Empirical Analysis and Privacy Implications in OAuth-based Single Sign-On Systems

Srivathsan G. Morkonda
Carleton University
Ottawa, Canada
srivathsan.morkonda@carleton.ca

Sonia Chiasson
Carleton University
Ottawa, Canada
chiasson@scs.carleton.ca

Paul C. van Oorschot
Carleton University
Ottawa, Canada
paulv@scs.carleton.ca

ABSTRACT

Single sign-on authentication systems such as OAuth 2.0 are widely used in web services. They allow users to use accounts registered with major identity providers such as Google and Facebook to login to a wide variety of independent services (relying parties). These services can both identify users and access a subset of the user's data stored with the provider. We empirically investigate the end-user privacy implications of OAuth implementations by relying parties around the world. We collect data on the use of OAuth-based logins in the Alexa Top 500 sites per country for five countries. We categorize user data made available by four identity providers (Google, Facebook, Apple, and LinkedIn) and evaluate popular services accessing user data from the SSO platforms of these providers. Many services allow users to choose from multiple login options (with different identity providers). Our results reveal that services request different categories and amounts of personal data from different providers, often with at least one choice undeniably more privacy-intrusive. We find that privacy-friendly login choices tend to be listed last, suggesting a dark pattern favoring options that release more user data. These privacy choices (and their privacy implications) are highly invisible to users. Based on our analysis, we consider challenges (e.g., opposing goals of stakeholders) in addressing these concerns and discuss ideas for further exploration.

CCS CONCEPTS

• **Security and privacy → Domain-specific security and privacy architectures**; **Human and societal aspects of security and privacy**.

KEYWORDS

OAuth; web single sign-on; empirical study; privacy; identity provider

ACM Reference Format:
Srivathsan G. Morkonda, Sonia Chiasson, and Paul C. van Oorschot. 2021. Empirical Analysis and Privacy Implications in OAuth-based Single Sign-On Systems. In *Proceedings of the 20th Workshop on Privacy in the Electronic Society (WPES '21), November 15, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3463676.3485600

WPES '21, November 15, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8527-5/21/11…$15.00
https://doi.org/10.1145/3463676.3485600

1 INTRODUCTION

An increasing number of web applications encourage users to log in to their services in exchange for a personalised experience. However, this comes with a usability problem for users having to administer a growing list of account credentials, e.g., to choose unique and strong passwords for accounts on each service. Managing large sets of credentials is a difficult task for users and can result in insecure practises such as reusing passwords and choosing weak passwords [41]. Many web authentication schemes have been proposed to improve usability and convenience. Federated single sign-on (SSO) schemes involve a trust relationship between an *identity provider* (IdP) and one or more other-party services (*relying parties* or RPs) that allow users to identify themselves on the service using login credentials registered with the IdP. The OAuth 2.0 protocol (for authorization) and the OpenID Connect protocol (for authentication) are federated SSO schemes used to establish trust in identity-related interactions between an IdP and an RP.

Instead of requiring users to create new credentials, RPs can use the OAuth 2.0 framework [26] to outsource user identification to an IdP with whom users are likely to have an existing account. As part of this transaction, the RP can also request access to additional user personal data stored with the IdP. Major identity providers (e.g., Facebook, Google, Microsoft) expose web APIs to grant RPs controllable access to protected user data stored on their platforms. With the user's permission, an IdP allows the RP access to one or more user-data attributes, which in some cases includes sensitive user data such as emails, contact information and documents in personal cloud storage—raising privacy concerns. Such user data allows the RP to extend functionality and personalise web content to the user. It also reduces implementation costs for the RP since they are outsourcing login-related tasks, including key management and credential verification, to the IdP.

Prior work (e.g., [13] [21]) involving OAuth-based SSO gives focus to security issues including leaks of user data from relying parties to other third-party actors due to implementation flaws. In contrast, we focus on privacy consequences for users of OAuth-based SSO that arise when RPs gain access to user data stored on IdP sites. In this study, we evaluate privacy practises in popular services using OAuth for SSO. Our contributions include:

- An empirical study of privacy in OAuth-based logins in the Alexa Top 500 sites for five countries. The study reveals considerable variation in how relying parties implement the different SSO options, in the data made available by identity providers, and in apparent trends across countries.
- An explication of how user choices, typically made without full facts, can result in release of considerably different amounts of user data leading to varying privacy implications.

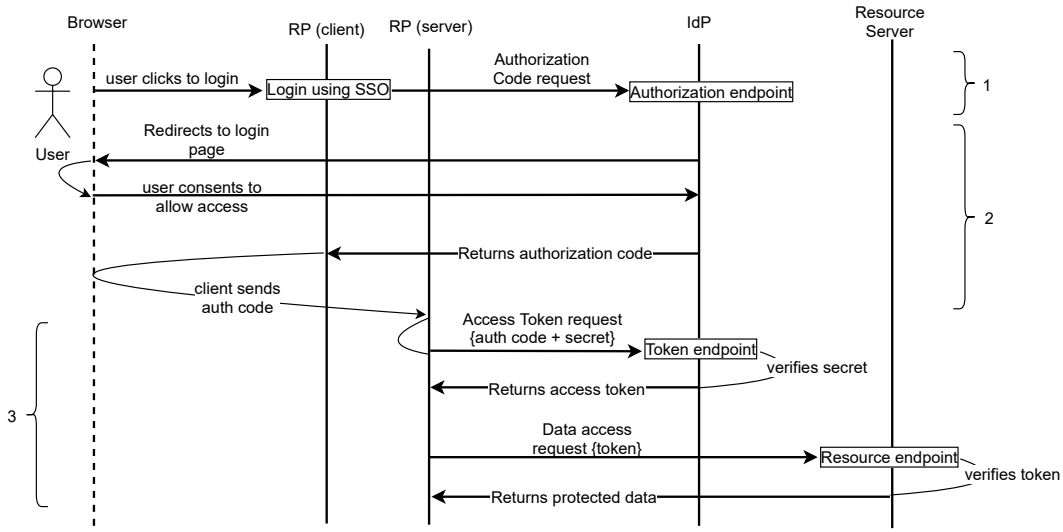

Figure 1: Procedure for OAuth 2.0 Authorization Code flow (based on [26]). Diagram for the Implicit flow is in Appendix C.

- *OAuthScope*, a tool to extract OAuth protocol request parameters from sites supporting SSO logins. The tool enables the collection and comparison of SSO data from four major IdPs (Google, Facebook, Apple, and LinkedIn).

Note: while some enterprise SSO providers (e.g., Microsoft) are popular among users, these are primarily used in closed systems using enterprise accounts. Since we target websites involving personal user accounts, we do not include these providers in our study.

The next section provides background on OAuth 2.0 framework. Section 3 introduces the OAuthScope tool. Section 4 presents the empirical study. Section 5 provides our classification of user data available in four identity providers. Results of the empirical study are reported in Section 6. Section 7 presents our analysis of privacy implications to SSO users. Section 8 describes related work. Further discussion and concluding remarks are provided in Section 9.

2 OAUTH 2.0 FRAMEWORK (BACKGROUND)

OAuth 2.0 [26] is a web resource authorization protocol popular in client-server deployments worldwide for granting applications access to protected resources without sharing the user's credentials. For example, a website prompts a user to log in and optionally allow access to specified user data by relying on their Google account rather than creating new credentials on that website. The website does not gain access to the user's Google credentials, but instead gains access to a subset of the user's data and trusts Google's verification of the user's credentials. The user is able to use the same Google account to log in on other websites that support Google as a SSO option. User data is referred to as *protected resources* and the user as the *resource owner* (RO) in OAuth 2.0 specification.

The OAuth 2.0 protocol is composed of several "grant types" that define how credentials are granted to RPs. The procedure for obtaining access to protected resources is defined by *OAuth flows* and each flow is designed to serve different use cases (and provide different levels of security). Since our study involves OAuth 2.0 use in web applications, we describe the steps involved in two flows commonly used in these applications.

2.1 Authorization Code Flow (Server-side flow)

In order for an RP to use OAuth 2.0, it must first register itself with an IdP to obtain a `client_id` (used for identification of the RP in requests). Additionally, in the case of confidential clients (RPs with the ability to securely store secrets), an associated `client_secret` is issued by the IdP. This allows an IdP to authenticate requests from an RP. We provide an overview of the three primary steps (labelled in Fig. 1) involved in the *authorization code flow* [26].

(1) **RP request to IdP**: The resource owner triggers the flow by clicking a login element, sending a request constructed by the RP to the IdP's *authorization endpoint*. The RP programs the request to add several parameters (as query components in the request URI) specifying the access request. To indicate this flow type, the RP must set the value of `response_type` to `code`. The request also includes the parameters `scope` and `redirect_uri`. The `scope` parameter lists one or more protected resources (e.g., name, email) the RP is requesting to access. The allowed values for this parameter are specified by individual IdPs based on the resources made available by the IdP. The `redirect_uri` is the endpoint the RP is requesting the IdP to redirect the RO, at the end of the flow. In order to redirect to the intended party, the OAuth 2.0 specification [26] requires the IdP to check if the URI specified in the request matches with the value registered by the RP.

(2) **IdP request to RO**: After receiving the RP's request, the IdP redirects the RO to a login page on its domain (example UIs included in Fig. 3) to login with the IdP account. If the RO is already logged in, the IdP displays a prompt to confirm resource access to the RP. In this process, the IdP provides the RO with a list of data attributes the RP is requesting to access and optionally provides the ability to deny RP the access to one or all of the requested attributes. If the RO approves the

request, the IdP issues an authorization code and redirects the RO back to the RP at the verified redirect endpoint. The code is included as a query component to the redirection URI. The standard does not specify what should happen if the RO denies access, leaving it to the IdP's discretion.

(3) **Token exchange**: From its server-side app (which is considered more secure than a channel from the RO's browser), the RP uses the `client_secret` and the authorization code to exchange for an *access token*. After verifying the code and RP's `secret`, the IdP issues an access token allowing the RP to access RO's resources within the IdP. Access tokens can vary in format (chosen by each IdP), but the purpose is to represent authorization information (e.g., allowed scope values, token expiry date). A commonly used self-contained format is the JSON Web Token (JWT) [27] that protects the integrity of information within the token.

When used by public clients (e.g., browser-based apps), the authorization code flow is susceptible to injection attacks where an attacker (on an infected client) intercepts a valid authorization code before exchanging it for an access token. The recommended countermeasure is the use of authorization code flow with Proof Key for Code Exchange (PKCE) [37].

2.2 Implicit Flow (Client-side flow)

The *implicit* grant type is simpler than the authorization code flow in that it allows the RP to directly obtain the access token without exchanging an authorization code. It is useful for browser-based (JavaScript) apps that lack the ability to securely store secrets. The RP initiates the implicit flow by sending an access request to the IdP with the parameter `response_type=token`. The IdP prompts the user with a login screen and, if access is granted, an access token is issued. This token is included in the URI fragment when redirecting the user to the RP. The redirect URI is verified by the IdP to ensure it matches the value registered by the RP. Since the access token is returned in the redirection URI, it is included in the user's browsing history and, therefore, the security of the access token relies on the security of the user's system. A malicious application with access to the user's browser could misuse the access token. Additionally, third-party scripts running within the RP site will be able to access the token. Appendix C gives a diagram for the implicit flow.

The OAuth 2.0 implicit flow was originally designed when browsers restricted apps to make requests only to its own domain [34]. This browser restriction prevented browser-based apps from using the authorization code grant since it requires sending a HTTP POST request to the IdP's authorization endpoint, which in many apps (not owned by the IdP) is different from the RP. The implicit flow in OAuth 2.0 offered a workaround that avoids using the POST request and includes the access token directly in the redirection URI. Though implicit flow provides the needed functionality, this is not recommended from a security perspective due to the risks associated with storing credentials in URIs [22]. Modern browsers now support Cross-Origin Resource Sharing (CORS) that enables a website to request resources from other permitted origins, removing the need for this workaround.

2.3 Authentication (OpenID Connect)

OAuth 2.0 can be adapted to allow an IdP to authenticate users to the RP. The OpenID Connect 1.0 [38] (OIDC) specification is designed for authentication and is built upon the OAuth 2.0 protocol. OIDC introduces a special value (`openid`) to the `scope` parameter for specifying intent to authenticate. The OIDC specification also extends OAuth 2.0's `response_type` parameter to define additional flows. If this parameter includes the value `id_token`, the OpenID Provider (OP) will issue an *ID token* to the RP after a successful authentication, encoded as a JWT [27] and containing key-value pairs (*Claims*) about the user's identity. *Claims* are digitally signed using JSON Web Signature (JWS). Since many RPs use both OIDC and OAuth 2.0, we refer to *OpenID Providers* as IdPs in our study. OIDC allows any combination of `id_token`, OAuth 2.0 values `code` and `token` for the `response_type` parameter [38]. Each combination refers to a *hybrid flow* that defines the values (access token, authorization code and ID token) included in IdP response and the endpoint (authorization and token) issuing the values.

Although the OIDC standard is built on top of OAuth 2.0 specification, some identity providers instead use custom modifications of OAuth 2.0 to provide authentication capabilities. In Section 5, we briefly describe these modifications as part of our analysis of the four identity providers.

2.4 Refresh Tokens

An OAuth 2.0 access token is issued for a limited lifetime and once it expires, the user is required to approve access again in order for a new access token to be issued to the RP. A common access token type is "bearer" token that allows use by any party in possession of the token. An unauthorized party in possession of such tokens can use it to access protected resources. While long-lived access tokens improve user experience by reducing the frequency of required logins, the longer life increases the risk of token leaks. The recommended practise is to combine the use of an access token with a *refresh token*, a string issued by the IdP that allows the RP to extend existing access without involving the user to approve access again. The RP can extend its access by exchanging the refresh token for a new access token with a scope (identical or lesser than the previously issued access token) defined by the IdP. In this exchange, the IdP validates the refresh token before issuing a renewed access token and optionally, a new refresh token. Due to the sensitivity of refresh tokens, the specification restricts its use to only server-side flows such as the authorization code flow [26]. Other protection mechanisms against token theft are discussed in Appendix B.

3 THE OAUTHSCOPE TOOL

We designed and built OAuthScope, a web tool that scans and extracts OAuth 2.0 protocol-related parameters from authorization requests made by relying parties to identity providers. We provide a list of URLs as input, and OAuthScope visits each URL in a headless browser setup from a local server. It scans each site to locate OAuth-based SSO requests to any of the four providers in our study (i.e., Google, Facebook, Apple, and LinkedIn). For each SSO option available on the RP, OAuthScope simulates the user action to trigger an OAuth 2.0 request, which initiates an authorization request by the RP to the IdP's authorization endpoint. OAuthScope captures

the parameters included in the request, including the flow type and `scope` parameter that specifies the protected resources for which the application intends to obtain access. OAuthScope uses *Selenium WebDriver* [39], an open-source browser automation framework built primarily for automated in-browser testing of web applications. OAuthScope uses the *WebDriver* framework for identifying HTML elements on an RP site and simulating web page navigation to extract OAuth 2.0 protocol data from supported IdP sites. We summarize the primary steps performed by OAuthScope as follows:

(1) **Identify login element**: After loading a website's landing page, OAuthScope searches for potential login elements based on a set of predefined match criteria and simulates a user click if a target element is found. Most RPs display login forms in either a new page or a new `iframe` within the landing page. We consider both cases and switch the WebDriver context as necessary. Our tool captures a screenshot of the login page, for use in manual verification of correctness.
(2) **Identify SSO elements**: On the new page (or `iframe`), it performs a search for HTML elements that potentially lead to a login page of one of the four IdPs in our study. OAuthScope identifies SSO elements based on element texts and HTML tag values commonly found in RPs pointing to IdPs. After obtaining potential SSO login elements, our tool triggers each target element and checks if the resulting page's URL matches with a list of predefined endpoints for each IdP in our study. We built this list to contain OAuth 2.0 endpoints published on each IdP's developer documentation pages.
(3) **Extract OAuth 2.0 parameters**: The OAuth 2.0 framework specifies that protocol parameters should be added as query components of the request URI (an example is listed in Appendix A) initiated by an RP. OAuthScope extracts these parameters encoded in the link as key-value pairs. Finally, the parameters and login screenshots are persisted to a database for further analysis.

We note that the current version of OAuthScope does not automatically ensure identification of all SSO elements in all websites. A manual inspection of the web pages to identify any skipped IdP logins allows for ensuring that our dataset (as defined) is complete. To facilitate this process, our tool captures screenshots of the login pages and of the landing page for any site where our tool did not find a login element, and stores the screenshots to a database. We use these images to identify any SSO options missed by the automated scan and manually feed the SSO login links to OAuthScope for extraction of the protocol parameters.

Although full automation of data collection is not required for our evaluation, we did iteratively improve our match criteria based on strings and tags found in RP sites during our data collection process. This gradually reduced the need for manual involvement in data collection. OAuthScope includes a web front-end (included in Appendix D) for displaying and filtering entries in the dataset.

4 EMPIRICAL STUDY: OVERVIEW

We conducted a study of SSO systems in popular sites that implement the OAuth 2.0 framework for identifying users and accessing user data stored with different IdPs. Alexa's Top 500 sites [3] provide a snapshot of the most visited sites in a given country based on 1-month traffic analysis. In an initial exploration, we manually scanned the top 500 sites in a first target country and found that three major IdPs (Google, Facebook, and Apple) are predominantly supported as SSO options. In addition to these providers which primarily contain personal data of users, we also included LinkedIn as an IdP given its popularity as a platform for sharing professional data. Similar to LinkedIn SSO, Log in with Twitter is offered by a number of top 500 US sites. Twitter does not use the standard OAuth scope parameter for indicating the user data released to sites and for this reason, is not included in our analysis.

For each of these four IdPs, we collected OAuth 2.0 protocol-related data from the Alexa Top 500 sites in five countries: Australia, Canada, Germany, India, and the United States. Our aim was to diversify geographically and include countries with different privacy regulations. To simplify analysis, we did not include certain countries (e.g., China) as the popular IdPs supported by the top sites differed from other countries. We describe our data collection procedure below.

4.1 Research Questions

The goal of this study is to understand the privacy implications for users opting to use OAuth-based SSO to log in to the top websites. To achieve this, we pursue the following research questions:

RQ1: What categories of user data do relying parties request from SSO providers? (Sec. 6.2, 6.3)
RQ2: How prevalent is each SSO scheme in popular relying parties across the five countries? (Sec. 6.4)
RQ3: If a relying party supports multiple SSO options, how do they differ in terms of requested user-data attributes? (Sec. 6.3)

Classifying RPs by site categories may offer additional insight into the types of user data obtained by similar RP services. However, in this approach, many sites could be associated with multiple categories (e.g., bloomberg.com could be labelled into both news and finance categories) making specific correlations imprecise. The diversity in multiple services offered within sites has led to the removal of Alexa's Top Sites by Category feature [4].

4.2 Data Collection

We found that site operators use different versions of their sites depending on the location from where the connection is initiated. To collect accurate representation of websites as served to local users in each country, we use a VPN service to connect to a server in the country when scanning and collecting data from sites. For each country included in our study, we first manually visit each of the Alexa Top 500 sites in the country and identify websites that support at least one of our four chosen IdPs. We then run each filtered site through OAuthScope to extract the OAuth 2.0 parameters for each of these IdPs supported by the website. As a cross-check, we noted the IdPs supported by each website during our initial manual scan and verified that OAuthScope collected all available data from each website. For any omissions, we manually obtained the IdP links and passed it to OAuthScope for extraction of the protocol parameters.

The motivation for manual verification is to conduct the privacy analysis on the complete set (i.e., dataset covering all four IdPs in 2500 sites). This ensures that websites with privacy-compromising

behaviors could not bypass detection by adding a CAPTCHA or using other means to avoid detection by automated tools including OAuthScope and other similar tools (e.g., [13] [47]).

In total, we gathered a dataset consisting of details from 815 RPs (Australia: 174; Canada: 159; Germany: 126; India: 172; US: 184). Our dataset consists of all RPs from each country that use at least one of the four IdPs; if a site appeared in the top 500 list of more than one country, it is included each time so that we could make direct comparisons across countries. As a limitation, we note that sites evolve over time (i.e., SSO options may change), and our dataset reflects a point-in-time snapshot of the top sites collected in July-September 2020.

5 API ANALYSIS OF IDENTITY PROVIDERS

User data available to third-party services through OAuth-backed APIs vary with each IdP. The diverse native services provided by IdPs lead to differences in the types of user data available within each IdP's SSO platforms. These differences make it difficult to objectively compare permissions across multiple IdPs at different granularity. To assist with this comparison, we reviewed OAuth-backed attributes relating to personal user data for each IdP and categorised them based on how the information could be used (or misused). We grouped items into five data categories: *basic* (online identity), (real world) *identity*, *personal*, *interests*, and *other sensitive*, as shown in Table 1.

Categories. Data in the *basic* category includes the attributes we consider least privacy-invasive (e.g., name, email address, profile image) that help RPs uniquely identify its users. Here, name refers to the user's full name or profile username, and in the case of Apple SSO, the user is allowed to choose a custom name to be shared with the specific RP being accessed. Data attributes that facilitate mapping users to real-world identities such as user's birthday, gender, mobile number and street address are grouped into the *identity* category. These are security-critical user data that, in the hands of adversaries, can be misused to impersonate users (e.g., obtain a new mobile SIM card). The *personal* category includes data that enable RPs to access the user's personal data including photos, videos and current location. Such data could also involve secondary individuals as part of the data being shared (e.g., a friend in the user's videos). We include data attributes (e.g., social content liked by users) that could exhibit a user's online behaviour in the *interests* category. Such data can possibly be used by online trackers and other scripts to study users. Finally, we group other user data such as email contents, contact lists and documents that could contain personal information under the *other sensitive* category.

We note that some user data can be associated with multiple categories. For example, a passport copy uploaded to cloud storage can be both *other sensitive*- and *identity*-related. In such cases, we include the attribute in the category we deemed more relevant. Our goal is to provide an overview of the types of user data accessed by RPs, and not a mutually-exclusive categorisation of the APIs. Table 1 groups relevant attributes from each provider into the five categories. We selected an initial list of attributes relevant to user data by reviewing OAuth documentation pages available on each IdP platform. Then, we refined this list to include all attributes requested by the top 500 sites in each of the five countries. In Fig. 3, we include screenshots of UI shown to SSO users of the four IdPs. We now discuss the four IdPs in the context of the data categories made available through their OAuth-backed APIs.

5.1 Google OAuth API

Google's *OAuth 2.0 Scopes* document [24] categorizes data attributes into APIs based on the service where each is normally used within Google's ecosystem. For example, the Gmail API groups attributes relevant to sending and receiving emails in a Gmail inbox. This is useful for third-party email clients (e.g., Mozilla Thunderbird), that externally manage users' emails. Although not relevant to our study, Google makes available several other attributes to RPs through OAuth. In addition to attributes found in our dataset, we included attributes related to personal user data from all Google APIs in the Scope document [24]. Google's OAuth 2.0 APIs classify a subset of their attributes as belonging to *sensitive* (different from *sensitive* category defined in our study) and *restricted* scopes. Most RP applications requesting access to any of these attributes must go through a verification process reviewed by Google [25].

Google's OAuth 2.0 platform includes two types of profile-related attributes. The `profile` [23] attribute under *basic* category shown in Table 1 includes only the user's name, email and public profile image (and is included by default in response to requests), whereas `userinfo.profile` includes all publicly available information from the user's Google profile and must be explicitly requested.

Supported flows. Google's SSO platform uses both OAuth 2.0 and OIDC specifications to support standard *flows*, including the implicit and authorization code flows. RPs specify the flow using the `response_type` parameter, as discussed in Section 2. Additionally, the value `permission` can be used for the parameter to specify use of implicit flow [23]. The platform also supports the OIDC attribute, `openid` [38], that allows the RP to obtain from Google an ID token containing fields that assert the user's identity.

5.2 Facebook OAuth API

Facebook's Graph API [16] is the platform for third-party applications to interact with a user's Facebook data. Applications implementing SSO with Facebook use the *Facebook Login* interface to identify users. This platform provides unique social user data (e.g., Facebook page likes and social posts created by the user) not available through other IdPs. We find that almost all of these attributes (as marked in Table 1) are requested for access by at least one site in our empirical study. Other permissions available in *Facebook Login* allow applications to view, create, edit and delete content on user-administered Facebook Pages. Although these permissions are available, we limit our analysis to user-data attributes (listed in Table 1) requested by RPs in the top 500 sites.

All SSO requests (including those that do not explicitly request the `public_profile` attribute) to *Facebook Login* require the user to allow the RP access to default fields (Facebook name and profile picture) [17]. RPs requesting permissions other than `public_profile` (user's name and profile picture), `email` and `pages_show_list` (list of Facebook Pages managed by the user) are required to undergo an approval process by Facebook [15].

Table 1: Comparison of data attributes in provider APIs. Default fields (applicable to Google and Facebook) are italicized. For further details explaining the entries, see Google [24], Facebook [17], Apple [7] and LinkedIn [29].

Data category	Google	Facebook	Apple	LinkedIn
Basic (online identifier)	email (address) *profile* openid	email (address) *public_profile*	email (address) name (as provided by user)	r_emailaddress name profilePicture headline
Identity (real world)	user.birthday.read user.addresses.read* user.gender.read* user.phonenumbers.read*	user_birthday user_hometown user_gender user_age_range instagram_graph_user_profile*		address birthDate phoneNumbers backgroundPicture
Personal	userinfo.profile photoslibrary* fitness* tasks*	user_location user_photos user_videos instagram_graph_user_media*		geoLocation
Interests	games* user.organization.read*	user_likes user_posts user_link		organizations positions educations projects certifications skills volunteeringInterests volunteeringExperiences
Other Sensitive	contacts drive gmail (email content) documents* spreadsheets* youtube*	user_friends		websites industryName courses testScores summary

*Data not requested (but available) by any site in our dataset.

Supported flows. *Facebook Login* supports the standard OAuth 2.0 authorization code flow (`code`) and the implicit flow (`token`). RPs can additionally include the `signed_request` parameter to obtain a user ID and the `granted_scopes` parameter for a list of permissions approved by the user. Although the OIDC specification is not supported by *Facebook Login*, authentication can be performed using *signed request* which returns a signed base64url encoded JSON object containing a user ID issued by Facebook.

5.3 Apple OAuth API

Apple introduced the *Sign in with Apple* framework in 2019 as a privacy-friendly alternative for SSO users wanting to use third-party web and mobile applications without disclosing their data. It allows RPs to authenticate SSO users and to optionally request access to the user's `name` and `email` through the scope parameter. When requested, *Sign in with Apple* allows users to either share their original email address or create an anonymous email address enabled by Apple's Private Email Relay Service. This service generates an anonymous email address unique to the user-RP pair and routes all email correspondence between the RP and user through this email, hiding the user's real email from the RP. *Sign in with Apple* also helps RPs distinguish real users from bots through a boolean-value real user indicator [7]. Applications on Apple's App Store using a third-party SSO service (e.g., Facebook, Google) are now required to offer *Sign in with Apple* as an SSO option [5].

Supported flows. *Sign in with Apple* only supports the authorization code flows and additionally, an ID token can be obtained (in a JWT object) for authentication purposes. Although the platform's documentation [7] lacks explicit mention of the OpenID standard, it closely follows OpenID conventions discussed in Section 2.3. *Sign in with Apple* does not support the implicit flow.

5.4 LinkedIn OAuth API

The *Sign In with LinkedIn* platform allows users to authenticate and authorize profile access to third-party applications. In addition to `r_emailaddress` (user's email address), *Sign In with LinkedIn* provides two scope parameters related to a user's LinkedIn profile. Each parameter groups several attributes, providing less granular access to user data compared to other IdPs. RPs can specify the `r_liteprofile` (or `r_basicprofile` in older API versions) scope to request access to the user's full name, profile picture (including image meta data) and profile headline. The `r_fullprofile` scope additionally includes all the other fields (listed in Table 1) appearing on the user's LinkedIn profile [29]. For each SSO request, *Sign In with LinkedIn* provides a user ID that allows RPs to identify users.

Supported flows. *Sign In with LinkedIn* defines two types of *consents* (analogous to OAuth flows): *member* and *application authorization*. We concern ourselves only with the former, where LinkedIn requires the use of the authorization code flow from OAuth 2.0. In contrast, *application authorization* uses OAuth 2.0's *client credentials flow* for systems requiring machine-to-machine authorization, without user involvement [26], and is outside the scope of this study. In LinkedIn documentation, we did not find explicit mention of the OpenID standard or whether RPs could request only the member ID, a unique identifier specific to the RP-user pair. However, RPs can authenticate users using the OAuth flows supported by the platform. After a successful login, the RP obtains an OAuth 2.0 access token that includes the user's LinkedIn member ID.

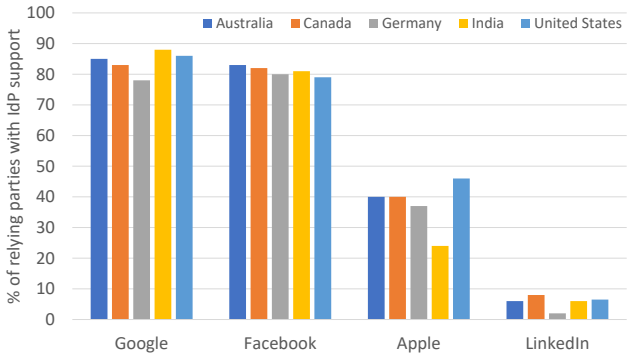

Figure 2: Percentage of RPs per country in our dataset that support each IdP.

6 EMPIRICAL RESULTS

In this section, we report the findings of our empirical study on the use of OAuth 2.0 to access user data in popular SSOs.

6.1 Distribution of Providers

We identify popular IdPs that are presented as SSO options in the top 500 sites across five countries. Our results show that Google and Facebook are the most popular SSO options in top sites across all five countries. Apple is the third most popular option, possibly due to the relatively recent introduction of Apple's SSO platform in 2019. As shown in Fig. 2, *Sign in with Apple* is less popular in India, consistent with Apple's lack of popularity in India [40]. However, recent requirements (discussed in 5.3) for apps on Apple's App Store could lead to an increase in the use of Apple SSO.

6.2 Comparing requested data across countries

We use our dataset to evaluate privacy differences (in user data released to RPs) in SSO options presented by RPs to users per country. RPs often provide different versions of their site to users in different countries (or regions) offering varying SSO features in each version. For example, Rakuten.com (a popular e-commerce site) lists three SSO options (Google, Facebook, and Apple) for US users and requests read access to the user's emails when signing in with Google. However, Rakuten.ca (for Canadian users) only provides two options (Facebook and Apple), while Rakuten.de (for German users) shows no SSO options and requires a site-specific account to shop on the site.

Comparing RPs in US and Germany. For a more specific comparison, we take each US site in our dataset and compare the user data released via SSO options presented to US users with that of users in Germany (using a VPN service), where stricter privacy laws such as the General Data Protection Regulation (GDPR) apply. In many cases, visiting a US site from Germany redirects to either a sub-domain or a different page within the site. Other sites use the same URL to serve different content to users in the two countries. We also search for the site's presence on the country-specific top-level domain and check site content against the US version to ensure they are from the same entity.

Table 2: SSO comparison of the US and Germany versions of RP sites. In all these sites, users in Germany are offered fewer or no login options compared to US users.

Relying Party	US	Germany
expedia.com	F G A -	F - - -
houzz.com	F G A -	F G - -
yelp.com	F G A -	- G A -
aol.com	F G - -	No SSO (requires site-specific login)
businessinsider.com	F G - L	No SSO (requires site-specific login)
marriott.com	F - - -	No SSO (requires site-specific login)
rakuten.com	F G A -	No SSO (requires site-specific login)
buzzfeed.com	F G A -	No login
chicagotribune.com	F G A -	No login
cnet.com	F - - -	No login
foodnetwork.com	F G A -	No login
nvidia.com	F G A -	No login
allrecipes.com	F G - -	"...Because of the General Data Protection Regulation (GDPR)...can still browse the site and view recipes, but you can't create an account or sign in"
grubhub.com	F G - -	"Grubhub food delivery is not available in your country."
people.com	F G - -	"...feature is not available in your location"
slickdeals.net	F G - -	"...does not support user accounts of EU/EEA citizens due to GDPR regulations"
usatoday.com	F G - -	Login page does not load

IdP: (**F**)acebook, (**G**)oogle, (**A**)pple and (**L**)inkedIn

Beyond site versions with top-level differences such as cookie consent prompts, we found 80 of 184 US sites presenting a different version to users in Germany. Of these, 17 sites (listed in Table 2) offered reduced functionality to users in Germany, such as fewer or no login choices. Among the 80 sites with different versions, we found zero instances where the RP's Germany site-version requested more data than the US version. While some sites present a login button that fail to load (or return an error) in Germany site-version, others present EU users with a dialog stating a lack of features that require an account due to GDPR regulations. To access RP sites as seen by users in Germany, we used a VPN service to appear to be visiting the site from Germany. We were unable to collect data on 4 sites that blocked access when attempting to visit using a VPN connection.

Country-specific differences. Table 3 provides a sorted list of the most requested attributes for each IdP per country. The majority of the RPs request one or more *basic* attributes (in blue) that help identify users (e.g., to display the user's name and profile picture on the RP site). Although relatively similar patterns emerge across countries, we do note some variation on particular attributes. For example, Google's `userinfo.profile` is requested more frequently in Germany and India, while LinkedIn's `r_basicprofile` and `r_fullprofile` are requested most frequently in India, followed by Australia, and not requested at all in Canada and Germany. While we cannot determine why these differences exist, we speculate that these variations could be a result of the IdP support offered by the top RPs in a given country (Fig. 2).

6.3 Comparing requested data across providers

For each RP in our dataset, we captured the login options presented to users along with the type of user data requested through each

Table 3: The percentage of RPs per IdP in our dataset that requests a particular scope attribute. Blue cells represent attributes from the *basic* category. Darker cells indicate a higher percentage of RPs making a given request.

IdP	Data Attribute	AUS	CAN	DEU	IND	USA
	profile*	100	100	100	100	100
G	email (address)	99	97	98	99	97
O	openid	55	54	53	64	62
O	userinfo.profile	11	11	17	17	10
G	user.birthday.read	2	1	2	1	1
L	contacts	1	1	1	2	1
E	gmail (email content)	0	0	1	0	1
	drive	0	0	0	1	0
	public_profile*	100	100	100	100	100
	email (address)	91	87	90	89	89
	user_birthday	9	12	14	12	8
F	user_friends	7	10	9	6	13
A	user_location	3	8	6	6	3
C	user_hometown	3	6	4	3	2
E	user_likes	2	4	3	3	3
B	user_gender	2	6	6	0	2
O	user_photos	1	2	2	2	3
O	user_link	1	2	2	2	0
K	user_posts	1	1	1	1	1
	user_age_range	1	3	0	1	0
	user_videos	1	0	1	1	1
APPLE	email (address)	100	100	94	100	99
	name (as given by user)	94	95	87	95	92
LINKEDIN	r_emailaddress	90	85	100	100	83
	r_liteprofile	90	85	100	80	75
	r_basicprofile	10	0	0	20	8
	r_fullprofile	10	0	0	10	8

*Attributes (scope values) included by the IdP by default

SSO option. To examine privacy differences in SSO choices offered by RPs, we search the US sites in our dataset and find 146 of 184 RPs with two or more available SSO options. For each of these RPs, we categorize the data requested with each option and filter out RPs that request the same category of attributes from each supported option (i.e., where all of an RP's SSO options request an equivalent amount of data). This results in a list of 43 RPs (shown in Table 4) that request varying categories of data across two or more SSO login choices (i.e., where an RP's SSO options each request different amounts of data).

Privacy choices on an RP site. We find that 42 of 43 RPs support at least one SSO login that requests only the *basic* data, suggesting that, in these cases, minimally privacy-invasive choices are available to users (as long as they are aware). For example, airbnb.com supports SSO logins with Facebook, Google and Apple. When a user logs in using Facebook, the site requests access to the user's hometown, location, Facebook page likes, birthday, and friends list. However, if the user logs in with Google or Apple, the request only includes *basic* attributes such as the user's name and email address. Additionally, most RPs present the most privacy-preserving choices as the last login option; for airbnb.com, Facebook is presented to users first. Table 4 highlights the issue: each RP is shown with its SSO options listed in the order presented to users. In most cases, the first option requests more data than others, suggesting the possibility of a dark pattern where SSO users are subtly nudged towards more privacy-invasive SSOs.

Table 4: US relying parties that support at least two SSO login options and that request different categories of data per option. Login options are shown in the order (earlier tends to request more data) that they are presented on the Relying Party's login page.

Relying Party	Option 1	Option 2	Option 3
aliexpress.com	F b i p - -	G b - - - -	- - - - - -
feedly.com	G b - p - -	F b - - - -	A b - - - -
hootsuite.com	F b - p n -	G b - - - -	A b - - - -
offerup.com	F b - - - s	G b - - - -	A b - - - -
poshmark.com	F b - - - s	A b - - - -	G b - - - -
quizlet.com	G b - - - -	F b i - - -	A b - - - -
slickdeals.net	F b - - - -	G b - p - -	- - - - - -
soundcloud.com	F b i - - -	G b - p - -	A b - - - -
vimeo.com	F b - - - -	G b - p - -	A b - - - -
wordpress.com	G b - p - -	A b - - - -	- - - - - -
airbnb.com	F b i p n s	G b - - - -	A b - - - -
allrecipes.com	F b - - - s	G b - - - -	- - - - - -
autotrader.com	F b - p - s	A b - - - -	- - - - - -
blizzard.com	F b - - - s	G b - - - -	A b - - - -
canva.com	G b - p - -	F b - - - -	A b - - - -
chess.com	F b - - - -	G b - p - -	A b - - - -
coursera.org	G b - - - -	F b i - - s	A b - - - -
dailymotion.com	F b i - - -	G b - - - -	- - - - - -
desmos.com	G b - p - -	A b - - - -	- - - - - -
dropbox.com	G b - - - s	A b - - - -	- - - - - -
epicgames.com	F b - - - s	G b - - - -	A b - - - -
expedia.com	A b - - - -	F b - - - -	G b - p - -
fiverr.com	F b i - n -	G b - - - -	A b - - - -
foodnetwork.com	A b - - - -	F b - p - s	G b - - - -
gamespot.com	F b - - - -	G b - p - -	- - - - - -
glassdoor.com	F b i p - -	G b - - - -	A b - - - -
goodreads.com	F b - - - s	G b - - - -	A b - - - -
groupon.com	F b i - - s	G b - - - -	- - - - - -
houzz.com	F b - - - s	G b - - - -	A b - - - -
imdb.com	F b i - - -	G b - - - -	A b - - - -
kickstarter.com	A b - - - -	F b - - - s	- - - - - -
loom.com	G b - p - -	A b - - - -	- - - - - -
meetup.com	F b - - - s	G b - - - -	A b - - - -
pinterest.com	F b i - n s	G b - - - -	- - - - - -
rakuten.com	G b - - - s	F b - - - -	A b - - - -
slideshare.net	L b i p n s	F b - - - s	- - - - - -
smartsheet.com	G b - p - -	A b - - - -	- - - - - -
theatlantic.com	F b i - - -	G b - - - -	- - - - - -
timeanddate.com	F b - - - -	G b - p - -	- - - - - -
tripadvisor.com	G b - - - -	F b i p n s	- - - - - -
trulia.com	F b - p - -	G b - - - -	- - - - - -
ultimate-guitar.com	G b - p - -	F b - - - -	A b - - - -
yelp.com	F b i - - -	G b - p - -	A b - - - -

IdP: (**F**)acebook , (**G**)oogle , (**A**)pple and (**L**)inkedIn
Data: (**b**)asic, (**i**)dentity, (**p**)ersonal, i(**n**)terests, (**s**)ensitive

Table 5 shows the individual attributes requested by RPs with each offered SSO option. This list includes the subset of all US RPs that request access to two or more non-*basic* attributes from at least one SSO option. We find that RPs request considerably more attributes with Facebook compared to other offered IdP SSOs (note the 10 columns in Table 5 for Facebook). Since Facebook SSO provides access to a variety of user data through its SSO platform (discussed in Sec. 5), RPs can request access to richer user data from the IdP. The privacy disparity in number of user data attributes released with each offered SSO suggests a need for informing privacy-aware users about these differences. Furthermore, many RPs (∗ and † in Table 5) use client-side OAuth flows that are vulnerable to access token misuse, as discussed in Sec. 2.2.

Table 5: Comparison of all data attributes requested by the subset of US RPs that request at least 2 non-*basic* permissions. This table presents a complementary view to Table 4 by highlighting the high number of attributes requested by RPs with Facebook compared to other IdPs.

Relying Party	F basic	F user_hometown	F user_location	F user_likes	F user_gender	F user_birthday	F user_friends	F user_photos	F user_video	F user_posts	G basic	G userinfo.profile	A basic	L basic	L r_fullprofile
aliexpress.com	•	•	•		•	•					•				
nba.com ∗	•			•		•	•								
tripadvisor.com ∗†	•	•	•	•			•	•			•				
airbnb.com	•	•	•	•		•	•				•		•		
dailymotion.com ∗†	•				•	•					•				
groupon.com ∗†	•	•					•				•				
pinterest.com ∗†	•			•		•	•				•				
glassdoor.com ∗	•		•			•					•		•		
imdb.com	•				•	•					•		•		
fiverr.com ∗†	•			•		•					•		•		
gofundme.com ∗	•						•	•							
yelp.com ∗†	•				•	•					•	•	•		
autotrader.com	•						•	•					•		
foodnetwork.com	•						•	•			•		•		
hootsuite.com	•							•	•	•	•		•		
soundcloud.com	•					•					•	•	•		
slideshare.net ∗	•						•							•	•

RPs using client-side OAuth flows are shown with ∗Facebook; †Google

IdP-specific differences. In situations where multiple IdPs offer access to broadly similar types of data (shown in Table 3), RPs are more likely to request this data from one IdP and not from the others. For example, 13% of US RPs in our dataset request access to user friends from Facebook SSO, but only 1% request user contacts with Google SSO. Additionally, an RP will gain access to varying amounts of data about a user depending on how much that user has shared with the IdP; an RP could receive a significantly richer profile about a SSO user who has "liked" hundreds of Facebook pages over many years compared to a user who has not "liked" any pages. Due to such differences, the actual data accessed by RPs could vary with individual users. Moreover, for the IdPs in our study, it is not possible for RPs to request subsets of data for a given attribute. Hence, we limit our analysis to data attributes as requested by RPs.

6.4 Use of OAuth 2.0 and related OIDC Flows

SSO protocol data collected in our study include information about which OAuth 2.0 flows were implemented by each RP. Table 6 summarizes the different OAuth 2.0/OpenID Connect flows used by the RPs in our dataset. Of particular concern is that a significant number of RPs use the less secure implicit flow. For example, between 29% (Germany) and 43% (US; India) of RPs use the implicit flow with Facebook SSO. Similarly, the implicit flow with Google SSO is used by between 17% (Germany) and 38% (India) RPs.

Using the implicit flow involves receiving access tokens directly into the user's browser from the IdP, suggesting that the safe storage (compared to the authorization code flow) of tokens issued to the RP is dependent on the user's security practises. Apple and LinkedIn do not support implicit flows on their SSO platforms, thus forcing all RPs to use more secure designs. Both Google and Facebook SSOs (shown in Table 6) support both the client-side and server-side flows. We find some cross-country differences, notably fewer RPs in Germany use less secure flows compared to RPs in other countries. This could be due to stricter data protection regulations.

7 PRIVACY IMPLICATIONS

In this section, we discuss user privacy implications from our evaluation of OAuth 2.0 and OpenID Connect use in popular SSO services.

7.1 Privacy choices

Our empirical results reveal that RPs often request differing amounts of user data (shown in Table 4) for each SSO option they offer. We also find that the privacy-friendly choices are typically presented last in the login screens. It is possible that RPs list popular SSO options earlier or encourage users to choose certain SSO options and request more user data to offer usability benefits (e.g., autofill forms). However this comes at the cost of privacy and, as privacy advocates, we argue that if a choice is inevitable between usability and privacy, the decision should be left to the user. To allow privacy-conscious users to make informed decisions, these differences in privacy and usability between SSO options should be made available before the user makes a login choice. Standard practice [44, §9.8] emphasizes that designs should be as simple as possible and provide users with a secure default path-of-least-resistance to complete authentication tasks. We argue that this path-of-least-resistance should also be privacy-preserving by default.

In addition to the type of data requested, privacy decisions by users may also depend on how the accessed data is used by RPs. As external observers, we have no means to assess an RP's handling or use of user data collected through its SSOs, thus we avoid speculating about possible uses beyond our knowledge. We do note that since a given RP frequently includes SSO options that request different amounts of data, it appears unlikely that all requested data is essential to provision core services since users logging in with the more privacy-friendly SSO can still use the RP's services.

IdP interface. Fig. 3 shows the design of SSO user interfaces presented to the user when they choose to login with an IdP. Information about permissions requested by the RP is presented differently by each IdP. If the user is not already logged in, some IdPs (i.e., Apple, Facebook, and LinkedIn) present their default login screens before showing the SSO screen. This means that, when the RP offers multiple SSO options, users can only view the requested permissions with a specific IdP after logging in with that IdP. To make an informed decision, users would need to click on all listed SSO options and complete authentication with each IdP in turn, to view and compare the complete list of permissions requested by the RP. This design can lead to privacy-conscious users sharing more data

Table 6: OAuth 2.0 and related OpenID Connect (OIDC) flows used by RPs per country. N = total number of sites (among top 500) offering the IdP per country (number of RPs); n = number of RPs using the given flow; % = percentage of RPs using a given flow (i.e., $\% = n/N * 100$). Hybrid flows represent multi-valued response types described in Section 2.3.

IdP	OAuth 2.0 / OIDC Flow	Response Type	Australia			Canada			Germany			India			US		
			N	n	%	N	n	%	N	n	%	N	n	%	N	n	%
Google	Authorization code	code	148	95	64	132	85	64	98	74	76	151	83	55	159	95	60
	Implicit	token	↑	4	3	↑	4	3	↑	0	-	↑	3	2	↑	2	1
		id_token		2	1		1	1		0	-		0	-		2	1
		id_token token		34	23		35	27		17	17		55	36		50	31
	Hybrid	code id_token	↑	0	-	↑	0	-	↑	0	-	↑	0	-	↑	0	-
		code token		0	-		0	-		0	-		0	-		1	1
		code id_token token		13	9		7	5		7	7		10	7		9	6
Facebook	Authorization code	code	147	102	69	134	90	67	103	73	71	144	82	57	148	84	57
	Implicit	token	↑	4	3	↑	5	4	↑	0	-	↑	2	1	↑	3	2
		token signed_request		41	28		39	29		30	29		60	42		61	41
Apple	Authorization code	code	70	34	49	64	30	47	47	30	64	42	16	38	85	35	41
	Hybrid	code id_token	↑	36	51	↑	34	53	↑	17	36	↑	26	62	↑	50	59
LinkedIn	Authorization code	code	9	9	100	11	11	100	3	3	100	10	10	100	11	11	100

than intended. While some IdPs allow users to opt-out of certain permissions (shown in Fig. 3), others require users to grant all the requested permissions. Given the differing amounts of private data requested by offered SSOs, we highlight the importance of presenting the requested permissions for all SSOs to users prior to their decision to login with a specific SSO provider.

Informing users. Usable SSO interfaces are essential in communicating SSO choices available on an RP site to users. While current IdPs theoretically obtain consent before sharing user data during SSO workflows, we argue that truly informed consent would enable interested users to adequately compare their options before making a decision and having some control over their privacy.

Android permission models involve device resources and are arguably more complex than SSO logins, but superficially they may appear similar from the user's perspective. For example, when signing in, a SSO user is presented with a list of resources similar to install-time permissions on Android and if the user is already signed in with an IdP, a dialog similar to Android's runtime permission request is shown. Prior research on Android UIs have shown that permission warnings are ineffective in informing users about the risks associated with granting resource access to applications [18]. Android permission warnings focus on resource access but lack useful information for users to understand the associated benefits and risks [35]. Even when users read permission warnings, they are unaware of risks and simply trust the marketplaces to have reviewed the hosted applications [28]. We can derive insights for SSO permission requests and, through our inspection, observe that the existing SSO UI designs similarly lack useful information for users to convey risks associated with sharing personal data.

Although specific user interface recommendations should be user tested, we suggest that users could be better informed when provided (e.g., in a second-level user interface) with a comparison of the differences between the requested data of each SSO. We do not necessarily suggest providing users with additional information, but rather making it easier for them to compare if they so choose.

7.2 OAuth 2.0 Implicit Flow

The OAuth 2.0 implicit flow was created due to past browser restrictions limiting websites to making requests only within its own domain [34]. This prevented JavaScript-based apps from using the authorization code flow since it involves making requests to the IdP domain, which is likely to be different when the RP and IdP are different entities. As discussed in Section 2.2, modern browsers support cross-origin requests, allowing the use of more secure flows.

Often privacy and security issues are closely related. Our analysis shows that many RPs still use the implicit flow that returns access tokens in the redirection URL. This is a security concern since URLs are persisted in users' browsing history and it increases the attack surface for access token leakage [22]. An access token providing broader access to user data means greater privacy risks created by design choices that are susceptible to RP's security choices. Access tokens provide access to specified resources for a limited duration. Users may not be aware of the risks associated with access token leaks, especially when the RP uses client-side flows. The potential damage to user privacy increases when the token's scope allows excessive access to sensitive user data. RPs can reduce the attack surface by requesting minimum access and using secure flows.

7.3 Offline data leaks

RPs and IdPs should clearly indicate the purpose of requested access to personal data before the user makes their choice. OAuth enables an RP to improve usability for users through customization based on user-data attributes from an IdP. If access is granted, the RP can download and persist user data for further processing. As

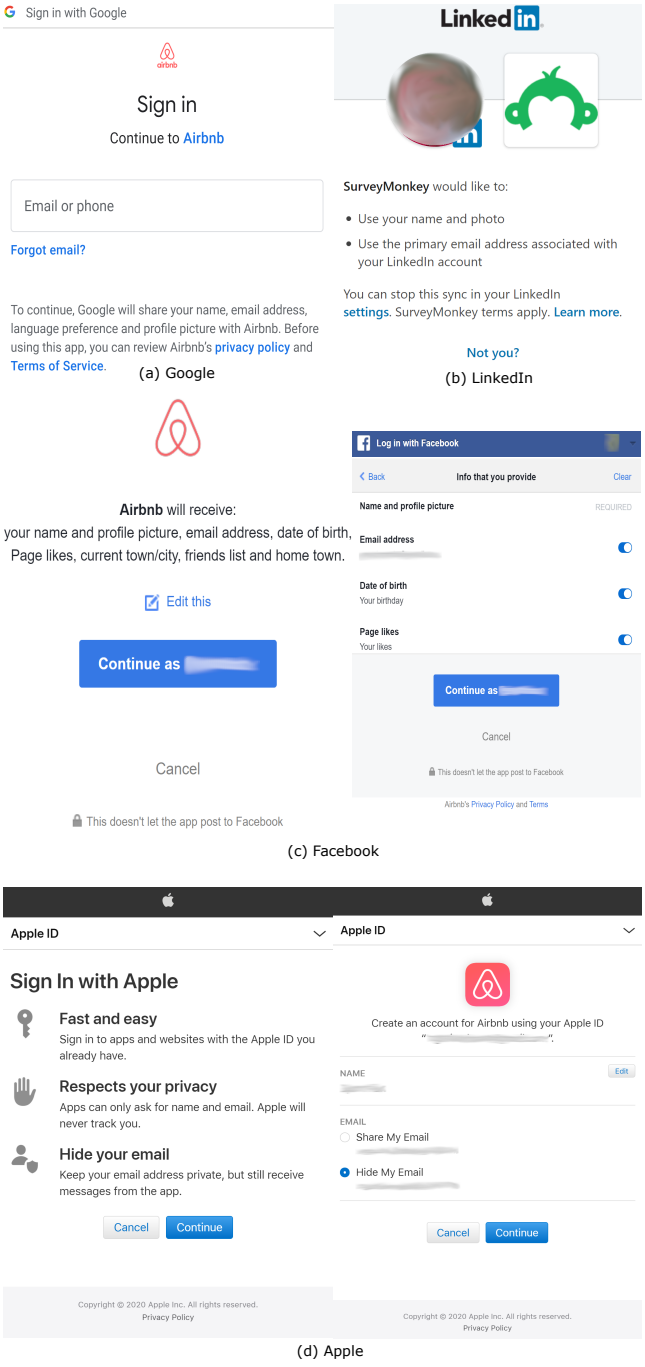

Figure 3: UI of SSO login forms on IdP sites. (a) Google and (b) LinkedIn do not allow users to alter fine-grained permissions granted to RPs. (c) Facebook allows users to selectively opt-out of non-*default* permissions requested by an RP. (d) Apple allows users to use a substitute name and anonymous email with an RP. In cases of (b), (c) and (d), the IdP presents its default login dialog before showing the SSO screens if the user is not already logged in, as discussed inline. Personal details are greyed out in these images.

mentioned in Section 5, many IdPs review RP apps that request access to sensitive user data. IdPs also provide an interface for users to revoke previously granted access to an RP, invalidating all access tokens issued to the RP. However, this does not prevent a rogue RP from misusing any user data already accessed. Since user data is processed on RP apps (not controlled by IdP), it is not possible for the user or IdP to be aware of any misuse of accessed data.

Without proper security measures, even a well-intentioned RP can be vulnerable to data breaches that increase the attack surface for users and their data. Although using OAuth ensures that the user's passwords are safe from an attack on RP, leaked access tokens can equally cause damage by allowing attackers to access user data [12]. It can be challenging for users to track attacks on RPs, and understanding the implications requires a mental model of the SSO system that many users lack [43]. Another challenge for users involves access to decommissioned accounts. RPs identify users by trusting the IdP's verification of user credentials. If a user has stopped using an RP or de-linked their IdP account with RP, it may not be possible for the user to later correspond with the RP (e.g., to demand deletion of personal data).

In Section 9, we extend discussion of privacy concerns to consider stakeholder-based challenges in improving user privacy and offer ideas for further exploration.

8 RELATED WORK

Zhou et al. [47] built SSOScan, a SSO security testing tool to automatically scan sites with Facebook SSO by simulating user interactions. We follow a similar methodology to simulate user actions leading to SSO login pages. However, our tool facilitates the collection of SSO parameters beyond a single IdP such as Facebook, enabling the comparison of SSO options available on RP sites. Additionally, instead of looking for security flaws, we examine privacy implications in RPs using any of four IdPs. Drakonakis et al. [13] built an auditing framework for evaluating login security in web apps, including sites that support SSOs with Facebook and Google. They simulate user interactions to automatically create accounts and analyse login requests to find security vulnerabilities such as data leaks to unauthorized parties. We also build on the Selenium [39] framework to automate user actions. As noted in our introduction, instead of privacy leaks to unauthorized parties, we evaluate privacy implications in sites that are explicitly granted access to the user's personal data protected by IdPs (although users might be unaware). While our tool is similar in some ways (in that, e.g., it automatically collects data from RPs) to previous research [13] [47], our empirical study differs by involving a complete dataset, as discussed in Section 4.2.

Mainka et al. [30] investigate malicious IdPs in OpenID implementations using a testing framework and identify four novel attack classes. Fett et al. [20] pursue formal analysis of the OAuth 2.0 standard and proofs of security properties for all OAuth 2.0 flow types. Chen et al. [11] evaluate the use of OAuth in mobile apps and reveal security vulnerabilities in several apps due to implementation flaws. In a 2012 field study of popular SSO systems, Wang et al. [45] analysed SSO web traffic through the browser and identified flaws in popular RPs and IdPs, allowing attackers to impersonate victims.

The Selenium-based OpenWPM platform, by Englehardt et al. [14], is designed for large-scale analysis of user tracking on the web.

Mainka et al. [31] analyse the OpenID Connect protocol and identify security flaws similar to vulnerabilities found in other SSO protocols. They implement a fully-automated evaluation tool to identify implementation flaws in OpenID Connect libraries. Bai et al. [8] provide a tool to automatically identify security vulnerabilities in implementations of web authentication protocols including OAuth-based SSO. Yang et al. [46] propose an OAuth 2.0 security testing framework and automatically evaluate four IdPs (Facebook, Sina, Renren and Tencent Weibo) and 500 top-ranked web apps in US and China. Their empirical study reveals web apps that lack adequate protection from security exploits.

Addressing challenges related to user awareness, AppCensus [1] (cf. [35]) uses dynamic analysis to reveal privacy implications of granting data access to Android apps. More recently, Apple introduced *privacy labels* [6] to highlight privacy practises to users of iOS apps. Narayanan et al. [33] discuss dark pattern designs in online services used to influence less-informed users into choices not in their best interest. Mathur et al. [32] investigate ~11K shopping sites and find 1,818 instances of dark patterns designed to increase user purchases. A 2014 user study by Robinson et al. [36] finds that users tend to misunderstand broad (as opposed to detailed) descriptions of Facebook Login permissions. Felt et al.'s [18] user studies evaluate the effectiveness of Android permissions and find majority of participants were unaware or did not look at permission warnings. Unlike mobile apps where users are given only one set of permissions, SSO users often have the choice (although hidden) to login to a given RP with a less privacy-intrusive alternative so user awareness could significantly impact decisions.

Sun et al. [43] empirically found users hesitant to adopt OpenID due to a lack of understanding and to concerns over releasing personal information. Many users held the misconception that their IdP credentials were shared with the RP. In 2012, Sun et al. [42] also evaluated OAuth 2.0 implementations by three major IdPs (Facebook, Microsoft, Google) and explored RPs supporting Facebook SSO to find several implementation decisions causing security concerns, including possible access token theft. Privacy implications discussed herein complement their work on security implications from identified vulnerabilities. Bonneau et al. [9] surveyed 35 password-replacement schemes and found that compared to other schemes, federated SSO systems offer more benefits across various usability, deployability and security properties. Alaca et al. [2] propose a framework to evaluate 14 web SSO schemes, including OAuth 2.0 and OpenID, and compare various properties including privacy benefits. They identify defining characteristics for each scheme and highlight priorities for stakeholders.

9 DISCUSSION & CONCLUDING REMARKS

OAuth-based systems provide many benefits such as flexibility and convenience to users. Services using OAuth benefit from reduced development costs related to outsourcing identity management. When an RP supports multiple SSO logins, users must commit to an SSO option (and in many cases, complete the authentication) before finding what user data will be requested by the RP. This design means that users never find out what data would be requested by other SSO options, and consequently, are not fully informed about available choices on the RP site. Our empirical results reveal privacy practises where popular RPs request vastly different amounts of user data from different IdPs, with at least one option unquestionably more privacy-intrusive than others. Moreover, we uncover a concerning trend in which privacy-friendly choices are presented to users as the last option on RP sites, similar to dark patterns found in website designs [32] [33]. SSO users are likely to make privacy decisions not in their best interest, due to the lack of information on available choices.

When granting RPs access to user data, users are not given information on the duration of the access. This lack of information, combined with an RP's ability to extend previously granted access without additional user involvement (Section 2.4), poses ongoing danger to user privacy. Further research is needed to mitigate risks related to allowing such continued access by RPs.

Addressing the privacy concerns identified by our analysis is challenging because of misaligned interests of different stakeholders. It is difficult to incentivize RPs to promote privacy-friendly choices (i.e., collect less user data) as it may not be in the company's business interest. The IdP might have its own agenda (e.g., encouraging adoption of its services) that may not align with that of privacy-conscious users or the RP. Partial solutions exist in some limited cases (e.g., signing into nytimes.com using Google SSO) where a Google dialog is presented to advise users of the data that would be shared with the RP if Google is chosen. This dialog is limited as it lacks a comparison between IdP options. Although this is only observed in limited circumstances (i.e., the user needs to be signed into Google on the same browsing window), the UI offers useful information, albeit incomplete, for a user signing in using SSO. Given the conflicting goals of stakeholders, a user-tested tool by a third-party might offer a better solution in providing relevant information for users to make informed decisions. These ideas for further exploration require separate user studies to examine usability and the effect on privacy decisions made by SSO users.

Another factor complicating privacy improvements is that there is no one-choice-fits-all solution because of variations in privacy preferences across users. For example, some users may prefer to share personal data (e.g., in exchange for using the product at no cost) while others may value privacy above extra functionality. This means that it is most likely that input is needed from users to specify preferences, compared to security solutions where the secure option can be preconfigured (no choice required from user). This is a classic tussle between stakeholders with diverging (in case of a privacy-conscious user and a privacy-invading RP, polar opposite) goals–one which we do not aim to resolve in this paper.

To the best of our knowledge, we offer the first in-depth analysis of OAuth-based SSO with a primary focus on user privacy as opposed to security. We argue that the first step in addressing the concerns identified by our analysis is awareness of the issues to encourage the community towards productive explorations about how to effectively support user privacy and about exposing dark patterns in this space. We hope that greater awareness by technically-savvy users and privacy enthusiasts, of the privacy implications identified through our work (Section 7), may result in further attention to privacy violations, further community-based monitoring, and a more privacy-friendly OAuth-based SSO ecosystem.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments. The second and third authors acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Canada Research Chairs & Discovery Grant programs.

REFERENCES

[1] Appcensus. https://www.appcensus.io/, 2020.
[2] F. Alaca and P. C. van Oorschot. Comparative Analysis and Framework Evaluating Web Single Sign-on Systems. *ACM Computing Surveys*, 53(5), 2020.
[3] Alexa. The top 500 sites on the web. https://www.alexa.com/topsites, 2021.
[4] Alexa. Top Sites by Category has been retired. https://support.alexa.com/hc/en-us/articles/360051913314, 2021.
[5] Apple. New Guidelines for Sign in with Apple. https://developer.apple.com/news/?id=09122019b, 2019.
[6] Apple. App privacy labels now live on the App Store. https://developer.apple.com/news/?id=3wann9gh, 2020.
[7] Apple. Sign in with Apple. https://developer.apple.com/documentation/sign_in_with_apple, 2021.
[8] G. Bai, J. Lei, G. Meng, S. S. Venkatraman, P. Saxena, J. Sun, Y. Liu, and J. S. Dong. AuthScan: Automatic Extraction of Web Authentication Protocols from Implementations. In *NDSS*, 2013.
[9] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *IEEE Symp. Security and Privacy*, 2012.
[10] B. Campbell, J. Bradley, N. Sakimura, and T. Lodderstedt. RFC 8705: OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens. https://datatracker.ietf.org/doc/html/rfc8705, 2020.
[11] E. Y. Chen, Y. Pei, S. Chen, Y. Tian, R. Kotcher, and P. Tague. OAuth Demystified for Mobile Application Developers. In *ACM CCS*, 2014.
[12] C. Cimpanu. Hackers stole GitHub and GitLab OAuth tokens from Git analytics form Waydev. https://www.zdnet.com/article/hackers-stole-github-and-gitlab-oauth-tokens-from-git-analytics-firm-waydev/, 2020.
[13] K. Drakonakis, S. Ioannidis, and J. Polakis. The Cookie Hunter: Automated Black-box Auditing for Web Authentication and Authorization Flaws. In *ACM CCS*, 2020.
[14] S. Englehardt and A. Narayanan. Online Tracking: A 1-million-site Measurement and Analysis. In *ACM CCS*, 2016.
[15] Facebook. App Review. https://developers.facebook.com/docs/app-review, 2021.
[16] Facebook. Graph API. https://developers.facebook.com/docs/graph-api/, 2021.
[17] Facebook. Permissions Reference. https://developers.facebook.com/docs/permissions/reference/, 2021.
[18] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android Permissions: User Attention, Comprehension, and Behavior. In *SOUPS*, 2012.
[19] D. Fett, J. Bradley, B. Campbell, T. Lodderstedt, and M. Jones. OAuth 2.0 Demonstration of Proof-of-Possession at the Application Layer. https://datatracker.ietf.org/doc/html/draft-fett-oauth-dpop-00, 2019.
[20] D. Fett, R. Küsters, and G. Schmitz. A Comprehensive Formal Security Analysis of OAuth 2.0. In *ACM CCS*, 2016.
[21] M. Ghasemisharif, A. Ramesh, S. Checkoway, C. Kanich, and J. Polakis. O Single Sign-Off, Where Art Thou? An Empirical Analysis of Single Sign-On Account Hijacking and Session Management on the Web. In *USENIX Security*, 2018.
[22] R. Gilbert. Information Exposure Through Query Strings in URL. https://owasp.org/www-community/vulnerabilities/Information_exposure_through_query_strings_in_url, 2021.
[23] Google. Google API for Authentication. https://developers.google.com/identity/sign-in/web/reference, 2021.
[24] Google. OAuth 2.0 Scopes for Google APIs. https://developers.google.com/identity/protocols/oauth2/scopes, 2021.
[25] Google. OAuth API verification FAQs. https://support.google.com/cloud/answer/9110914, 2021.
[26] D. Hardt. RFC 6749: The OAuth 2.0 Authorization Framework. https://tools.ietf.org/html/rfc6749, 2012.
[27] M. B. Jones, J. Bradley, and N. Sakimura. RFC 7519: JSON Web Token (JWT). https://tools.ietf.org/html/rfc7519, 2015.
[28] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. Sadeh, and D. Wetherall. A Conundrum of Permissions: Installing Applications on an Android Smartphone. In *Financial Cryptography and Data Security*. Springer, 2012.
[29] LinkedIn. Sign In with LinkedIn. https://docs.microsoft.com/en-us/linkedin/consumer/integrations/self-serve/sign-in-with-linkedin, 2018.
[30] C. Mainka, V. Mladenov, and J. Schwenk. Do Not Trust Me: Using Malicious IdPs for Analyzing and Attacking Single Sign-on. In *IEEE EuroS&P*, 2016.
[31] C. Mainka, V. Mladenov, J. Schwenk, and T. Wich. SoK: Single Sign-On Security — An Evaluation of OpenID Connect. In *IEEE EuroS&P*, 2017.
[32] A. Mathur, G. Acar, M. J. Friedman, E. Lucherini, J. Mayer, M. Chetty, and A. Narayanan. Dark Patterns at Scale: Findings from a Crawl of 11K Shopping Websites. *ACM Human-Computer Interaction*, 3(CSCW), 2019.
[33] A. Narayanan, A. Mathur, M. Chetty, and M. Kshirsagar. Dark Patterns: Past, Present, and Future. *ACM Queue*, 18(2), 2020.
[34] A. Parecki. Is the OAuth 2.0 Implicit Flow Dead? https://developer.okta.com/blog/2019/05/01/is-the-oauth-implicit-flow-dead, 2019.
[35] J. Reardon, Á. Feal, P. Wijesekera, A. E. B. On, N. Vallina-Rodriguez, and S. Egelman. 50 Ways to Leak Your Data: An Exploration of Apps' Circumvention of the Android Permissions System. In *USENIX Security*, 2019.
[36] N. Robinson and J. Bonneau. Cognitive Disconnect: Understanding Facebook Connect Login Permissions. In *ACM COSN*, 2014.
[37] N. Sakimura, J. Bradley, and N. Agarwal. RFC 7636: Proof Key for Code Exchange by OAuth Public Clients. https://tools.ietf.org/html/rfc7636, 2015.
[38] N. Sakimura, J. Bradley, M. B. Jones, B. de Medeiros, and C. Mortimore. OpenID Connect Core 1.0. https://openid.net/specs/openid-connect-core-1_0.html, 2014.
[39] Selenium. Selenium WebDriver. https://www.selenium.dev/documentation/en/webdriver/, 2021.
[40] M. Singh. Why Apple sells just 2.5% of India's smartphones. https://www.cnbc.com/2018/01/29/why-apple-sells-just-2-point-5-percent-of-indias-smartphones.html, 2019.
[41] E. Stobert and R. Biddle. The Password Life Cycle: User Behaviour in Managing Passwords. In *SOUPS*, 2014.
[42] S.-T. Sun and K. Beznosov. The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems. In *ACM CCS*, 2012.
[43] S.-T. Sun, E. Pospisil, I. Muslukhov, N. Dindar, K. Hawkey, and K. Beznosov. What Makes Users Refuse Web Single Sign-On? An Empirical Investigation of OpenID. In *SOUPS*, 2011.
[44] P. C. van Oorschot. *Computer Security and the Internet: Tools and Jewels*. Springer Nature, 2020.
[45] R. Wang, S. Chen, and X. Wang. Signing Me onto Your Accounts through Facebook and Google: A Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services. In *IEEE Symp. Security and Privacy*, pages 365–379, 2012.
[46] R. Yang, G. Li, W. C. Lau, K. Zhang, and P. Hu. Model-based Security Testing: An Empirical Study on OAuth 2.0 Implementations. In *AsiaCCS*, 2016.
[47] Y. Zhou and D. Evans. SSOScan: Automated Testing of Web Applications for Single Sign-On Vulnerabilities. In *USENIX Security*, 2014.

A OAUTH 2.0 AUTHORIZATION REQUEST

We provide an example authorization request discussed in Section 3. RPs specify OAuth 2.0 parameters in authorization requests to IdPs.

```
HTTP GET /authorizationEndpoint?
response_type=code
&scope=email%20profile
&redirect_uri=https%3A%2F%2Fclient%2Ecom%2Fcb
&client_id=lp4qazfnh1
&state=hnz3krb2mn
```

A brief description is included for each parameter in the request [26]:

- `authorizationEndpoint:` endpoint URI used by the RP for sending authorization requests to the IdP.
- `response_type:` specifies the OAuth flow type the RP intends to use with IdP.
- `scope:` a list of resources requested for access by the RP.
- `redirect_uri:` user is redirected to this endpoint after completing interactions with the IdP. For security reasons, this value must match the endpoint registered with the IdP during RP's app registration.
- `client_id:` a unique string issued to RP during registration.
- `state:` a unique (non-guessable) string generated by RP and included in the authorization request. The IdP returns the value when redirecting the user back to RP. To mitigate cross-side request forgery (CSRF) attacks, the RP application must ensure that the returned value is equal to value included in the initial request.

B MITIGATING OAUTH 2.0 ACCESS TOKEN THEFT

Bearer tokens (e.g., typical access tokens in OAuth 2.0) are susceptible to token theft as they can be used by any party in temporary possession of the token. Protection against such attacks include using mutual Transport Layer Security (mTLS) where the tokens are bound to the client's X.509 certificate (self-signed or signed by a trusted CA certificate), restricting their use to the client in possession of the certificate's private key [10]. An alternate mechanism to protect against token thefts is DPoP, or Demonstration of Proof of Possession which utilizes public/private key pairs to bind access and refresh tokens to the key pair. The client needs to prove possession of the private key in order to use an access token bound to the corresponding public key [19].

C OAUTH 2.0 IMPLICIT FLOW

As an extension to background provided in Section 2, Fig 4 lists the process for the OAuth 2.0 implicit flow. Returning access tokens in the redirection URIs is a security concern as the tokens are typically retained in the user's browsing history, increasing the attack surface for unauthorized access token use outside the RP application.

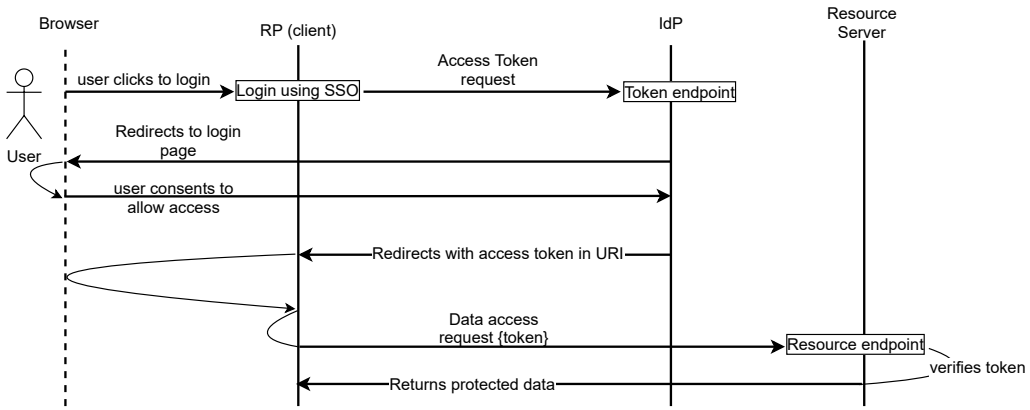

Figure 4: Procedure for the OAuth 2.0 Implicit flow (derived from [26]).

D OAUTHSCOPE

Fig. 5 is a screenshot of OAuthScope described in Section 3 and lists identified OAuth 2.0 parameters for each RP. This UI is used for analysis of data collected by OAuthScope.

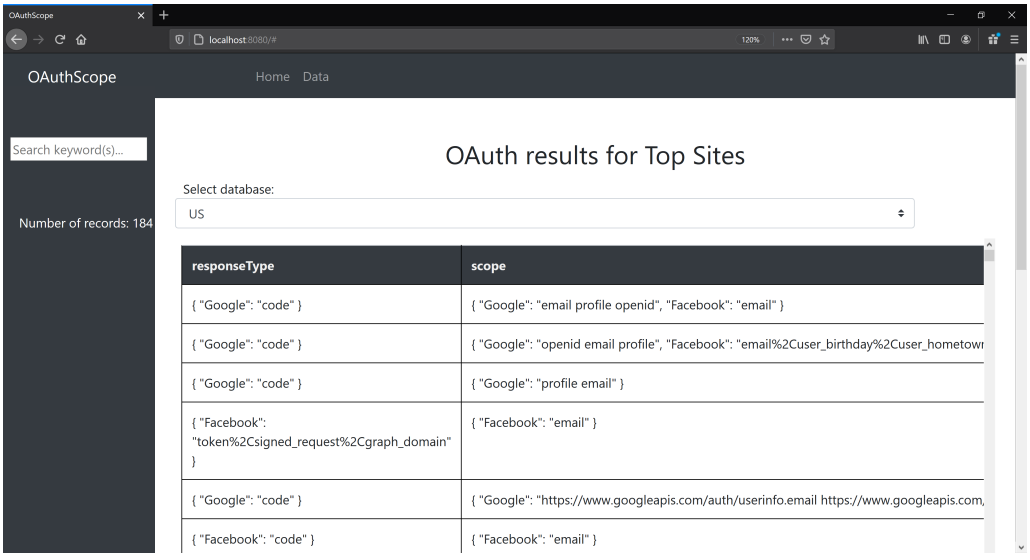

Figure 5: Screenshot of OAuthScope tool listing OAuth 2.0 parameters included in authorization requests from top US RPs.