Final Report for OPC Contributions Program 2024-2025

“Privacy Concerns in Social Login Ecosystems”

Mohammad Mannan and Amr Youssef

Concordia Institute for Information Systems Engineering (CIISE)
Concordia University, Montreal

Abstract

Federated Single Sign-On (SSO) is a widely used authentication method that delegates user login to Identity Providers (IdPs) such as Google and Facebook. While convenient, SSO raises privacy and security concerns, particularly, as we observed, when permissions vary across different platforms (web vs. mobile, even different versions of an app). Existing work on SSO logins completely lacks the exploration of such variances, and their privacy consequences, even though many users may use a service both via web and mobile platforms. This study examines such discrepancies at scale, alongside an analysis of dangerous permissions specifically requested on websites and Android apps. We developed a framework to automate SSO logins on both platforms, systematically measuring permission discrepancies. Our analysis, based on 661 and 318 successful logins using Google and Facebook SSO, respectively, across both the Android app and its corresponding website for the same service, reveals a 12.58% discrepancy in Facebook SSO permissions and a 3.48% discrepancy in Google SSO permissions between web and Android platforms. These findings, along with our analysis of top-5K Tranco websites, indicate that Android apps tend to request more intrusive permissions, underscoring the need for incremental authorization mechanisms to minimize unnecessary data exposure.

1 Introduction

Federated Single Sign-On (SSO) has emerged as a widely adopted authentication strategy, permitting websites to delegate the login process to established Identity Providers (IdPs) such as Google, Facebook, and Apple. By employing SSO protocols such as OAuth and OpenID Connect, websites allow users to access applications using their existing IdP accounts. This approach effectively integrates the user's account on the new platform with their pre-existing online identity, thus removing the need for users to manage distinct credentials for each site, leading users to prefer social logins over website-specific registration mechanisms. For example, a survey conducted by LoginRadius [44] indicates that 73.69% of individuals aged 18-25 prefer using social logins over other login and registration methods. On the flip side, through SSO, websites, referred to as Relying Parties (RPs), can access more comprehensive user profiles by requesting additional data from users, such as their birthday, location, and interests.

Privacy and security concerns of social logins have long been significant issues for users [15]. Consequently, considerable research has been focused on analyzing such issues in public SSO-supported services, and SSO protocols [47], with several frameworks [37, 49, 38, 17, 23, 22] developed to measure these issues. Specifically, Dimova et al. [14] assessed the privacy implications of OAuth authentication by examining the SSO permissions requested by various IdPs, finding that 18.53% of websites using OAuth request at least one non-minimal permission (largely unnecessary, not requested by other IdPs). Moreover, Morkonda et al. [34] discovered that popular RPs request varying amounts of user data from different IdPs, with some being significantly more privacy-intrusive, a phenomenon comparable to dark patterns in website design. In subsequent work, they introduced SPEye [35], a browser extension prototype that extracts and displays permission request information from SSO login options in RPs, focusing on three major IdPs. Several past user studies (e.g., [8, 9]) also revealed that users frequently grant permissions without fully understanding the scope of data being shared with the RPs.

A significant gap in previous research is the lack of privacy analysis on SSO permissions for mobile apps, and more critically, the discrepancies (if any) in permissions between web and mobile platforms. This is important as many users rely on mobile apps for accessing online services, and users also switch between mobile and web services at least for specific applications (e.g., checking notifications on the app and more involved usage on the website). Users may assume that logging into a mobile app with a specific IdP results in consistent data access as logging into the corresponding website; however, existing work in SSO privacy does not shed light into such specific issue. As SSO implementations on mobile apps also differ from those on websites [12, 26] (although transparent to users), privacy issues need a closer look on both platforms.

To address this gap, we develop *SSO-Scoper*, a framework designed to automate Google and Facebook social logins on websites and Android mobile apps. We choose Android due to its popularity compared to other mobile platforms (e.g., iOS), and Google and Facebook IdPs, as they are most commonly supported by websites (see e.g., [14, 7, 23]. We use *SSO-Scoper* to automatically identify, login, and collect requested permissions by RPs for a given set of website domains (top sites from the Tranco [39] list) and downloaded apps (top apps from Google Play). After collecting the list of permissions for top apps and websites, we perform various privacy analyses, including: generate statistics about the permissions, especially the more sensitive ones (beyond the minimum scopes allowed by the IdPs); and systematically compare the permissions requested on web and Android platforms for the

same services, and identify the discrepancies (if any) between web vs. app.

Our seemingly straightforward approach encountered several challenges, including: the complexities of UI automation (e.g., finding the correct login buttons) in websites (see the example in appendix A.4), and specifically in Android apps, due to the numerous ways that developers implement UI in websites and apps; the lack of an obvious mapping between an app and its corresponding website (if exists); Captcha challenges and other UI banners on some sites; and the variations in SSO login implementations across IdPs. We adequately addressed these challenges to enable our large-scale analysis. For instance, while our tool relies on text-based searches to locate SSO-related buttons, it currently cannot identify IdP logos/images on websites or apps. To mitigate Google reCAPTCHA triggers during domain login searches, we used proxy servers to rotate *SSO-Scoper*'s IP addresses. For preventative UI banners on websites (e.g., cookie consent pop-ups and ads), we installed two browser extensions to handle these interruptions.

Our main contributions and notable findings include:

1. We design and implement *SSO-Scoper*, the first such SSO permission measurement tool for automating SSO login on websites and Android apps, using Facebook and Google IdPs. We will open-source our tool to further future research in this area. We use *SSO-Scoper* to log into 1716 and 678 apps using Google and Facebook SSO, respectively (chosen from a dataset of 25K popular apps). For corresponding websites, we successfully logged into 661 and 318 using Google and Facebook SSO, respectively. To supplement this, we also logged into 733 and 265 websites from top 5K Tranco sites using Google and Facebook SSO, respectively. In total, we successfully logged into 1286 and 523 websites with Google and Facebook SSO, respectively (from a total of 6322 websites). After successful logins, we collect all the requested permissions and perform our analysis.

2. We observed that Android apps in general request more permissions than websites. For Google, 1,828 permissions were requested by 1,716 apps (1.06 permissions/app) vs. 740 permissions on 733 websites (1.01 permissions/website). For Facebook, 1,525 permissions were requested by 678 apps (2.25 permissions/app) vs. 554 permissions on 265 websites (2.09 permissions/website). Similarly, the number of permissions requested from Facebook is generally higher than those requested from Google.[1] Such trend underlines the importance of evaluating SSO permissions for apps.

3. We identified the use of 34 non-minimal Google SSO permissions (all permissions except profile info) on 6,322 websites and 138 permissions on 1,716 apps. For Facebook, there were 118 non-minimal permissions (beyond name, profile picture, and email address) on 6,322 websites, and 226 permissions on 678 apps. These permissions are more privacy-intrusive, and in many cases, not essential for users (as observed in our manual analysis, see Sec. 6).

4. Surprisingly, for the same service offered via a website and Android app, the app generally requests more intrusive permissions than the website (the opposite is also

[1] We conducted a statistical analysis with a 95% confidence level. For the comparison between the average number of permissions requested from Google (apps vs. websites), a p-value of 0.0048 was calculated. For the average number of permissions requested from Facebook (apps vs. websites), the p-value was 0.0357. Both p-values indicate statistically significant results. Additionally, when comparing the average number of permissions requested from Facebook and Google using the same platform, the p-values were less than 0.0001, further confirming significant differences in both cases.

true in a few cases). Considering all the apps and websites offering the same service, for Facebook, we identified these permission discrepancies in 12.58% of the RPs (40/318), and for Google, 3.48% (23/661) of the RPs. When a service requests different sets of permissions on its web and mobile versions, users who access both platforms may unintentionally grant the more intrusive permissions, even if a single login on the more demanding platform (web or mobile) is performed. Such potential oversharing has not been reported on past work due to their focus on websites alone.

5. As noted in the official Google and Facebook documentations and previous studies [14], users typically have the option to deny any requested permission during login, except for the default permissions. However, our analysis revealed that injecting additional permissions (permissions that have not been reviewed previously by the IdP) from the client side introduces a significant attack vector that malicious RP developers could exploit to bypass the IdP's app review process. Specifically, our attack on Facebook's SSO permissions was successful, prompting Facebook to address the vulnerability and reward us with a bounty. For Google, the results were more nuanced. While mitigation mechanisms were already in place and the behavior aligns with the intended design, the attack surface remains partially exploitable, as discussed in Sec. 7.

Ethics and Disclosure. Our experiments primarily involved logging into websites and Android apps. We used test accounts with email addresses containing the keyword "test" to clearly indicate their purpose as non-personal, experimental accounts. Throughout our automatic and manual analyses, we strictly avoided actions that could interfere with the normal operation of the websites or mobile apps. No malicious or heavy data requests were sent, and we limited our interactions to essential login and permission analysis tasks, minimizing any potential impact on the services being tested. For the 14 case study apps mentioned in Sec. 6, we contacted each app's developer, using the contact information available on their Google Play Store page, to report our findings, and inquire about the observed discrepancies. We received responses only from Smule, Badoo, and Cupid Media. Smule's response indicated that the mandatory permissions remain the same across both platforms, while the optional permissions differ. Badoo team stated that both the app and web versions only require members to share their Facebook name and profile picture—although they ask for additional non-minimal permissions. Cupid Media explained that the Android app requests additional information, like gender and birthday, to streamline the user experience by auto-populating profiles, while the web platform only requires basic authentication. Additionally, to responsibly address the risks associated with permission adjustments in Sec. 7, we disclosed our findings to both Facebook and Google. Facebook acknowledged the vulnerability, awarded us a bounty, and has since implemented a patch at the time of writing this paper.

2 Background

OAuth [21] is an open standard for access delegation that enables websites or apps to obtain limited access to user information without exposing user credentials. This standard was developed to provide a method for third-party applications to request access to protected resources hosted by service providers like Google, Facebook, and Apple. The access is granted by users through a consent-based mechanism, where they authorize the third-party service to access their data without sharing their login credentials. Over time, OAuth has

become a fundamental protocol for modern web and mobile apps, enabling secure third-party access to user resources hosted at popular services like Facebook/Google.

Table 1: Summary comparison for logins and platform coverage in related measurement studies

Ref	Year	Focus	IdPs	Platform	Target Size	# FB Logins	# Google Logins
[49]	2014	security	f	web	17,913	1,660	-
[43]	2019	security	f +2	mobile	550	128	-
[32]	2021	privacy	f G +2	web	2,500	676	688
[16]	2022	security	f	web	100K	1,900	-
[7]	2023	SSO prevalence	f G +7	web	10K	293	339
[14]	2023	privacy	f G +33	web	100K	4,743	3,400
[23]	2024	security	f G +10	web	1M	18,560	21,473
Our work	2025	privacy	f G	**mobile**	**21,163**	**678**	**1,716**
				web	6,322	523	1,286

During SSO login, users are typically prompted to grant specific permissions to the requesting application, such as access to their profile information, email address, and other personal data. These SSO permissions are often presented in a dialog box, where users can review and modify the scope of access before proceeding. The granularity of permissions allows users to control which aspects of their data are shared. If a user wishes to revoke or edit these permissions after the initial login, they can do so through the IdP website, which provides a centralized interface to manage the granted permissions, including revoking access entirely or adjusting the permissions to limit the data shared with the application.

In the context of implementing SSO using OAuth 2.0, developers are required to register their applications with an IdP such as Google or Facebook. This registration process results in the issuance of a unique application identifier, known as an *app ID*, which is used to identify the application during the OAuth authorization flow. Typically, developers provide information such as the application's name, website domain, and redirect URL during registration. Often, the application's requirements remain consistent across web and mobile platforms, necessitating the same set of SSO permissions for both web and mobile users. In such cases, developers usually opt for a single app ID with uniform permissions across platforms.

However, when different SSO permissions are required for web and mobile platforms, developers have two possible strategies. The first strategy is to use a single app ID for both platforms, adjusting the permissions in the client-side code to meet the specific needs of each platform. Alternatively, developers may register two separate app IDs–one for the web and one for the mobile platform–allowing for platform-specific control over settings, SSO permissions, and security configurations, thus addressing the distinct requirements of each platform more effectively. Regardless of the chosen strategy, IdPs recommend developers to adopt incremental authorization when requesting SSO permissions [19, 28]. This method enhances user trust and privacy by requesting permissions only when they are required for a specific functionality, rather than requesting all permissions upfront; however, such incremental permission request is yet to be adopted by RPs (c.f. [14]).

3 Related Work

The research on SSO systems, especially regarding automated login and social login usage, has been extensive due to rising concerns about security and privacy issues [42, 18, 37, 38, 22, 25, 5, 10, 48, 41]. Most work primarily however examined SSO implementations on websites, not mobile apps. Below we discuss example studies more relevant to our work.

One of the first tools developed in this domain was SSOScan [49], which aimed to uncover SSO-related vulnerabilities (e.g., access token misuse, user credential leakage) in websites that used Facebook as the IdP. Out of the 1660 sites with Facebook SSO (taken from top 20k websites), over 20% were found to be vulnerable. More recently, in the similar vein, Ghasemisharif et al. [16] introduced SAAT, a tool designed to assess account and session management practices on websites using Facebook SSO, and reveal security issues such as the lack of implementing re-authentication by most RPs to prevent compromise from hijacked IdP cookies. Jannett et al. [23] proposed SSO-MONITOR, a framework aimed at continuously monitoring/archiving the security and implementation of SSO systems on websites. From 89k SSO authentication flows on the top 1M websites, the authors found 33k violations of OAuth security best practices and 339 severe security vulnerabilities (e.g., 30 username and password leaks).

In terms of SSO security analysis, Shi et al. [43] assessed SSO implementation in Android apps with support for Facebook, WeChat, and Sina Weibo IdPs. Their study primarily identified vulnerabilities stemming from incorrect SSO implementations by testing and analyzing network traffic. Out of 23,936 apps, they successfully examined 550 apps, and found that 397 of them had flawed SSO implementations.

On the privacy analysis of SSO permissions, Dimova et al. [14] examined unnecessary data collection practices in 6211 SSO-supported websites (chosen from the CrUX top 100K websites, over 30 different IdP services). Their findings revealed that when websites request a non-minimal scope of user data, much of the information collected is often excessive (as apparent from the support of alternative SSO options like Apple that allow access to very little user data).

To understand variations in the permissions requested by websites for different IdPs (Google, Facebook, Apple, and LinkedIn), Morkonda et al. [32] developed OAuthScope, a tool for semi-automated scanning and analysis of OAuth 2.0 parameters and permissions. By checking the SSO login options on popular websites (Alexa top 500 from five countries), they revealed that websites request different categories and amounts of personal data from different IdP providers. Their work is focused on identifying privacy concerns, including dark patterns in the placement and ordering of SSO login buttons, often nudging users toward selecting IdPs that requested more permissions than others.

Apart from security or privacy issues, Ardi and Calder [7] examined the prevalence of SSO logins on top 10K CrUX websites using nine different IdPs, including Google and Facebook. They found that 51% of these websites offer a login option, and about 30% of the top 10K sites allow login via 3rd-party IdPs.

In terms of user studies focusing SSO login usage, recent work by Balash et al. [8] found that 89% of their 432 survey participants have used Google SSO at least once to log into 3rd-party apps/services. In their second survey with 214 participants, they used a browser extension to collect information about apps that have access to users' Google accounts, and surveyed users about their awareness and understanding of such access. Their findings include: most participants were not concerned about third-party apps' access to their Google account, although a significant number of participants could not fully understand what

an app can do with a specific permission (e.g., "view personal info"). Majority of the participants also reported not to review what services have access to their Google account.

In a 2013 study by Bauer et al. [9], reported that participants' understanding of the information IdPs shared with RPs was not influenced by the content of consent dialogs displayed by the IdPs, or how much information was being shared with RPs. Participants were also generally unaware of RPs' access rights (e.g., durations, frequencies) to user data. Recently, Morkonda et al. [33]conducted a 200-participant study and found that 55% of participants preferred an SSO login option as their initial login choice, and 28% of participants decided to change their login choice after viewing the comparative IdP permissions.

Research Gaps. As apparent from the above discussion, there is significant research in SSO security, privacy, and usability—mostly around the use of SSO logins for websites. Surprisingly, no privacy measurement study has been done on mobile SSO privacy issues. Consequently, our study explores privacy-sensitive permissions requested by Android apps, as well as, covers the use of non-minimal permissions in both websites and apps, and reveals the discrepancies between permission requests for the same services offered via websites and apps.

Table 1 provides a summary of relevant studies closely related to our research. For each study, the table indicates the successfully analyzed IdPs, the successfully tested dataset size, and the final number of successful logins. Our work contributes to this domain by offering a side-by-side analysis of websites and Android apps using the two most common IdPs, Facebook and Google.

4 Methodology

This section outlines the methodology we adopted for automating logins on apps and websites, as well as for analyzing the requested permissions on each platform. We detail the detection and login techniques utilized by *SSO-Scoper* along with its approach to permission analysis; see Fig. 1 for an overview. The framework comprises two primary components for automating social logins on apps and websites, along with a third component dedicated to extracting and analyzing requested permissions.

Both Facebook and Google SSO login processes allow users to edit the permissions shown during login, enabling them to proceed with the minimal default permissions, which typically include only the public information of the SSO account and the user's email address. However, in our experiment, we assumed that users do not alter the presented permissions in the login pop-up and proceed with them (c.f. [8, 9]).

4.1 SSO Logins on Android Apps

The *SSO-Scoper* component for Android app automation is designed to analyze screen elements, specifically targeting login buttons to identify available SSO options, and then execute the login process using our SSO test accounts. The orchestration and management module processes a list of predefined IdPs, specifically Facebook and Google, and conducts separate analyses for each IdP. We utilize text-based keyword searches to locate relevant buttons within the authentication process. We first identify login or registration buttons on the screen and then interact with these buttons to find Google/Facebook SSO options. If an SSO button is detected, the tool initiates the login procedure. The authentication process is divided into three distinct phases: login search, SSO search, and SSO login. Depending

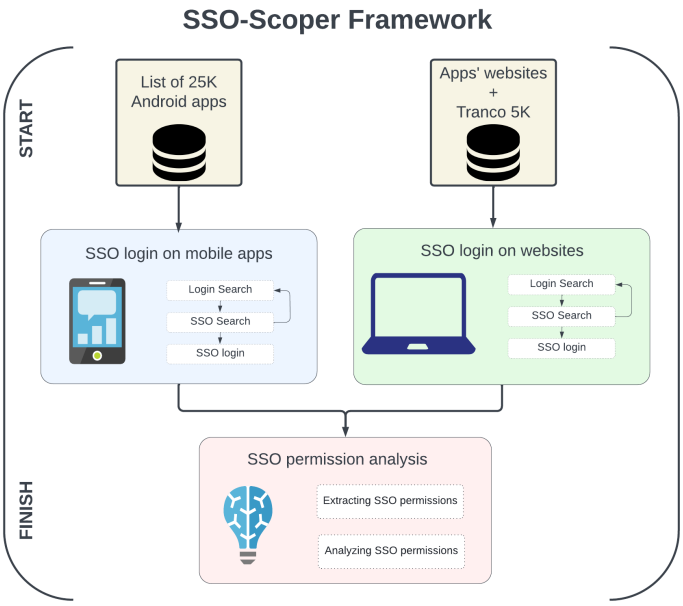


Figure 1: *SSO-Scoper* overview

on the execution of each phase, a specific set of keywords is searched within the screen elements. In the login search method, a set of hardcoded strings (e.g., register, login; for the full list, see Table 6 in the appendix), derived from a manual inspection of 50 random apps, is used to identify login or signup buttons. If these buttons are found, we then search for the IdP names (Facebook, Google) within the text of the screen elements. We also perform this search on the app's start page, as the manual analysis of 50 apps revealed that some apps present SSO login options directly on their start page. During the SSO search phase, if any screen element's text attribute contains the IdP name, the corresponding button is clicked, and the SSO login process begins, searching for our test IdP account name or email on the screen. To ensure a successful login into an application, the IdP accounts are logged into on the device. For Facebook, the Facebook app is installed and logged in, allowing it to open and request permissions when logging in to other apps using Facebook SSO. For Google, a Google account is signed in on the device, so that when logging in with Google SSO, a dialog box displaying the Google email appears. In both cases, the tool identifies the account name or email, and selects it to complete the login process.

After a successful login, the app data is erased from the device, and it is re-launched in a fresh state for the analysis of the next IdP. When both IdPs are tested, the tool uninstalls the installed app and removes its associated data from the device. The process then continues with the installation and analysis of the next app. After the tool has finished running, a list of successfully logged-in apps, along with the permissions granted to them, is automatically extracted from the IdP accounts and saved. This information is subsequently used for our analysis.

We developed this module on top of the ThirdEye framework [40] to leverage its capabilities in app execution, orchestration, and UI interaction. In addition to making modifications

to the existing code-base to suit our requirements, we added approximately 600 lines of code to the UI interactor module to implement the SSO search and login processes.

4.2 Mapping of Android Apps and Websites

For our comparison between apps and websites SSO permissions for the same services, it is essential to map Android apps to their corresponding websites. Each app's Google Play Store listing includes two URL fields: the website URL, which refers to the official domain, and the privacy policy URL. Since some apps do not have the website field populated by the developer, we also collect the privacy policy URLs (assuming that may lead to the corresponding service's website). By leveraging the Python library "tldextract" [24], we extract the domain from the privacy policy URL and use it as the corresponding website for the mobile app. This process of retrieving website and privacy policy URLs is automated via the Google Play API [36]. The final output consists of websites associated with Android apps, with the privacy policy domain used for apps without a website URL.

We manually compared the website and privacy policy domains of 100 randomly selected apps. In 79 cases, both the website and privacy policy fields matched. For 12 cases, however, the website domain differed from the privacy policy URL domain: the website field referred to the app's official site, while the privacy policy field either pointed to a static landing page—common for entertainment apps—an unrelated website used solely for hosting legal documents, or a shortened URL such as `bit.ly`. For 9 apps, the website field was left blank on the app's Google Play Store page, with the privacy policy field containing the app's website.

4.3 SSO Logins on Websites

To automate social logins on websites, we utilized the Python library "undetected_chromedriver" [46], an optimized Selenium WebDriver designed to bypass detection by potential antibot systems. This approach, previously adopted by Pham et al. [37], was complemented by using the latest version of the Chrome browser for user interface automation.

SSO-Scoper begins by processing a list of website domains (collected from apps as described in Sec. 4.2, augmented with Tranco top-5K sites). For each domain, a Google search is performed using the "login" keyword to locate the login page. The first result that matches the input domain is selected, and the tool attempts to locate the SSO login button on this page. If the button is not found, the tool sequentially examines the second search result and the root domain webpage. However, it does not consider any other links from the search results, as our manual analysis of 100 websites indicated that the login page typically appears within the first two search results.

Once a webpage is selected and opened, *SSO-Scoper* initiates a heuristic, text-based search, scanning for SSO-related regex-based keywords (see Table 7 in the appendix), within all attributes of page elements. The search prioritizes buttons first, followed by all other elements, while disregarding those with zero height or width to limit the search space and optimize performance. The list of keywords, priorities, and filters was developed from a manual analysis of 100 websites. If the tool locates these clickable keywords, it proceeds to the login phase, where it searches for the SSO account name or email address on the screen. If no SSO-related elements are found, a third phase (login search) begins, targeting keywords related to login or registration (see Table 8 in the appendix). Once a login button

is identified, *SSO-Scoper* resumes its search for SSO login buttons and proceeds with the login process if the related buttons are detected.

After each click on SSO buttons, *SSO-Scoper* calls a method to analyze the current page, and determines if SSO login can be performed. The main functionality of this method is to check if the IdP URLs used for SSO login initiation are contained in the actual URL login page that has been opened in the browser. For Facebook SSO, this pattern includes the presence of "facebook.com/login" or "facebook.com/privacy/consent" in the URL. For Google, it includes "/v3/signin/identifier?", "/o/oauth", "/gsi/select?", or "/oauth/google". The presence of these strings within the URL guides the tool to proceed with the login method.

One key aspect of our approach is the use of a fixed, pre-configured Chrome profile to facilitate the login process and address issues such as non-English websites and preventative pop-ups. In this profile, both Facebook and Google accounts are logged in, along with the installation of two Chrome extensions to further simplify the interaction with websites. The first extension, "Accept All Cookies" [1], is a Google Chrome extension that automatically accepts cookie consent on various forms of notifications or pop-ups, minimizing user interaction with the website. The second extension, "AdGuard Adblocker" [2], is employed to block advertisement pop-ups on web pages.

To address the issue of non-English languages on some websites, the Chrome profile is configured to automatically translate all content into English. This ensures that, immediately after a page loads, it is translated, allowing the tool to accurately identify login and SSO-related buttons. Based on a manual analysis of 20 non-English websites, we observed that the translation process typically completes in under 2 seconds. As a result, *SSO-Scoper* is programmed to pause for 3 seconds before processing the webpage. Additionally, the tool detects and avoids social media pages and links that might be mistakenly identified as SSO login links during analysis.

A primary challenge we encountered during this stage was frequently triggering Google's reCAPTCHA when searching for website login pages. To enhance the human-like behavior of our automation and reduce Captcha triggers, we implemented pauses between actions and used Selenium's Python module, "Action Chains" [4], to simulate mouse movements. Additionally, we set up four dedicated proxy servers for our experiment and configured Selenium WebDriver to rotate the proxy IP after analyzing every 20 domains.

Finally, a list of successfully logged-in websites' SSO names and their granted SSO permissions is automatically extracted from the IdP accounts and saved. This information is subsequently used to compare the SSO permissions requested on the corresponding websites.

4.4 Collection and Analysis of SSO Permissions

Facebook imposes a rate limit on viewing app permissions, locking the account temporarily if a certain threshold is reached. To avoid such issues, we extract all app permissions from our test Facebook and Google accounts after completing the login process for all apps and websites (i.e., not after each app/site testing). Each app appears in the Facebook and Google dashboards under its configured app SSO name (configured SSO name by the app's developer).

To map SSO names across the web and Android platforms and compare SSO permissions for each app and its corresponding website, we employed two approaches; app ID comparison, and fuzzy string matching using Levenshtein distance.

For Facebook, each app is assigned a unique app ID—a numeric string included in the query string of the URL displaying permissions. Because this app ID is unique and consistent

for each service, we used it to compare app SSO permissions across two different profiles, one associated with the web experiment and the other with the mobile experiment.

For Google, since the app IDs are not accessible through the Google dashboard, we employed the "fuzzywuzzy" [13] Python library, which uses Levenshtein distance to measure the differences between sequences of app SSO names. To ensure the accuracy of the final mapping, we manually verified the mapped SSO names across both platforms.

5 Results

In this section, we present our findings on the prevalence of SSO logins in Android apps and websites, followed by an analysis of SSO permissions and their discrepancies across the two platforms. Note that our experiments were conducted from December 2023 to September 2024. For testing apps, we utilized Pixel 4 and Pixel 6 Android devices running rooted Android 12 images, alongside a desktop running Ubuntu 22.04 to orchestrate the execution of the target apps.

5.1 Prevalence of SSO Logins on Android Apps

We began by collecting a dataset of 25K popular Android package names from various sources, including Androidrank [3], AndroZoo [27], and the Google Play Store. During analysis, 3,837 apps failed to install on our devices due to various reasons such as incompatible versions or geographic region restrictions. Of the remaining apps, *SSO-Scoper* successfully logged into 678 apps (3.20%) using Facebook as the IdP and 1716 apps (8.11%) using the Google IdP; see Table 2.

The remaining apps either did not support login via Facebook or Google SSO, or their login processes were too complex for us to navigate. This complexity often stemmed from lengthy login flows with non-standard keywords for login-related buttons, or the presence of advertisements during app startup or before the login process. We manually installed and evaluated 50 apps to assess the efficiency of *SSO-Scoper*. Out of these apps, 8 supported login with Facebook SSO, and 11 supported Google SSO. *SSO-Scoper* successfully logged into 5 apps using Facebook SSO and 7 using Google SSO. For the remaining apps, *SSO-Scoper* failed to log in due to the challenges mentioned.

Table 2: Summary of app installation and login success

# Total Android apps	25,000
# Successfully installed and analyzed	21,163/25,000 (84.65%)
# Successful login with Facebook	678/21,163 (3.20%)
# Successful login with Google	1,716/21,163 (8.11%)

In terms of permissions distribution, as expected, most apps (except a few games) request the default minimal permissions. For Facebook SSO, 99.71% of the apps requested "Name and profile picture" and 91.89% requested "Email address", and for Google SSO, 98.86% of the apps requested "See your profile info". Other commonly requested permissions include: "Birthday", "Gender", and "Photos" (for Facebook); and "Create, edit, and delete your

Google Play Games activity", "See and download your exact date of birth", and "See, create, and delete its own configuration data in your Google Drive" for Google. See Figures 6 and 7.

To compare Facebook vs. Google SSO permissions requested by the same apps, we identified 424 apps with successful SSO logins using both IdPs. In 55 apps (12.97%), Facebook permissions were more privacy-intrusive than Google permissions; and in 8 apps (1.89%), Google permissions were more privacy-intrusive than Facebook. In one app ("Fotka"), both Facebook and Google requested different non-minimal permissions (Facebook SSO requested for "Name and profile picture", "Gender", "Email address", "Birthday", "Current city", and "Hometown", while Google SSO requested "See your profile info" and "View Google Photo Library"). Note that for Google SSO results, in some cases, we used the exact permission names as appear in a user's Google account dashboard.

5.2 Prevalence of SSO Logins on Websites

We perform our tests on two sets of websites: domains that are collected from our Android apps (to compare between apps with websites), and Tranco top-5K websites (for general websites).

We extracted 1724 unique websites corresponding to the tested apps with successful IdP logins. In total, *SSO-Scoper* successfully logged into 318 websites using Facebook SSO and 661 websites using Google SSO, which corresponds to 46.90% and 38.52% successful login rates for Facebook and Google, respectively. To validate our results, we manually checked 100 randomly-selected URLs for Facebook SSO and 100 URLs for Google SSO. Surprisingly, for Facebook, only 58 of these URLs offered Facebook SSO as a login method, while 22 were static websites with no login capability, often serving as simple landing pages for mobile apps, especially common in the entertainment and gaming categories. Therefore, the true success rate of our tool is estimated at 75.86% (44/58) for Facebook SSO. In contrast, for Google SSO, 30 websites were static pages with no login option. Of the remaining URLs, 56 websites supported Google SSO, and *SSO-Scoper* successfully logged into 48 of them, achieving a success rate of 85.71%.

From the Tranco top 5K websites, 1,170 domains (23.4%) did not have login pages (as from our search results). Among the remaining websites, it successfully logged into 733 using Google SSO and 265 websites using Facebook. To assess the success rate, we randomly selected 100 websites from the top 3K Tranco domains where *SSO-Scoper* did not complete the login process. Of these 100 domains, the tool failed on 10 sites: 4 required Captcha or a confirmation button before login, and 6 used extensive customization with non-standard button names.

Overall, we assessed a total of 6,322 websites, including the top 5K Tranco sites and the websites corresponding to the Android apps. See Figures 4 and 5 for the distribution of permissions. Top 3 common permissions for Facebook are "Birthday", "Gender", and "Current City"; note that the "Current City" permission takes precedence over the "Photos" permission, which ranks higher for Android apps. For Google, the top three most common website permissions differ significantly from those on Android: "See and download your exact date of birth", "See and download your contacts", and "See your age group".

To compare Facebook vs. Google SSO permissions requested by the same websites, we identified 254 websites with successful SSO logins using both IdPs. In 23 sites (9.05%), Facebook permissions were more privacy-intrusive than Google permissions. In one case (0.39%), Google permissions were more privacy-intrusive than Facebook. On two websites, both Facebook and Google requested different non-minimal permissions.

5.3 Discrepancy of SSO Permissions Across Web and Android Apps

For Facebook SSO, we identified 40 services with different permissions between the web and Android platforms. Among these, TikTok was the only service where the Android app's *app ID* differed from that of the website. 20 additional permissions were requested exclusively by the websites, while 38 extra permissions were requested solely by the apps. For Google SSO, we found permissions discrepancies in 23 cases. Among these, 22 additional permissions were requested exclusively by the Android apps, while 14 extra permissions were requested only by the websites. The statistics of apps with permission discrepancies are shown in Table 3. Based on these findings, the Android platform typically requests more permissions than the web platform (see Fig. 2 and Fig. 3), underscoring the varying privacy practices across different SSO implementations.

Table 3: Breakdown of successful SSO logins and permissions discrepancies across Android apps and corresponding websites

	Facebook	**Google**
# Successful login on apps	678	1716
# Successful login on the apps' websites	318/678 (46.90%)	661/1716 (38.52%)
# Different permissions	40/318 (12.58%)	23/661 (3.48%)

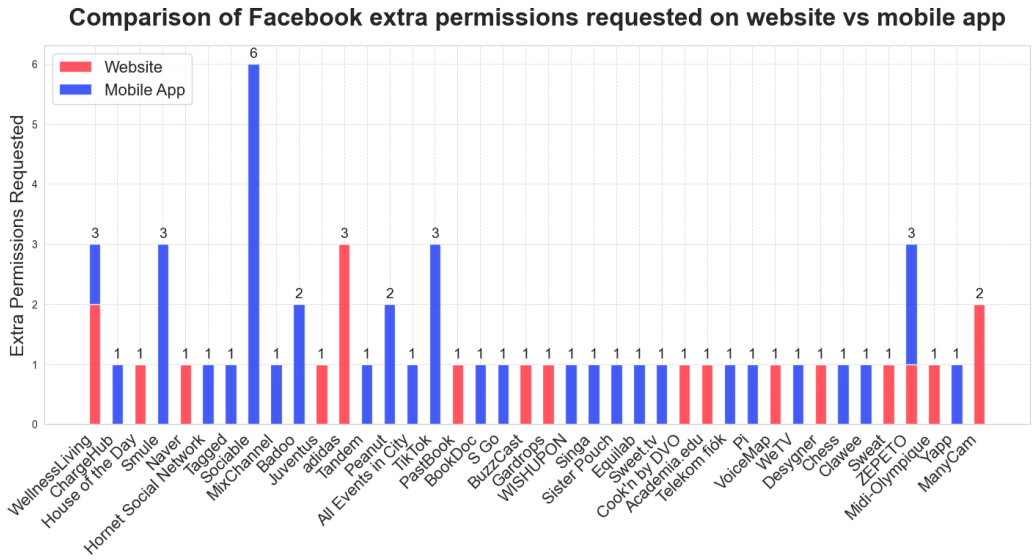


Figure 2: SSO permissions discrepancies: extra permissions requested for Facebook SSO

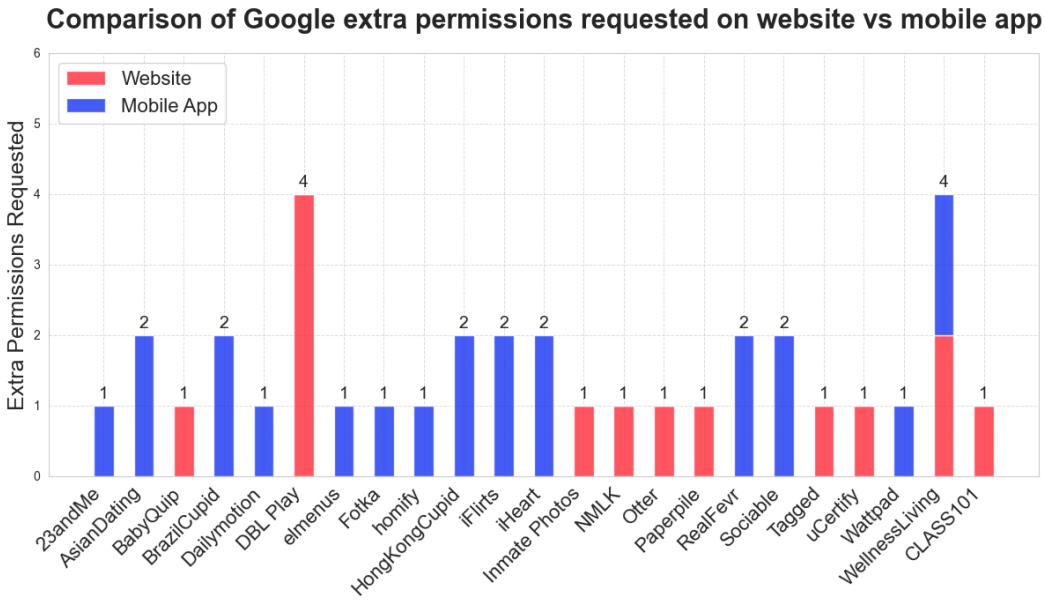


Figure 3: SSO permissions discrepancies: extra permissions requested for Google SSO

Table 4: Privacy-intrusive permissions requested by Facebook and Google on web and Android platforms

		Android App	Website	Example App/Website (#DL)
Facebook	Photos	32	14	Tinder (100M+)
	Friends list	30	11	StarMaker (100M+)
	Page likes	3	1	Sociable (1M+)
	Publish videos to timeline	0	1	`Manycam.com`
Google	Gmail Emails (full access)	5	0	Yahoo Mail (100M+)
	Google Calendar (full access)	5	2	TypeApp mail (1M+)
	Google Calendar (read access)	2	2	Lich Van Nien 2024 (5M+)
	Contacts (full access)	2	0	Microsoft Outlook Lite (10M+)
	Contacts (read access)	2	4	Truecaller (1B+)
	Youtube Account (full access)	1	0	AutoGuard Dash Cam (1M+)
	Youtube Account (read access)	2	0	AmpMe (10M+)
	Google Fit Physical Activity (write access)	1	0	Yoga Club (100K+)
	Google Drive (read access)	1	0	Microsoft Outlook Lite (10M+)
	Google Photos (read access)	1	1	Fotka (1M+)
	Google Classroom Information (read access)	1	1	ThingLink (100K+)
	Personal Phone numbers (read access)	1	2	`Class101.net`
	Street Addresses (read access)	0	1	`Dbl.id`

5.4 Privacy-Intrusive Permissions

Throughout this study, we encountered several revealing and potentially dangerous permissions requested during login by websites and mobile apps using Facebook and Google social login options. These permissions extend beyond basic profile access, posing significant privacy risks by requesting access to more sensitive data. While some services may require these permissions for specific functionalities, IdPs like Google and Facebook recommend developers adopt incremental authorization [28, 19]. This approach ensures that permissions

are requested only when the user activates the related feature (e.g., importing Facebook photos into the app), reducing unnecessary access to sensitive information for features users may choose not to use; see Table 4.

In the case of Facebook, the following less common but highly intrusive permissions were observed in our experiments: Photos, Friends list, Page likes (Allows the RP to view a list of all Facebook Pages a user has liked, potentially disclosing personal interests, affiliations, and political views), Publish videos to timeline (Grants the RP the ability to publish live videos to a user's timeline, group, event, or Page)

For Google, we identified the following permissions that grant extensive access to personal information: Gmail Emails (full access), Google Calendar (full access), Google Calendar (read access), Contacts (full access), Contacts (read access), Youtube Account (full access), Youtube Account (read access), Google Fit Physical Activity (write access) which access individual activities like walking, running, number of calories burned, step count or any workout activity that other apps have added to Google Fit. It also access information about physical habits that may be sensitive and could be used to make assumptions about users' fitness. Google Drive (read access), Google Photos (read access), Google Classroom Information (read access) which include access to users' Google classes and the list of students (rosters) , Personal Phone numbers (read access), Street Addresses (read access).

6 Case Studies

Here we discuss the results of our manual analysis of 14 RPs (out of 60 unique cases with discrepancies, see Sec. 5.3), where there are significant permission differences between an app and its corresponding website. We provide a summary of the discrepancies in Table 5; for details of these services, see appendix A.3.

We found only minor differences in functionality (not in core features) between web and Android versions of these services. This raises questions about the necessity of the extra permissions requested, as these differences do not seem to justify the additional data access.

Furthermore, our review of the privacy policies for these fourteen services revealed only in six cases, the extra permissions are mentioned in the privacy policy page, while eight of them provided only broad descriptions of data collection, such as basic permissions for email and name, without disclosing the additional permissions requested on the more intrusive platform. For example, the ZEPETO app requests access to the "Friends list" only on its mobile version, yet this is not mentioned in its privacy policy. Additionally, while all the services investigated had a single privacy policy covering all versions of their service, none specified platform-specific permissions for web and app versions. This lack of transparency leaves users unaware of the permissions requested on different versions of the app, raising concerns about the justification for such requests.

With the exception of two cases (Cupid Media, and ManyCam), the other apps provide an Apple SSO option for login on their websites or iOS versions. As Apple's documentation states [6], Apple only shares the user's name and/or email with RPs. Therefore, offering Apple SSO indicates that the service can operate with minimal SSO permissions.

We further compared SSO login options with manual registration by creating user accounts on both the web and Android versions of each service. With the exception of the Sociable app, which only offers SSO options for login and registration, for 9/14 services, the SSO login option was found to be more intrusive than manual registration. In 2 cases, both options had comparable privacy levels. Only for Cupid Media, manual registration required

more information (country and city on the app) than Google SSO login (no Facebook SSO support).

Table 5 provides a summary for all services. The permissions listed in the "Extra Permissions" column are those that were requested exclusively on either the website or Android app, with the corresponding platform not requiring the same permissions. Note that we disclosed our observations to all the developers of these apps as mentioned in the Introduction.

Table 5: Extra permissions requested on either the mobile platform or website for notable services

Application	#DL	SSO Type	Platform	Extra Permissions
TikTok	1B+	f	mobile	Email, Age range, Friends list
Smule	100M+	f	mobile	Email, Age range, Friends list
Badoo	100M+	f	web	Birthday, Gender
Zepeto	100M+	f	mobile	Email, Friends list
iHeart	50M+	G	mobile	Birthday, Gender
adidas	50M+	f	web	Birthday, Gender, Age range
Chess	50M+	f	web	Friends list
Tagged	50M+	f	mobile	Photos
		G	web	Contacts (read access)
Desygner	5M+	f	web	Photos
Sociable	1M+	f	mobile	Email, Birthday, Gender, Friends list, Page likes, Photos
		G	mobile	Birthday, Gender
ManyCam	1M+	f	web	Email, Publish video to timeline
AsianDating **BrazilCupid** **HongKongCupid**	1M+	G	mobile	Birthday, Gender
WellnessLiving **Achieve**	100K+	f	web	Gender, Timeline link
		G	mobile	Birthday, Google Calendar (full access)
		G	web	Secondary Google Calendars (full access), Google Calendar (read access)
InmatePhotos	100K+	G	web	Google Photos

7 Privacy Risks from Permission Adjustments

In both Google and Facebook IdPs, users have the right to deny any permissions beyond the default (which includes only basic profile information) during login, or afterward through the IdP's dashboard. Dimova et al. [14] also explored the removal of non-minimal permissions as a way to reduce privacy exposure (most websites were found to function properly without the extra permissions). While this option is available to users, we examine a potential abuse of it—to forcibly increase SSO permissions by modifying the OAuth flow from the client side by a malicious RP. For testing purposes, we selected certain websites that use Google and Facebook SSO and attempted to inject extra permissions into the OAuth scopes transferred in the requests. Importantly, these tests were conducted on existing websites rather than personal test applications, and all findings were responsibly disclosed to the respective IdPs.

As documented [20, 29], both Google and Facebook require developers to undergo a review process when requesting non-minimal permissions to ensure the necessity and non-malicious intent of the developers. However, we found that this review process can be bypassed in Facebook's case, allowing a malicious developer to request more permissions than those allowed in the review process.

We noticed that by injecting extra permissions in the "params[steps]" parameter within Facebook SSO requests, a developer can forcibly request additional permissions beyond those initially approved during the Facebook review process. This means a developer could release an app with minimal permissions or less intrusive permissions to avoid or facilitate Facebook's review process, but later request additional permissions from users to gain access to their resources. Importantly, the consent screen still displays *all* requested permissions to the user, regardless of the initial approval. We responsibly reported this finding to Facebook as a lack of permission validation on Facebook's side—i.e., not checking the requested permissions against the reviewed permissions. Facebook confirmed the vulnerability and addressed it accordingly, awarding us a bounty in recognition of our responsible disclosure.

In contrast, Google categorizes its SSO scopes into three broad levels: non-sensitive, sensitive, and restricted. The review process is conducted based on these categories. If an application is granted access to non-sensitive scopes (e.g., gender), it can access all scopes within this category (e.g., birthday, street address, and classroom rosters). Similarly, applications validated for sensitive scopes (e.g., contacts) can access any other scopes in the sensitive or non-sensitive categories. For restricted scopes, such as those related to Gmail or Google Drive, access implicitly includes all lower-tier categories. However, Google's documentation does not explain this categorization, nor did they provide clarification when directly queried. Therefore, we observed that developers can verify their application for a specific non-sensitive scope, such as a user's birthday, and later request additional permissions within the same category, such as the user's street address.

Additionally, attempts to inject a sensitive scope into the OAuth flow on a service configured for non-sensitive scopes result in a non-preventative error message. This error message reveals the developer's Gmail address, submitted as contact information during the SSO setup. In contrast, attempts to inject a restricted scope result in a stricter response, with Google blocking the OAuth flow entirely and preventing the user from proceeding.

These variations in behavior suggest that Google has implemented some safeguards to handle unforeseen permission requests, potentially mitigating associated risks to a certain extent. We responsibly disclosed our findings to Google, and their response confirmed that these behaviors are intentional.

8 Conclusion

This study analyzes discrepancies in permissions requested by Google and Facebook SSO across Android and web platforms. As part of this work, we developed *SSO-Scoper*, an automated framework for SSO logins, enabling a systematic comparison of permissions between platforms and identifying more privacy-intrusive requests. Our findings show that SSO permissions differ significantly: Facebook SSO generally requests more intrusive permissions than Google SSO, and Android apps demand more permissions than web apps. Specifically, we observed a 12.58% discrepancy in Facebook SSO and a 3.48% discrepancy in Google SSO permissions between web and Android platforms. We also uncovered potential risks, including permission validation gaps in Facebook and inconsistent behaviors in Google's

review process that could allow permissions to bypass checks. These findings highlight the need for incremental authorization, where permissions are requested only when necessary, to enhance user privacy.

References

[1] Accept all cookies (2024), https://chromewebstore.google.com/detail/accept-all-cookies/ofpnikijgfhlmmjlpkfaifhhdonchhoi, version 1.0.3

[2] Adguard adblocker (2024), https://chromewebstore.google.com/detail/adguard-adblocker/bgnkhhnnamicmpeenaelnjfhikgbkllg, version 4.4.22

[3] Adnroidrank (2024), https://www.androidrank.org/

[4] Selenium 4.25.0 documentation (2024), https://www.selenium.dev/selenium/docs/api/py/webdriver/selenium.webdriver.common.action_chains.html

[5] Al Rahat, T., Feng, Y., Tian, Y.: Oauthlint: An empirical study on oauth bugs in android applications. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 293–304. IEEE (2019)

[6] Apple: Request an authorization to the sign in with apple server (2024), https://developer.apple.com/documentation/sign_in_with_apple/request_an_authorization_to_the_sign_in_with_apple_server

[7] Ardi, C., Calder, M.: The prevalence of single sign-on on the web: towards the next generation of web content measurement. In: Proceedings of the 2023 ACM on Internet Measurement Conference. pp. 124–130 (2023)

[8] Balash, D.G., Wu, X., Grant, M., Reyes, I., Aviv, A.J.: Security and privacy perceptions of third-party application access for google accounts. In: 31st USENIX security symposium (USENIX Security 22). pp. 3397–3414 (2022)

[9] Bauer, L., Bravo-Lillo, C., Fragkaki, E., Melicher, W.: A comparison of users' perceptions of and willingness to use google, facebook, and google+ single-sign-on functionality. In: Proceedings of the 2013 ACM workshop on Digital identity management. pp. 25–36 (2013)

[10] Benolli, M., Mirheidari, S.A., Arshad, E., Crispo, B.: The full gamut of an attack: An empirical analysis of oauth csrf in the wild. In: Detection of Intrusions and Malware, and Vulnerability Assessment: 18th International Conference, DIMVA 2021, Virtual Event, July 14–16, 2021, Proceedings 18. pp. 21–41. Springer (2021)

[11] Ceci, L.: Most popular dating apps worldwide in june 2024, by number of monthly downloads (2024), https://www.statista.com/statistics/1200234/most-popular-dating-apps-worldwide-by-number-of-downloads/

[12] Chen, E.Y., Pei, Y., Chen, S., Tian, Y., Kotcher, R., Tague, P.: Oauth demystified for mobile application developers. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. pp. 892–903 (2014)

[13] Cohen, A.: Fuzzywuzzy (2020), https://pypi.org/project/fuzzywuzzy/

[14] Dimova, Y., Van Goethem, T., Joosen, W.: Everybody's looking for ssomething: A large-scale evaluation on the privacy of oauth authentication on the web. Proceedings on Privacy Enhancing Technologies (2023)

[15] Gafni, R., Nissim, D.: To social login or not login? exploring factors affecting the decision. Issues in Informing Science and Information Technology **11**(1), 57–72 (2014)

[16] Ghasemisharif, M., Kanich, C., Polakis, J.: Towards automated auditing for account and session management flaws in single sign-on deployments. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 1774–1790. IEEE (2022)

[17] Ghasemisharif, M., Ramesh, A., Checkoway, S., Kanich, C., Polakis, J.: O single sign-off, where art thou? an empirical analysis of single sign-on account hijacking and session management on the web. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 1475–1492 (2018)

[18] Göçer, B.D., Bahtiyar, Ş.: An authorization framework with oauth for fintech servers. In: 2019 4th International Conference on Computer Science and Engineering (UBMK). pp. 536–541. IEEE (2019)

[19] Google: Incremental authorization (2024), `https://developers.google.com/identity/protocols/oauth2/web-server#incrementalAuth`

[20] Google: Oauth app verification (2024), `https://support.google.com/cloud/answer/13463073`

[21] Hardt, D.: The oauth 2.0 authorization framework (2012), `https://datatracker.ietf.org/doc/html/rfc6749`

[22] Jannett, L., Mladenov, V., Mainka, C., Schwenk, J.: Distinct: identity theft using in-browser communications in dual-window single sign-on. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 1553–1567 (2022)

[23] Jannett, Louis and Westers, Maximilian and Wich, Tobias and Mainka, Christian and Mayer, Andreas and Mladenov, Vladislav: SoK: SSO-Monitor - The current state and future research directions in single sign-on security measurements. In: 2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P) (2024). https://doi.org/TBD

[24] John.Kurkowski: Tldextract (2024), `https://pypi.org/project/tldextract/`

[25] Li, W., Mitchell, C.J., Chen, T.: Oauthguard: Protecting user security and privacy with oauth 2.0 and openid connect. In: Proceedings of the 5th ACM workshop on security standardisation research workshop. pp. 35–44 (2019)

[26] Liu, X., Liu, J., Wang, W., Zhu, S.: Android single sign-on security: Issues, taxonomy and directions. Future Generation Computer Systems **89**, 402–420 (2018)

[27] du Luxembourg, U.: Androzoo (2016), `https://androzoo.uni.lu/`

[28] Meta: Facebook login best practices (2024), `https://developers.facebook.com/docs/facebook-login/best-practices`

[29] Meta: Permissions / login review (2024), https://developers.facebook.com/docs/facebook-login/guides/permissions/review/

[30] Meta: Permissions reference for meta technologies apis (2024), https://developers.facebook.com/docs/permissions

[31] Meta: Permissions with facebook login (2024), https://developers.facebook.com/docs/facebook-login/guides/permissions/

[32] Morkonda, S.G., Chiasson, S., van Oorschot, P.C.: Empirical analysis and privacy implications in oauth-based single sign-on systems. In: Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society. pp. 195–208 (2021)

[33] Morkonda, S.G., Chiasson, S., van Oorschot, P.C.: Influences of displaying permission-related information on web single sign-on login decisions. Computers & Security **139**, 103666 (2024)

[34] Morkonda, S.G., van Oorschot, P.C., Chiasson, S.: Exploring privacy implications in oauth deployments. arXiv preprint arXiv:2103.02579 (2021)

[35] Morkonda Gnanasekaran, S., Chiasson, S., Van Oorschot, P.: "sign in with... privacy": Timely disclosure of privacy differences among web sso login options. ACM Transactions on Privacy and Security (2025)

[36] Olano, F.: google-play-api (2024), https://github.com/facundoolano/google-play-api

[37] Pham, T.H., Vo, Q.H., Dao, H., Fukuda, K.: Ssologin: A framework for automated web privacy measurement with sso logins. In: Proceedings of the 18th Asian Internet Engineering Conference. pp. 69–77 (2023)

[38] Philippaerts, P., Preuveneers, D., Joosen, W.: Oauch: Exploring security compliance in the oauth 2.0 ecosystem. In: Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses. pp. 460–481 (2022)

[39] Pochat, V.L., Van Goethem, T., Tajalizadehkhoob, S., Korczyński, M., Joosen, W.: Tranco: A research-oriented top sites ranking hardened against manipulation. arXiv preprint arXiv:1806.01156 (2018)

[40] Pourali, S., Samarasinghe, N., Mannan, M.: Hidden in plain sight: exploring encrypted channels in android apps. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 2445–2458 (2022)

[41] Rahat, T.A., Feng, Y., Tian, Y.: Cerberus: Query-driven scalable vulnerability detection in oauth service provider implementations. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 2459–2473 (2022)

[42] Sadqi, Y., Belfaik, Y., Safi, S.: Web oauth-based sso systems security. In: Proceedings of the 3rd International Conference on Networking, Information Systems & Security. pp. 1–7 (2020)

[43] Shi, S., Wang, X., Lau, W.C.: Mossot: An automated blackbox tester for single sign-on vulnerabilities in mobile applications. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. pp. 269–282 (2019)

[44] Soni, R.: Loginradius releases consumer identity trend report 2022, key login methods highlighted (2022), https://www.loginradius.com/blog/identity/loginradius-consumer-identity-trend-report-2022/

[45] Team, B.: Most popular apps (2024), https://backlinko.com/most-popular-apps

[46] UltrafunkAmsterdam: undetected_chromedriver (2024), https://pypi.org/project/undetected-chromedriver/

[47] Wang, K., Bai, G., Dong, N., Dong, J.S.: A framework for formal analysis of privacy on sso protocols. In: Security and Privacy in Communication Networks: 13th International Conference, SecureComm 2017, Niagara Falls, ON, Canada, October 22–25, 2017, Proceedings 13. pp. 763–777. Springer (2018)

[48] Wei, H., Hassanshahi, B., Bai, G., Krishnan, P., Vorobyov, K.: Moscan: A model-based vulnerability scanner for web single sign-on services. In: Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis. pp. 678–681 (2021)

[49] Zhou, Y., Evans, D.: Ssoscan: automated testing of web applications for single sign-on vulnerabilities. In: 23rd USENIX Security Symposium (USENIX Security 14). pp. 495–510 (2014)

A Appendix

A.1 Detection Keywords for SSO Buttons

This appendix presents the keywords used by *SSO-Scoper* to detect SSO-related buttons in both apps and websites for login automation.

Table 6: Keywords for login detection in apps

Keywords
"register", "login", "sign up", "signup", "don't have an account? sign up", "create an account", "join", "join now", "log in", "sign in", "log in to your account", "login/signup", "profile", "login/register", "login or signup", "login or register", "register/login", "profile", "user", "continue", "options"

A.2 Distribution of SSO Permissions Across Websites and Android Apps

Fig. 4 and Fig. 5 show the distribution of Facebook and Google permissions across websites, while Fig. 6 and Fig. 7 show the distribution of Facebook and Google permissions across websites .

Table 7: Keywords for SSO button detection

SSO Provider	Keywords
Google SSO	"google", "gmail", "google+"
Facebook SSO	"facebook", "fb[*]?login", "fb[*]?sign"

Table 8: Regular expressions used for login detection in websites

Keywords
``"^(Log|Sign)[\s]?in$", "^Log in to your account$", "^Login(/|or)(SignUp|Register)$", "^Register/Log[\s]?in$", "^sign[\s]?up$", "^Profile$", "^Don't have an account? sign up$", "^Create an Account$", "^(Join|Register)[\s]?Now[\s]?$"```

A.3 Example Cases

Below we provide details of the RPs with significant discrepancies in SSO permissions between their apps and websites.

TikTok. This app is the third most popular social media app worldwide [45] with over 1 billion downloads from the Google Play Store, employs different Facebook app IDs for handling its mobile and web applications separately. During our experiments, we observed that while TikTok only requests the "Name and profile picture" permission on its website, it requests three additional permissions—"Email address", "Age range", and "Friends list"—on its Android app. The "Friends list" permission pertains to the user's list of friends who also use TikTok. When logging in with Google SSO, the app requests only the minimal permission "See your profile info" on both the web and Android platforms.

Smule: Karaoke Songs & Videos. Smule is a popular entertainment app with over 100 million downloads, supporting both Facebook and Google SSO on its mobile and web platforms. Our analysis revealed discrepancies in the permissions requested between the web and Android platforms when using Facebook SSO. On Android, Smule requests access to three additional data fields: "Email address", "Age range", and "Friends list", whereas on the web platform, it only requests the "Name and profile picture" permission. Additionally, when using Google SSO, the app requests only the minimal permission, "See your profile info". We reached out to the Smule support team regarding the discrepancies in permissions. Their response indicated that the mandatory permissions remain the same

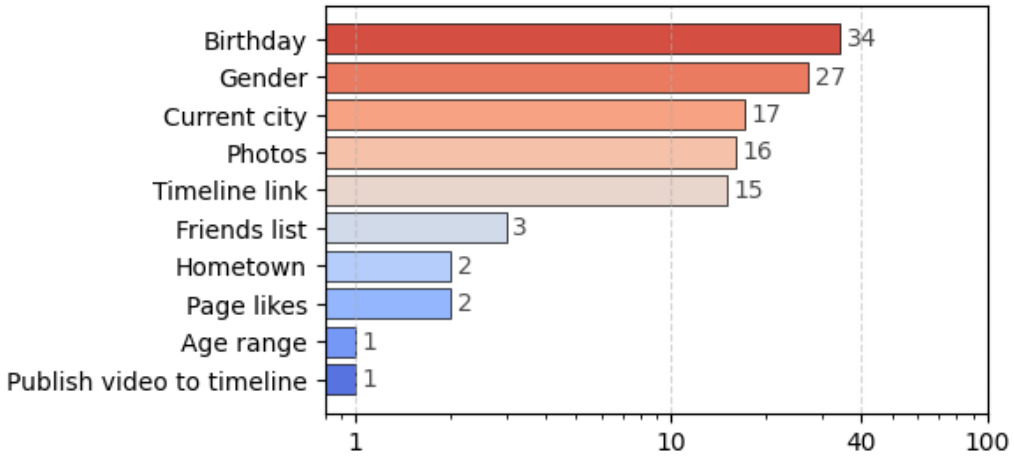


Figure 4: Distribution of Facebook permissions in websites without the two minimal permissions of "Name and profile picture" and "Email"

across both platforms, while the optional permissions differ. They clarified that it is up to the user's discretion to grant additional permissions, as they are not mandatory. According to Facebook's specifications [31], the only mandatory permission is the "Name and profile picture" field, allowing users to deny any additional permissions by modifying them during the login process. However, since these extra permissions are still presented as default requirements during login, our experiment did not account for users deliberately modifying permissions.

Badoo Dating App: Meet & Date. Badoo is recognized as the fourth most popular dating app worldwide, according to Statista's 2024 report [11]. Unlike most apps, Badoo requests more permissions when users log in using Facebook SSO on a web browser. The additional permissions include the user's "Birthday" and "Gender" from their Facebook profile, while this information is not required on the Android app or during the Google SSO login process. When using Google SSO, the app only requests the minimal permission, "See your profile info". In response to our inquiry about the differing permissions across the two platforms, they replied, stating that "both the app and web versions only require members to share their Facebook name and profile picture. Members can then optionally choose to share their email address, gender, and birthday from Facebook if they wish." However, their response did not justify the current permissions, and they did not address our follow-up question regarding the reason behind the existing difference.

ZEPETO: Avatar, Connect & Live. ZEPETO is a virtual role-playing game that allows users to create digital avatars, boasting over 100 million downloads on the Google Play Store. While the app requests only the "Name and profile picture" permission when logging into the web version via Facebook SSO, the Android version requires two additional permissions: "Email address" and "Friends list". The functionality review revealed that user registration can only be completed through the mobile app, forcing users to grant the extra permissions regardless of their necessity, as the additional permissions appear unnecessary for the app's functionality.

iHeart: Music, Radio, Podcasts. iHeart is a well-known music, radio, and podcast app

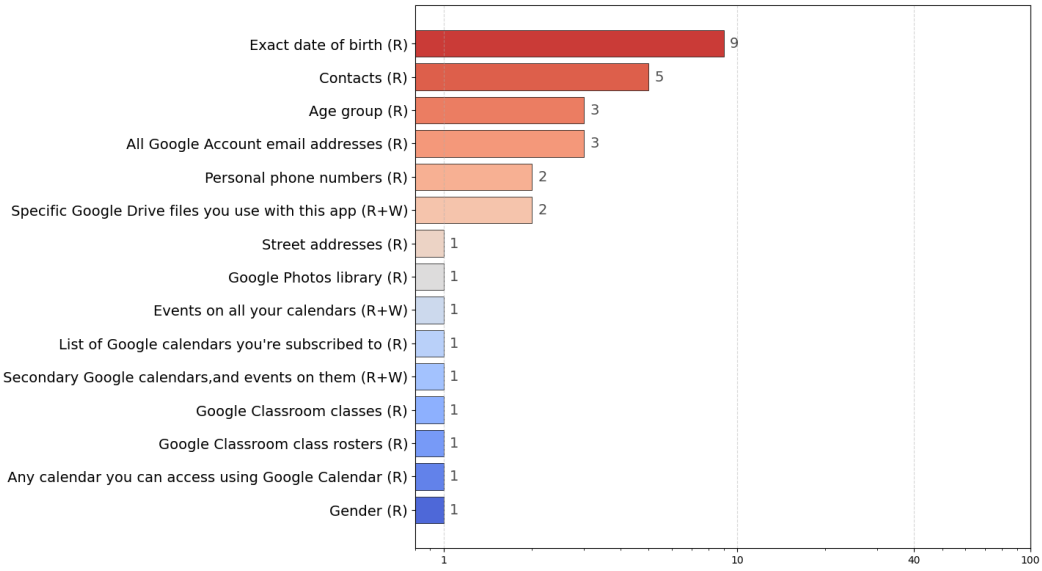


Figure 5: Distribution of Google permissions in websites without the minimal permission "See your profile info". The labels in parentheses indicate whether the permission involves *read* (R) and/or *write* (W) access.

with over 50 million downloads from the Google Play Store. This app requests minimal permissions from users on its web platform during Google SSO login. However, when using the Android app or logging in with a Facebook account, the app consistently requests access to the user's "Birthday" and "Gender", regardless of the platform.

adidas: Shop Shoes & Clothing. Adidas is a renowned shopping brand that offers web and mobile apps for online shopping, with over 50 million downloads. When using Google SSO to log in to this app, both the web and Android app platforms require only minimal mandatory permissions, while the permissions for "Age Group" and "Exact Date of Birth" are presented as optional, allowing the user to choose whether to grant them. In contrast, when using Facebook SSO on the web platform, the same permissions—"Age Group" and "Birthday"—along with the user's "Gender", are requested as mandatory, whereas these permissions are not requested on the Android app.

Chess - Play and Learn. Chess is a free, unlimited chess game with over 50 million downloads. This gaming app requests access to its users' list of friends who also use the app, only when a user logs in with their Facebook account on a web browser. In contrast, when using the mobile app or logging in with Google SSO, the app requests only the minimal permissions.

Tagged - Meet, Chat & Dating. Tagged is a dating app with over 50 million users. On the Android app of this service, the "Photos" permission is requested, granting read access to the photos a user has uploaded to Facebook [30], which may include personal and sensitive images. In contrast, logging in with Google SSO on the web results in an additional permission, "See and download your contacts", which provides read access to all of the user's Google contacts. Google specifies that this permission allows the app to view and make a copy of the user's Google Contacts, which may include names, phone numbers,

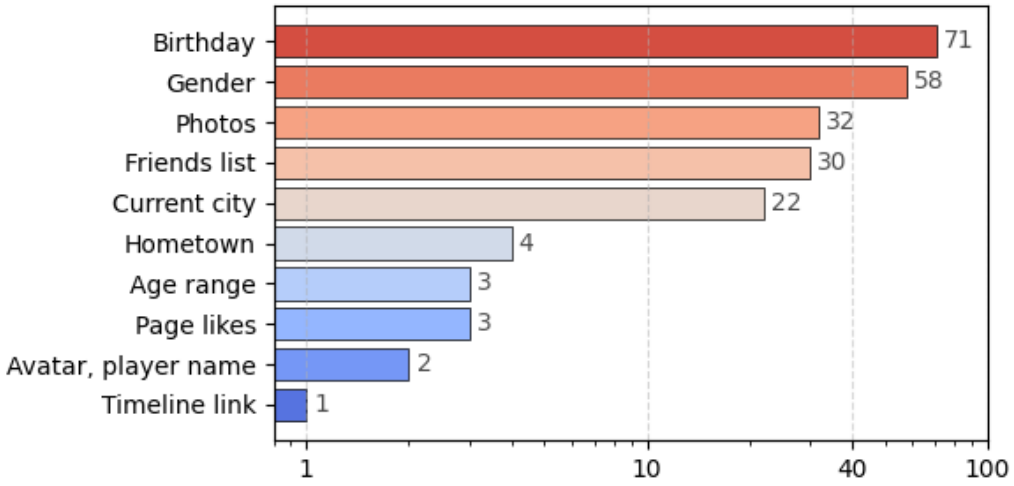


Figure 6: Distribution of Facebook permissions in Android apps without the two minimal permissions of "Name and profile picture" and "Email"

addresses, and other information about the user's acquaintances.

Desygner: Graphic Design Maker. Desygner is a business marketing app with over 5 million downloads from the Google Play Store. While the app requests minimal permissions when logging in with Google SSO, it requests access to the user's Facebook photos when the user chooses to log in with Facebook on a web browser. This permission ("Photos") is not required when the user uses the same SSO option to log in on an Android device. Regarding functionality, both the app and website offer similar functionality, allowing users to upload or import photos from Facebook. However, the web version requests the "Photos" permission during login but requires re-authentication later to access photos, while the Android app requests the permission only when the user navigates to the Gallery to upload images.

Sociable - Social Games & Chat. Sociable is a social gaming app with over 1 million downloads. This app exhibits several discrepancies in its requested permissions across web and mobile platforms, as well as between Facebook and Google SSOs. Overall, the app requests more permissions on its Android app compared to its website version. When using Facebook SSO, the app requests sensitive additional permissions, such as access to the user's photos posted on Facebook ("Photos") and a list of all Facebook Pages the user has liked ("Page likes"), among other permissions; see Table 5. In contrast, when using Google SSO, the app requests access to the user's birth date and gender exclusively on the Android platform.

After reviewing the functionality of Sociable, we found that full registration and core features like gaming are only available on the mobile app, with the website encouraging users to install the app for the complete experience. Despite extensive permission requests via Facebook and Google SSO, these permissions were not visibly utilized in either version.

ManyCam - Easy live streaming. ManyCam is a virtual camera and live streaming software with over 1 million downloads on the Google Play Store. This app requests minimal permissions when logging in with Google SSO on both Android and web platforms, as well

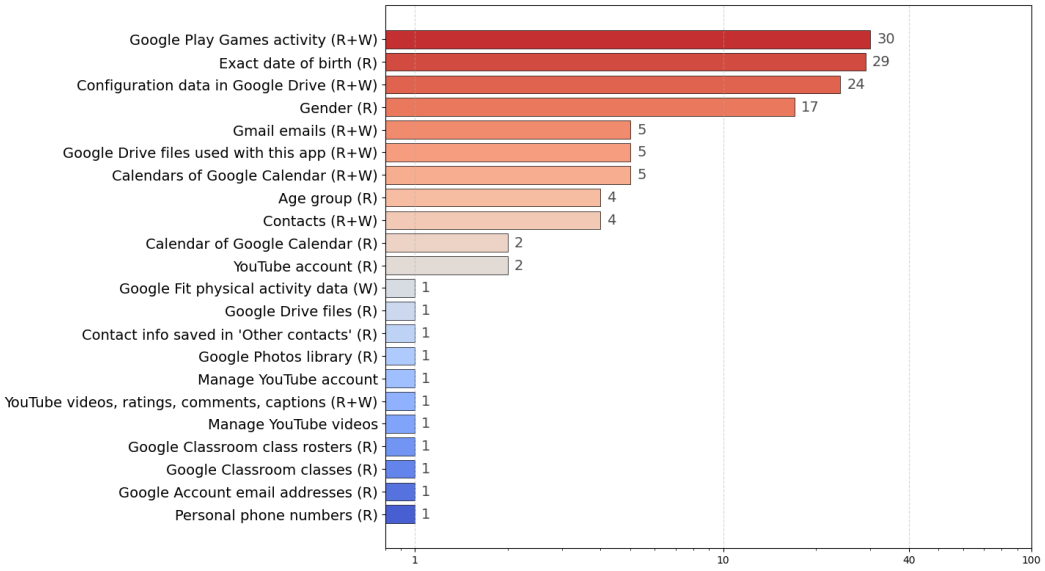


Figure 7: Distribution of Google permissions in Android apps without minimal permission "See your profile info". The labels in parentheses indicate whether the permission involves *read* (R) and/or *write* (W) access.

as during Facebook login on the Android app. However, when using Facebook SSO on the web, it requests permission to publish live videos to the user's timeline, group, event, or page ("Email address", "Publish video to your timeline on your behalf"). The ManyCam mobile app enables video streaming to Facebook, requesting permissions "Publish video to your timeline on your behalf", "Create and manage content on your Page", "Read content posted on the Page", and "Show a list of the Pages you manage". only when the user decides to initiate a stream. In contrast, the website serves as an administrative panel with no recording capabilities, making the permission to publish videos to the Facebook timeline excessive and unnecessary on the web platform.

AsianDating: Asian Dating, BrazilCupid: Brazilian Dating, and HongKongCupid Hong Kong Dating These three popular region-based dating apps are owned by "Cupid Media" and share the same theme and configurations. When logging in with Google SSO on the Android app, these apps require permissions for the user's birthday and gender, whereas these permissions are not requested on the web platform. In response to our inquiry, the vendor explained that the Android app requests additional information, like gender and birthday, to streamline the user experience by auto-populating profiles, while the web platform only requires basic authentication. However, despite this explanation, both the website and the Android app ask users for this information directly, rendering the granted permissions unnecessary.

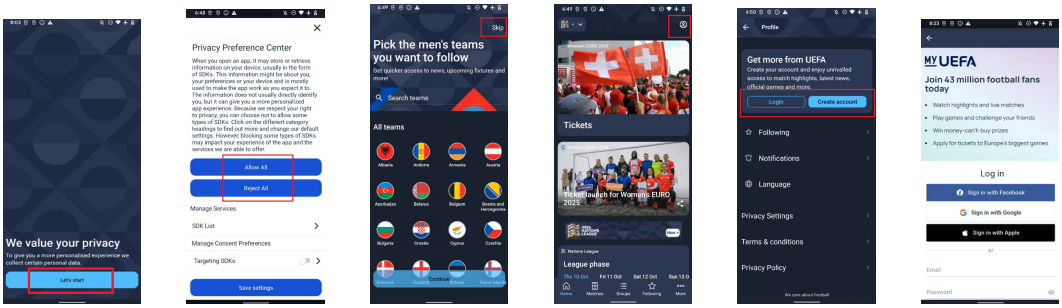
WellnessLiving Achieve. WellnessLiving is a software solution designed for art and sports studios, supporting both Google and Facebook SSO login on its web and mobile platforms. This app requests different permissions depending on the platform. When logging in with Facebook SSO, the Android app requests only "Name and profile picture" and "Email address". However, on the web browser, it additionally requests the "gender" and a

"Timeline link" to access the user's profile link. When a user chooses to log in using Google SSO, the app requests more extensive permissions on the Android platform, including access to the user's birthday ("Exact Date of Birth") and full access to all of their Google calendars ("Google Calendar, see, edit, share, and permanently delete all the calendars you can access using Google Calendar"). This permission allows the app to make changes to the user's calendars, as well as any calendar they can access via Google Calendar, including creating, changing, or deleting calendars, updating individual calendar events, modifying settings such as who can view the events, and altering who the calendar is shared with. This is a sensitive permission, as a user's calendar may contain personal contacts and private appointments. In contrast, when logging in with Google SSO on the WellnessLiving website, the service requests the following permissions: "Make secondary Google calendars, and see, create, change, and delete events on them" and "See the list of Google calendars you're subscribed to". Although Google OAuth scopes do not provide a detailed explanation for this permission, the general description suggests that the app requests full access to all of the user's calendars. Functionality testing revealed that logging into the app was not possible due to restricted access for specific accounts, though we successfully logged in on the website. However, the requirement to complete medical information forms prevented a full review of the features, leaving our functionality assessment incomplete and inconclusive.

Inmate Photos: Photos to Jail Inmate Photos is an app designed for delivering photos and pictures to inmates. While this app requests access to users' photos during both Google and Facebook SSO login, it notably does not request this permission when logging in with Facebook SSO on the Android app.

A.4 Example of Complex Login Path in Android App

Figure 8 shows an example of an Android app where the navigation path to the login is too complex for *SSO-Scoper* to detect effectively.



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4 (e) Step 5 (f) Step 6

Figure 8: Login process of the "Nations League & Women's EURO" Android app, requiring five clicks on buttons with uncommon keywords to reach the login screen. This complex navigation path causes *SSO-Scoper* to fail in detecting and completing the login.