Nationales Testinstitut für Cybersicherheit

Technische Sicherheitsanalyse Mobile App «Temu»

Begutachtung der Sicherheitsrisiken aus Schweizer Perspektive

Version	1.0
Datum	05. Dezember 2024
Klassifikation	Öffentlich
Autoren	Tobias Castagna, Patrik Fabian, Andreas Leisibach, Dilip Many, Fabio Zuber
Verantwortlich	Tobias Castagna

Inhaltsübersicht

1 Management Summary

1.1 Ausgangslage und Hintergrund

Der Online-Handel ist bei vielen Schweizerinnen und Schweizern sehr beliebt und immer mehr Menschen verlagern grosse Teile ihrer Einkäufe ins Internet. Während in der Vergangenheit vor allem bei inländischen Anbietern oder im benachbarten Ausland bestellt wurde, gewinnen Online-Händler aus Fernost wie Temu, Ali Express oder Shein zunehmend an Bedeutung. Temu, ein Tochterunternehmen der chinesischen PDD Holdings Inc., hat sich eine bedeutende Position im Schweizer Online-Handel erarbeitet: Ihre mobile App gehört mittlerweile zu den beliebtesten in den App-Stores von Google und Apple.

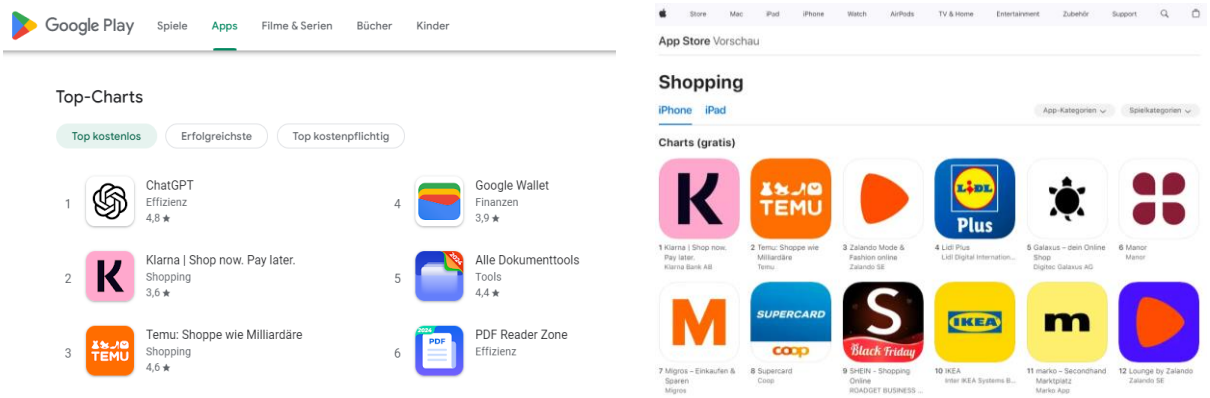

Abbildung 1: Google Play und Apple Store, Stand 31. Oktober 2024

Dass die Temu-App in der Schweiz rege genutzt wird, zeigen nicht nur die zahlreichen Downloads, sondern auch die vielen Lieferungen. Ein Beleg dafür ist die Flut von Paketen aus Fernost, die den Flughafen Zürich erreichen. Täglich passieren Hunderttausende von Sendungen den Flughafen, die meisten davon von Anbietern wie Temu und Shein [1].

Insbesondere in Zusammenhang mit Temu wurden in jüngster Zeit Bedenken in Bezug auf die Cybersicherheit laut: Mehrere Organisationen und Behörden auf der ganzen Welt haben den Vorwurf erhoben, dass die Temu-App ihre Nutzenden ausspioniert oder ein Verhalten aufweist, das mit Malware vergleichbar ist [2]. Diese Vorwürfe sind jedoch häufig nicht belegt oder basieren weitgehend auf einer viel zitierten Veröffentlichung von Grizzly Research LLC [3]. Diese Investmentgesellschaft mit Sitz in den USA ist für ihre Short-Selling-Berichte bekannt. Sie hat ein wirtschaftliches Interesse an fallenden Aktienkursen und ist daher nicht neutral [4]. Derartige Berichte beruhen oft auf spekulativen oder schwer überprüfbaren Informationen.

Gleichzeitig kursieren im Internet Zertifikate, die der App angeblich ein hohes Sicherheitsniveau attestieren. Diese wurden jedoch von Temu selbst beauftragt und sind bei näherer Betrachtung oberflächlich und wenig aussagekräftig [5].

Für die Nutzerinnen und Nutzer ist es schwer zu erkennen, ob eine App ein problematisches Verhalten aufweist. Dafür sind spezifische technische Kenntnisse und Fähigkeiten erforderlich. Dies ist insofern problematisch, als dass nahezu alle Schweizerinnen und Schweizer internetfähige Mobiltelefone besitzen, ja sogar fast ständig mit sich führen. Diese Geräte verfügen über Sensoren, die grundsätzlich eine präzise und lückenlose Überwachung ermöglichen. Bösartige mobile Apps könnten beispielsweise Standortdaten erfassen und ein Bewegungsprofil erstellen [6]. Über das

Mikrofon und die Kamera können vertrauliche Gespräche oder Informationen über die Umgebung aufgezeichnet werden.

Um eine sachliche Diskussion über die Risiken von mobilen Apps zu ermöglichen, besteht der Bedarf, eine fundierte und unabhängige technische Sicherheitsanalyse solcher Apps durchzuführen.

Ziel des vorliegenden Berichts ist es, die behaupteten Risiken der Temu-App zu verifizieren oder zu widerlegen und bisher unbekannte Risiken zu identifizieren. Darauf aufbauend werden Empfehlungen für einen sicheren Umgang mit der Temu-App erarbeitet. Es kommen verschiedene Analysemethoden wie dynamische Verhaltensanalysen, Netzwerkverkehrsanalysen und Reverse Engineering zum Einsatz.

Die vorliegende Untersuchung wurde vom NTC aus eigener Initiative, mit eigenen Ressourcen und somit ohne externe Einflussnahme durchgeführt. Der Fokus liegt ausschliesslich auf Themen im Zusammenhang mit Cybersicherheit und Nutzerüberwachung. Andere oft kritisierte Aspekte im Zusammenhang mit Temu werden nicht berücksichtigt. Dazu gehören die Qualität und Sicherheit der Produkte sowie bestimmte problematische Geschäftspraktiken. Beispiele dafür sind die Vermeidung der Mehrwertsteuer [7] oder die fehlende Beteiligung an den Entsorgungs- oder Recyclingkosten über die vorgezogene Recyclinggebühr [8]. Diese Themen liegen ausserhalb der Expertise des NTC.

1.2 Zusammenfassende Einschätzung

Im Rahmen der Analyse konnten keine eindeutig kritischen Sicherheitslücken oder Beweise für eine Benutzerüberwachung in der Temu-App festgestellt werden. Im Allgemeinen entsprechen das beobachtete Verhalten und die angeforderten Berechtigungen dem, was von einer typischen E-Commerce-Anwendung erwartet werden kann. Im Vergleich zu anderen beliebten Apps von Mitbewerbern wie Amazon [9], AliExpress [10] oder Shein [11] verfügt die Temu-App [12] sogar über weniger und unproblematischere Berechtigungen.

Die Überprüfung der App hinsichtlich des Zugriffs auf die für die Benutzerüberwachung besonders geeigneten Gerätesensoren wie Mikrofon, Kamera und GPS ergab, dass die App hierfür überwiegend keine Berechtigungen besitzt. Sie wird vom Betriebssystem am Zugriff gehindert. Dort, wo sie über die notwendigen Berechtigungen verfügt, wie z.B. bei den Standortdaten, wurden die Sensordaten nicht zu unerwarteten Zeitpunkten abgefragt. Zugriffe auf die Sensordaten erfolgten nur, wenn sie durch nachvollziehbare Benutzerinteraktionen initiiert wurden. Es wurden keine Hintergrundprozesse identifiziert, die ohne Wissen des Benutzers auf diese Sensoren zugreifen.

Trotz des Fehlens unmittelbarer Beweise für böswillige Aktivitäten wurden ungewöhnliche Beobachtungen gemacht. Diese "Red Flags" sind kritisch zu betrachten. Sie wurden in keinem der bisher bekannten Berichte erwähnt.

So verfügt die App über die Fähigkeit, JavaScript-ähnlichen Code dynamisch nachzuladen. Dadurch kann das Verhalten der App zur Laufzeit geändert werden. Dies gibt den Entwicklern die Flexibilität, Funktionalitäten und Inhalte anzupassen, ohne das ursprüngliche App-Paket über den App Store aktualisieren zu müssen. Auf diese Weise kann die App ihr Verhalten an bestimmte Bedingungen oder Umgebungen anpassen. Dies macht es schwierig, alle möglichen Verhaltensweisen vorherzusagen oder zu überprüfen. Die Fähigkeit, Code dynamisch nachzuladen, ist an sich nicht besonders ungewöhnlich – auch andere Apps tun dies. Auffällig ist die Verwendung einer unbekannten, proprietären JavaScript-Laufzeitumgebung, die bei keiner anderen App beobachtet wurde.

Kritisch zu bewerten ist auch die Verwendung von zusätzlichen Verschlüsselungsschichten an mehreren Stellen. Sie ergänzen die üblichen Verschlüsselungen, wie z.B. TLS bei der Kommunikation mit den Backend-Servern. Solche Massnahmen können legitim sein und dazu beitragen, den Schutz der Benutzerdaten vor unberechtigtem Zugriff zu erhöhen. Es besteht jedoch die Gefahr, dass die Verschlüsselung zur Verschleierung unerwünschter Datenübertragungen genutzt wird. Im Rahmen der Überprüfung war es nicht immer möglich, diese Verschlüsselungen aufzubrechen und die übertragenen Daten zu analysieren. Es kann daher nicht abschliessend beurteilt werden, welche Daten von der App verarbeitet und an die Server von Temu gesendet werden.

Da grundsätzlich nur das Vorhandensein von Schwachstellen oder Benutzerüberwachung nachgewiesen werden kann, nicht aber deren Abwesenheit, ist eine pauschale Unbedenklichkeitserklärung nicht möglich. Eine Überwachung der Nutzenden durch die App wäre technisch grundsätzlich möglich. Die App könnte versteckte Überwachungsfunktionen enthalten, die nur unter bestimmten Bedingungen, etwa an bestimmten Orten oder zu bestimmten Zeiten, nachgeladen oder aktiviert werden. Dies gilt jedoch grundsätzlich auch für viele andere Apps. Im Fall von Temu wurden keine konkreten Anzeichen dafür beobachtet.

Es ist zudem zu beachten, dass Temu und die Muttergesellschaft PDD chinesischem Recht unterliegen. Dieses gewährleistet aus europäischer Perspektive keinen angemessenen Datenschutz [13]. Die Standards in China unterscheiden sich stark von jenen in der EU und der Schweiz, wo Gesetze wie die DSGVO die Rechte der Nutzenden schützen. Staatliche Stellen haben in China leichteren Zugang zu persönlichen Daten und Unternehmen sind oft verpflichtet, Daten an Behörden weiterzugeben.

Insgesamt zeigen die Ergebnisse der Analyse, dass die im viel zitierten Grizzly-Bericht [3] beschriebenen Risiken oft übertrieben, in einen falschen Kontext gestellt oder fachlich falsch sind (siehe Relativierungen in *Kapitel 4.4* weiter unten). Sie zeichnen ein deutlich negativeres Bild als es aus Sicht NTC sachlich gerechtfertigt wäre.

Zusammenfassend wird empfohlen, den Einsatz der Temu-App kritisch zu hinterfragen. Dies gilt insbesondere im geschäftlichen und behördlichen Kontext oder bei besonders exponierten Personen. Diese Empfehlung gilt grundsätzlich auch für andere Apps, deren Nutzen im geschäftlichen und behördlichen Kontext begrenzt ist. Werden solche Apps dennoch eingesetzt, sollten technische und organisatorische Massnahmen getroffen werden. Dazu gehören die Vergabe der nur unbedingt notwendigen Berechtigungen [14], die Verwendung eines aktuellen Betriebssystems und die Beschränkung der Nutzung der App auf das notwendige Minimum. Alternativ könnte der Zugriff auf den Dienst auch über den Browser des mobilen Endgerätes erfolgen. Hier sind die Angriffsfläche und die Möglichkeiten zur permanenten Benutzerüberwachung geringer.

2 Umfang und Einschränkungen der Sicherheitsanalyse

In diesem Abschnitt wird der Umfang der durchgeführten Sicherheitsanalyse beschrieben. Dabei wird auch auf die selbst auferlegten sowie die technischen und ressourcenbedingten Einschränkungen eingegangen. Es folgt eine Übersicht über die wichtigsten Punkte, gefolgt von einer detaillierten Erläuterung.

2.1 Umfang der Analyse im Überblick

Bei der Überprüfung wurde der Fokus auf Risiken für die individuelle Privatsphäre sowie auf Überwachung und Spionage gesetzt. Weiterführende detaillierte Analysen wie beispielsweise Langzeitverhaltensbeobachtungen wurden nicht durchgeführt.

Die bei der Analyse berücksichtigten Testfälle sind in *Kapitel 5* aufgelistet. Die folgende Auflistung beschreibt den groben Umfang der Analyse:

- Kommunikation zwischen den mobilen Apps und dem Temu Backend
- Angefragte Berechtigungen der Apps und Zugriff auf Sensoren wie Kamera, Mikrofon und GPS
- Dekompilierter Quellcode der Mobile-Apps (Reverse Engineering)

Folgende Bereiche und Aspekte wurden in dieser Analyse bewusst **nicht** untersucht:

- Die öffentliche Webseite von Temu sowie andere nicht aufgeführte Plattformen
- Die Wirksamkeit der von den Betriebssystemen angebotenen Schutzfunktionen, insbesondere die Einhaltung der Berechtigungen einer App
- Mögliche Weitergabe der erhobenen (personenbezogenen) Daten durch PDD an Dritte, auch an den chinesischen Staat
- Sicherheit und Qualität der Produkte, welche über Temu erworben werden können
- Psychologische Faktoren, Dark Patterns usw., die dazu verleiten sollen, Produkte bei Temu zu bestellen

Das folgende Diagramm zeigt eine schematische Übersicht all jener Komponenten, die Teil der vorliegenden Sicherheitsanalyse sind.

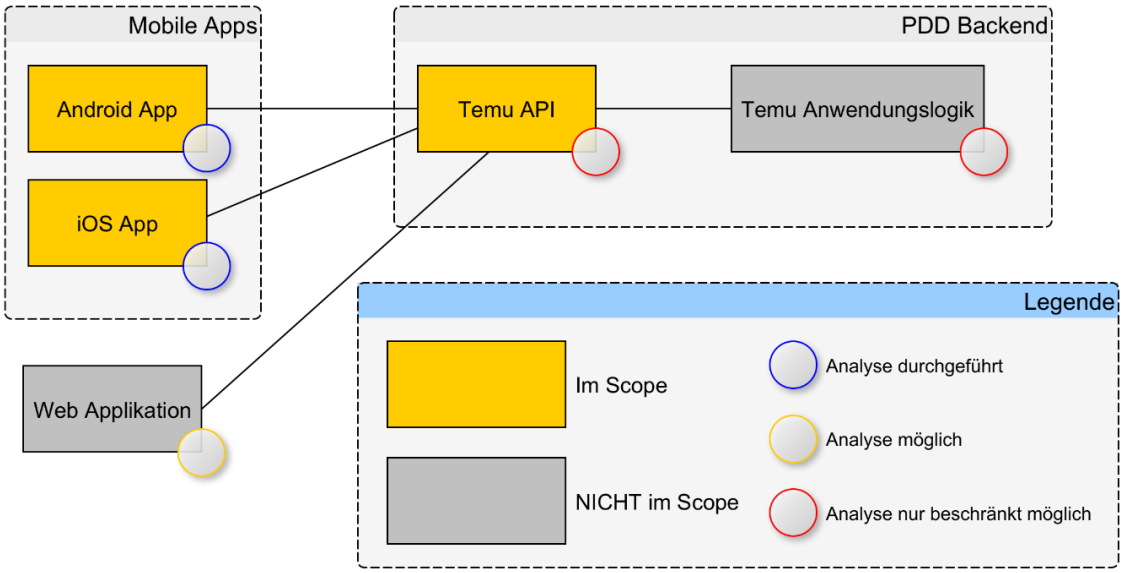

Abbildung 2: Übersicht der Komponenten von Temu

2.2 Umfang der Analyse im Detail

Die Überprüfung wurde auf Initiative des NTC durchgeführt. Das NTC hat die für die Überprüfung erforderlichen Ressourcen zur Verfügung gestellt und die Ziele, den Umfang und die Rahmenbedingungen der Überprüfung festgelegt. Es gibt keinen externen Auftraggeber. Temu und die Muttergesellschaft PDD hatten keine Kenntnis von der Überprüfung und konnten daher keinen Einfluss darauf nehmen.

Die Überprüfung fand zwischen September und November 2024 statt und wurde hauptsächlich von einem Kernteam von drei Testexperten des NTCs durchgeführt. Insgesamt wurden rund 60 Personentage für Recherche, Analyse und Dokumentation aufgewendet. Bei der Analyse wurde auf möglichst realitätsnahe Testbedingungen geachtet ohne besondere Schutzmassnahmen, wie sie beispielsweise durch restriktiv konfigurierte Mobile Device Management (MDM) Lösungen oder stark gehärtete Endgeräte möglich wären.

Die zur Verfügung stehende Zeit und Ressourcen wurden genutzt, um kritische und praxisrelevante Risiken zu analysieren. Weiterführende Beobachtungen des Langzeitverhaltens über Wochen, Monate oder Jahre wurden nicht durchgeführt. Auch Untersuchungen unter besonderen Bedingungen oder an sensiblen Orten, wie z.B. im Bundeshaus oder an militärischen Standorten, fanden nicht statt. Solche weitergehenden Untersuchungen könnten auf mögliche fortgeschrittene Überwachungstechniken hinweisen, sofern diese vorhanden sind.

Es ist im Allgemeinen nicht bekannt, was mit den übermittelten Daten bei PDD geschieht. Eine Überprüfung ist ohne eine umfassende Kooperation des Anbieters nicht möglich. Es kann daher keine Aussage darüber getroffen werden, wie die Daten und Metadaten nach der Übermittlung an PDD weiterverarbeitet werden. Auch die physischen Standorte der Backend-Server wurden nicht näher betrachtet, da dies kein entscheidendes Kriterium ist, um eine Aussage darüber zu treffen, an wen die Daten letztendlich gesendet werden bzw. wer Zugriff auf die Daten hat. Relevant ist jedoch, dass es sich beim Datenverarbeiter PDD um ein Unternehmen unter chinesischer Gesetzgebung handelt.

Die Wirksamkeit der von den Betriebssystemen angebotenen Schutzfunktionen, insbesondere die Durchsetzung der Berechtigungen einer App, wurde nicht überprüft. Bei der Bewertung wurde vorausgesetzt, dass diese, wie von den jeweiligen Herstellern in der Dokumentation dokumentiert, voll wirksam sind.

Bei der Überprüfung wurde der Fokus auf Risiken für die individuelle Privatsphäre sowie auf Überwachung und Spionage bei der Nutzung der Temu-App auf Android- und iOS-Mobilgeräten gelegt. Die Temu-Webseite sowie andere Plattformen wurden nicht berücksichtigt.

Um eine lückenlose Analyse des Netzwerkverkehrs zu ermöglichen, erfolgte sämtlicher Netzwerkverkehr über WLAN. Die Geräte waren nicht mit SIM-Karte ausgerüstet und konnten nicht über das Mobilfunknetz kommunizieren.

Um einen vertieften Einblick in die Funktionsweise der Apps zu ermöglichen, wurden gerootete bzw. «jailbroken» Android- und iOS-Mobilgeräte verwendet. Die Tabelle zeigt, welche Geräte für die Überprüfung verwendet wurden:

Modell	Betriebssystem-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

Die Überprüfung wurde an folgenden Versionen der Temu-App durchgeführt:

- Android Versionen: `2.95.0, 2.97.0, 2.99.0`
- iOS Version: `2.97.0`

Die Apps wurden in der Schweiz über den offiziellen App Store des jeweiligen Betriebssystems bezogen:

- 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*

Es sei an dieser Stelle ausdrücklich darauf hingewiesen, dass es sich bei der Überprüfung um eine Momentaufnahme handelt. Allfällige Anpassungen der App vor oder nach der Überprüfung sind nicht erfasst. Gleiches gilt für allfällige App-Varianten, die in anderen Ländern oder Sprachregionen zum Einsatz kommen oder aus anderen Quellen bezogen werden.

3 Befundliste

Im Folgenden werden alle Befunde aufgeführt und in eine von vier Kategorien gruppiert: Hohe Risiken, Mittlere Risiken, Tiefe Risiken und Relativierungen. Alle Befunde werden im Detail in *Kapitel 4* behandelt.

3.1 Hohe Risiken

Befunde in dieser Kategorie bergen je nach Anwender und Umständen ein hohes Risiko für die Anwender. Es wird empfohlen, das individuelle Risiko schnellstmöglich zu analysieren und bei Bedarf geeignete Massnahen zu treffen, um das Risiko auf ein akzeptiertes Mass zu reduzieren.

3.2 Mittlere Risiken

Befunde in dieser Kategorie bergen je nach Anwender und Umständen ein mittleres Risiko für die Anwender. Es wird empfohlen, das individuelle Risiko zu analysieren und bei Bedarf geeignete Massnahen zu treffen, um das Risiko auf ein akzeptiertes Mass zu reduzieren.

3.3 Tiefe Risiken

Befunde in dieser Kategorie bergen je nach Anwender und Umständen ein geringes Risiko für die Anwender. Es wird empfohlen, das individuelle Risiko abzuschätzen und bei Bedarf geeignete Massnahen zu treffen, um das Risiko auf ein akzeptiertes Mass zu reduzieren. Vorher sollten jedoch die Risiken der anderen Kategorien beachtet werden.

3.4 Relativierungen

Die in dieser Kategorie aufgeführten Befunde relativieren Erkenntnisse aus anderen Berichten zu Temu. Sie stellen aus Sicht des NTC für die Nutzenden entweder nur ein geringes oder kein Risiko dar.

4 Befunde im Detail

4.1 Hohe Risiken

Befunde in dieser Kategorie bergen je nach Anwender und Umständen ein hohes Risiko für die Anwender. Es wird empfohlen, das individuelle Risiko schnellstmöglich zu analysieren und bei Bedarf geeignete Massnahen zu treffen, um das Risiko auf ein akzeptiertes Mass zu reduzieren.

4.1.1 Erstellung von Java Klassen durch proprietären JavaScript Interpreter

Die Temu-App beinhaltet einen proprietären Interpreter, welcher es erlaubt, zur Laufzeit JavaScript Code zu interpretieren und auszuführen. Der Hersteller kann daher über seine Server jederzeit individuelle Code-Pakete an die App ausliefern, um die App beliebig anzupassen und zu erweitern. Dabei werden die App-Stores der jeweiligen Plattformen und allfällige Kontrollen umgangen.

4.1.1.1 Hintergrund

Im dekompilierten Quellcode der App befindet sich eine Laufzeitumgebung, die JavaScript-Code von einem Backend empfängt, interpretiert und ausführt. Dabei ist die Laufzeitumgebung auch in der Lage, Klassen in anderen Programmiersprachen (Java für Android, Swift für iOS) aus dem Code zu erzeugen.

Dies erlaubt es dem Anbieter, das Verhalten dynamisch zu ändern. So kann potenziell Code selektiv an bestimmte Endbenutzer oder in bestimmten Umgebungen übermittelt werden.

Ein möglicher Motivationsgrund für diese Funktion ist, dass Temu auf diese Weise schnell Änderungen oder Patches ohne ein komplexes und zeitaufwändiges Update der App bereitstellen kann. Allerdings kann Code, der während der Laufzeit geladen wird, schwieriger getestet werden, da es unvorhersehbar ist, welchen Code die App für andere Benutzende zu einem bestimmten Zeitpunkt ausführen wird. Dies erschwert es, Bugs oder Schwachstellen im Vorfeld zu identifizieren.

Eine eigens entwickelte JavaScript-Laufzeitumgebung bietet den Entwicklern von Temu mehr Flexibilität bei der Gestaltung der unterstützten Funktionen. Durch diese Eigenentwicklung kann der Funktionsumfang im Vergleich zu bestehenden JavaScript Interpretern gezielt auf das Wesentliche reduziert werden, was die Performance optimiert. Gleichzeitig können Interaktionen zwischen JavaScript und tieferliegenden Systemschnittstellen, wie etwa den Zugriff auf Sensoren, effizienter gestaltet werden. Warum genau keine Standard-Laufzeitumgebung verwendet wird, konnte in der Analyse nicht geklärt werden.

4.1.1.2 Nachweis

Die Mobile-App von Temu lädt `.lego` Dateien, welche JavaScript oder JavaScript-ähnlichen Quellcode enthalten. Zum Beispiel wird eine Datei von `https://static.kwcdn.com/fs-lg/lg/CameraPreNotifyPopup--a5f6b6f1c879d09bcb4c531b4882eea7.lego` geladen, welche die React JavaScript Bibliothek verwendet. Die Abbildung 3 zeigt einen Teil der Datei zur Veranschaulichung.

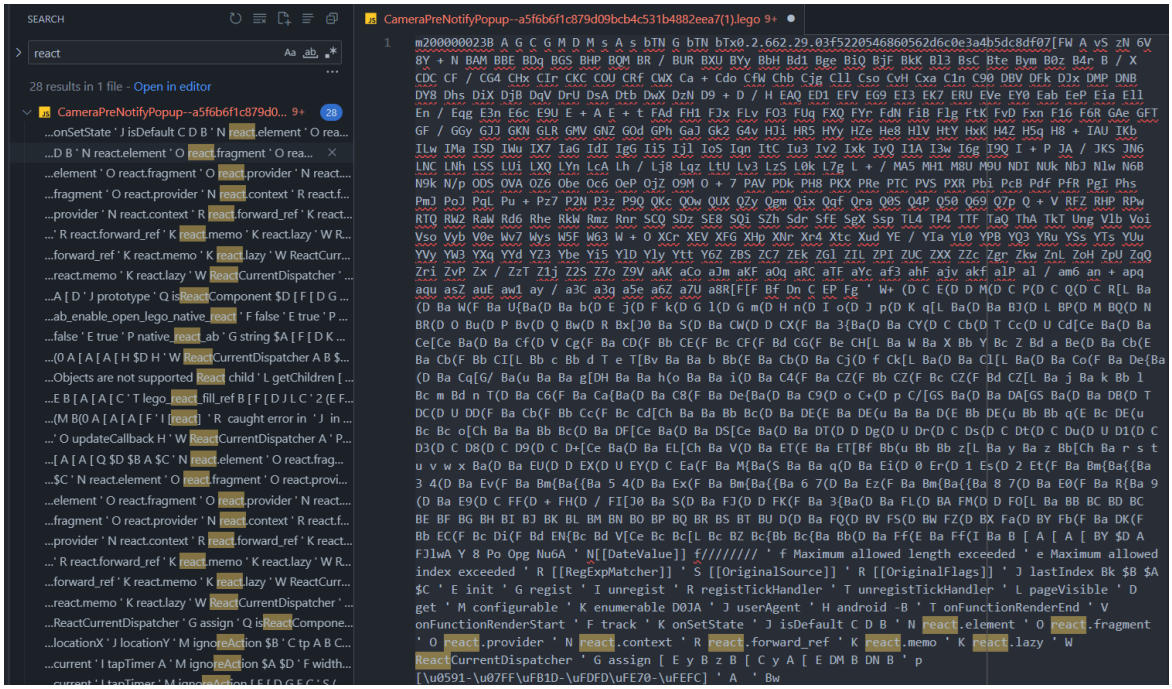

Abbildung 3: Beispiel einer .lego Datei

Der Inhalt der .lego Datei wird in der Android-App durch die Klasse `com.einnovation.whaleco.lego.m2.impl.m2.core.M2FunctionManager` behandelt und unter anderem an die `createCustomElement` Funktion übergeben:

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

Innerhalb der `createCustomElement` Funktion wird durch den Aufruf von `class.ForName(String className)`[1] ein Klassen-Objekt zurückgegeben. Der Name der Klasse, welche retourniert wird, stammt aus dem Lego-Dateiinhalt. Somit ist es der App möglich beliebige Objekte von Java Klassen zu erstellen und über Reflection statische Methoden der Klassen aufzurufen.

```
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);
    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
```

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

```
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);
}
```

Ob die Klassen zu einem späteren Zeitpunkt auch instanziiert werden, konnte in der zur Verfügung stehenden Zeit nicht bestimmt werden. Das daraus resultierende Risiko ist aber unverändert; Code kann dynamisch zur Laufzeit entgegengenommen und interpretiert werden.

Die Strings des Lego-Codes, welche durch die Android-App verarbeitet werden, können durch das folgende Frida Skript ausgegeben werden. Die Strings können helfen einen Überblick über die Funktionalität des übermittelten Codes zu erlangen.

```
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;
    }

});
```

Das Ergebnis dieses Skripts:

```
└─$ 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
```

Bei der iOS-App werden Lego-Skripte ebenfalls eingesetzt. Im Rahmen des durchgeführten Reverse Engineering konnten jedoch innerhalb der verfügbaren Zeit keine detaillierten Erkenntnisse über die Fähigkeiten der Laufzeitumgebung auf iOS-Geräten gewonnen werden.

4.1.1.3 Auswirkungen und Einschätzung

Eine Vielzahl von Apps nutzt heutzutage JavaScript, um dynamisch die integrierte App-Funktionalität anzupassen und zu erweitern. Die Android App von Shein verwendet z.B. die Laufzeitumgebung QuickJS[2] um Code zu interpretieren.

Zur Laufzeit geladener Code stellt eine Herausforderung für Sicherheitsanalysen dar. Es ist unvorhersehbar, welcher Code zu einem bestimmten Zeitpunkt für verschiedene Benutzer ausgeführt wird. Diese Dynamik erschwert die frühzeitige Identifikation von

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

Sicherheitslücken oder unerwartetem Verhalten, da nicht alle möglichen Codepfade vorhersehbar und umfassend getestet werden können.

4.1.1.4 Empfehlungen

Für Endbenutzer gibt es zum Zeitpunkt der Analyse keine direkte Möglichkeit den dynamischen Inhalt zu blockieren. Während der iOS Lockdown Modus JavaScript JIT für Safari deaktiviert, hat dies keinen Einfluss auf die Apps von Dritten [15]. Bei Android gibt es ebenfalls keine Möglichkeit die Ausführung von dynamischen Inhalten zu blockieren. Ansonsten kann die App präventiv deinstalliert und bei Bedarf über die Webseite bestellt werden.

Es wird empfohlen, weitere Analysen und Prüfung der Ergebnisse durch weitere Sicherheitsforschende zu tätigen, um ein vollständigeres Bild der potenziellen Sicherheitsrisiken zu erhalten.

4.1.2 Verschlüsselte Daten zum System werden übermittelt

Die Mobile-App sendet verschlüsselte Inhalte an ein Backend, welche Details über das Gerät enthalten, auf dem die App ausgeführt wird.

4.1.2.1 Hintergrund

Die Temu-App versendet verschlüsselte JSON-Daten an ein Backend System. Diese Daten können beispielsweise für Telemetrie oder zum Erfassen von Nutzerstatistiken genutzt werden. Sie erlauben es allerdings auch, Benutzende ohne Temu-Konto eindeutig zu identifizieren und protokollieren, ob das verwendete Gerät möglicherweise zum Analysieren der Temu-App genutzt werden kann.

4.1.2.2 Nachweis

Die Mobile-Apps für Android und iOS senden HTTP POST Anfragen zu einem `phantom` genannten API-Endpunkt.

```
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"}
```

Um die Daten vor der Verschlüsselung abzufangen, wurde das folgende Frida Skript mit der Android-App verwendet:

```
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);
        }
});
```

Die folgenden Daten zeigen den Inhalt einer Anfrage exemplarisch. Die in Rot markierten Zeilen bieten potenziell die Möglichkeit Benutzende und Endgeräte auch ohne Temu Konto eindeutig zu identifizieren.

```
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
```

Die in Grün markierten Einträge `development_enabled`, `ae` (ADB-Debugging enabled) und `secure_lock` sind mögliche Indikatoren für eine Prüfstand-Erkennung. Das Erheben dieser Daten wurde in anderen Apps ebenfalls beobachtet und kann legitime Gründe haben, um Missbrauch oder Manipulation von Daten zu erkennen und zu verhindern.

Die `user_env2` Daten, welche oben Violett hervorgehoben wurden, sind zusätzlich verschlüsselt. Den genauen Inhalt dieser Daten konnte in der zur Verfügung stehenden Zeit nicht bestimmt werden. Beim Reverse Engineering der Android App wurde festgestellt, dass der Inhalt aus der nativen Bibliothek `libUserEnv.so` stammt. Die Interaktion zwischen der nativen Bibliothek und dem Java-Code wurde mit dem folgenden Frida Skript abgefangen. So ist es möglich, den Arbeitsspeicher-Inhalt der relevanten Funktion auszugeben.

```
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();
});
```

Beispiel-Ausgabe des Skripts:

{"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 der zur Verfügung stehenden Zeit konnte die Verschlüsselung der Daten und der Inhalt des Arbeitsspeichers nicht genauer analysiert werden.

4.1.2.3 Auswirkungen

Die Details, welche übertragen werden, geben Auskunft in welchem Umfeld die App ausgeführt wird. Die Daten beinhalten keine Informationen zur Person oder dem Konto, welche die App verwendet. Allerdings konnte der Inhalt der `user_env2` Daten welche zusätzlich verschlüsselt sind, in der zur Verfügung stehenden Zeit nicht entschlüsselt werden.

Des Weiteren erlaubt das übermittelte `install_token`, die `uuid` und die `google_adid` eine eindeutige Identifizierung einer Installation.

4.1.2.4 Empfehlungen

Es wird empfohlen, weiterführende Analysen der Daten von `user_env2` durchzuführen, um den genauen Inhalt und die Art der darin gespeicherten Daten zu bestimmen. Diese Analyse könnte wichtige Informationen darüber liefern, ob und in welchem Umfang sensible oder personenbezogene Daten in diesem Feld enthalten sind.

Um das Tracking über Werbe-ID zu reduzieren, gibt es die Möglichkeit die `google_adid` (Google Advertising ID) als auch die `IDFA` (Identifier for Advertisers) auf Apple-Geräten zurückzusetzen. Allerdings ist es Temu in Theorie auch möglich, mehrere Werbe-IDs zu einem Endgerät zuzuweisen, wenn sich die übrigen Umgebungsparameter nicht verändert haben.

4.1.3 Genauer Standort kann angefragt werden

Die App ist in der Lage, den genauen Standort des Smartphone abzufragen.

4.1.3.1 Hintergrund

In Android und iOS gibt es die Möglichkeit, nur den ungefähren anstatt des genauen Standorts zu ermitteln. Der Grizzly-Bericht unterstellt Temu aber den genauen Standort abzufragen, ohne ein berechtigtes Interesse für diese Daten zu haben [3]. Die chinesische Regierung wiederum habe ein Interesse an den Standortdaten insbesondere von Regierungsmitarbeitern, Polizisten, Expats und Angehörigen von unterdrückten Minderheiten. Temu soll der Regierung die Daten liefern und diese auch weiterverkaufen und Benutzer generell ausspionieren.

Der Grizzly-Bericht behauptet zudem, dass die App bezüglich Standortbestimmungs-Berechtigung ein manipulatives Design verwendet [3]. Dabei soll die App beim Starten der Kamera-Suche die Berechtigung für die Standortbestimmung abfragen. Die Idee ist, dass zu diesem Zeitpunkt die Benutzenden eher eine Berechtigungs-Anfrage erwarten könnten (z.B. weil die Kamera den Standort für Geotagging brauchen könnte) und diese dann nicht genau lesen und die Berechtigung leichtfertig erteilen. Dieses Verhalten konnte im Rahmen der Tests jedoch nicht bestätigt werden.

4.1.3.2 Nachweis

Die App hat sowohl in der Android- als auch der iOS-Version die Möglichkeit, die Berechtigung für die genaue Standortabfrage einzufordern. Dies ist in der Android-Version in der `AndroidManifest.xml` ersichtlich:

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

Bei iOS ist dies in der `Info.plist` ersichtlich:

```
[…]
        <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>
[…]
```

Bevor die App auf diese Daten zugreifen kann, muss aber die Berechtigung bei den Benutzenden eingefordert werden. Diese Abfrage ist während dem Test aber nie erfolgt. Auch bei der Verwendung der kameragestützten Produkt-Suche wurde die Berechtigung nicht angefragt.

In der App, wird wie in der Abbildung 4 zu sehen, darauf hingewiesen, dass u.A. in der Schweiz von dieser Berechtigung nicht Gebrauch gemacht wird. Hingegen bei Benutzenden z.B. aus dem Mittleren Osten werde diese abgefragt, um bei der Erfassung der genauen Versandadresse zu unterstützen.

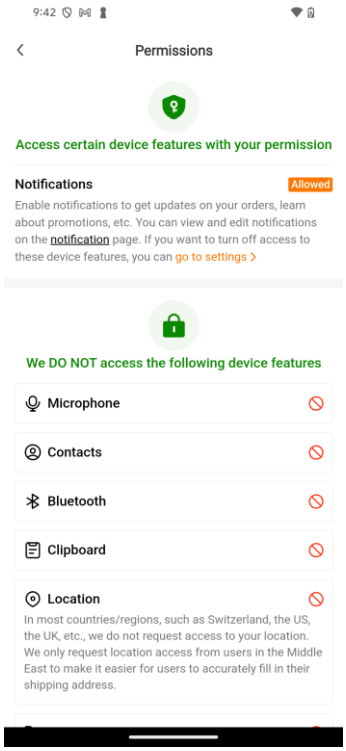

Abbildung 4: Berechtigungshinweis bezüglich Standortabfrage

4.1.3.3 Auswirkungen

Während der Tests erfolgten keine Standortabfragen. Die Möglichkeit, die Berechtigung um den genauen Standort abzufragen ist jedoch vorhanden. Daher liegt es an den Benutzenden, allfällige Anfragen, welche diese einfordern, zu bemerken und zu prüfen. Auch andere Android und iOS Apps wie z.B. Shein[11], AliExpress[10] und Amazon[9] verlangen die Berechtigung für die Standortbestimmung.

Wie die Standortdaten allenfalls genutzt würden, konnte im Rahmen dieser Analyse nicht bestimmt werden, da keine genauen Standortdaten angefragt wurden.

4.1.3.4 Empfehlungen

Es wird empfohlen, auf Aufforderungen zur Vergabe von Berechtigungen zu achten und diese nur dann zu geben, wenn ersichtlich ist, wozu diese benötigt werden und dies gewünscht ist. Zudem ist es ratsam, regelmässig den Privatsphärenbericht des Betriebssystems zu überprüfen, um festzustellen, ob die App nicht unbemerkt von Rechten Gebrauch gemacht hat.

Insbesondere Angehörige der Armee auf Stützpunkten, Journalisten und andere exponierte Personen sollten diese Empfehlungen beherzigen.

Generell wird auch empfohlen die aktuelle Version des Betriebssystems einzusetzen.

4.2 Mittlere Risiken

Befunde in dieser Kategorie bergen je nach Anwender und Umständen ein mittleres Risiko für die Anwender. Es wird empfohlen, das individuelle Risiko zu analysieren und bei Bedarf geeignete Massnahen zu treffen, um das Risiko auf ein akzeptiertes Mass zu reduzieren.

4.2.1 Eigene DNS-Implementierung

Die Mobile-App löst gewisse verwendete Domänennamen über einen eigenen Dienst auf.

4.2.1.1 Hintergrund

Zur Auflösung bestimmter Domänennamen nutzen die Android- und iOS-Apps einen eigenen Service, der über eine feste IP-Adresse per HTTPS angesprochen wird.

Diese Art der Namensauflösung könnte möglicherweise dazu dienen, zu verhindern, dass Werbung durch Werbeblocker blockiert wird.

4.2.1.2 Nachweis

Der Temu-eigene DNS-Service wird von den Mobile-Apps per HTTPS-GET Anfrage an die IP-Adresse `20.15.0.9` kontaktiert.

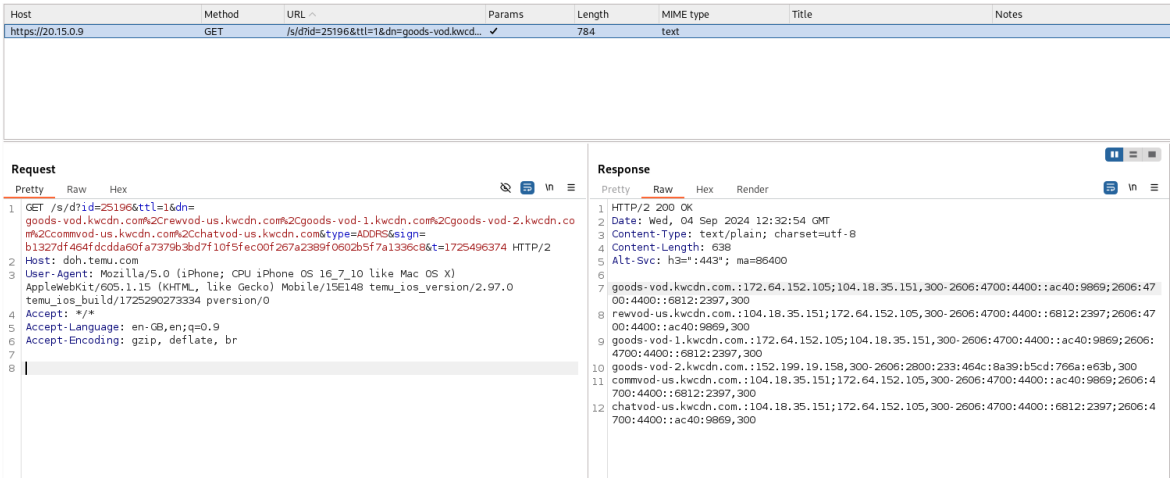

Abbildung 5: Anfrage Domainnamenauflösung

Die folgenden Domänen wurden, bei der Untersuchung, durch den eigenen Server aufgelöst:

```
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
```

Diese Hostnamen wurden auf die folgenden IP-Adressen aufgelöst:

```
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 Auswirkungen

Es besteht die Möglichkeit, dass Temu die aufgelösten IP-Adressen dynamisch wechselt. DNS und IP basierende Netzsperren werden somit umgangen. Solche Netzsperren werden unter anderen auch vom Bund für das Sperren von illegalen Glückspielangeboten verwendet. Durch den dynamischen Wechsel lassen sich zudem Werbeblocker umgehen, die Werbung und Tracking anhand von Domainnamen oder IP-Adressen erkennen und blockieren würden.

Dieses Verfahren wurde vereinzelt auch bei IoT Geräten beobachtet, wird durch die Testexperten aber nicht als gängige Praxis angesehen.

4.2.1.4 Empfehlungen

Für diesen Befund sind keine Massnamen notwendig.

Benutzende haben in der Praxis keine Möglichkeit, dieses Verhalten der Temu-App zu unterbinden.

4.2.2 Standortbestimmung per WebView

Mit Hilfe von WebViews kann der Standort der Benutzenden der App bestimmt werden.

4.2.2.1 Hintergrund

Die Android-App verwendet für die Darstellung von bestimmten Inhalten eine WebView. Diese WebView nutzt eine Brücken-Klasse, welche Zugriff auf Sensordaten und andere Java-Funktionalitäten ermöglicht. Dies erlaubt beispielsweise, dass der Standort der Benutzenden abgefragt werden kann.

4.2.2.2 Nachweis

Die Java Brückenschnittstelle `com.einnovation.whaleco.web.meepo.extension.jsapi.BridgeImpl` erlaubt es von WebViews aus, Java-Methoden aufzurufen. Die Brücken-Klasse macht es möglich, dass JavaScript-Code, welcher von Temu Servern geladen wird, auf Sensordaten und andere Java-Schnittstellen zugreifen kann.

```
@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;
}
```

Im dekompilieren Code der Methode `c()` werden rund 190 Methoden aufgelistet, welche dadurch für WebViews zur Verfügung gestellt werden. Darunter befindet sich auch `AMLocation.get`, welche es erlaubt den Standort anzufragen und diesen anschliessend zu übermitteln.

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

Die `AMLocation.get()` Methode findet sich im dekompilierten Temu Quellcode unter `com.whaleco.temu.address_map.jsapi` und ruft einen Dienst zum bestimmen des Standorts auf:

```
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());
}
```

Die iOS-App verwendet ebenfalls WebViews zum Anzeigen von dynamischen Inhalten. In der zur Verfügung stehenden Zeit konnte nicht geklärt werden, ob es der WebView in der iOS-App ebenfalls möglich ist auf Sensordaten und andere Schnittstellen zuzugreifen.

4.2.2.3 Auswirkungen

Potenziell kann über JavaScript-Code, welcher von einem Backend geladen wird, der Standort der Benutzenden abgefragt werden. Da die JavaScript-Brücke in der WebView Zugriff auf gerätespezifische Funktionen hat, wäre es prinzipiell möglich, Standortdaten abzufragen und diese an ein Backend zu übermitteln. In den durchgeführten Tests konnte jedoch keine solche Standortabfrage beobachtet werden: Weder im Privacy Report der Betriebssysteme noch im Datenverkehr zum Backend.

Mehr zur Standortabfrage findet sich in *Kapitel 4.4.0*.

4.2.2.4 Empfehlungen

Es wird empfohlen der App keinen Zugriff auf den Gerätestandort zu erteilen. Da diese Berechtigungen laut der Temu Datenschutzrichtlinie, für Personen in der Schweiz nicht verwendet wird, kann sie ohne Einschränkungen der Bedienfreundlichkeit verweigert werden.

Des Weiteren wird empfohlen, eine aktuelle Version von Android und iOS zu verwenden, welche es erlauben Zugriffe von Apps auf sensible Sensordaten aufzuzeigen.

4.2.3 Übermittlung verschlüsselter Daten mit Gerätedetails

Die Applikation sendet verschlüsselte Daten mit Details zu dem verwendeten Smartphone an das Backend.

4.2.3.1 Hintergrund

Die Android- und iOS-Anwendungen verschlüsseln bestimmte Daten mithilfe einer eigenen Verschlüsselungslogik und senden diese an den API-Endpunkt `eu.pftk.temu.com/pmm/api/pmm/api`. Durch Reverse Engineering der Android App konnte festgestellt werden, dass für die Verschlüsselung AES-GCM eingesetzt wird.

4.2.3.2 Nachweis

Die verschlüsselten Daten können auf Android mittels Frida durch das Einhaken in die Verschlüsselungslogik entschlüsselt werden.

Für die Analyse der verschlüsselten Daten kam folgendes Frida-Skript zum Einsatz. **Hinweis**: Da die Klassennamen in der App obfuskiert werden, variieren sie je nach 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;
    };

});
```

Die Daten werden Hex-kodiert ausgegeben und können beispielsweise über Cyberchef dekodiert werden: https://gchq.github.io/CyberChef/#recipe=From_Hex('Auto')

Es folgt ein Auszug aus den dekodierten Daten.

```
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
[…]
```

In den übermittelten Daten wurden keine personenbezogenen Daten festgestellt.

4.2.3.3 Auswirkungen

Diese Verschlüsselung bietet die Möglichkeit, persönliche Daten im Datenverkehr zu verbergen. Bei der durchgeführten Analyse konnten allerdings keine sensiblen Daten im verschlüsselten Datenverkehr festgestellt werden.

4.2.3.4 Empfehlungen

In der Praxis haben Nutzende keine Möglichkeit, dieses Verhalten der mobilen App zu verhindern. Bei weiteren Analysen zukünftiger Versionen soll sicherzustellen, dass keine sensiblen Informationen an das Backend übermittelt werden.

4.3 Tiefe Risiken

Befunde in dieser Kategorie bergen je nach Anwender und Umständen ein geringes Risiko für die Anwender. Es wird empfohlen, das individuelle Risiko abzuschätzen und bei Bedarf geeignete Massnahen zu treffen, um das Risiko auf ein akzeptiertes Mass zu reduzieren. Vorher sollten jedoch die Risiken der anderen Kategorien beachtet werden.

4.3.1 Abfrage laufender Prozesse

Die Mobile App erstellt eine Liste laufender Prozesse.

4.3.1.1 Hintergrund

Die Temu-App prüft bei der Ausführung, ob bestimmte Apps auf dem Mobiltelefon installiert sind. Das beobachtete Verhalten kann damit zusammenhängen, dass die Temu-App mehr oder weniger Schaltflächen (z.B. zum Teilen von Inhalten) anzeigt, je nachdem, welche anderen Apps installiert sind. Diese Information könnte im ungünstigsten Fall potenziell sensible Informationen über den Nutzer preisgeben (z.B. Zugehörigkeit zu einer bestimmten Ethnie oder Religion).

Zum jetzigen Zeitpunkt ist jedoch unklar, ob die Informationen an die Backend-Server gesendet und von Temu ausgewertet werden. Aus diesem Grund wird der Befund in diesem Dokument als geringes Risiko eingestuft.

4.3.1.2 Nachweis

Der folgende Auszug aus der Info.plist -Datei der iOS Temu-App zeigt die zum Zeitpunkt der Untersuchung aktuelle Liste der Drittanbieter-Apps, welche von der Temu-App ermittelt werden können:

```
<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>
```

Tatsächlich wurde während der Verwendung der Temu-App auf dem Gerät des NTC-Analysten das Vorhandensein der folgenden Drittanbieter-Apps ermittelt:

```
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://
```

Zur dynamischen Analyse des Verhaltens der Temu-App wurde das folgende Frida-Skript verwendet:

```
/*
$ 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) {
}
});
```

Unter Android werden die folgenden Abfragen gemäss AndroidManifest.xml gemacht.

```
<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>
```

Unter Android wird die Methode `getRunningAppProcesses` aufgerufen. Diese kann genutzt werden, um auszulesen, welche Prozesse momentan laufen. Die Android-Entwicklungsdokumentation erwähnt, dass dies eine Debug-Methode ist, und im Produktionsbereich nicht verwendet werden sollte [16]. Aus Zeitgründen wurde dies unter Android nicht näher untersucht.

Im Rahmen dieser Analyse konnte keine Anfrage identifiziert werden, die eine Liste der laufenden Prozesse an ein Backend übermittelt.

4.3.1.3 Auswirkungen

Die Vorbedingungen für die Möglichkeit zur Abfrage der Drittanbieter-Apps schafft Temu mit den Einträgen in der `Info.plist`-Datei. Diese wird unter anderem im Rahmen des Apple App Store-Reviews geprüft. Scheinbar bestehen beim Betreiber des App Stores keine Bedenken, dass die eingeräumten Berechtigungen von Temu missbraucht werden könnten.

Eine Übermittlung der Informationen über installierte Drittanbieter-Apps an ein Temu-Backend wurde im Rahmen der Untersuchung nicht festgestellt. Es kann jedoch nicht ausgeschlossen werden, dass diese Informationen in codierten oder verschlüsselten Inhalten der HTTP-Anfragen enthalten sind.

Werden diese Informationen tatsächlich nur lokal auf dem Mobiltelefon verwendet, so ist das Verhalten der Temu-App unbedenklich. Problematisch wird es erst, wenn die Informationen an ein Temu-Backend übermittelt werden. Dies konnte jedoch im Rahmen dieser Untersuchung nicht ausgeschlossen werden.

Andere Apps wie Shein, Aliexpress oder Amazon verlangen vergleichbare Berechtigungen für das Erkennen fremder Apps.

4.3.1.4 Empfehlungen

Benutzende haben in der Praxis keine Möglichkeit, dieses Verhalten der Temu-App zu unterbinden.

4.3.2 Multi-Faktor-Authentisierung wird nicht erzwungen

Temu verlangt bei der Registrierung eines Kontos lediglich eine E-Mail-Adresse und ein Passwort oder eine Telefonnummer. Eine Multi-Faktor-Authentisierung ist optional.

4.3.2.1 Hintergrund

Temu möchte die Registrierung und das Einloggen in das Temu-Konto so einfach wie möglich gestalten. Aus diesem Grund wird angenommen, dass Temu auf eine tiefgehende Identitätsprüfung und standardmässig auch auf komplexere MFA-Faktoren verzichtet. Für Angreifer wäre es potenziell möglich, durch SIM-Swapping die Telefonnummer von Temu-Benutzenden zu übernehmen und so Zugriff auf deren Konten zu erhalten.

Die Aktivierung der von Temu optional angebotenen "2-Stufen-Verifizierung" (Wortlaut Temu) schützt vor dieser Art von Angriffen.

4.3.2.2 Nachweis

Wie in der Abbildung 6 zu sehen, verlangt Temu bei der Registrierung eines Kontos lediglich eine E-Mail-Adresse und ein Passwort oder eine Telefonnummer. Eine Registrierung über andere soziale Medien ist ebenfalls möglich, wurde aber in dieser Untersuchung nicht berücksichtigt.

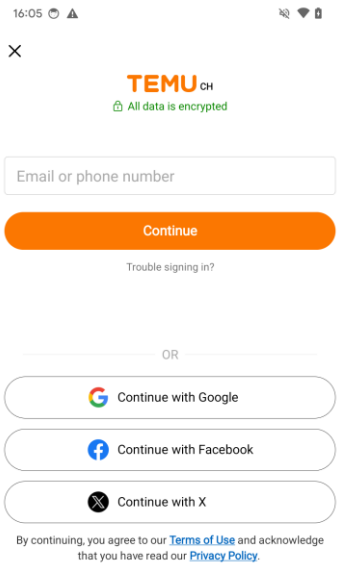

Abbildung 6: Temu Registrierung

Die Abbildung 7 zeigt die Möglichkeit einen weiteren Faktor für die Anmeldung einzurichten.

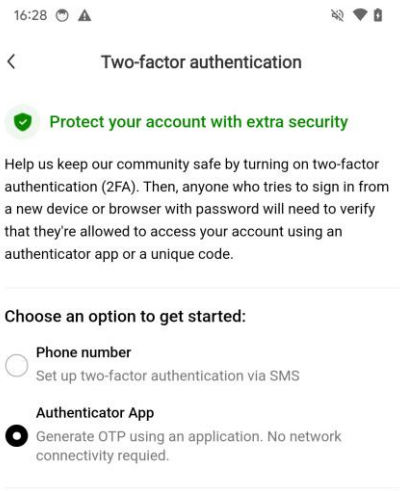

Abbildung 7: Temu MFA Option

4.3.2.3 Auswirkungen

Wenn die 2-Stufen-Verifizierung nicht explizit aktiviert wird, besteht ein erhöhtes Risiko der Kontenübernahme durch Phishing, Wiederverwendung von bekannten Passwörtern und weitere Angriffe der Zugriffsdaten. Dieses Risiko gilt nicht nur für Temu, sondern für alle Apps, welche lediglich den Empfang einer SMS-Textnachricht oder E-Mail benötigen.

4.3.2.4 Empfehlungen

Es wird allen Benutzenden empfohlen, in den Konteneinstellungen die Option für 2-Stufen-Verifizierung via Authenticator App zu aktivieren.

Zudem wird generell die Verwendung von starken und einmaligen Passwörtern empfohlen.

4.3.3 Verschlüsselte Konfigurationsdateien

Temu verwendet verschlüsselte Konfigurationsdateien.

4.3.3.1 Hintergrund

Die Android- und iOS-Apps nutzen verschlüsselte Konfigurationsdateien, vermutlich um sensible Daten wie API-Tokens besser zu schützen. Zusätzlich ermöglicht die Konfiguration Rückschlüsse auf die Geschäftslogik, die durch die Verschlüsselung ebenfalls besser abgesichert wird.

Obwohl die Verschlüsselung einen Schutz darstellt, kann die Konfiguration dennoch ausgelesen werden. Dies erfordert jedoch Reverse Engineering und gestaltet den Zugriff aufwändiger.

4.3.3.2 Nachweis

In der Android-App finden sich unter `/assets/abc/data` und `/assets/config/data` verschlüsselte Konfigurationsdateien. Es wird eine AES-CBC Verschlüsselung verwendet. Die verwendeten Parameter lassen sich durch Reverse Engineering ermitteln:

Der folgende Code setzt den Schlüssel zusammen.

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

Der folgende Code setzt den Initialvektor zusammen.

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

Die untenstehende Funktion entschlüsselt mit Hilfe von Initialvektor und Schlüssel die Konfigurationsdatei.

```
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;
        }
    }
```

Die Verschlüsselung verwendet folgende Parameter:

- **Algorithmus:** `AES CBC`
- **Schlüssel:** `_amm_config_key_`
- **Initial Vektor (IV):** `213344114255776201593163165249 75`

Die Entschlüsselung kann auch mit diesem Cyberchef- «Rezept» durchgeführt werden: 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)

Es folgt ein Auszug des entschlüsselten Dateiinhaltes, welcher unter Android rund 180 Zeilen umfasst:

```
{
  "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\"}}}}"
    },
[…]
```

Die iOS-App enthält deutlich mehr Konfigurations-Einträge (ca. 1300 Zeilen), welche teilweise sogar JavaScript-Code enthalten. In manchen Fällen ist es sinnvoll, eine Konfigurationsdatei mit Programmiercode zu erstellen, um dynamische Werte oder logische Entscheidungen basierend auf bestimmten Bedingungen zu ermöglichen.

```
[…]
"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]})}}());"
    },
[…]
```


Die `/assets/config/data` Konfigurationsdatei hat genau den gleichen Inhalt, verwendet für die Verschlüsselung jedoch andere Parameter:

- **Algorithmus:** `AES CBC`
- **Schlüssel (HEX-Wert):** `42132b322f063b161d4a7303745a596e`
- **Initial Vektor (IV, HEX-Wert):** `22505c58440b433c09520446547b7e12`

Die Entschlüsselung kann auch mit diesem Cyberchef- «Rezept» durchgeführt werden:

https://gchq.github.io/CyberChef/#recipe=AES_Decrypt(%7B'option':'Hex','string':'42132b322f063b161d4a7303745a596e'%7D,%7B'option':'Hex','string':'22505c584440b433c09520446547b7e12'%7D,'CBC','Raw','Raw',%7B'option':'Hex','string':''%7D,%7B'option':'Hex','string':''%7D)

4.3.3.3 Auswirkungen / Einschätzung

In der zur Verfügung stehenden Zeit konnte nicht vollständig geklärt werden, wo diese Konfigurations-Einträge überall verwendet werden. Anhand der Namen und Werte der Konfigurations-Einträge wird angenommen, dass die Konfiguration zum Steuern von WebView Inhalten genutzt wird.

Durch die Verschlüsselung der Konfigurationsdatei wird die Analyse erschwert.

4.3.3.4 Empfehlungen

Benutzende haben in der Praxis keine Möglichkeit, dieses Verhalten der Temu-App zu unterbinden. Eine tiefergehende Prüfung von weiteren Sicherheitsforschenden, wo diese Konfigurationswerte verwendet werden, wird empfohlen.

4.3.4 Erkennen von erhöhten Privilegien und Debugging

Die Mobile App prüft, ob ein Mobiltelefon Debugging Optionen aktiviert hat und «gerootet» oder «jailbroken» ist.

4.3.4.1 Hintergrund

Die Temu-App sammelt Informationen über die Betriebsumgebung. Diese können Rückschlüsse darüber erlauben, ob die App auf einem Testgerät ausgeführt wird. Solche Methoden können von Herstellern genutzt werden, um den Programmablauf an diese Umgebung anzupassen, z.B. durch Unterdrückung von Verhalten, das geheim gehalten werden soll.

4.3.4.2 Nachweis

In der dekompilierten iOS-App finden sich Strings, die auf eine «Jailbreak»-Erkennung hindeuten. Die Abbildung 8 zeigt einen Ausschnitt der Datenstruktur, in welcher die Ergebnisse gespeichert werden.

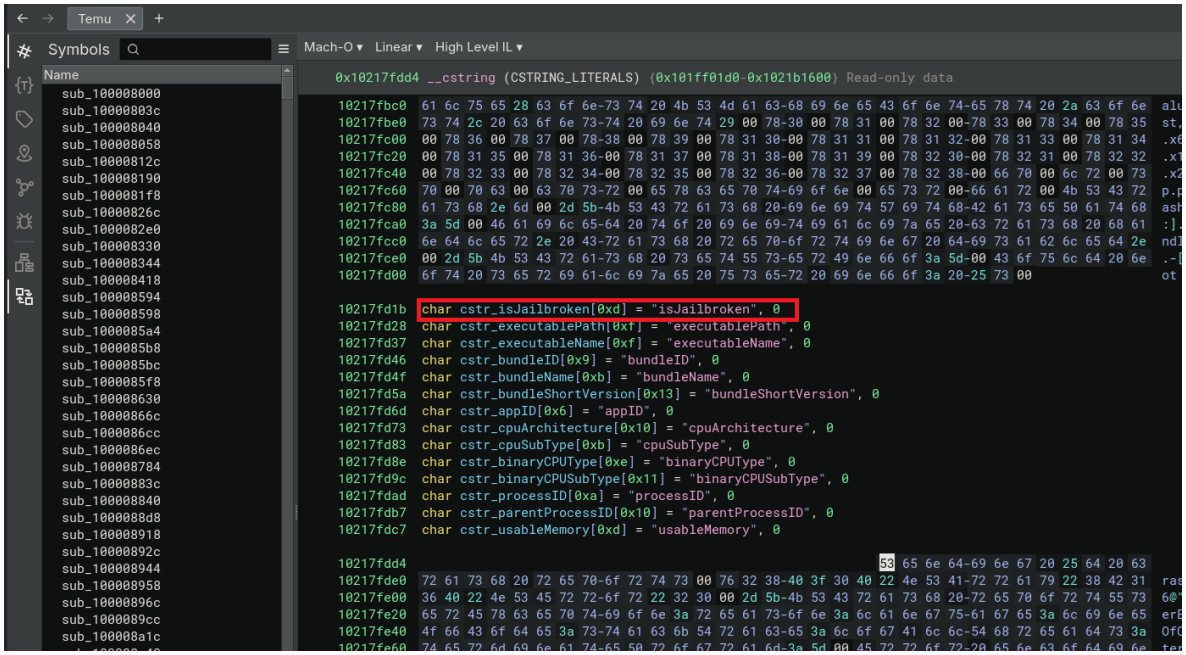

Abbildung 8: Jailbreak Erkennung unter iOS

Wie in der Abbildung 9 zu sehen ist, wurden unter Android in der Bibliothek `libUserEnv.so` Strings gefunden, welche auf eine Root Erkennung hindeuten.

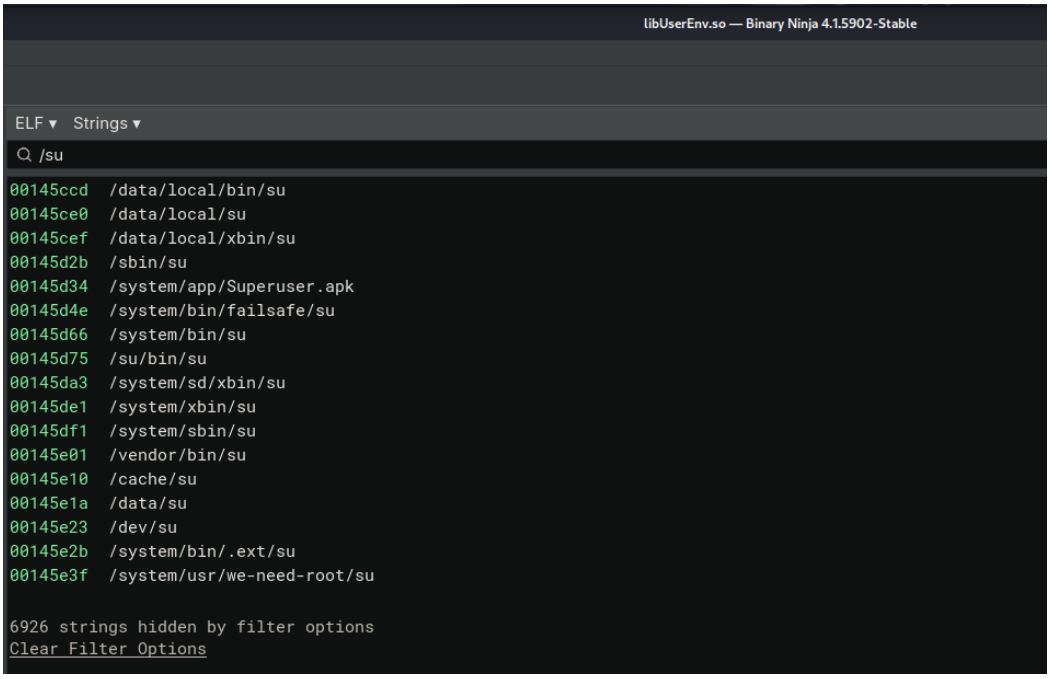

Abbildung 9: Strings zum Ermitteln von gerootet Geräten

Im Rahmen dieser Analyse konnte keine Nachricht an ein Backend gefunden werden, welche das Resultat der Root / Jailbreak Erkennung and ein Backend übermittelt.

Des Weiteren ermittelt die App, ob Debugging Optionen auf dem Gerät aktiviert sind. Diese Details werden auch an ein Backend übermittelt. Diese Übermittlung ist in *Kapitel 4.1.2* genauer beschrieben.

4.3.4.3 Auswirkungen

Es ist unklar, wofür die oben erwähnten Informationen von Temu verwendet werden. Daher kann die Vermutung, dass es sich hierbei um die Erkennung eines Prüfstandes handelt, nicht ausgeschlossen werden. Ist dies der Fall, könnte ein solcher Prüfstand-Modus bestimmtes Verhalten der Temu-App verbergen.

Auch andere Mobile Apps wie z.B. Aliexpress oder Shein haben eine Erkennung von Root / Jailbreak implementiert.

4.3.4.4 Empfehlungen

Es wird empfohlen, tiefergehende Untersuchungen durchzuführen, um ein abweichendes Verhalten der Temu-App in Prüfstand-Situationen ausschliessen zu können.

4.4 Relativierungen

Die in dieser Kategorie aufgeführten Befunde relativieren Erkenntnisse aus anderen Berichten zu Temu. Sie stellen aus Sicht des NTC für die Nutzer entweder nur ein geringes oder kein Risiko dar.

4.4.1 Auslesen und Übermitteln fremder App-Logs

Für die getestete Version konnte keinen Beweis gefunden werden, dass die App Ereignis-Protokolle anderer Apps auslesen kann.

4.4.1.1 Hintergrund

Unter Android ist es möglich mittels Logcat Ereignisse der Applikation zu protokollieren. Dies kann genutzt werden, um Fehler oder auch andere von den Entwicklern als wichtig erachtete Ereignisse zu protokollieren. Dabei sollte es vermieden werden, sensitive Daten wie z.B. Passwörter zu protokollieren. Die Einhaltung dieser Best-Practices liegt an den Entwicklern der jeweiligen App.

Der Grizzly-Bericht behauptet, dass die Temu-App die Protokolldaten anderer Apps ausliest, um das Benutzerverhalten auszuspionieren [3]. Jedoch ist es seit Android 4.1, welches schon seit 12 Jahren auf dem Markt ist, grundsätzlich nicht mehr möglich auf Protokolldaten von fremden Apps zuzugreifen. Ausgenommen sind nur spezielle privilegierte und OEM-Apps [17]. Daher sollte dies der Temu-App im Normalfall gar nicht möglich sein.

4.4.1.2 Nachweis

Es wurde festgestellt, dass die Temu-App eigene Protokolldaten an Backend-Server versendet, jedoch nicht fremde Protokolldaten. Seit Android 4.1 ist der Zugriff auf fremde Protokolldaten grundsätzlich nicht mehr möglich. Ausnahmen sind nur möglich, wenn die App z.B. vorinstalliert wurde und die Berechtigung zum Auslesen von System-Protokolldaten ausweist. Apps, welche über den Play Store publiziert werden, ist es nicht gestattet die Berechtigung zu verwenden.

Die Abbildung 10 zeigt exemplarisch die Datenübertragung der App.

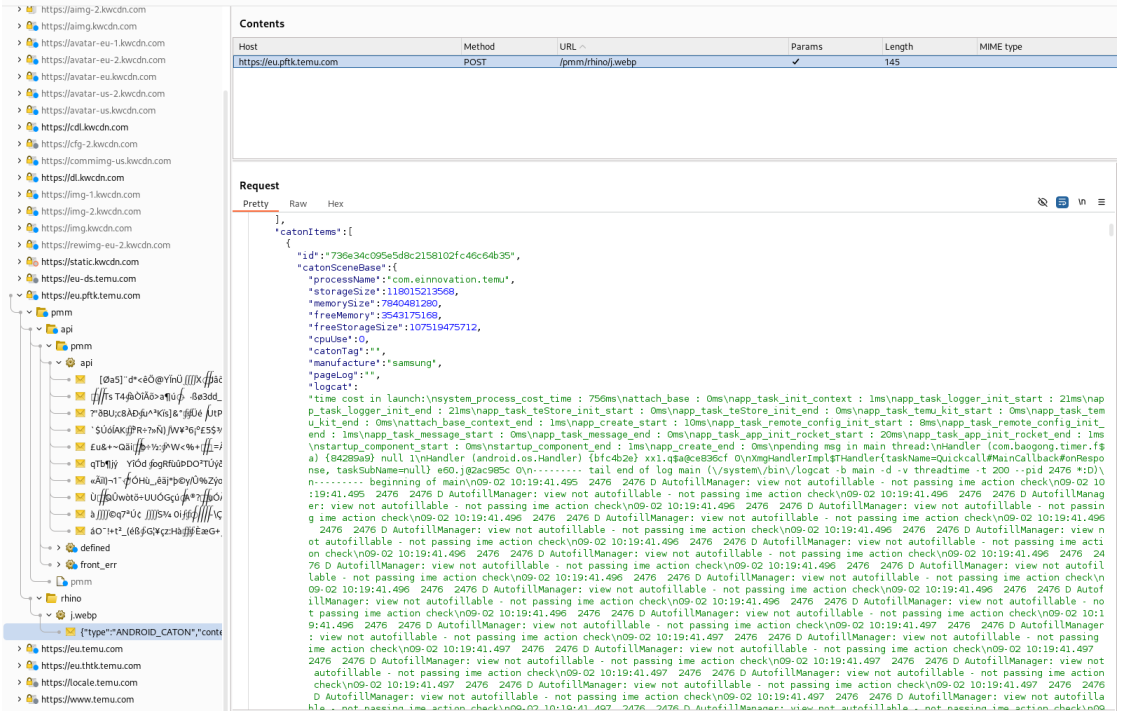

Abbildung 10: Protokolldaten werden übermittelt

Die übertragenen Daten werden in einer nativen Bibliothek mit der Bezeichnung `libcrash.so` gesammelt. Die Abbildung 11 zeigt den Befehl, welcher zum Abfragen der Ereignisprotokolle genutzt wird.

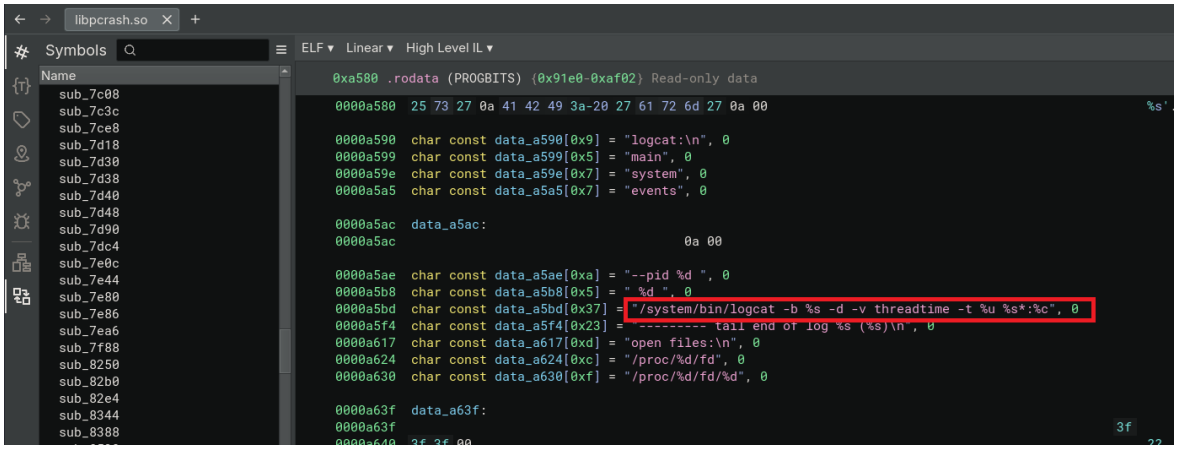

Abbildung 11: Logcat Befehl zum Sammeln von Ereignisprotokollen

4.4.1.3 Auswirkungen

Die App sendet Protokolldaten der eigenen App an Backend-Server.

4.4.1.4 Empfehlungen

Es wird empfohlen Temu nur aus offiziellen Quellen und auf aktuellen Betriebssystemen zu verwenden.

4.4.2 Auslesen und Übermitteln von System-Dateien

Die Android-App sammelt und übermittelt Daten zur CPU des Endgerätes an ein Backend des Anbieters.

4.4.2.1 Hintergrund

Im Grizzly-Bericht wird behauptet, dass die App sensible System-Ereignisprotokolle und System-Informationen sammelt und an Temu-Server sendet [3].

4.4.2.2 Nachweis

Es gibt Indizien, dass die App mittels nativer Bibliotheken auf System-Dateien zugreift, um CPU-Informationen (Anzahl und jeweils Taktfrequenz) und die Seriennummer der SoC abzufragen. Ersteres kann legitime Gründe haben, um z.B. eine Ahnung über die Leistungsdaten der Geräte der Nutzenden zu haben, um die App zu optimieren. Das zweite kann zur Benutzeridentifizierung gebraucht werden, wenn der Benutzer nicht angemeldet ist oder kein Konto hat und allenfalls die Werbe-ID nicht zur Verfügung steht oder eine Alternative dazu gewünscht ist.

Diverse Strings in einer Programmbibliothek, welche auf Zugriffe auf CPU-Informationen hindeuten. Die Abbildung 12 und Abbildung 13 zeigen die Pfade, welche durch die App abgerufen werden.

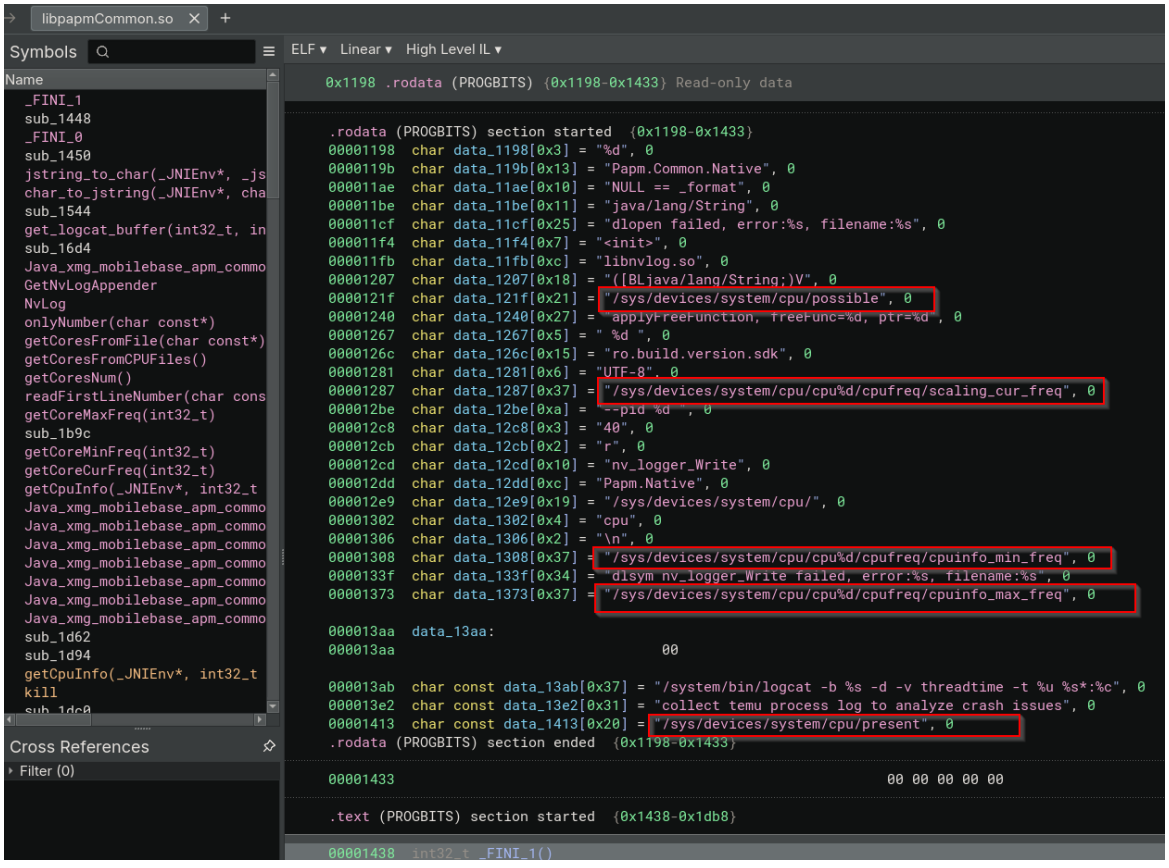

Abbildung 12: Pfade zu CPU Metadaten

Pfad in der `libsecure_lib.so` Programmbibliothek für den Zugriff auf die CPU-Seriennummer:

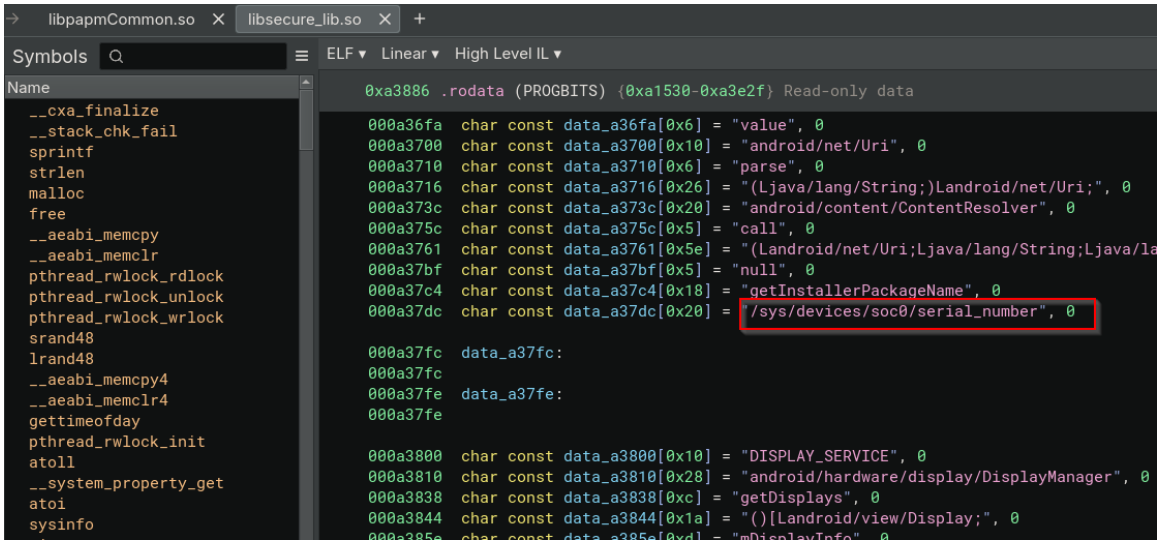

Abbildung 13: Zugriff auf SoC Seriennummer

Im Rahmen dieser Analyse konnte keine weiteren Indizien für Zugriffe auf das sysfs-Pseudo-Dateisystem gefunden werden. (Dieses wird verwendet, um System-Informationen abzufragen und allenfalls mit den nötigen Rechten auch Konfigurationen vorzunehmen.)

4.4.2.3 Auswirkungen

Es wurden keine Zugriffe auf heikle Daten festgestellt. Daher gibt es auch keine besonderen Auswirkungen. Das Sammeln von CPU-Architekturdaten kann nützlich sein bei der Fehlerbehebung und zum Erstellen von relevanten Performance Metriken.

4.4.2.4 Empfehlungen

In diesem Bereich sind keine besonderen Massnahmen nötig.

4.4.3 Screenshots von anderen Apps

Es wurden keine Hinweise darauf gefunden, dass Screenshots von anderen Apps gemacht werden.

4.4.3.1 Hintergrund

Der Grizzly-Bericht stellt fest, dass bestimmte Methoden (`getWindow()`, `getDecorView()`, `getRootView()`) von der Android Temu-App benutzt werden, um angeblich die Aktivitäten der Benutzenden in anderen Apps auszuspionieren [3].

Diese Methoden werden zwar aufgerufen, aber diese Methoden geben die View bzw. Window der eigenen App zurück und haben legitime Use-Cases.

4.4.3.2 Nachweis

Die Temu-App verwendet die im Grizzly-Bericht genannten Methoden:

```
    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 sind Apps primär aus Aktivitäten (Activities) aufgebaut, welche die Nutzung gewisser Funktionalitäten der App ermöglichen. Eine Aktivität wiederum hat ein Fenster (Window). Der Fensterinhalt besteht aus einer Ansicht (View), die hierarchisch aus Ansichten aufgebaut werden kann [18]. Daher wird schlussendlich in der obigen Methode mittels `activity.getWindow()` das Fenster der Aktivität ermittelt und danach mittels den Methoden `peekDecorView()` (ähnlich zu `getDecorView()`) und `getRootView()` das Wurzelelement der Ansicht ermittelt [19]. Diese Vorgehensweise ist unbedenklich.

Der Code der Temu-App ist nicht ausreichend, um ein Bildschirmfoto einer App zu erstellen. Dazu müsste noch die Methode `draw()` auf der `rootView` angewendet werden, um die Ansicht zu rendern[19]. Dies ist hier nicht der Fall.

Ein zentraler Punkt ist zudem, dass all dies nicht auf Aktivitäten von fremden Apps angewendet werden kann. Der Zugriff auf fremde Apps ist nur erlaubt, falls diese Aktivitäten explizit exportieren und somit für andere Apps zugänglich machen [20].

Um Bildschirmfotos von anderen Applikationen zu erstellen, gibt es seit Android 5.0 (API-Level 21) offiziell die MediaProjection-Schnittstelle [21]. Bei Benutzung dieser wird den Benutzenden eine Meldung angezeigt, in der bestätigt werden muss, dass der Bildschirminhalt aufgezeichnet werden darf. Somit ist eine unbemerkte Aufnahme nicht möglich. Die geprüfte Version zeigte keinerlei Anzeichen dafür, dass Temu die MediaProjection-Schnittstelle verwendet.

Eine weitere Möglichkeit um das Bild anderer Apps zu erfassen, wäre die PixelCopy API, welche seit Android 7.0 (API Level 24) existiert [22]. Auch diese wird gemäss Analysen in der Temu-App nicht verwendet.

4.4.3.3 Einschätzung

Die Verwendung der im Grizzly-Bericht erwähnten Methoden ist legitim. Diese können z.B. dazu dienen die Grösse des Fensters zu eruieren. Die Methoden können keinen Zugriff auf fremde App-Anzeigen gewähren, sofern die anderen Apps dies nicht explizit erlauben. Daher geht davon auch keine Gefahr aus.

Es gibt auch keine Anzeichen dafür, dass die Temu-App mittels anderen bekannten Mechanismen, Media-Projection- oder PixelCopy-Schnittstellen, auf den Bildschirminhalt von anderen Apps zugreift.

4.4.3.4 Empfehlungen

Als präventive Massnahme wird wiederum empfohlen die Berechtigungen der App zu prüfen. Zudem kann dem Privatsphären-Bericht entnommen werden welche Berechtigungen die App in letzter Zeit benutzt hat. Des Weiteren ist es ratsam, Meldungen resp. Bestätigungen für Bildschirmaufzeichnungen (oder andere Berechtigungen) aufmerksam zu lesen und nur zu bestätigen, falls der Nutzen klar und erwünscht ist.

4.4.4 Speicherung von MAC-Adressen

Gemäss Grizzly-Bericht speichert die Temu-App die MAC-Adresse vom Mobilgerät [3].

4.4.4.1 Hintergrund

Der Grizzly-Bericht beschreibt, dass die Temu-App die MAC-Adresse des Mobilgeräts ausliest und speichert. Ebenfalls wird behauptet, dass diese MAC-Adresse potenziell dazu genutzt werden, eine DDoS-Attacke gegen das Gerät über das Internet durchzuführen [3].

4.4.4.2 Nachweis

Im Rahmen dieser Analyse konnte das Auslesen der MAC-Adresse im dekompilierten Programmcode sowie in der Kommunikation der Android und iOS-Apps nicht nachgewiesen werden. Möglicherweise wurde die Funktionalität von Temu entfernt.

4.4.4.3 Auswirkungen

Selbst wenn die MAC-Adresse ausgelesen würde, könnte sie – anders als im Grizzly-Bericht beschrieben – nicht für eine DDoS-Attacke über das Internet genutzt werden. Geräte sind nur innerhalb desselben lokalen Netzwerks über ihre MAC-Adressen erreichbar. Zudem vergeben iOS und Android standardmässig für jedes Netzwerk zufällige MAC-Adressen, sodass sich die Adressen je nach verwendetem Netzwerk ändern.

4.4.4.4 Empfehlungen

In diesem Bereich sind keine Massnahmen nötig.

4.4.5 Re-Kompilierung von Programm-Paketen

Gemäss dem Grizzly-Bericht kann die Temu Android App dynamisch neue Programm-Pakete kompilieren [3].

4.4.5.1 Hintergrund

Im Grizzly-Bericht heisst es, die Temu-App für Android nutze dynamische Kompilierung über den Befehl `cmd package compile`, ausgeführt durch `runtime.exec()`, wodurch sie direkt auf dem Gerät des Nutzers neue Programme erstellen könne. Diese Methode verstecke die ausführbaren Dateien vor Sicherheits-Scans und ermögliche es der App, die Prüfungen des Google Play Stores möglicherweise unbemerkt zu umgehen.

4.4.5.2 Nachweis

Diese Funktion ist in der geprüften Version von Temu nicht vorhanden. Zudem ist der Vorwurf aus einem weiteren Grund unbegründet: Mit dieser Android-Funktionalität lässt sich nur das schon vorhandene Programmpaket neu kompilieren, etwa um es für den verwendeten Prozessor zu optimieren und so eine bessere Performance zu erzielen [23].

4.4.5.3 Empfehlungen

In diesem Bereich sind keine Massnahmen nötig.

4.4.6 Berechtigungen der Android App

Gemäss dem Grizzly-Bericht verwendet die Android App fragwürdige Berechtigungen [3].

4.4.6.1 Hintergrund

Der Grizzly-Bericht beschreibt, dass die Applikation die Berechtigungen `CAMERA`, `RECORD_AUDIO`, `WRITE_EXTERNAL_STORAGE`, `INSTALL_PACKAGES` und `ACCESS_FINE_LOCATION` im Programmcode verwendet, ohne dass diese im Android Manifest deklariert sind. Zudem wird erwähnt, dass die Applikation die Berechtigungen `INTERNET` und `WAKE_LOCK` einfordert, im Gegensatz zu anderen vergleichbaren Apps wie Shein, TikTok oder eBay.

4.4.6.2 Nachweis

Der Zugriff auf Systemfunktionen, die Berechtigungen erfordern, aber nicht im Android-Manifest aufgeführt sind, ist nicht möglich [24]. Die Berechtigungen `CAMERA`, `RECORD_AUDIO`, `WRITE_EXTERNAL_STORAGE` und `INSTALL_PACKAGES` sind nicht im Manifest deklariert und können somit auch nicht verwendet werden.

Die Berechtigungen `ACCESS_FINE_LOCATION`, `INTERNET` und `WAKE_LOCK` werden im Manifest der Android-App deklariert; dies ist jedoch gängige Praxis und wird – anders als im Grizzly-Bericht dargestellt – auch von anderen Apps wie Galaxus [25], Shein [11], TikTok [26] und eBay [27] genutzt.

Auch die Applikation für iOS verwendet nur erklärbare Berechtigungen:

- `NSLocationWhenInUseUsageDescription`: Lokalisierung
- `NSSupportsLiveActivities`: Anzeige von Live Activities
- `NSCameraUsageDescription`: Kamera
- `UIRequiresFullScreen`: Anzeige auf ganzem Bildschirm

4.4.6.3 Empfehlungen

Es wird empfohlen Temu auf aktuellen Betriebssystemen zu verwenden.

5 Testfälle

In diesem Abschnitt werden alle Testfälle vorgestellt, die während der Sicherheitsanalyse betrachtet wurden. Befunde, die aus einem bestimmten Testfall hervorgehen, sind unter der Kurzbeschreibung des Testfalls verlinkt. Wenn kein Befund verlinkt ist, wurde im Zeitrahmen der Analyse keine relevante Schwachstelle gefunden. Wenn die Testfälle nur für eine Teilmenge der Komponenten gelten, werden die entsprechenden Komponenten explizit aufgeführt.

5.1 Netzwerk-Kommunikation

Die Testfälle zur Kommunikation fokussieren sich auf die Daten, die zwischen der Temu-App und dem Temu Backend ausgetauscht werden, basierend auf einer zu erwartenden Nutzung der Temu-App durch legitime Benutzende ohne missbräuchliche Absichten.

TF 1 Verschlüsselte Datenübertragung

Wird der Netzwerkverkehr verschlüsselt übertragen?

Risiko: Wird der Netzwerkverkehr nicht verschlüsselt übertragen, so ist es für Angreifer einfach möglich, den Inhalt der Kommunikation mitzulesen oder zu manipulieren.

Befunde: *4.1.2*

TF 2 Verschlüsselte Datenübertragung mit sicheren Protokollen

Wird der Netzwerkverkehr mittels sicherer Protokolle verschlüsselt?

Risiko: Wird der Netzwerkverkehr mit unsicheren Protokollen übertragen, erleichtert dies den Angreifern, den Inhalt der Kommunikation mitzulesen oder zu manipulieren

TF 3 Benutzerkonto anlegen und Erstidentifikation

Welche Möglichkeiten für eine Konten-Registrierung werden angeboten? Wie wird die Identität der Benutzenden überprüft?

Risiko: Wird die Identität der Benutzenden nicht ausreichend geprüft, so ist es Angreifern möglich, Konten zu übernehmen oder unter einem fremden Namen zu erstellen. Dadurch können sensible Daten eingesehen oder falsche Informationen verbreitet werden.

Befunde: *4.3.2*

TF 4 Temu-App im Vordergrund

Während die Temu-App im Vordergrund läuft, ist es ihr möglich, mit bereits erteilten Berechtigungen wie z.B. für die genaue GPS-Ortung Daten zu erheben. Wird diese Möglichkeit übermässig genutzt und Daten an ein Temu Backend gesendet?

Risiko: Werden unerwartet Daten an ein Temu Backend gesendet, ohne dass ein erkennbarer Nutzen für die Benutzenden besteht, kann dies ein Hinweis auf Benutzerüberwachung sein.

Befunde: *4.1.2*, *4.1.3*, *4.2.2*

TF 5 Temu-App im Hintergrund

Werden unerwartete Daten an ein Temu Backend gesendet, während die Temu-App im Hintergrund aktiv ist?

Risiko: Eine unerwartete Datenübertragung im Hintergrund kann ein Hinweis auf eine Benutzerüberwachung sein.

5.2 Privatsphäre und Datenschutz

Bei den nachfolgend aufgeführten Tests zu Privatsphäre und Datenschutz wurde besonderer Fokus darauf gelegt, wie die Temu-App mit den Berechtigungen der mobilen Betriebssysteme umgeht. Dabei wurde insbesondere überprüft, welche Berechtigungen wann angefragt werden und ob die erhaltenen Daten an das Temu Backend übermittelt werden.

TF 6 Geolocation Tracking

Wird die aktuelle Position des Gerätes erfasst und an das Temu Backend übermittelt? Wird die GPS Position zwingend für den Gebrauch der App benötigt?

Risiko: Positionsdaten können verwendet werden, um den aktuellen Standort zu ermitteln oder Bewegungsprofile zu erstellen. Je mehr und genauere Datenpunkte zur Verfügung stehen, desto genauer können die Benutzenden überwacht werden. Bewegungsprofile können Rückschlüsse auf Wohnort, Arbeitsort, Vorlieben und Gewohnheiten etc. ermöglichen.

Befunde: *4.1.3*, *4.2.2*

TF 7 Zugriff auf Kontakte

Ist der Zugriff auf die Kontakte zwingend nötig, um die Temu-App verwenden zu können? Zu welchen Zeitpunkten werden die Kontakte angefragt und übermittelt?

Risiko: Das Sammeln von Kontaktdaten ermöglicht es Temu, Nutzerprofile und Beziehungsmuster von unbeteiligten Dritten - also Personen, die Temu nicht nutzen - zu erstellen.

TF 8 Zugriff auf Kalender

Benötigt die Temu-App Zugriff auf den Kalender der Benutzenden?

Risiko: Kalenderinformationen können sensitive Informationen über den Tagesablauf und die Aktivitäten der Benutzenden enthalten und eignen sich daher gut für die Überwachung. Termine können darüber hinaus wertvolle Zusatzinformationen enthalten, wie z. B. die teilnehmenden Personen, deren Kontaktdaten, Standorte, Aktivitäten usw.

TF 9 Zugriff auf Kamera

Zu welchen Zeitpunkten wird auf die Kamera zugegriffen? Werden die Daten direkt an das Temu Backend übertragen?

Risiko: Die Kamera kann dazu benutzt werden, unbemerkt Bilder oder Videos

aufzunehmen. Diese Bildinformationen können Aufschluss darüber geben, in welcher Umgebung sich die Benutzenden gerade befinden, welche Personen sich in der Nähe aufhalten oder welche Tätigkeiten ausgeführt werden. Unter Umständen können die Bilder auch als Erpressungsmittel eingesetzt werden.

TF 10 Zugriff auf Mikrofon

Zu welchen Zeitpunkten wird auf das Mikrofon zugegriffen? Werden die Daten direkt an das Temu Backend übertragen?

Risiko: Das Mikrofon kann für unbemerkte Tonaufnahmen verwendet werden. Diese Aufnahmen können Aufschluss über die aktuelle Umgebung geben, in der sich die Benutzenden befinden, und den Inhalt von - möglicherweise vertraulichen - Gesprächen offenbaren. Unter Umständen können die Aufnahmen auch als Erpressungsmittel eingesetzt werden.

TF 11 Zugriff auf externen Speicher (Android)

Wann wird auf den externen Speicher zugegriffen? Werden nur die ausgewählten Dateien gelesen oder auch auf andere Inhalte?

Risiko: Die App könnte ohne Wissen und ausdrückliche Zustimmung der Benutzenden auf Daten zugreifen und diese lokal verarbeiten oder an das Temu Backend übertragen. Dabei kann es sich unter anderem um vertrauliche und personenbezogene Daten handeln.

TF 12 Zugriff auf das lokale Netzwerk

Wird der Zugriff auf das lokale Netzwerk angefragt?

Risiko: Die Berechtigung erlaubt die Interaktion mit lokalen Netzwerkgeräten, wie z.B. Smart Home-Geräten. Unter Umständen könnte es möglich sein, solche lokalen Netzwerkgeräte zu steuern oder sensible Informationen auszulesen.

TF 13 Screenshots von fremden Apps

Versucht die App Bildschirminhalte anderer Apps zu machen?

Risiko: Durch das Aufzeichnen von Inhalten anderer Apps können sensible Informationen wie Zugangsdaten, Nachrichten, Bankdaten oder persönliche Informationen gesammelt werden.

Befunde: *4.4.3*

TF 14 Sammeln von Systeminformationen

Werden Informationen zum Mobiltelefon an ein Temu Backend gesendet? Werden Informationen über installierte oder ausgeführte Apps gesammelt und an das Temu Backend übertragen?

Risiko: Die Übertragung von Systeminformationen ermöglicht die Erstellung eines Benutzerprofils und die Überwachung der Benutzeraktivitäten. Dies ist insbesondere dann von Bedeutung, wenn Benutzende mehrere Profile (z. B. privat und beruflich) auf demselben Gerät verwenden.

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

TF 15 Datenauszug nach Datenschutzgesetz

Wenn Nutzerinnen und Nutzer von ihrem Auskunftsrecht Gebrauch machen und einen vollständigen Datenauszug anfordern, enthält dieser dann auch nur zweckmässig erhobenen Daten?

Risiko: Die Erhebung und Speicherung von Daten sollte nur in dem Umfang erfolgen, wie es für den jeweiligen Zweck erforderlich ist.

TF 16 Abfrage der Zwischenablage

Greift die Temu-App ungefragt auf die Zwischenablage zu? Werden die Daten aus der Zwischenablage automatisch an das Temu Backend übertragen?

Risiko: Die Zwischenablage kann vertrauliche Inhalte wie Passwörter enthalten.

TF 17 Verwendung eines integrierten Browsers

Verwendet die App einen integrierten Browser? Wenn ja, wozu dient dieser? Verfügt er über besondere Funktionen?

Risiko: Ein integrierter Browser könnte um nahezu beliebige Funktionen erweitert werden. Dies könnte z.B. die Aufzeichnung der angezeigten Inhalte oder der Benutzereingaben ermöglichen. Damit könnten potenziell sensible Daten erfasst und an das Temu Backend übertragen werden.

Befunde: *4.2.2*

TF 18 Verwendung der App im «Lockdown» Modus (iOS)

Kann die iOS-App im Lockdown Modus verwendet werden?

Risiko: Der Lockdown-Modus reduziert die Erhebung und Verarbeitung von Daten durch Apps, was für Nutzer mit erhöhten Datenschutzanforderungen wichtig ist. Eine App, welche nicht im Lockdown-Modus nicht funktioniert, kann dazu führen, dass mehr Nutzerdaten preisgegeben oder exponiert werden.

5.3 Sicherheit der Mobile-Apps

TF 19 Unterschiede zwischen iOS und Android

Gibt es signifikante Unterschiede im Verhalten der Android und iOS-Apps?

Risiko: Unterschiedliches Verhalten kann dazu führen, dass bei der App auf einer Plattform die Datenschutzrichtlinien weniger strikt umgesetzt werden als auf der Anderen.

TF 20 Anzeichen für das Ausnützen von Sicherheitslücken

Finden sich im dekompilierten Code der Apps Anzeichen dafür, dass die App bekannte Sicherheitslücken ausnützt? (Wurde explizit geprüft, da dies im Grizzly-Bericht so behauptet wird [3])

Risiko: Durch das Ausnutzen von Schwachstellen kann eine App, potenziell auf sensible Informationen zugreifen oder Malware installieren, was für den Nutzer schwerwiegende Konsequenzen haben kann, wie Datenverlust oder Identitätsdiebstahl.

Literatur

[1] M. Kälin, «Temu, Shein und Co. liefern täglich Hunderttausende Päckli in die Schweiz», Blick. Zugegriffen: 5. November 2024. [Online]. Verfügbar unter: 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]. Verfügbar unter: 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». Zugegriffen: 28. Oktober 2024. [Online]. Verfügbar unter: 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». Zugegriffen: 8. November 2024. [Online]. Verfügbar unter: https://grizzlyreports.com/performance-history/
[5] DEKRA Testing and Certification, S.A.U., «Security Evaluation Report», Feb. 2024. Zugegriffen: 5. November 2024. [Online]. Verfügbar unter: https://appdefensealliance.dev/reports/com.einnovation.temu_1706872227165973.pdf
[6] J. Schmidli, K. Schumacher, und P. Albisser, «Tracking mit Ortungsdiensten - Der Spion in unseren Handys», Schweizer Radio und Fernsehen (SRF). Zugegriffen: 15. November 2024. [Online]. Verfügbar unter: 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. Zugegriffen: 5. November 2024. [Online]. Verfügbar unter: https://www.watson.ch/!345235115
[8] B. Egger, «Beschwerde gegen Temu beim SECO eingereicht», HANDELSVERBAND.swiss. Zugegriffen: 5. November 2024. [Online]. Verfügbar unter: https://handelsverband.swiss/news/beschwerde-gegen-temu-beim-seco-eingereicht/
[9] Exodus, «Report for in.amazon.mShop.android.shopping 28.18.2.300», εxodus. Zugegriffen: 7. November 2024. [Online]. Verfügbar unter: 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. Zugegriffen: 7. November 2024. [Online]. Verfügbar unter: https://reports.exodus-privacy.eu.org/en/reports/ru.aliexpress.buyer/latest/
[11] Exodus, «Report for SHEIN», εxodus. Zugegriffen: 6. November 2024. [Online]. Verfügbar unter: https://reports.exodus-privacy.eu.org/de/reports/519955/
[12] Exodus, «Report for com.einnovation.temu 3.11.0», εxodus. Zugegriffen: 7. November 2024. [Online]. Verfügbar unter: 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». Zugegriffen: 8. November 2024. [Online]. Verfügbar unter: 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). Zugegriffen: 15. November 2024. [Online]. Verfügbar unter: 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? Zugegriffen: 18. Oktober 2024. [Online]. Verfügbar unter: https://www.lookout.com/blog/apple-lockdown-mode
[16] Android Developers, «ActivityManager», Activity Manager | Android Developers. Zugegriffen: 29. Oktober 2024. [Online]. Verfügbar unter: https://developer.android.com/reference/android/app/ActivityManager
[17] Android Developers, «READ_LOGS Permission», Android Developers. Zugegriffen: 7. November 2024. [Online]. Verfügbar unter: https://developer.android.com/reference/android/Manifest.permission#READ_LOGS
[18] Android Developers, «Introduction to activities», Android Developers. Zugegriffen: 6. November 2024. [Online]. Verfügbar unter: https://developer.android.com/guide/components/activities/intro-activities
[19] Android Developers, «View», Android Developers. Zugegriffen: 6. November 2024. [Online]. Verfügbar unter: https://developer.android.com/reference/android/view/View
[20] Android Developers, «<activity>», Android Developers. Zugegriffen: 6. November 2024. [Online]. Verfügbar unter: https://developer.android.com/guide/topics/manifest/activity-element
[21] Android Developers, «Media projection | Android media», Android Developers. Zugegriffen: 6. November 2024. [Online]. Verfügbar unter: https://developer.android.com/media/grow/media-projection
[22] Android Developers, «PixelCopy», Android Developers. Zugegriffen: 7. November 2024. [Online]. Verfügbar unter: https://developer.android.com/reference/android/view/PixelCopy
[23] Android Open Source Project, «Implement ART just-in-time compiler», Android Open Source Project. Zugegriffen: 7. November 2024. [Online]. Verfügbar unter: https://source.android.com/docs/core/runtime/jit-compiler#force-compilation-of-a-specific-package
[24] Android Developers, «App manifest overview | Android Developers». Zugegriffen: 6. November 2024. [Online]. Verfügbar unter: https://developer.android.com/guide/topics/manifest/manifest-intro
[25] Exodus, «Report for com.galaxusapp 4.13.0», εxodus. Zugegriffen: 6. November 2024. [Online]. Verfügbar unter: https://reports.exodus-privacy.eu.org/en/reports/com.galaxusapp/latest/
[26] Exodus, «Report for Tiktok», εxodus. Zugegriffen: 6. November 2024. [Online]. Verfügbar unter: https://reports.exodus-privacy.eu.org/de/reports/449993/
[27] Exodus, «Report for com.ebay.mobile 6.182.0.1», εxodus. Zugegriffen: 6. November 2024. [Online]. Verfügbar unter: https://reports.exodus-privacy.eu.org/de/reports/517988/