Silver Surfers on the Tech Wave: Privacy Analysis of Android Apps for the Elderly

Pranay Kapoor, Rohan Pagey, Mohammad Mannan, and Amr Youssef

Concordia University, Montreal, Quebec, Canada
p_apoo@live.concordia.ca, r_pagey@live.concordia.ca,
m.mannan@concordia.ca, youssef@ciise.concordia.ca

Abstract. Like other segments of the population, elderly people are also rapidly adopting the use of various mobile apps, and numerous apps are also being developed exclusively focusing on their specific needs. Mobile apps help the elderly to improve their daily lives and connectivity, and their caregivers or family members to monitor the loved ones' well-being and health-related activities. While very useful, these apps also deal with a lot of sensitive private data such as healthcare reports, live location, and Personally Identifiable Information (PII) of the elderly and caregivers. While the privacy and security issues in mobile applications for the general population have been widely analyzed, there is limited work that focuses on elderly apps. We shed light on the privacy and security issues in mobile apps intended for elderly users, using a combination of dynamic and static analysis on 146 popular Android apps from Google Play Store. To better understand some of these apps, we also test their corresponding IoT devices. Our analysis uncovers numerous security and privacy issues, leading to the leakage of private information and allowing adversaries to access user data. We find that 95/146 apps fail to adequately preserve the security and privacy of their users in one or more ways; specifically, 15 apps allow full account takeover, and 9 apps have an improper input validation check, where some of them allow an attacker to dump the database containing elderly and caregivers' sensitive information. We hope our study will raise awareness about the security and privacy risks introduced by these apps, and direct the attention of developers to strengthen their defensive measures.

Keywords: Elderly Privacy · Android Apps Privacy and Security

1 Introduction

The adoption of mobile devices is forcing the elderly to navigate the treacherous waters of a complex digital world [5], wherein online threats can even translate into offline harm. While over 53% of all elderly own a smartphone [2], and are keenly adopting mobile technology [15, 7], several studies have shown that older adults are more vulnerable to security and privacy threats than the general population [16]. According to US FBI and FTC, cybercrimes against older adults in the US have increased five times since 2014, costing over $650 million in yearly losses [4]. A combination of low self-efficacy, mistrust and lack of awareness and understanding of security hazards [18] makes the elderly reluctant to adopt cyber-secure habits, hence vulnerable.[1]

Applications for the elderly offer various services such as care-giving, e-learning, and improving physical and mental health (e.g., apps for exercise and fitness). While these apps might be used daily by the elderly, their inherent privacy and security implications are not fully known. Weaknesses in elderly apps may expose sensitive private data, sometimes on a large scale, and endanger users' safety (online and in the real world). Recent studies [10] have revealed several security and privacy issues in Android apps, but most large-scale research has been done on apps used by the general population (also see Sec. 6). A few studies have exposed privacy issues in only one particularly vulnerable group (e.g., elderly or children) on a small scale. The work on elderly groups is limited to the study of elderly behavior concerning their privacy and security.

In this paper, we perform an in-depth analysis of 146 prominent elderly Android apps. We define a list of pertinent security and privacy related issues for these apps, and analyze them for such issues (e.g., security vulnerabilities, backend issues, presence of third-party trackers, and insecure data transmission). We also analyze three IoT devices to better understand the corresponding apps and their security implications. We combine the use of several existing tools that enable dynamic and static analysis to perform a wide range of security and privacy tests.

Contributions and Notable Findings.

1. We design a hybrid approach of dynamic and static analysis for evaluating security and privacy issues in elderly apps (and their corresponding IoT devices). We inspect the apps' web traffic for personally identifiable information (PII) leakage, access control issues, improper authentication management, improper input validation, dangerous third-party library permissions, and the presence of third-party trackers.
2. We apply our analysis framework to 146 Android apps (and the IoT devices corresponding to three apps). Overall, 95/146 apps fail to adequately protect the security and privacy of users due to one or more vulnerabilities.
3. 4/146 Android apps (*GoldenApp*, *POC EVV*, *Senior Discounts*, *Damava*) do not properly authenticate their server API endpoints, allowing illegitimate access to view and obtain sensitive data such as elderly users' physical address, email, health reports, and private messages on the platform.
4. 15/146 Android apps (e.g., *40 Plus Senior Dating*, *All Well Senior Care*, *Seniority*) allow an attacker to easily compromise the account of elderly users and caregivers.
5. 9/146 (*Senior Dating*, *Empowerji*, *GoldenApp*, *Caring Village*, *EZ Care*, *Generations Homecare System*, *EllieGrid*, *Seniority*, *Tricella Health*) Android apps have improper input validation with injection attack vulnerabilities such as SQL injection, allowing an adversary to dump and modify the

[1] The term "vulnerable user" means a person "at-risk" due to his/her particular circumstances, and not to be confused with an app having a security "vulnerability".

application's database. We are assigned CVE-2022-30083 [6], with a CVSS score of 9.8/10 (critical), for the code injection issue we found in *EllieGrid*.

6. 16/146 Android apps transmit PII via HTTP to their client-side servers (e.g., *Empowerji*, *GoldenApp*), while 8/146 apps transmit PII (6/146 via HTTP and 2/146 via HTTPS) to various third-party domains.

2 Potential Privacy and Security Issues and Threat Model

Potential Security and Privacy Issues. We primarily consider two types of data that can be leaked over the network: (1) personally identifiable information (PII) and (2) smartphone device information and usage. A PII leak is any data leak which can be used to identify an individual (e.g., email ID, location/address, password, date of birth, health data, unique device serial number). Device information and usage is the combination of the device data (e.g., manufacturer, model, OS, API level, IP address, screen, battery, cellular carrier, free memory/disk, language, time zone, orientation), and user interaction (e.g., session time, button clicks, visited web pages). Device information and usage leaks can be used to identify an individual or a group of individuals. We tested the most prominent vulnerability types from the OWASP top 10 for Android, based on their CVSS scores. From that base knowledge, we define the following list of potential security and privacy issues to evaluate elderly apps.

1. Improper authentication management: The ability of an attacker to gain access to a user's account (unauthorized login).
2. Improper access control: To be able to gain or observe other users' data on a given platform without their authorization.
3. Improper input validation: Possible injection attacks (e.g., SQL injection and code injection) resulting from missing/inadequate input validation, which may compromise sensitive user data.
4. Vulnerable backend: The use of remotely exploitable outdated server software, and misconfigured or unauthenticated backend service (e.g., Firebase).
5. Plaintext transmission of authentication secrets (e.g., passwords and session IDs), which can be easily captured by a network attacker to gain unauthorized access to user accounts.
6. Insecure PII, device information and usage transmission: PII and device information and usage from the client-end is sent without encryption (i.e., plain HTTP).
7. Data transmission to third-party: Any PII and device/usage information and usage data transmitted from the client side to third-party domains/ trackers, or library providers.
8. Inadequate security configurations: Android apps with misconfigured backend HTTP web servers (e.g., lack of Cross-Origin Resource Sharing or improper flash cross-domain policy), which may lead to large-scale attacks.

9. Dangerous permissions (e.g., Write External Storage, Access Fine Location) automatically acquired by a third-party library when requested by the elderly app, or by a malicious app using the same signed certificate third-party library as the elderly app developer.

Threat Model. We consider three attacker types with varying capabilities: (1) On-device attacker: a malicious app with limited permissions on the user's device. (2) On-path attacker: an attacker who is placed between the user's smartphone and its server. This attacker can eavesdrop, modify, and behave like a man-in-the-middle attacker between the user's device and the app's backend server. (3) Remote attacker: any attacker who can connect to an app's backend server. Our threat model does not consider attacks requiring physical access to the device.

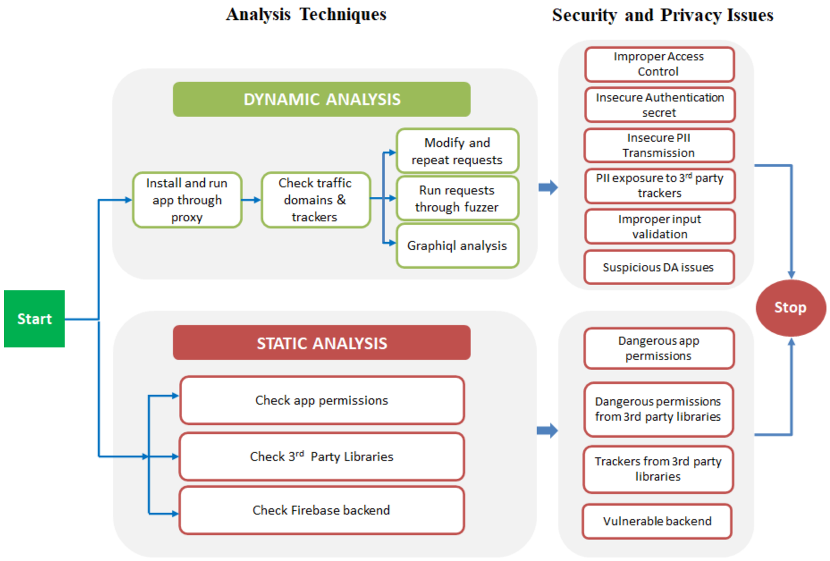
Analysis Techniques
Security and Privacy Issues
DYNAMIC ANALYSIS
Install and run app through proxy
Check traffic domains & trackers
Modify and repeat requests
Run requests through fuzzer
Graphiql analysis
Improper Access Control
Insecure Authentication secret
Insecure PII Transmission
PII exposure to 3rd party trackers
Improper input validation
Suspicious DA issues
Start
Stop
STATIC ANALYSIS
Check app permissions
Check 3rd Party Libraries
Check Firebase backend
Dangerous app permissions
Dangerous permissions from 3rd party libraries
Trackers from 3rd party libraries
Vulnerable backend

Fig. 1. Overview of our Methodology

Ethical Considerations and Responsible Disclosure. We test vulnerabilities only against accounts that we own and we do not interact with the data of any legitimate user. We do not use an existing vulnerability to exfiltrate data or pivot to other systems, i.e., we stop our analysis when we have enough evidence of a vulnerability and its impact. We also refrain from running any automated scanners that might bombard the servers to cause denial of service. As part of the responsible disclosure, we contacted the developers of our vulnerable apps to share our detailed proof-of-concept and explain to them the related security consequences. 7/35 developers contacted us back, where 2/7 were automated replies to acknowledge our email, and 5/7 developers acknowledged the issues and forwarded them to their respective security teams.

3 Analysis Methodology

In this section, we explain how we perform our static and dynamic app analysis, and also how we select our Android test apps.

3.1 App Selection

We search Google Play Store for elderly apps (and also screen the best apps for older adults [12]), with relevant keywords.[2] The search was conducted on May 20, 2021, which provided us with 500 apps for further consideration. We shortlist the apps based on the criteria that the apps are specifically designed for elderly users, their caregivers and relatives. We exclude apps that required financial account details or verified identities (e.g., bank accounts, credit card numbers, social security numbers). We manually screen each app to check if it satisfies our key requirements. Our final dataset contains 146 apps. We found that 24/146 apps have a companion IoT device, where 5/24 apps are pill managing apps and 19/24 are elderly tracking apps. We purchased 3/24 IoT devices (available without any subscription and deliverable to our location) to better understand their functionality. Altogether, these apps have been downloaded 20.8M+ times, with a range between 10M+ (*NeuroNation*) to 1000+ downloads (*CareGo* IoT companion app). Note that each caregiver/EVV app may indirectly serve (and have access to) hundreds or thousands of elderly people.

3.2 Dynamic Analysis of Traffic Flow

We perform dynamic testing of the apps to simulate the real world usage for the apps so that we can observe the apps as they were intended. We set up test environments for each app (creating user accounts, setting up the IoT device, etc), emulate user actions for 20 to 60 minutes depending on the feature-set of the app, collect traffic from the elderly apps and the IoT devices (up to 24 hours), and then perform our analysis (explained further in this section). Fig. 1 illustrates our methodology. We use Burp Suite[3] for manual dynamic analysis. Burp Suite is an integrated platform for security testing of web and mobile applications, using its various extensions. We also notice that some our apps use GraphQL [14]; note that the use of graph analytics is driving many important business applications from social network analysis to machine learning. To analyze GraphQL APIs, we use the official GraphQL IDE called GraphiQL [11] to test the network traffic on the apps using GraphQL. In-depth dynamic analysis with Burp Suite and GraphiQL[4] helps us find relevant security and privacy issues in our test apps.

The four main components for our dynamic analysis include the following: (1) *Proxy*, an intercepting proxy that lets us see and modify the contents of requests

[2] The keywords include: “elderly”, “old”, “senior”, “dementia”, “Alzheimer’s”, “retirement”, “senior dating”, “pension”, “seniority”, “caregiver”, “memory”, “maturity”, “retiree”, “Electronic Visit Verification”, “EVV”, “senior health”, “memory games”.

[3] https://portswigger.net/burp/releases/professional-community-2021-12-1

[4] https://github.com/graphql/graphiql

and responses while they are in transit. We use this component to analyze the complete network traffic of the app to check for insecure session management, insecure PII transmission to the app as well as to any third-parties, and look out for any suspicious activity from the app. (2) *Intruder*, a fuzzer used to run a set of values through an input point and perform brute-force attacks and testing rate limiting on apps. We use this component to enumerate user IDs (integer values), list of passwords and API endpoint parameters. (3) *Repeater* lets us send requests repeatedly with manual modifications to check for injection attacks and servers' response to unexpected values or requests. (4) *Decoder* lists the common encoding methods like URL, HTML, Base64, Hex, etc., when looking for chunks of data in values of parameters or headers.

We install each test app from Google Play Store and run it through Burp proxy. We analyze every request and response of the app's APIs (or any included third-party libraries) to the app server and to any third-party domain and tracker. We identify the known third-party trackers using EasyList and EasyPrivacy [8] filtering rules. We differentiate the requests with weak authentication, like the ones which are missing authentication headers or cookies, as they are more likely to be exploitable. This differentiation is done by inspecting the HTTP request headers and searching for the presence of session headers. We also identify the requests responsible for user login/logout or any transmission of user data. We pass these requests through Burp components to check for security and privacy issues. The requests transmitted via GraphQL are analyzed using GraphiQL. In particular, we first use an introspection query to read the GraphQL documentation. Then we inspect the whole documentation to read the available API calls (queries and mutations). Vulnerabilities in GraphQL are found by probing and tampering with the queries and mutations.

We assess the collected traffic to check for PII and transmitted authentication secrets, or leakage of PII to third-party domains that can be leaked via the request URL, Referer, HTTP Cookie, and requests' payload. If encoded data is observed, we use the decoder component in Burp to check for any suspicious data that is being transmitted to the domain. We also analyze the traffic to check for API endpoints with improper access control. APIs with weak authentication are checked first. We conclude that an app has improper access control if we can retrieve any other user's data (on the given app, tested using our own accounts) by changing the existing requests sent from the app to its backend server.

To check improper input validation, we follow the OWASP manual [19] to test for injection attacks to see how the apps respond to unexpected modified requests. We check for SQL injection, code injection and cross-site scripting (XSS) attacks. Any sensitive data observed is immediately deleted from our databases, and we only record the type of data that the vulnerabilities exposed.

IoT Device Analysis. For each of the selected IoT devices, we test the companion apps, radio communications and the embedded device. We test the companion apps by following the same dynamic and static analysis process as for other apps. For the radio communications, we analyze Bluetooth and WiFi communications, and we do this by inspecting the packets sent between the IoT device and smartphone. To analyze the underlying embedded device, we pop open the IoT device and analyze the functionality of the different components. We look at the debug ports and try to exploit them to gain further access to the device.

3.3 Static Analysis: Library, App Code, and Firebase

Our static analysis aims to complement the dynamic analysis to understand the apps' intended flow so that we can correlate that with our dynamic analysis to look for any suspicious behavior or weak security measures (e.g., bad input sanitization, unprotected Firebase services, etc.) which can potentially lead to privacy or security issues. We target the following components:

Third-Party Libraries. Third-party libraries are widely used by Android app developers to build new functionalities and integrate external services. For an in-depth library analysis for our elderly apps, we use LiteRadar.[5] We run the tool using our custom Python script, with the APK file to be tested, so that we can automate the data (e.g., library names, type, permissions used, etc.) collection process. We analyse the libraries in terms of their permissions and purpose.

Firebase Analysis. We analyze the Firebase configuration for security issues by performing an automated analysis using Firebase Scanner [23]. Critical misconfigurations can allow attackers to retrieve all the unprotected data stored on the cloud server and we followed a similar approach to Appthority's work [1] on scanning apps for Firebase misconfigurations.

Static Code Analysis. Mobile Security Framework[6] (MobSF) is an automated, open-source, all-in-one mobile application (Android/iOS/Windows) pen-testing framework capable of performing fast static, dynamic, and malware analysis of Android, iOS, and Windows mobile applications [24]. So, we use MobSF for static analysis of 146 apps to check for vulnerabilities related to sensitive information logged or hard-coded in files, improper usage of SQLite databases, insecure implementation of SSL, and WebView implementation. We also check the Manifest file of each app to obtain their permissions.

4 Results

Following the methodology in Sec. 3, we tested 146 Android apps for elderly people, between October 2020 and December 2021. For dynamic analysis, we ran the apps on a Samsung Galaxy M02 (SM-M022G) phone with Android 10. We report our findings in this section, with an overview of the top 30/146 apps with the most security and privacy issues in Table 1.

4.1 Improper Authentication Management

We found that 15/146 apps have authentication management vulnerabilities. Prominent examples include the following: In *Empowerji*, *40 Plus Senior Dating*,

[5] https://github.com/pkumza/LibRadar/blob/master/docs/QuickStart.md
[6] http://opensecurity.in/mobilesecurity-framework/

Table 1. Overall Results for 30/146 Elderly Apps with Maximum Security Flaws. Legend: ○: On-device Attacker ◐: On-path Attacker ●: Remote Attacker

App Name / Security Flaw	Improper Authentication Mgt.	Insecure Session Management	Insecure PII Transmission	PII Exposure to Third-party (3P)	Device Info. & Usage Exposure to 3P	Improper Input Validation	Improper Access control	Security Misconfigurations	File Path Manipulation
40 Plus Senior Dating (v9.8)	◐	◐	◐	◐	◐			◐	
Empowerji (v5.7)	◐	◐	◐		◐	●		◐	
GoldenApp (v3.2)	◐	◐	◐		◐	◐	●		
Senior Safety App (v9.7)	◐	◐	◐	◐				◐	
POC EVV (v3.2)	◐	◐	◐		◐		◐		
EZ Care (v0.0.7)	◐	◐	◐			●			
BrickHouse TrackView (v1.5.8)	◐	◐	◐		◐				
Family1st (v1.0.1)	◐	◐	◐		◐				
X-GPS Monitor (v2.10.4)	◐	◐	◐		◐				
DAGPS (v21100901)	◐	◐	◐		◐				
Tricella Health (v2.15.6)	●				◐	◐			
Oscar Senior/Enterprise (v6.8.2)			◐	◐				◐	
FlirtMatures Dating (v1.0)	◐	◐			◐				
Caring Village (v0.16.5)			◐			◐		◐	
Senior Discounts (v2.2)					◐		○	◐	
Seniority E-commerce app (v.1.0.2)	●			◐		◐			
Cougar Dating (v1.1.3)	◐		◐					◐	
Over 40 Dating Mature (v1.0)					◐				◐
Generations Homecare System (v3.3.3)						◐		◐	
Alzheimer's Daily Companion (v1.0.7)		◐	◐						
Big Launcher (v1.4)		◐		◐					
HelpAge SOS (v1.0.27)		◐	◐						
Carelinx (v3.0.1)				◐				◐	
Damava (v1.2.4)			●				●		
Doulikesenior (v1.5.1)					◐			◐	
Homage (v5.0.8)				◐				◐	
EllieGrid (v3.4.1)						●			
All Well Senior Care (v2.15.0)	●								
Mobile Caregiver+ (v2.0.35)					◐				
401(K) - Retirement Planning (v2.5)					◐				

GoldenApp, *EZ Care*, *FlirtMatures Dating*, *POC EVV* and *Cougar Dating*, the login credentials are sent in plaintext over HTTP, so any on-path attacker sniffing the traffic can get the user login credentials (e.g., *Empowerji* leaks name, email ID, password and phone number; *POC EVV* leaks the 6-digit user ID, a 4-digit PIN for login and the private messages sent between the caregiver and his/her supervisor). For *All Well Senior Care*, *Seniority* and *Tricella Health*, we successfully performed an OTP brute-force attack (on our test account). This is possible as these apps do not implement any rate limiting and the OTPs consist of 4 or less numerical digits, which can easily be enumerated (even for the worst-case scenario, where we could easily try all 10000 requests for a 4-digit number); we also verified that full account takeover by a remote attacker takes only trivial efforts. In *All Well Senior Care*, the attacker can obtain the user's health data (e.g., heart rate, blood pressure, etc.), wellness data (wake up time, steps taken, etc.), see all the hourly updates the user is providing to her caregiver, the location of the user, all the health charts which are saved on the user's account, and even the private messages of the user with their caregiver or their care group (containing multiple users in one group). Wherein user information (e.g., address, phone numbers, credit card details) can be obtained in the *Seniority* app due to improper authentication management. During our retesting, we also noticed that *Senior Safety App* fixed its issues in a software update.

4.2 Insecure Session Management

We found 10/146 apps that had their session IDs sent in plaintext over HTTP. For example, *POC EVV* exposes its session ID in plaintext over HTTP, so an on-path attacker can replay a request from this app and perform an account takeover. Also, 8/146 apps did not use any authentication secret. For example, *GoldenApp* does not make use of any authentication secret for accessing any resource (which also leads to improper access control issues which is explained further in Sec. 4.4). The app's authorization mechanism is purely based on supplying a mobile number, where there is no verification from the server's end regarding which mobile number is tied to which user. An adversary can change the mobile number from the request and log into the replaced number's account. Although the victim's number is not leaked anywhere, an on-path attacker can still see the mobile number as the communications are over HTTP. For our testing, we used only our own test phone numbers. After changing the number, the attacker can impersonate the victim, e.g., to request home services on the user's behalf. Apps like *FlirtMatures Dating* send their session IDs in plaintext over HTTP; any on-path attacker can sniff these secrets, and potentially takeover a user's account, also allowing the attacker to access user's sensitive information.

4.3 PII Exposure, Data Sharing with Third-parties and Trackers

We found that 16/146 apps send plaintext PII to their servers. Examples include: *POC EVV* (login code, login PIN, session ID during login), *40 Plus Senior Dating* (email ID and password during login), *Empowerji* (full name, email ID, password, mobile number and city), *GoldenApp* (username, mobile number, user address), and *EZ Care* (username and password during login and the private messages sent and received between a doctor and the user).

Table 2. Top 10 Trackers that Receive Traffic from 146 Elderly Apps

Tracker	# Apps
crashlytics.com	35
doubleclick.net	22
googlesyndication.com	12
google-analytics.com	8
googletagmanager.com	6
appsflyer.com	6
flurry.com	4
googleadservices.com	4
onesignal.com	3
branch.io	3

Moreover, out of the 16 apps that send plaintext PII to their own servers, 6 of them also send PII in plaintext over HTTP to third-party domains/trackers. Examples include: *Oscar Senior* (email ID, user name and profile picture sent to googleapis, and geolocation to onesignal's API endpoint), *Big Launcher* (exact geolocation to openweathermap.org), *Carelinx* (email ID to intercom.com), *40 Plus Senior Dating* (email ID, user name and profile picture sent to googleapis), *Senior Dating* (user name and password sent to googleapis).

18/146 apps send device information and usage data in plaintext (6/146 over HTTP and 12/146 over HTTPS) to third-party domains. The most common parameters include phone model and OS build version. *Empowerji* sends CPU build, Android version and firmware version to AppsFlyer (third-party domain). *Homage*, *EZ Care* and *All Well Senior Care* send WiFi, cellular information, signal strength, and a flag to check if the device is rooted or not. *Seniority* sends email ID, device information (phone model and OS build), and the product details (that the user adds to the shopping cart or buys on the app) to a third-party analytics tracker (wzrkt.com) over HTTPS.

We found that 115/146 apps communicate with 341 third-party (non-tracker) domains:[7] 66 apps communicate with Googleapis.com domains, 43 apps with Firebase sub-domains and 29 apps with Facebook domains. 72/146 apps had traffic through at least one Google domain. We found 39 unique tracker domains with 137 occurrences across 76/146 apps (see Table 2). The top 3 prevalent trackers are Crashlytics (35/146), DoubleClick (22/146) and Google Syndication (12/146). Crashlytics is a crash reporting software that helps identify bugs in the apps and report the user's activity to the app developers so they can take appro-

[7] A domain is considered to be a third-party domain if an app from a developer connects to it to enable third-party functions. Thus, the domain certificate owner is not the same as the developer of the app.

priate measures to ensure that users do not stop using their app. DoubleClick is a Google ad service. In 9 apps, we detect 10 or more third-party domains and trackers (*Senior Discounts*, *Big Keyboard & Notifications*, *Free Chat & Senior Dating*, *Senior Dating by Lauber*, *Over 40 - Find People 50*, *40 Plus Senior Dating*, *NeuroNation*, *Ianacare*, *Oscar Senior*). These apps could expose elderly users to potential voluminous in-app advertisements, and extensive tracking.

4.4 Improper Access Control

We found 4/146 apps with improper access control. *GoldenApp's* access control issues are due to insecure session management. As there are no authentication tokens or cookies in the requests, an attacker can replay the requests (even modify them) to create accounts in other users' names which can lead to misrepresentation or identity theft for the user. *POC EVV* contains a 5-digit "dcsId" parameter as the user ID in the requests which can be changed (by a remote attacker) to get other users' data (e.g., phone number, home and office address, zip code). *Senior Discounts plus Coupons* has a 6-digit parameter for the user ID that can be modified to get any other user's email ID. *Damava* also has a similar issue where an attacker can fetch the user details using a GraphQL query and then modify the user ID to get other users' data (e.g., email ID, address, criminal record). The information disclosed in *Damava* could result in a full account takeover for both the patient as well as a caregiver. We also found that the appointment details query and mutation do not implement any access control in *Damava*; an adversary can view, modify and cancel any elderly patient's appointment. Moreover, given the appointment and caregiver details, the attacker can also impersonate a caregiver to harm the patient.

4.5 Improper Input Validation

9/146 apps are vulnerable to various injection attacks such as SQL/code injection, cross-site scripting. For example, *Senior Dating by Lauber*, *GoldenApp*, *Caring Village* and *Generations Homecare System* are vulnerable to reflected cross-site scripting attacks. An attacker can execute malicious JavaScript code to fetch elderly users' detail or to phish them. We note that for this attack to work, a victim would first need to click on a malicious link crafted by the attacker. *Empowerji*, *EZ Care* and *Tricella Health* are vulnerable to SQL injection attacks, allowing an adversary to view, modify and delete any elderly user's data. *EllieGrid* and *Seniority* are vulnerable to code injection. For this attack, we added a JavaScript sleep function in the request body and then observed the response time. When there was a delay of 10 seconds for the response after the sleep command of 10 seconds, we confirmed the code injection vulnerability. This is a very serious issue that can lead to complete compromise of the application's data and functionality, and the server that's hosting the application [3]. Due to ethical reasons, we limit our attack in detecting this vulnerability. As there is no authentication secret on *EllieGrid* requests, the attacker can perform this attack remotely by constructing and sending the modified requests to the app's server.

4.6 Server-side Security Misconfigurations

We found 16/146 apps with various security misconfigurations. Apps such as *Doulikesenior*, *Carelinx*, *Pension Status Search Old Age Widow Handicap* and *Homage* transmit HTTP requests to modify an object via unprotected GET requests, and thus are vulnerable to Cross-Site Request Forgery (CSRF) attacks, mostly executed via sharing/clicking a malicious link. We found that *Over 40 Dating Mature* has a file path manipulation vulnerability where we placed user-controllable data (the file path on the app's server) into the URL path of the app's request that might be used on the server to access local resources (which may be within or outside the web root). With this vulnerability, an attacker can modify the file path to access different resources, which may contain sensitive information. For legal and ethical reasons we did not test/validate this attack.

4.7 Dangerous App Permissions

Dangerous permissions grant an app access to personal user data (e.g., user's location), or control over the user's device. They are only granted after explicit user consent. We found a total of 598 dangerous permissions in 118/146 apps, i.e., an average of 5 dangerous permissions per app. See Table 3. *Ianacare* (caregiver app) and *Life Assure* (companion app for a tracking device) had the maximum of 11 dangerous permissions (Call Phone, Camera, Write External Storage, Read External Storage, Read Calendar, Write Calendar, Read Contacts, Write Contacts, Read Phone State, Access Coarse Location, Access Fine Location, Get Accounts, Record Audio). Access Fine Location permission is needed if an app wants to know detailed information about the user's location, and respond accordingly. This is often used with advertising and location-based and social-network services like Facebook. Read Calendar allows an application to read the user's calendar data. Calendar events can, and often do contain contact information. The top 2 dangerous permissions found were Write External Storage (92 apps) and Read External Storage (91 apps). Rarely used permissions found were Read Call Log (*BIG Phone for Seniors*), Receive SMS (*Homedoctor Protección Senior*), Get Tasks (*DAGPS*) and Write Call Log (*BIG Phone for Seniors*).

84/146 apps required Access Fine Location and 75/146 apps required Access Coarse Location permission. 61/146 apps asked for Camera permission, such as *Petralex*, *Walk to End Alzheimer's*, *GoutDietRecipes*, *Seniority*, *Aveanna EVV*, and *401(K) - Retirement Planning*. Apps with a significantly high number of risky permissions include *Ianacare*, *401(K) - Retirement Planning*, *Oscar Senior*, *Senior Safety App*, *CrescendoConnect*, *Trusted Senior Care* and *ClearCareGo*.

4.8 Third-Party Libraries and Permissions

Types of Libraries. We found 122 unique third-party libraries and a total of 1008 libraries in 127/146 apps, for various purposes: app development (93/122), analytics (6/122), advertisements (6/122), and social networking (2/122). We

Table 3. 598 Dangerous Permissions Asked by 118/146 Elderly Apps

Dangerous Permission	# Apps
Write External Storage	92
Read External Storage	91
Access Fine Location	84
Access Coarse Location	75
Camera	61
Record Audio	44
Read Phone State	39
Read Contacts	29
Call Phone	27
Get Accounts	22
Write Settings	9
Read Calendar	8
Write Calendar	8
Write Contacts	5
Get Tasks	1
Read Call Log	1
Receive SMS	1
Write Call Log	1

found 34/146 apps with *Facebook* social media library and 14/146 apps with advertisement libraries, mainly *Google Ads* (9 apps) and *Unity3d Ads* (3 apps).

Libraries by App Category. A high number of total third-party libraries were found in 26 caregiver apps (218/1008 libraries), 20 EVV apps (162/1008), 16 location tracking apps (150/1008), 12 dating apps (107/1008) and 6 apps for Alzheimer's (55/1008). This shows that the elderly who may need caring, or are unwell, or socially active may be more prone to privacy and data security issues arising via these third-party libraries. 48/146 apps have 10 or more unique third-party libraries. Examples of apps with a high number (>14) of unique libraries are *Knee Arthritis Exercises* (24), *Theora Link* (22), *Walk to End Alzheimer's* (19), *My SOS Family Emergency Alerts* (18), *Doulikesenior* (17), *Pension Status Search Old Age Widow Handicap* (16), *Tracki GPS* (15), and *Empowerji* (15).

Kinds of Permissions Asked. We found 3 unique signature[8] permissions asked by the libraries (Dump, Write APN Settings, Write Secure Settings). We found the Dump permission for example used by *Firebase* (64/146 apps), *Glide* (41/146 apps) and *Facebook* (32/146 apps) third-party libraries. Write Secure Settings permission was asked predominantly by *Google Mobile Services* (62/146 apps) and *Firebase* (62/146 apps). This Development Aid library permission allows an application to read or write the secure system settings. This permission should only be seen on Android system apps (and possibly wireless carriers or hardware manufacturer pre-installed apps) [26]. Write APN Settings permission was asked by *Google Play* library (4/146 apps).

[8] Enables communication between multiple apps of the same developer. Only granted if the requesting app is signed with the same certificate.

4.9 Static Code Analysis

Static code analysis with MobSF shows that 93/146 apps can read/write to External Storage; 84/146 apps execute raw SQL queries which may expose them to SQL Injection attacks; 71/146 apps use weak hash functions; 36/146 apps have insecure WebView implementation; and 12/146 apps have insecure SSL implementation, a critical security issue. Apps with all these five concerns are *Alzheimer's Disease Pocketcard*, *Big Keyboard & Notifications*, *Over 40 - Find People 50*, *Pill Reminder & Medicine App*, *Doulikesenior*, *Senior Safety App* and *Silver 50 Dating.*

4.10 Apps with an IoT Device

We acquired IoT devices that operate with 3/24 IoT companion apps: *EllieGrid*, *Carego Alphahom*, *Tuya SOS*, and tested them to understand the relationship between the device and app. We analyzed the app behavior for the remaining 21/24 IoT companion apps, to the extent possible without the IoT device, and found issues in 7/24 apps. *X-GPS Monitor*, *Family1st*, *BrickHouse TrackView* and *DAGPS* are IoT companion apps which help family members track their elderly loved ones via IoT devices. All these 4 apps had 3 main issues: (1) improper authentication management, allowing an on-path attacker to sniff the username and password for full account takeover, (2) insecure session management, i.e., there is no use of authentication secrets in their requests, allowing an attacker to replay the requests, and (3) insecure PII transmission, leaking username and password in plaintext over HTTP. An on-path attacker can exploit these issues to track the exact location of users wherever they go with their IoT device due to full account takeover.

Tricella Health and *EllieGrid* are smart pill organizers which make medication management easier and are specifically designed to help the elderly by reminding them to take their pills on time (they consist of a pillbox and an app). *Tricella Health* has improper authentication management, where it is vulnerable to a remote OTP (3-digit number) brute-force attack during login and registration, leading to full account takeover. It also has improper input validation where its login requests are vulnerable to SQL injection attacks. These issues can lead to a remote attacker changing the user's medications causing the user to take the wrong medications, skip doses or overdose.

EllieGrid's physical pillbox is designed to store pills and receive reminders as ring notifications. The reminders and medications can be set up in its companion app. We found two major vulnerabilities in *EllieGrid.* Firstly, it offers a functionality to alert the elderly user's caregiver via email and phone, when the pillbox is not opened on time. We note that there is no access control present in this functionality, and a remote adversary can completely tamper with the associated caregiver's detail by using the caregiver profile setup option, which is present on the app UI. An adversary can enumerate a caregiver's ID by brute-forcing, and then supply it to modify the caregiver's email and send the alerts to an attacker under his control, which would allow him to track the elderly users' activities and collect their pill taking habits; additionally, the legitimate caregiver will not receive any further notifications from the pillbox. Secondly, the *EllieGrid* solution offers a paid plan with additional functionalities for the elderly, such as viewing weekly adherence reports and adding a caregiver. Specifically, we found a parameter *subscriptionTypeId* from the user profile API, which sets the value of the current plan. An adversary can set this parameter's value to *premium* and upgrade their *EllieGrid* account for free.

We also found some vulnerabilities by following the static analysis approach in *Carego Alphahom*, which provides a personal alarm system for the elderly. In particular, the app is vulnerable to the Janus vulnerability [17], in which an adversary can prepend a malicious DEX file to an APK file while keeping its signature unaffected. Android versions 5.0 - 8.1 accept the file as a valid APK.

4.11 Firebase Analysis

84/146 Android apps use Google Firebase as a backend service and we found 4/84 apps whose Firebase DB was exposed publicly. For ethical reasons and to protect other customers' privacy, we created elderly accounts on the four apps. Then, we updated the Firebase scanner to automatically search for our test data in its response and record the leaked information from our own account. 2/4 apps (*CogniFit* and *Carely*) fixed this issue during the time of our testing. For *UnitedHealthcare EVV Tennessee* and *Amerigroup EVV Tennessee*, at the time of testing, we could not see any sensitive data being stored on their databases.

5 Limitations

As Google Play Store does not have a defined "Elderly" or "Senior" app category, our app search is limited to the keywords used. A major limitation we faced during our dynamic analysis was the inability to create accounts for 67/115 of our apps because the companies either make accounts for the users beforehand (and provide access information) or the apps will validate the user's information (e.g., medical insurance numbers, and organization email IDs, which we cannot provide in our test accounts) before creating the account. This was most applicable for the EVV and caregiver apps. For those apps, we conduct a limited dynamic analysis of pre-login application behaviors. We also did not test any paid apps.

6 Related Work

Slane et al. [25] collected seniors' perspectives on technological devices and applications to show how seniors protect their personal information, and what knowledge, tools, and support they would need in order to consider new functions or devices. Huckvale et al. [13] assessed 79 clinically safe medical/health apps used by chronic and unwell persons, and found that 23/79 of apps sent unencrypted PII over the Internet, 63/79 apps communicated directly with third-party services and 53/79 of apps had some form of privacy policy. However, this work does not specifically study elderly users, or analyze the backend security issues. Frik et al. [9] identified a range of complex privacy and security attitudes and needs specific to *older adults*, along with common threat models, misconceptions, and mitigation strategies. They showed how older adults' limited technical knowledge, experience, and declining abilities amplify vulnerability to certain risks. Oliveira et al. [20] showed that *older women* were the most vulnerable group to phishing attacks in a study of 158 internet users. Razaghpanah et al. [21] identified 2,121 third-party advertising and tracking services at the traffic level, of which 233 were previously unknown to other popular advertising and tracking blacklists. Their analysis of the privacy policies of the largest advertising and tracking service providers showed rampant sharing of harvested data with subsidiaries and third-party affiliates. Ren et al. [22] analyzed 512 apps for privacy leaks over time across three dimensions (PII leaks, HTTPS adoption, and domains contacted) independently, and found that app privacy gets worse as users upgrade apps and all apps leak at least one type of PII.

In contrast to the above work, we take an in-depth look at the security and privacy threats in Android apps used by the elderly. We also analyze traffic flows, PII or device information and usage leaks, dangerous permissions used by apps and third-party libraries, and backend security issues of high severity, using various tools for both dynamic and static analysis. Our initial framework also included Lumen Privacy Monitor for dynamic analysis of test apps, but we removed it from the framework as we found that Lumen did not uncover several security flaws as compared to Burp Suite. Even though Lumen would show leaks, there was a layer of uncertainty as to how a leak was transmitted (over HTTP or HTTPS) and where it was leaked (to the client-side product itself or to some third-party domains). Also, Lumen did not work reliably on newer Android versions and only worked best below Android 7. Hence, we decided to use a more manual approach with Burp Suite.

7 Conclusion

We presented a comprehensive analysis of 146 Android apps that are intended to assist elderly people. Our methodology included dynamic analysis of traffic domain flows, trackers, leaks, and permissions, static analysis of third-party libraries for risky permissions and vulnerable backend issues using various automated as well as manual tools. We reveal individually many red flags in 30/146 apps and how they are most likely to be a security risk. But also, in a wider sense, we have noticed trends in apps' permissions and domain flows which show us how some companies, third-party libraries, or permissions dominate the segments. This is why we think the analysis should not stop here, as we can delve even deeper to find more flaws and vulnerabilities. This will create a safe environment for the elderly to have the peace of mind that their new smartphones are safe and they have one less thing to worry about.

Acknowledgements

This work was partly supported by a grant from the Office of the Privacy Commissioner of Canada (OPC) Contributions Program.

References

1. Arghire, I.: Thousands of mobile apps leak data from Firebase databases (2018), online article. https://www.securityweek.com/thousands-mobile-apps-leak-data-firebase-databases
2. Bengfort, J.: Senior care and mobility: Why smartphones and tablets make sense. (2019), online article. https://healthtechmagazine.net/article/2019/11/senior-care-and-mobility-why-smartphones-and-tablets-make-sense
3. Choi, H., Kim, Y.: Large-scale analysis of remote code injection attacks in Android apps. Security and Communication Networks **2018**, 1–17 (04 2018). https://doi.org/10.1155/2018/2489214
4. CNBC.com: Here's how online scammers prey on older Americans, and what they should know to fight back (Nov 2019), online article. https://www.cnbc.com/2019/11/23/new-research-pinpoints-how-elderly-people-are-targeted-in-online-scams.html
5. Columbus, L.: Roundup of internet of things forecasts. (2017), online article. https://www.forbes.com/sites/louiscolumbus/2017/12/10/2017-roundup-of-internet-of-things-forecasts/?sh=4f00f1d11480
6. CVE.mitre.org: Cve-2022-30083 (May 2022), https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-30083
7. Davidson, J., Schimmele, C.: Evolving internet use among canadian seniors. statistics canada research paper series. (2019), online article. https://www150.statcan.gc.ca/n1/pub/11f0019m/11f0019m2019015-eng.htm
8. Easylist.to: Easylist. (2022), online article. https://easylist.to/
9. Frik, A., Nurgalieva, L., Bernd, J., Lee, J.S., Schaub, F., Egelman, S.: Privacy and security threat models and mitigation strategies of older adults. In: Proceedings of the Fifteenth USENIX Conference on Usable Privacy and Security. p. 21–40. SOUPS'19, USENIX Association, USA (2019)
10. Gibler, C., Crussell, J., Erickson, J., Chen, H.: Androidleaks: Automatically detecting potential privacy leaks in android applications on a large scale. In: International Conference on Trust and Trustworthy Computing. pp. 291–307. Springer (2012)
11. Github.com: graphiql (January 2022), https://github.com/graphql/graphiql
12. Hoyt, J.: Senior citizen apps. (2020), online article. https://www.seniorliving.org/cell-phone/apps/
13. Huckvale, K., Prieto, J.T., Tilney, M., Benghozi, P.J., Car, J.: Unaddressed privacy risks in accredited health and wellness apps: a cross-sectional systematic assessment. BMC medicine **13**(1), 1–13 (2015)
14. Jindal, A., Madden, S.: Graphiql: A graph intuitive query language for relational databases. In: 2014 IEEE International Conference on Big Data (Big Data). pp. 441–450. IEEE (2014)
15. Kakulla, B.N.: Older adults keep pace on tech usage. AARP Research (2020), https://www.aarp.org/research/topics/technology/info-2019/2020-technology-trends-older-americans.html
16. Maaß, W.: The Elderly and the Internet: How Senior Citizens Deal with Online Privacy, pp. 235–249. Springer Berlin Heidelberg (2011)
17. Medium.com: Exploiting apps vulnerable to janus (cve-2017–13156)., online article (26 March 2021). https://medium.com/mobis3c/exploiting-apps-vulnerable-to-janus-cve-2017-13156-8d52c983b4e0
18. Morrison, B., Coventry, L., Briggs, P.: How do older adults feel about engaging with cyber-security? Human Behavior and Emerging Technologies **3**(5), 1033–1049 (2021)

19. Muscat, I.: What are injection attacks (April 2019), online article. https://www.acunetix.com/blog/articles/injection-attacks
20. Oliveira, D., Rocha, H., Yang, H., Ellis, D., Dommaraju, S., Muradoglu, M., Weir, D., Soliman, A., Lin, T., Ebner, N.: Dissecting spear phishing emails for older vs young adults: On the interplay of weapons of influence and life domains in predicting susceptibility to phishing. In: Proceedings of the 2017 chi conference on human factors in computing systems. pp. 6412–6424 (2017)
21. Razaghpanah, A., Nithyanand, R., Vallina-Rodriguez, N., Sundaresan, S., Allman, M., Kreibich, C., Gill, P., et al.: Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In: The 25th Annual Network and Distributed System Security Symposium (NDSS 2018) (2018)
22. Ren, J., Lindorfer, M., Dubois, D.J., Rao, A., Choffnes, D., Vallina-Rodriguez, N., et al.: Bug fixes, improvements,... and privacy leaks. In: The 25th Annual Network and Distributed System Security Symposium (NDSS 2018) (2018)
23. Sahni, S.: Firebase scanner., online article (28 February 2018). https://github.com/shivsahni/FireBaseScanner
24. Shirke, K.: Mobile security framework (mobsf) static analysis (Jan 2019), online article. https://medium.com/@kshitishirke/mobile-security-framework-mobsf-static-analysis-df22fcdae46e
25. Slane, A., Pedersen, I., Hung, P.C.K.: Involving seniors in developing privacy best practices: Towards the development of social support technologies for seniors. in: Office of the privacy commissioner of canada., online article (2020). https://www.priv.gc.ca/en/opc-actions-and-decisions/research/funding-for-privacy-research-and-knowledge-translation/completed-contributions-program-projects/2019-2020/p_2019-20_03/
26. XDA-developers.com: Android permissions & security explained, online article. https://forum.xda-developers.com/t/android-permissions-security-explained.2312066/