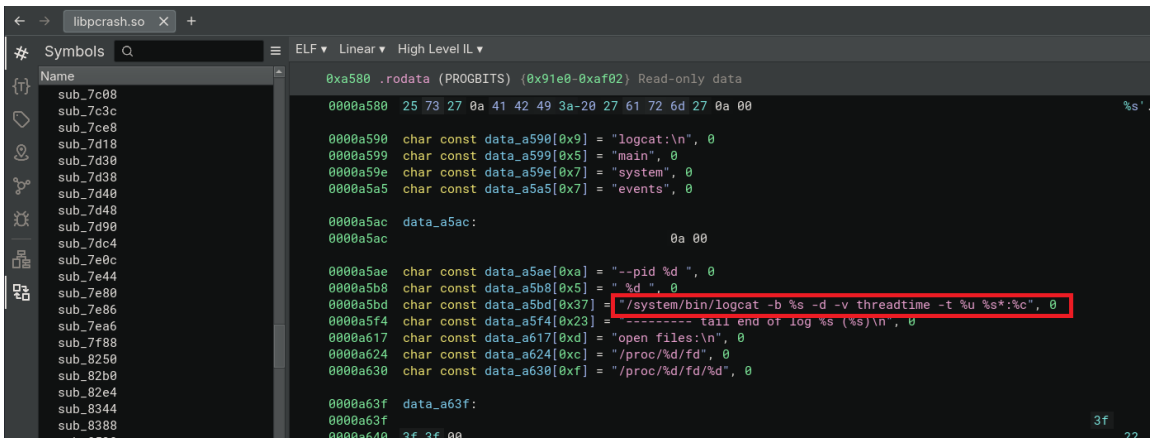

Figure 11: Logcat command to collect event logs

4.4.1.3 Impact

The app sends log data from its own app to backend servers.

4.4.1.4 Recommendations

It is recommended to install Temu only from trusted, official sources - such as the Google Play Store, Apple App Store - and ensuring that the device is running the latest operating system version.

4.4.2 Read out and transmit system files

The Android app collects and transmits data on the CPU of the end device to a TEMU backend.

4.4.2.1 Background

The Grizzly report claims that the app collects sensitive system event logs and system information and sends it to Temu's servers [3].

4.4.2.2 Evidence

There are indications that the application utilizes native libraries to access system files in order to retrieve information such as the CPU's number of cores, clock frequency, and the SoC (System on Chip) serial number. While accessing CPU performance data may have legitimate purposes, such as optimizing the application's performance based on the user's device capabilities, obtaining the SoC serial number raises potential privacy concerns. This unique identifier could be used to track users who are not logged in, when an Advertising ID is unavailable, or as an alternative method of user identification.

Various strings in a programme library indicate access to CPU information. Figure 12 and Figure 13 show the paths that are queried by the app.

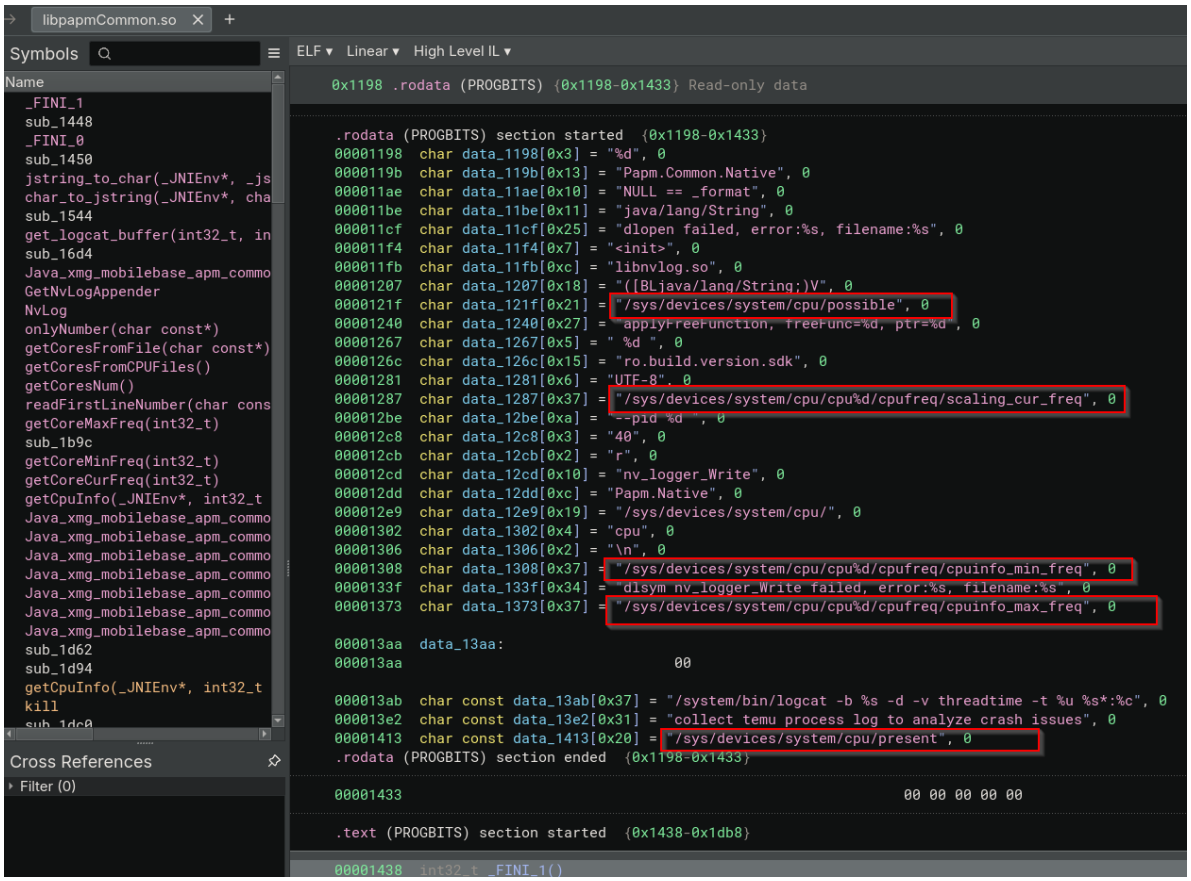

Figure 12: Paths to CPU metadata

Path in the `libsecure_lib.so` programme library for accessing the CPU serial number:

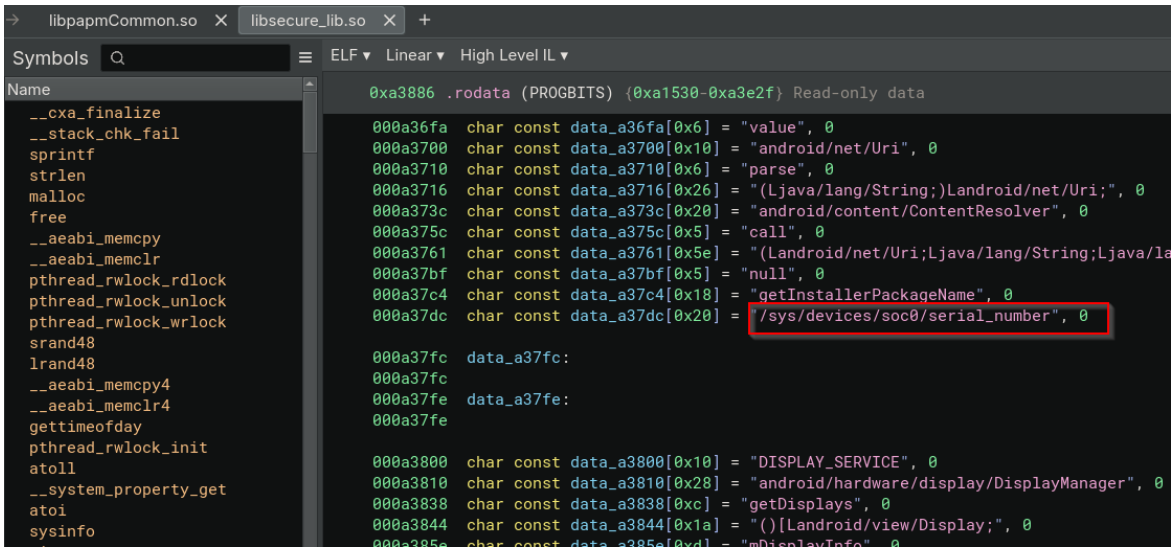

Figure 13: Access to SoC serial number

No further evidence of access to the sysfs pseudo file system could be found during this analysis. (This is used to query system information and, if necessary, to carry out configurations with the required rights).

4.4.2.3 Impact

No access to sensitive data was detected. Therefore, there is no particular impact. Collecting CPU architecture data can be useful for troubleshooting and creating relevant performance metrics.

4.4.2.4 Recommendations

There is no need for any specific measures in this area.

4.4.3 Screenshots of other apps

No evidence was found that screenshots are taken of other apps.

4.4.3.1 Background

The Grizzly report notes that certain methods, such as `getWindow()`, `getDecorView()`, `getRootView()`, are used by the Android Temu app to spy on users' activities in other apps [3].

These methods indeed are called, but these methods return the view or window of their own app and have legitimate use cases.

4.4.3.2 Evidence

The Temu app uses the methods mentioned in the Grizzly report:

```
    public final void h(Activity activity, n52.g gVar) {
        View peekDecorView;
        Window window = activity.getWindow();
        if (window == null || (peekDecorView = window.peekDecorView()) == null) {
            return;
        }
        View rootView = peekDecorView.getRootView();
        if (rootView == null) {
            x42.f.j("Papm.Leak.Repair", "repairActivity viewRoot is null, return. " +
activity);
            return;
        }
        k(rootView, gVar);
    }
```

In Android, apps are primarily made up of activities, which allow you to use certain features of the app. An activity has a window. The content of the window consists of a view, which can be hierarchically structured from views [18]. Therefore, in the method shown above, the window of the current activity is determined using `activity.getWindow()` and then the View root element is determined using the methods `peekDecorView()` (similar to `getDecorView()`) and `getRootView()`[19]. This approach is unproblematic.

The code of the Temu app is not sufficient to take a screenshot of an application. This would require the `draw()` method to be applied to the `rootView` to render the view [19]. This is not the case here.

Another key point is that none of these methods can be applied to activities of third-party apps. Access to third-party apps is only allowed if they explicitly export activities and thus make them accessible to other apps [20].

To take screenshots of other applications, the MediaProjection interface has been available since Android 5.0 (API level 21) [21]. When using this, the user is prompted to confirm that the screen content may be recorded. This means that it is not possible to record the screen without letting the user know. In the version tested, there was no indication that Temu is using the MediaProjection interface.

Another option for capturing the images of other apps would be the PixelCopy API, which has existed since Android 7.0 (API level 24) [22]. According to analyses in the Temu

app, this is not used either.

4.4.3.3 Assessment

The use of the methods mentioned in the Grizzly report is legitimate. For example, they can be used to determine the size of the window. The methods cannot access the displays of other apps unless the other applications explicitly allow it. Therefore, no danger arises from it.

There is also no indication that the Temu app accesses the screen content of other apps using other known mechanisms, such as media projection or pixel copy APIs.

4.4.3.4 Recommendations

As a preventive measure, it is recommended to review the app's permissions. In addition, the privacy report displays which permissions the app has used recently. It is also advisable to read messages or confirmations for screen capture (or other permissions) carefully and only confirm if the purpose is clear and desired.

4.4.4 Storage of MAC addresses

According to the Grizzly report, the Temu app stores the MAC address of the mobile device [3].

4.4.4.1 Background

The Grizzly report describes how the Temu app reads and stores the MAC address of the mobile device. It is also claimed that this MAC address could potentially be used to launch a DDoS attack against the device over the internet [3].

4.4.4.2 Evidence

No evidence for the collection of the MAC address was identified during this analysis. Neither in the decompiled code nor in the communication between the Android and iOS apps. It is possible that this functionality has been removed by Temu.

4.4.4.3 Impact

Even if the MAC address could be read, it could not be used for a DDoS attack over the internet, in contrast to what is described in the Grizzly report. Devices can only be accessed via their MAC addresses within the same local network. In addition, iOS and Android assign random MAC addresses for each network by default, so the address changes depending on the network being used.

4.4.4.4 Recommendation

No measures are necessary in this area.

4.4.5 Re-compiling of programme packages

According to the Grizzly report, the Temu Android app can dynamically compile new program packages [3].

4.4.5.1 Background

The Grizzly report states that the Temu app for Android uses dynamic compilation via the `cmd package compile` command, executed by `runtime.exec()`, which allows it to create new programs directly on the user's device. This method hides the executable files from security scans and potentially allows the app to bypass Google Play Store checks without being detected.

4.4.5.2 Evidence

This feature is not present in the tested version of Temu. Furthermore, the claim is unjustified for another reason: this Android functionality only allows the existing program package to be recompiled, for example to optimise it for the processor used and thus achieve better performance [23].

4.4.5.3 Recommendations

No measures are necessary in this area.

4.4.6 Android app permissions

According to the Grizzly report, the Android app uses questionable permissions [3].

4.4.6.1 Background

The Grizzly report shows that the app uses the permissions `CAMERA`, `RECORD_AUDIO`, `WRITE_EXTERNAL_STORAGE`, `INSTALL_PACKAGES`, and `ACCESS_FINE_LOCATION` in the programme code without these being declared in the Android manifest. It also mentions that the app demands the permissions `INTERNET` and `WAKE_LOCK`, in contrast to other comparable apps such as Shein, TikTok or eBay [3].

4.4.6.2 Evidence

It is not possible to access system features that require permissions but are not listed in the Android manifest [24]. The `CAMERA`, `RECORD_AUDIO`, `WRITE_EXTERNAL_STORAGE`, and `INSTALL_PACKAGES` permissions are not declared in the manifest and therefore cannot be used.

The permissions `ACCESS_FINE_LOCATION`, `INTERNET`, and `WAKE_LOCK` are declared in the manifest of the Android app; but this is common practice and, contrary to the Grizzly report, is also used by other apps such as Galaxus [25], Shein [11], TikTok [26] and eBay [27].

The iOS app only requests permissions that are explainable and relevant to its functionality as well:

- `NSLocationWhenInUseUsageDescription`: Localization
- `NSSupportsLiveActivities`: Display of live activities
- `NSCameraUsageDescription`: Camera
- `UIRequiresFullScreen`: Display in full screen

4.4.6.3 Recommendations

It is recommended to use Temu on up-to-date operating systems.

5 Test cases

This section presents all the test cases considered during the security analysis. Findings resulting from a specific test case are linked below its brief description. If no findings are linked, no relevant vulnerabilities were identified within the timeframe of the analysis. For test cases that apply only to a subset of components, the respective components are explicitly listed.

5.1 Network communication

The communication test cases focus on the data exchanged between the Temu app and the Temu backend, based on an expected use of the Temu app by legitimate users without malicious intent.

TC 1 Encrypted data transmission

Is network traffic encrypted?

Risk: If the network traffic is not encrypted, it is easy for attackers to read or manipulate the content of the communication.

Findings: *4.1.2*

TC 2 Encrypted data transmission with secure protocols

Is network traffic encrypted using secure protocols?

Risk: When network traffic is transmitted using insecure protocols, it is easier for attackers to read and manipulate the content of the communication.

TC 3 Create user account and initial identification

What account registration options are offered? How is the identity of the users verified?

Risk: If the identity of users is not sufficiently verified, it is possible for attackers to create accounts under someone else's name or take control over others' accounts. This allows sensitive data to be viewed or false information to be spread.

Findings: *4.3.2*

TC 4 Temu app in the foreground

While the Temu app is running in the foreground, d, it is possible to collect data with permissions already granted, such as for precise GPS location. Is this capability overused by sending data to a Temu backend?

Risk: If data is unexpectedly sent to a Temu backend without any apparent benefit to users, this can be an indication of surveillance.

Findings: *4.1.2*, *4.1.3*, *4.2.2*

TC 5 Temu app in the background

Is unexpected data sent to a Temu backend while the Temu app is active in the background?

Risk: Unexpected data transmission in the background can be an indication of surveillance.

5.2 Privacy and Data Protection

In the privacy and data protection testcases listed below, particular focus was placed on how the Temu app handles the permissions of the mobile operating systems. Concretely, which permissions are requested and when, and whether the data received is transmitted to the Temu backend.

TC 6 Geolocation tracking

Is the current geolocation of the device captured and transmitted to the Temu backend? Is the GPS position required for the use of the Temu app?

Risk: Geolocation data can be used to create movement profiles. The more numerous and precise the data points are available, the more accurately users can be monitored. Movement profiles can enable conclusions to be drawn about place of residence, place of work, preferences, and habits, etc.

Findings: *4.1.3*, *4.2.2*

TC 7 Contacts access

Is access to the contacts mandatory in order to use the Temu app? At what times are the contacts requested and transmitted?

Risk: Collecting contact information enables Temu to create user profiles and relationship patterns of uninvolved third parties - persons who do not use Temu.

TC 8 Calendar access

Does the Temu app need access to the users' calendar?

Risk: Calendar information can contain sensitive information about the users' daily schedule and activities, and is therefore well suited for monitoring. Appointments can also contain valuable additional information, such as the persons participating, their contact details, locations, activities, etc

TC 9 Camera access

At what times is the camera used? Is the data transferred directly to the Temu backend?

Risk: The camera can be used to record images or videos unnoticed. This image information can provide information about the environment in which users are currently located, which persons are in the vicinity, or which activities are being carried out. Under certain circumstances, the images can also be used for blackmailing.

TC 10 Microphone access

At what times is the microphone used? Is the data transmitted directly to the Temu backend?

Risk: The microphone can be used for unnoticed audio recordings. These recordings can provide information about the current environment in which the users are located and reveal the content of – possibly confidential – conversations. Under certain circumstances, the recordings can also be used for blackmailing.

TC 11 Access to external storage (Android)

When is the external storage accessed? Are only the selected files read or also other contents?

Risk: The Temu app can access and process locally stored data and transfer it to a Temu backend without the knowledge and explicit consent of the user. Among other things, this data may include confidential and personal details.

TC 12 Local network access

Is access to the local network requested?

Risk: The permission allows interaction with local network devices, such as smart home devices. Under certain circumstances, it could be possible to control such local network devices or read out sensitive information.

TC 13 Screenshots of third-party apps

Does the app try to capture the screen of other apps?

Risk: Capturing content from other apps can be used to collect sensitive information such as credentials, messages, banking data or personal information.

Findings: *4.4.3*

TC 14 Collect System Information

Is information about the smartphone sent to a Temu backend? Is information about installed or running apps collected and transmitted to the Temu backend?

Risk: The transfer of system information enables the creation of a user profile and the surveillance of user activities. This is particularly important if users use multiple profiles (e.g. private and professional) on the same device.

Findings: *4.1.2, 4.2.3, 4.3.4, 4.4.1, 4.4.4*

TC 15 Data extract according to Federal Act on Data Protection

If users exercise their right to information and request a complete data export, does the export contain user data that was collected without a legitimate purpose?

Risk: Data should only be collected and stored to the extent necessary for the purpose in question.

TC 16 Clipboard access

Does the Temu app access the clipboard without user interaction? Is the data from the clipboard automatically transferred to the Temu backend?

Risk: The clipboard may contain confidential information such as passwords.

TC 17 Using an integrated browser

Does the app use an integrated browser? If so, what is its purpose? Does it have any unusual behavior?

Risk: An integrated browser can be extended by almost any functionality. This could, for example, enable the recording of displayed content or user input. This could capture potentially sensitive data and transmit it to a Temu Backend.

Findings: *4.2.2*

TC 18 Using the app in 'Lockdown Mode' (iOS)

Can the iOS app be used in Lockdown Mode?

Risk: The Lockdown Mode reduces the collection and processing of data by apps, which is important for users with heightened data protection requirements. An app that does not work in Lockdown Mode can result in more user data being disclosed or exposed.

5.3 Security of mobile apps

TC 19 Differences between iOS and Android

Are there significant differences in the behaviour of the Android and iOS apps?

Risk: Different behaviour can result in the app implementing less strict privacy policies on one platform compared to the other.

TC 20 Indicators of vulnerability exploitation

Does the decompiled code of the apps show any indications that the app exploits known vulnerabilities? (This was explicitly checked, as the Grizzly report claims [3]).

Risk: Exploiting vulnerabilities can allow an app to potentially access sensitive information or install malware, leading to severe consequences for the user, such as data loss or identity theft.

References

[1] M. Kälin, "Temu, Shein und Co. liefern täglich Hunderttausende Päckli in die Schweiz," Blick. Accessed: Nov. 05, 2024. [Online]. Available: https://www.blick.ch/wirtschaft/temu-shein-co-ueberfluten-die-schweiz-mit-billigware-taeglich-landen-bis-zu-einer-halben-million-paeckli-aus-asien-in-zuerich-id20076890.html

[2] N. Kaufman, "Shein, Temu, and Chinese e-Commerce: Data Risks, Sourcing Violations, and Trade Loopholes," Apr. 2023, [Online]. Available: https://www.uscc.gov/sites/default/files/2023-04/Issue_Brief-Shein_Temu_and_Chinese_E-Commerce.pdf

[3] Grizzly Research, "We believe PDD is a Dying Fraudulent Company and its Shopping App TEMU is Cleverly Hidden Spyware that Poses an Urgent Security Threat to U.S. National Interests – Grizzly Research LLC." Accessed: Oct. 28, 2024. [Online]. Available: https://grizzlyreports.com/we-believe-pdd-is-a-dying-fraudulent-company-and-its-shopping-app-temu-is-cleverly-hidden-spyware-that-poses-an-urgent-security-threat-to-u-s-national-interests/

[4] Grizzly Research, "Our Track Record – Grizzly Research LLC." Accessed: Nov. 08, 2024. [Online]. Available: https://grizzlyreports.com/performance-history/

[5] DEKRA Testing and Certification, S.A.U., "Security Evaluation Report," Feb. 2024. Accessed: Nov. 05, 2024. [Online]. Available: https://appdefensealliance.dev/reports/com.einnovation.temu_170687222716597 3.pdf

[6] J. Schmidli, K. Schumacher, and P. Albisser, "Tracking mit Ortungsdiensten - Der Spion in unseren Handys," Schweizer Radio und Fernsehen (SRF). Accessed: Nov. 15, 2024. [Online]. Available: https://www.srf.ch/news/schweiz/tracking-mit-ortungsdiensten-der-spion-in-unseren-handys

[7] S. Ehrbar, "Temu liefert weniger Mehrwertsteuer ab als die Schweizer Konkurrenz – warum das legal ist," watson.ch. Accessed: Nov. 05, 2024. [Online]. Available: https://www.watson.ch/!345235115

[8] B. Egger, "Beschwerde gegen Temu beim SECO eingereicht," HANDELSVERBAND.swiss. Accessed: Nov. 05, 2024. [Online]. Available: https://handelsverband.swiss/news/beschwerde-gegen-temu-beim-seco-eingereicht/

[9] Exodus, "Report for in.amazon.mShop.android.shopping 28.18.2.300," εxodus. Accessed: Nov. 07, 2024. [Online]. Available: https://reports.exodus-privacy.eu.org/en/reports/in.amazon.mShop.android.shopping/latest/

[10] Exodus, "Report for ru.aliexpress.buyer 8.20.595.1660743," εxodus. Accessed: Nov. 07, 2024. [Online]. Available: https://reports.exodus-privacy.eu.org/en/reports/ru.aliexpress.buyer/latest/

[11] Exodus, "Report for SHEIN," εxodus. Accessed: Nov. 06, 2024. [Online]. Available: https://reports.exodus-privacy.eu.org/de/reports/519955/

[12] Exodus, "Report for com.einnovation.temu 3.11.0," εxodus. Accessed: Nov. 07, 2024. [Online]. Available: https://reports.exodus-privacy.eu.org/en/reports/com.einnovation.temu/latest/

[13] Bundesamt für Justiz, "Schweizerische Anerkennung von Staaten, die einen angemessenen Datenschutz gewährleisten." Accessed: Nov. 08, 2024. [Online]. Available: https://www.bj.admin.ch/bj/de/home/staat/datenschutz/internationales/anerkennung-staaten.html

[14] S. Meier, "So schützen Sie Ihr Handy vor Tracking," Schweizer Radio und Fernsehen (SRF). Accessed: Nov. 15, 2024. [Online]. Available: https://www.srf.ch/news/schweiz/anleitung-gegen-tracking-drei-einfache-

schritte-um-ihre-handy-daten-zu-schuetzen
[15] C. Hebeisen, "What Is Lockdown Mode for iOS and iPadOS and Why Should I Care?," What Is Lockdown Mode for iOS and iPadOS and Why Should I Care? Accessed: Oct. 18, 2024. [Online]. Available: https://www.lookout.com/blog/apple-lockdown-mode
[16] Android Developers, "ActivityManager," Activity Manager | Android Developers. Accessed: Oct. 29, 2024. [Online]. Available: https://developer.android.com/reference/android/app/ActivityManager
[17] Android Developers, "READ_LOGS Permission," Android Developers. Accessed: Nov. 07, 2024. [Online]. Available: https://developer.android.com/reference/android/Manifest.permission#READ_LOGS
[18] Android Developers, "Introduction to activities," Android Developers. Accessed: Nov. 06, 2024. [Online]. Available: https://developer.android.com/guide/components/activities/intro-activities
[19] Android Developers, "View," Android Developers. Accessed: Nov. 06, 2024. [Online]. Available: https://developer.android.com/reference/android/view/View
[20] Android Developers, "<activity>," Android Developers. Accessed: Nov. 06, 2024. [Online]. Available: https://developer.android.com/guide/topics/manifest/activity-element
[21] Android Developers, "Media projection | Android media," Android Developers. Accessed: Nov. 06, 2024. [Online]. Available: https://developer.android.com/media/grow/media-projection
[22] Android Developers, "PixelCopy," Android Developers. Accessed: Nov. 07, 2024. [Online]. Available: https://developer.android.com/reference/android/view/PixelCopy
[23] Android Open Source Project, "Implement ART just-in-time compiler," Android Open Source Project. Accessed: Nov. 07, 2024. [Online]. Available: https://source.android.com/docs/core/runtime/jit-compiler#force-compilation-of-a-specific-package
[24] Android Developers, "App manifest overview | Android Developers." Accessed: Nov. 06, 2024. [Online]. Available: https://developer.android.com/guide/topics/manifest/manifest-intro
[25] Exodus, "Report for com.galaxusapp 4.13.0," εxodus. Accessed: Nov. 06, 2024. [Online]. Available: https://reports.exodus-privacy.eu.org/en/reports/com.galaxusapp/latest/
[26] Exodus, "Report for Tiktok," εxodus. Accessed: Nov. 06, 2024. [Online]. Available: https://reports.exodus-privacy.eu.org/de/reports/449993/
[27] Exodus, "Report for com.ebay.mobile 6.182.0.1," εxodus. Accessed: Nov. 06, 2024. [Online]. Available: https://reports.exodus-privacy.eu.org/de/reports/517988/