# USER PRIVACY IN OAUTH-BASED SINGLE SIGN-ON SYSTEMS

by

Srivathsan Morkonda Gnanasekaran

A thesis submitted to
the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

Computer Science

at

CARLETON UNIVERSITY

Ottawa, Ontario
January, 2024

## Abstract

Web Single Sign-On (SSO) login is a popular alternative to password-based login in current authentication systems. SSO services enable users to use accounts registered with *identity providers* (IdPs) such as Google and Facebook to login on multiple *relying party* (RP) websites. Common web SSO deployments are based on the OAuth 2.0 authorization standard which enables RPs to both authenticate users and access a subset of a user's personal information from an IdP. This thesis pursues three goals related to user privacy in OAuth-based web SSO implementations. First, we build *OAuthScope*, a tool that extracts OAuth protocol data from RP sites. We use it to conduct an empirical investigation of privacy implications for users of OAuth implementations in RP websites most visited by users across five countries. We categorize user data made available by four IdPs (Google, Facebook, Apple, and LinkedIn) and evaluate the types of user data accessed by RPs through these IdPs. Our results reveal considerable variations in the categories and amounts of user data accessed by RPs, including differences across site versions in different countries. Second, to improve the transparency of user data accessed by RPs, we design and implement *SSOPrivateEye (SPEye)*, a browser extension tool to inform users about the privacy consequences of choosing SSO login options. SPEye extracts information about permission requests made by RPs to enable users to compare SSO options before making a login choice. Third, we conduct a user study to identify factors that influence participants when choosing from SSO and non-SSO login options. We compare login decisions made by participants before and after viewing comparative information on the user data accessed by RPs through different SSO choices. We find that usability preferences and inertia influence a majority of login decisions when presented with a list of SSO and non-SSO choices, while privacy-related reasons were most common after participants viewed the user data requested through each SSO choice. Through these three goals, we highlight and tackle SSO privacy issues affecting users, and provide insights on further improving privacy in OAuth systems.

# Acknowledgements

I would like to express my sincere gratitude to my supervisors, Professor Sonia Chiasson and Professor Paul Van Oorschot, for their continuous guidance, support, and patience during the last four years. Their invaluable mentorship and keen attention to detail were instrumental in guiding the contributions in this thesis.

I would like to thank my colleagues at the Carleton Computer Security Labs (CCSL and CISL) and the Carleton's Human Oriented Research in Usable Security (CHORUS) lab for the stimulating discussions and feedback, which also helped make my PhD journey lively and memorable. I would also like to thank Professor David Barrera for general advice on research and career.

I also thank the members of my PhD committee, Professor Carlisle Adams (University of Ottawa), Professor Hala Assal (Carleton University), Professor Hassan Khan (University of Guelph), and Professor Anil Somayaji (Carleton University), for their insightful comments.

Finally, I would like to thank my family and friends for all their support and motivation throughout my studies. To my parents, Gnanasekaran and Indumathi, and my sister Srinimisha, thanks for encouraging me to pursue a PhD. To my wife Nivethini, thank you for always being there and keeping me motivated throughout this journey. This thesis would not have been possible without their love and support.

# Table of Contents

# List of Tables

# List of Figures

# List of Acronyms

**API** Application Programming Interface

**CA** Certification Authority

**CCPA** California Consumer Privacy Act

**CORS** Cross-Origin Resource Sharing

**CSRF** Cross-Site Request Forgery

**CSS** Cascading Style Sheets

**DOM** Document Object Model

**EU** European Union

**GDPR** General Data Protection Regulation

**HTML** HyperText Markup Language

**HTTP** Hypertext Transfer Protocol

**IdP** Identity Provider

**IETF** Internet Engineering Task Force

**IoT** Internet of Things

**JSON** JavaScript Object Notation

**OIDC** OpenID Connect

**PII** Personally Identifiable Information

**RFC** Request for Comments

**RP** Relying Party

**SDK** Software Development Kit

**SIM** Subscriber Identity Module

**SOAP** Simple Object Access Protocol

**SSO** Single Sign-On

**TEE** Trusted Execution Environment

**TLS** Transport Layer Security

**UI** User Interface

**URI** Uniform Resource Indicator

**URL** Uniform Resource Locator

**VPN** Virtual Private Network

# Chapter 1

# Introduction

Managing large sets of credentials is a difficult task for many users and can result in insecure login practices such as password reuse and choice of weak (easy to guess) passwords [99]. Many password-replacement schemes have been proposed to improve security and convenience. Among such proposals, web single sign-on (SSO) protocols have emerged as dominant in enterprises and consumer web services alongside traditional password-based login methods. Web SSO improves login systems by offering benefits across various usability, deployability, and security properties [13]. Many thousands of websites offer SSO services such as "Sign in with Google" and "Sign in with Facebook" to encourage users to create accounts on their sites and to make it easier for users to login. Users of these sites rely on the security and privacy of SSO services to protect their login accounts and personal data stored in these accounts.

Web SSO schemes aim to improve login security by reducing the number of passwords users need to choose and remember on individual websites. These schemes leverage user accounts registered with a single *identity provider* (IdP) to enable users to log into one or more third-party web services.[1] A user on a third-party site (e.g., `Airbnb.com`) selecting SSO login with an IdP (e.g., Facebook) is redirected to the IdP's login page where they are prompted to login using their IdP account credentials, and the IdP verifies and conveys the user's identity to the site. This removes the need for users to create new accounts using fresh passwords and they can instead use a single SSO account across different websites that support SSO login. As the third-party site relies on the IdP for authenticating users, it is referred to as a *relying party* (RP) site. SSO login is also convenient because once users login to their IdP accounts, they can login to multiple RP sites without having to re-enter their IdP credentials for each RP site login. The convenience and quick login on different sites

---

[1]We use "third parties" to refer to data shared with entities other than users and IdPs.

1
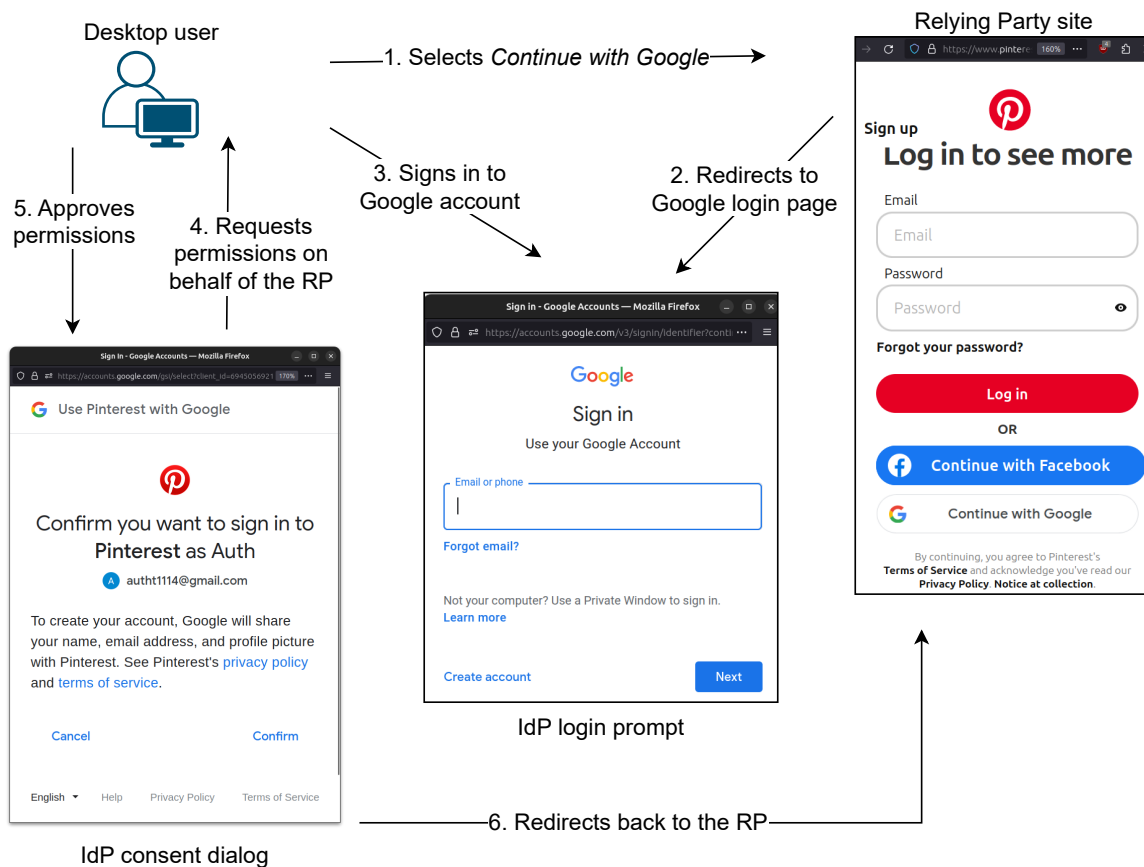
Figure 1.1: Example of an SSO login workflow involving a desktop user signing into a Relying Party (RP) (e.g., `Pinterest.ca`) through an IdP (e.g., Google Sign-In). The IdP consent dialog displays the user data permissions granted to the RP site.

may encourage users to use SSO login options.

Web SSO is also desirable to SSO service providers including the RPs and the IdPs for their business interests. It reduces implementation costs for the RPs as they outsource login-related tasks (e.g., key management, credential verification) to the IdPs, and reduces potential support costs associated with passwords. The IdPs benefit from an increased presence of their SSO services on different RP sites as it potentially increases the use of IdP accounts and other services provided by the IdPs.

Some web SSO schemes also enable RPs to access selected personal information of users through their IdP accounts. OAuth 2.0 [53], a standard web authorization protocol, is widely used in web SSO login systems and supports delegated access of user data to third-party RP sites (Section 2.2 gives details). Figure 1.1 shows an

example of a typical SSO login workflow in OAuth-based systems. When a user on an RP site (e.g., `pinterest.ca`) selects an SSO login option (such as "Continue with Google"), the site redirects to a Google (IdP) login page where the user is prompted to login to their Google account. As part of the SSO login transaction, the RP can request access to the user's personal information stored with the IdP. The IdP displays a consent dialog with the requested permissions and asks the user to grant access to the RP. If the user approves, the IdP releases access to the selected user data to the RP and redirects back to the RP site, completing the SSO login workflow.

Major IdPs (including Facebook [33] and Google [51]) expose web APIs to grant RPs controllable access to protected user personal data stored on their platforms. With the user's consent (as illustrated in Figure 1.1), the IdPs allow RPs access to one or more user data APIs through SSO login options. In some RP sites, the released user data includes potentially sensitive personal information such as email messages, personal calendars, and photos and documents in personal cloud storage. Personal information accessed through these APIs are designed to enable RPs to offer extra functionality (such as a third-party scheduling app with access to the user's personal calendar). Though the IdP APIs might be intended for benign use cases, such user data APIs could be misused by RPs and lead to potential privacy issues.

This thesis explores privacy in real-world web SSO deployments to develop solutions that help address common privacy issues in web SSO systems. To achieve this: first, we conducted empirical studies to understand how web SSO systems are implemented, and the SSO privacy practices used by RP web services; then, we designed and implemented a browser extension to help inform users about privacy consequences of choosing SSO login options; and we performed a user study to understand how users make login decisions, and to evaluate the impact of displaying privacy-related information on login decisions.

## 1.1 Scope

User privacy in web SSO systems focuses on the disclosure of personal information of users through SSO services. Web SSO login systems involve different RP sites, IdP services, browser applications, and SSO protocols. In this thesis, we focus on privacy

consequences in web SSO services due to "explicitly" granted access to personal information through SSO login accounts. Vulnerabilities in these components can lead to privacy leaks of user information to unauthorized parties; however, exposure of personal information due to unintended security weaknesses is out of scope.

Web SSO privacy consequences depend on the type of information made available by different IdP services and the permissions requested by individual RP sites. The practical privacy exposure varies across individual users depending on the amount of personal information stored in their IdP accounts. However, this thesis focuses on the first aspect: the privacy consequences related to the security and privacy practices of RPs and IdPs.

We also note that the primary focus of this thesis is on OAuth-based web SSO schemes widely used in consumer RP websites, where SSO services typically involve delegating access to personal information of users to third parties. Thus, our scope does not include other SSO schemes (such as SAML [77] or Kerberos [75]) which are popular in enterprise environments and typically do not involve authorization of access to personal information; similarly, we exclude password managers which can be viewed as a form of SSO login [1].

## 1.2   Motivation

SSO login options have become increasingly common in many websites over the past decade [58]. With the ubiquitous OAuth 2.0 [53] authorization standard, web SSO login systems have extended from primarily authenticating SSO users to also delegating user data access to third-party RP sites. This disclosure of user personal information to third parties raises privacy concerns about data sharing in web SSO systems [10], and motivates the work presented in this thesis.

Previous privacy research (e.g., [24,45,56,84]) involving OAuth-based SSO focuses on security issues in vulnerable SSO implementations, including privacy leaks of user personal information to unauthorized third parties. There has been little research on privacy consequences for users of OAuth-based SSO that arise when RPs gain access to user data stored in IdP accounts through methods allowed by the protocols. This thesis investigates the research gap of understanding privacy consequences in web

SSO systems, and explores ways to inform users about such consequences of choosing SSO login options.

Existing user studies [10, 11, 26, 87, 101] related to SSO users' privacy and security perceptions have so far focused on SSO mental models and privacy preferences in individual IdP platforms. However, the growing number of login options in RP websites [70] motivate us to examine how users make login decisions within RP sites when multiple options are available. This thesis also explores privacy attitudes by examining the factors that influence users' login preferences in RP sites that offer multiple SSO and non-SSO login options.

## 1.3 Research Goals

The main goal of this research is to understand how user privacy is impacted in web SSO systems, and to develop solutions to address relevant privacy issues. The objectives of this thesis can be summarized as follows.

G1: Analyzing what types of user data RP websites request in OAuth-based web SSO systems.

G2: Developing tools to inform users about the privacy issues in web SSO systems.

G3: Understanding how users make login decisions in websites that provide more than one SSO login option, and studying the effect of displaying privacy-related login information on RP websites prior to selecting an option.

## 1.4 Contributions and Thesis Outline

The main contributions of this thesis are as follows.

- *Literature review of research on the privacy of web SSO systems (Chapter 2).* We studied existing research literature on OAuth-based web SSO protocols and real-world OAuth deployments. We reviewed the design of the OAuth 2.0 protocol to also identify inherent privacy risks (described in Section 2.2.4). Chapter 2 also provides background on web SSO schemes.

- *The OAuthScope tool (Section 3.2).* We built the *OAuthScope* tool to automatically extract information on the use of OAuth from sites supporting SSO logins; this aids the empirical analysis in the following item. This tool enables the collection and comparison of SSO data from four major identity providers (Google, Facebook, Apple, and LinkedIn) on the web.

- *Empirical analysis of privacy in web SSO systems (Section 3.3).* We conducted the first in-depth empirical analysis of privacy consequences in OAuth-based web SSO systems by studying OAuth user data requests in the Alexa Top 500 sites per country for five countries (Australia, Canada, Germany, India, and the United States). The analysis revealed considerable variation in how sites implement the different SSO login options, including apparent differences across countries with different privacy laws.

  - An initial step of this analysis involved designing a cross-provider categorization of user data APIs (described in Section 3.1) made available by four major identity providers (Google, Facebook, Apple, and LinkedIn). We note that our IdP API categorization has been used by Dimova et al. [23] in a recent large-scale privacy study of web SSO systems.

- *Empirical analysis of client-side SSO software patterns (Section 3.4).* We conducted an empirical analysis of OAuth-based client-side implementations in the Tranco top 500 sites. We identified four software patterns used by popular relying party sites to implement web SSO login options with any of three major identity providers (Facebook, Google, or Apple). For each of the four patterns, we provide strategies allowing a tool to automatically extract RP site data useful for privacy and security analysis.

- *The SSOPrivateEye browser extension (Chapter 4).* We built the *SSOPrivateEye (SPEye)* tool, a Chrome browser extension displaying real-time privacy information about web SSO permission requests. This information is presented in a standard format, just-in-time (before the user decides to log in to an RP site). We also show the usefulness of this tool by building it to recognize SSO

login options with three major IdPs: Facebook, Google, and Apple. We designed the tool such that further IdPs (that use standard SSO protocols) can be added without major structural changes.

- The approach (described in Section 4.2.1) takes into account privacy differences in permissions prompted to users in locations where different privacy laws apply.

- *User study of web SSO login preferences (Chapter 5).* We conducted a user study to understand factors that influence participants' login decisions in websites with multiple SSO and non-SSO login options. We found that usability preferences and inertia (i.e., habituation) were among the dominant factors influencing login decisions.

  - We also examined the impact of displaying SPEye UIs that show permission differences across login options on participants' login decisions. After participants viewed permission-related information, many prioritized privacy over other factors, changing their login decisions to more privacy-friendly alternatives. These results are discussed in Sections 5.3.2 and 5.4.3.

## 1.5   Related Publications

Chapter 3 includes work that has been published at the following peer-reviewed conference [70]:

- **Srivathsan G. Morkonda**, Sonia Chiasson, and Paul C. van Oorschot. 2021. Empirical Analysis and Privacy Implications in OAuth-based Single Sign-On Systems, *20th Workshop on Privacy in the Electronic Society (WPES)*, ACM

Chapter 5 consists of materials available in the following journal publication [72]:

- **Srivathsan G. Morkonda**, Sonia Chiasson, and Paul C. van Oorschot. Influences of Displaying Permission-related Information on Web Single Sign-On Login Decisions. *Computers & Security* (to appear; Volume 139, April 2024).

Portions of Chapter 3 and Chapter 4 include materials from work under submission for journal publication [71]. A comparable version is filed in the following technical report:

- **Srivathsan G. Morkonda**, Sonia Chiasson, and Paul C. van Oorschot. "Sign in with ... *Privacy*": Timely Disclosure of Privacy Differences among Web SSO Login Options, Aug 2023. Technical report available at: `https://arxiv.org/abs/2209.04490`

# Chapter 2

# Background and Related work

This chapter presents background on common SSO schemes, and relevant research on the privacy, security, and usability of OAuth-based SSO systems. We summarize the main OAuth 2.0 concepts which are used in the remaining chapters of this thesis, and highlight inherent privacy risks in the OAuth 2.0 protocol.

## 2.1 Single Sign-On Overview

An SSO system consists of one or more entities that use a standard SSO scheme to enable users authenticate to one or more relying party (RP) applications using an account registered with an identity provider (IdP). A group of one or more affiliated RP and IdP entities is known as an *SSO federation*. The trust agreement between the members of a federation to enable SSO login (from an IdP domain to an RP application) is referred to as *federated identity management*. In an SSO federation system, the IdPs hold users' identity information including account credentials, and the RPs trust the IdP verification of user credentials to identify users on their applications. Many IdPs also store a collection of protected data resources linked with user accounts, useful for enabling SSO users to delegate access of their data to RPs.

SSO schemes can be described based on their use cases within different application domains. A typical closed enterprise network uses SSO login to enable employees to login across various applications within its domain through a centralized authentication server (e.g., as in the Kerberos [75] authentication protocol). The central server stores the users' credentials and might also store additional information about users to facilitate authorization of access to different computing resources.

Beyond closed authentication domains, web SSO authentication is also used in web enterprise and in web consumer applications. An enterprise SSO system on the web consists of an identity federation where members support SSO authentication of users

from one member domain to other domains using a member IdP. This is convenient and cost-effective as it enables cross-domain access to applications without requiring re-authentication of users. In consumer applications, SSO systems typically involve independent (i.e., not necessarily affiliated) third-party RPs that enable users to login using supported IdP accounts and grant RPs access to their personal information. These applications differ from enterprise SSO as they typically involve personal user accounts and personal information released to third parties.

Next, we describe key web SSO protocols, including the OAuth 2.0 authorization framework and the OAuth-based OpenID Connect (OIDC) standard used for SSO authentication. We explain the basic concepts of OAuth 2.0, and highlight key differences in other SSO schemes. We focus more on OAuth-based protocols as they are more relevant to our focus on privacy in web SSO systems.

## 2.2   OAuth 2.0

OAuth 2.0 [53] is a web resource authorization protocol popular in client-server deployments worldwide for granting third-party RP websites access to protected data resources without revealing the user's credentials. For example, an RP site prompts a user to login using Google (i.e., the IdP) and grant access to specified user data by relying on their Google account rather than creating new credentials on that website. The site 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 also log in to other websites that support the Google SSO login option. User data is referred to as *protected resources* and the user is identified as the *resource owner* (RO) in the OAuth 2.0 specification. Beyond read access to user data, OAuth 2.0 can also be used to delegate *write* access, allowing RPs to make changes to the IdP account on behalf of the user. For example, OAuth 2.0 is used for authorizing third-party email clients to retrieve and send email messages from a user's email (e.g., Microsoft Outlook) IdP account inbox.

The OAuth 2.0 protocol is composed of several "grant types" that define how credentials are granted to RPs. The procedures for obtaining access to protected resources is defined by *OAuth flows* and each flow is designed to serve different use cases
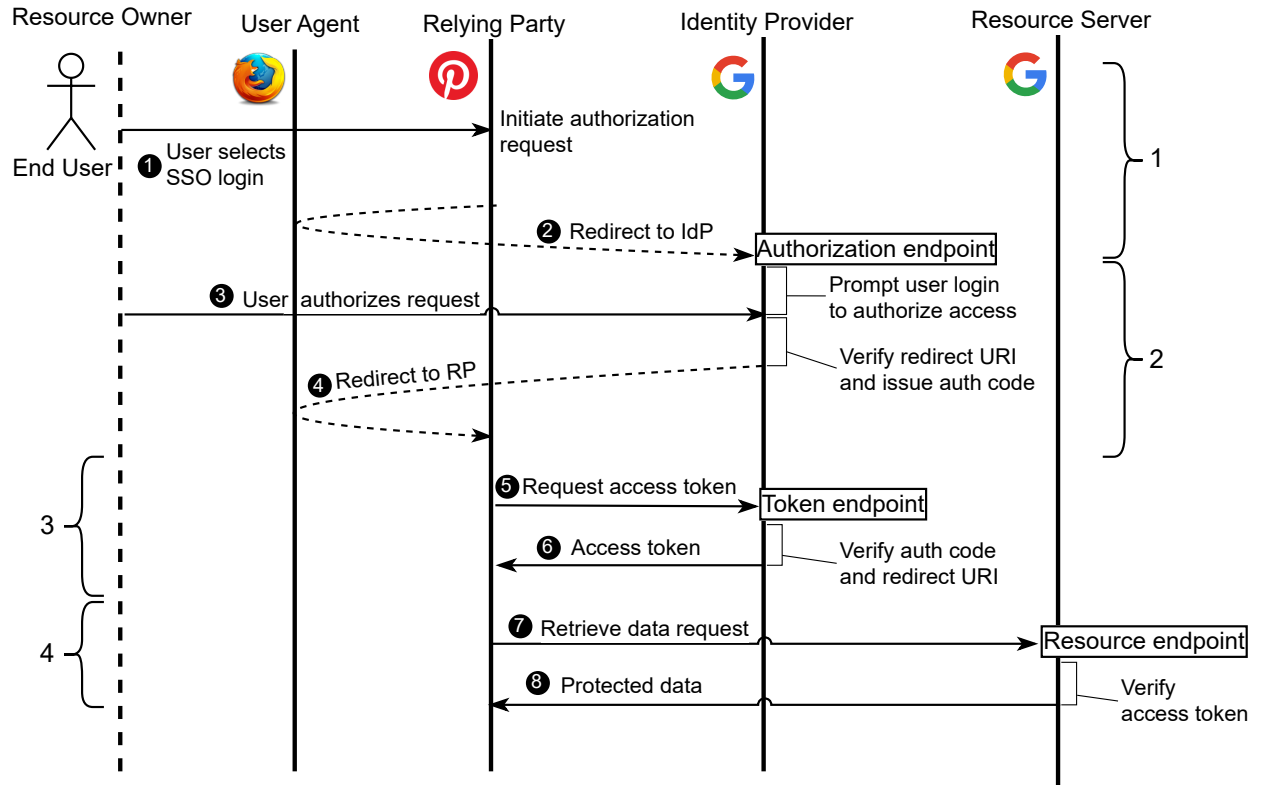
Figure 2.1: Procedure for OAuth 2.0 Authorization Code flow, with example entities.

(and provide different levels of security). Since this thesis primarily involves OAuth 2.0 use in web applications, we describe the steps involved in two flows commonly used in web apps.

### 2.2.1   Authorization Code (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 login 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 four primary steps (labelled in Fig. 2.1) involved in the *authorization code flow* [53].

1. **RP request to IdP**: In Step ❶, the resource owner selects an SSO login option at an RP site which initiates the SSO login workflow by sending an

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

Listing 1: Example of an OAuth 2.0 authorization request with query parameters.

*authorization request* (Listing 1 gives an example). The RP site constructs the request by including several protocol parameters (as query components in the request URI) that inform the IdP about the RP's request to access user data. To specify the authorization code flow type, the RP must set the value of `response_type` to `code`. The `scope` parameter specifies the data resources (i.e., IdP APIs) the RP wants to access. The allowed values for this parameter are specified by individual IdPs based on the resources they choose to make available. The `redirect_uri` specifies the endpoint to which the IdP should redirect the user agent once the request has been authorized (or denied). In order to redirect to the correct RP, the OAuth 2.0 specification [53] requires the IdP to check if this URI included in the request matches with the value registered by the RP. The authorization request also includes a `state` parameter, a unique (non-guessable) string generated by the RP, to protect against CSRF (Cross-Site Request Forgery) attacks (described in the next step).

2. **IdP request to RO**: In Step ②, the RP redirects the user agent to the IdP's *authorization server* (Figure 3.1 includes example UIs) where the RO (end-user) is prompted to authenticate using their IdP credentials. Then, the RO is asked if they want to grant the RP access to the requested data (Step ③). If the RO completes login and approves the requested access, the IdP redirects the user agent back to the RP along with a fresh *authorization code* value (Step ④) and the RP's `state` value (from the authorization request), included as query components to the redirection URI. Before redirection to the RP, the IdP must verify that the redirection URI specified in the SSO request matches

a value provided by the RP during its registration with the IdP. This ensures that the authorization codes are delivered to the correct RP URL. To mitigate CSRF attacks, the RP must verify that the returned `state` value is equal to the value included in the initial request. The standard does not specify what should happen if the RO denies access, leaving it to the IdP's discretion.

3. **Token exchange**: The RP needs to exchange the authorization code for an access token before gaining access to user data, typically from its server-side app (which is considered more secure than a channel from the RO's browser). In Step ⑤, the RP sends the authorization code (along with an optional `client_secret` for RP authentication) to the IdP's *token exchange* endpoint where the exchange request is verified. If the authorization code (and the RP's `client_secret`) is valid, the IdP issues a fresh access token and returns it to the RP (Step ⑥). 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) [57] that protects the integrity of information within the token.

4. **Data access**: The obtained access token grants its bearer (i.e., the RP) access to data resources approved by the RO. The RP makes an access request (Step ⑦) to the *resource server* (RS) to retrieve resources. This server is typically also owned by the IdP, though it may be hosted in a different domain than the authorization server. If the access token is valid, the RS returns the requested data to the RP (Step ⑧).

When used by public clients (e.g., browser-based web apps, where the RP app code is fully available on a browser, and thus cannot store the `client_secret` value safely), 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 an OAuth 2.0 extension known as Proof Key for Code Exchange (PKCE) [89].
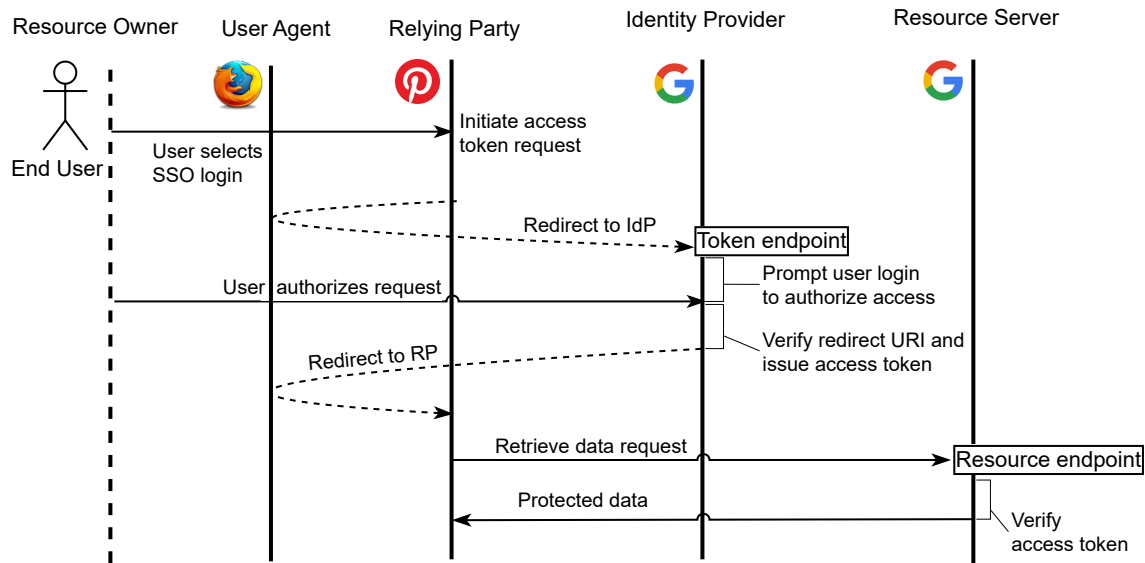
Figure 2.2: Procedure for the OAuth 2.0 Implicit flow, with example entities.

### 2.2.2 Implicit (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, similar to the authorization code flow (Sec. 2.2.1). Since the access token is returned in the redirection URI, it is retained 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 also be able to access the token. Fig. 2.2 gives a diagram for the implicit flow.

The OAuth 2.0 implicit flow was originally designed when browsers restricted apps to making requests only to its own domain [80]. 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 the implicit flow provides the necessary functionality, it is not recommended from a security perspective due to the risks associated with storing credentials in URIs [46]. Modern browsers now support CORS (Cross-Origin Resource Sharing) that enables a website to request resources from other permitted origins, removing the need for this workaround.

### 2.2.3   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 for a new access token to be issued to the RP. A "bearer" token is commonly issued to allow access by any party in possession of the token. An unauthorized party in possession of such a token 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 practice is to combine an access token with a *refresh token* [53], a string issued by the IdP allowing the RP to extend existing access without involving the user. 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 their use to only server-side flows such as the authorization code flow [53].

### 2.2.4   OAuth 2.0 Privacy Risks

Privacy risks related to OAuth-based systems can occur by users unintentionally revealing personal information to either RPs or IdPs. For example, although OAuth 2.0 [53] removes the need to re-authenticate for each RP login request, each request is still redirected to the IdP endpoint. This design reveals the identities of the RP websites visited by users to the IdP, including metadata such as time of visits. Such

information can be used by IdPs to build profiles of users and track across RP websites [69]. Another risk involves an RP accessing tokens to retrieve and store user personal information for processing outside the SSO login system. Using subsequent refresh tokens [53], the RP can continue its access silently (i.e., without prompting the user again) and retrieve new data generated by users on their IdP accounts after the initial login. Such ongoing background access poses privacy risks that may not be apparent to users. In Section 3.3, we explore other privacy issues related to our research goal on privacy consequences for users of OAuth-based RP sites.

### 2.2.5 Extensions to OAuth 2.0 and Future OAuth-based Protocols

OAuth 2.0 [53] is a complicated framework, consisting of various flows and mechanisms to support authorization, and it leaves many of the implementation details to the RPs and IdPs. Its framework design has contributed to several OAuth 2.0 extensions (in separate RFCs) that target a multitude of use cases and security mechanisms [79]; for example, there are currently 27 OAuth 2.0-related extension documents produced by the IETF (Internet Engineering Task Force) Working Group for OAuth.[1] Developers are expected to understand OAuth 2.0 along with its extensions to identify the most relevant flows and extensions for their applications. This complexity often leads to developers making incorrect security assumptions (e.g., about IdP SDKs [107]) and leads to major vulnerabilities in OAuth 2.0 implementations (discussed later in Section 2.5). These issues have also influenced work on future OAuth protocols, as described below.

**Example OAuth 2.0 Extensions**

To illustrate the complexity of OAuth 2.0 further, we describe two OAuth extensions, each of which is an alternate security mechanism proposed as mitigation for access token thefts.

**Extension 1.** Bearer tokens (e.g., typical OAuth 2.0 access tokens) 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

---

[1]https://datatracker.ietf.org/wg/oauth/

(mTLS) [16] where the tokens are bound to the RP client's X.509 certificate (self-signed or signed by a trusted CA), restricting their use to the RP client in possession of the certificate's private key.

**Extension 2.** A separate alternate security mechanism to protect against token thefts is Demonstration of Proof of Possession (DPoP) [39], which utilizes public/private key pairs to bind access and refresh tokens to the key pairs. The client needs to prove possession of the private key in order to use an access token bound to the corresponding public key.

Depending on individual applications and relevant threats, developers are required to choose from such OAuth 2.0 extensions and implement security mechanisms correctly to ensure sufficient protection against common attacks.

**Future OAuth Protocols**

Deploying OAuth 2.0 with adequate security protections involves navigating a complicated set of OAuth 2.0 and related extension specifications. This complexity has motivated various protocol improvements currently being developed by the OAuth Working Group to simplify implementations.

OAuth 2.1 [54] is an ongoing effort to combine several OAuth 2.0 extensions related to security best practices into a single spec to help guide secure implementations of OAuth 2.0. For example, it mandates the use of the PKCE [89] as protection against authorization code injection attacks. It is not a replacement for OAuth 2.0 as it does not change the original data structures and maintains the main OAuth 2.0 flows.

Grant Negotiation and Authorization Protocol (GNAP) [86] is another OAuth 2.0-related IETF Internet Draft currently being pursued as a '*next-generation OAuth protocol*'. It is an effort to fix protocol flaws identified in OAuth 2.0 deployments and extend its current set of use cases to more complex deployment environments. For example, GNAP aims to support authorization in larger set of native applications (apps designed with a focus on certain target OS or device type), such as apps in IoT (Internet of Things) devices with limited UI capabilities. It also aims to improve user experience in mobile apps by introducing new *interaction modes* offering different UI mechanisms for users to authenticate with an IdP and login to RP apps.

Table 2.1: A high-level comparison of popular web SSO schemes

| SSO scheme | Origin | Security goal | Main artifacts | Deployment target |
|---|---|---|---|---|
| SAML 2.0 | 2005 | Authentication | Assertions | Affiliated services |
| OAuth 2.0 | 2012 | Authorization | Access Tokens | Third-party services |
| OpenID Connect | 2014 | Authentication | ID Tokens | Third-party services |
| GNAP | 2023[†] | Authorization | Access Tokens | Native applications |

[†]In development stage

## 2.3   Other Web SSO Protocols

For context, we now describe other common web SSO schemes, including the OAuth-based OpenID Connect. While OAuth focuses on authorization, the following schemes target authentication. Table 2.1 provides a high-level comparison of the web SSO schemes discussed herein.

### 2.3.1   OpenID Connect

The OpenID Connect [90] specification (OIDC) was developed as an authentication extension to OAuth 2.0. Before this specification, it was possible for RPs to adapt OAuth to perform user authentication in SSO login if they applied customizations. The OIDC specification formalized commonly used customizations and introduced features to help make user authentication easier. It introduces the *ID token*, a value issued by an IdP (also referred to as an OpenID Provider) to convey information about a user's identity to an RP. Since many RPs use both OIDC and OAuth 2.0, we refer to OpenID Providers as IdPs in this thesis. Although OIDC is built on top of OAuth 2.0, some IdPs instead use custom modifications (details are included in Section 3.1) of OAuth 2.0 to provide authentication capabilities.

OIDC extends OAuth 2.0 to define a set of six OIDC flows for RPs to obtain ID tokens and access tokens when requested. It allows any combination of `id_token`, OAuth 2.0 values `code` and `token` for the `response_type` parameter. 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 those values. Similar to OAuth 2.0, each OIDC flow begins with an

initial *authentication request* from RP to IdP with the `response_type` parameter indicating the OIDC flow type. Authentication requests use the same parameters as authorization requests in OAuth 2.0 but with special values. For example, the OIDC specification uses the `scope` parameter with the value "openid" to request identity-related information. Each flow then redirects the user agent to the IdP's login page where the user is prompted to grant the RP access to basic information such as their name and email address. If the user consents, the IdP then redirects the user agent back to the RP (at the specified redirect URI). Depending on the OIDC flow, the IdP might redirect to the RP with an authorization code (e.g., as in the authorization code flow) or directly return the ID token (e.g., in the implicit flow).

Common data returned in an ID token include a unique identifier for the user, an expiry time for the token, and an IdP identifier. A standard ID token uses the JSON Web Token (JWT) [57] data structure to represent a digitally signed JSON object that contains verifiable *claims* about an authenticated user. Standard claims about the user include basic profile info such as name, email address, and profile picture. ID tokens enable the RP to verify a user's identity based on the claims returned by the IdP. The claims contained in the ID tokens could reveal sensitive personal information depending on specific IdP implementations.

### 2.3.2 Security Assertion Markup Language (SAML)

Security Assertion Markup Language (SAML) [77] is a web SSO authentication framework based on the XML format used for exchanging identity information between an IdP and an RP. The framework consists of multiple specifications that define message structures and mechanisms for implementing multi-domain web SSO. The SAML 2.0 *core* specification [29] defines a set of *assertions*, which are signed statements made by an IdP to convey information about a *subject*'s identity to an RP. The assertion statements are primarily about the subject's identity but can also include authorization information that help an RP make access decisions (e.g., whether the subject is authorized to access a particular resource). The core specification also describes the structure of *protocol messages* used for assertion requests and responses between

SAML parties. SAML 2.0 *bindings* specify mechanisms for transmitting protocol messages and assertions over existing web protocols such as HTTP and SOAP (Simple Object Access Protocol), enabling authentication communications between different web SSO contexts.

SAML *profiles* define combinations of various SAML framework components for use in particular deployment scenarios, the most common being the *Web Browser SSO profile* [28]. This profile involves a typical web SSO workflow where a user authenticated on an IdP platform (e.g., a banking site) needs to access a partner RP service (e.g., to make transactions using another financial application). The RP receives SAML assertions from the IdP to automatically authenticate and identify users on its platform. A pre-established agreement between the IdP and the RP enables the use of cross-domain identifiers for user accounts.

SAML is more flexible compared to other SSO schemes (including OAuth 2.0 protocols) as the SAML framework provides components that can be combined to serve many use cases. However, many protocol details (such as how protocol messages are transmitted over the internet) need to be specified and agreed upon by the federation members before deployment. While this flexibility might be desirable for some service providers, it increases implementation complexity and reduces scalability [1]. Many SAML implementations provide software to simplify SSO deployments of SAML. Shibboleth [94] is an open-source SAML implementation popular in academic and governmental organizations. SAML-based SSO is also widely used in commercial software, including in Microsoft's Active Directory Federation Services (ADFS).

## 2.4  Privacy of OAuth and OpenID Connect Systems

In this section, we provide an overview of existing privacy research related to OAuth-based web SSO systems. Differences between our works and other related studies are detailed in the respective chapters, once our work has been presented.

Early privacy measurements [37, 105] of OAuth systems suggest practices of over-requesting of permissions by RPs. Many existing proposals to improve privacy depend on protocol-level changes that are difficult to implement. While some practical tools (e.g., Shehab et al. [93]) exist, they do not cover IdPs in current OAuth deployments.

### 2.4.1 Privacy Measurements

Dimova et al. [23] conducted a large-scale privacy measurement of the OAuth 2.0 permissions requested through web SSO login options on 100K websites. Their work extended our empirical study (Section 3.3) to a larger number of websites, and built on our categorization of IdP APIs (Section 3.1). Järpehult et al. [58] developed a tool based on the Selenium browser automation framework [92] to track differences in RP permission requests and IdP usage over a nine-year period. Farooqi et al. [35] designed a privacy tool to detect third-party apps on Facebook that abused their access to users' personal information. Similar to the OAuthScope tool (Section 3.2), these tools use browser automation to scan RP sites and extract OAuth 2.0 protocol information.

Wang et al. [105] analyzed the privacy practices of third-party apps on Facebook, including the personal information about users requested by these apps. Felt and Evans [37] examined the permissions in 150 Facebook apps and found many that requested user data that was not essential for their services.

### 2.4.2 Privacy-Enhancing Proposals

Increasing concerns about privacy issues have motivated researchers to propose privacy-enhancing protocols. Fett et al. [40] proposed *SPRESSO*, an OAuth client-side forwarder that masks the RP's identity from the IdP. Hammann et al. [52] proposed *POIDC*, a protocol extension to OpenID Connect 1.0 to hide RP identity from the IdP by masking the RP's client ID using hash functions. Dey and Weis [22] proposed *PseudoID* to enable users to login to RPs using pseudonyms. Xu et al. [112] proposed *MISO* to hide the RP's identity by introducing a middle component hosted in a TEE (Trusted Execution Environment). These existing proposals that aim to hide the RP's identity from the IdP involve protocol changes or modifications to the RP and/or IdP software, requiring buy-in from various stakeholders. Shehab et al. [93] proposed *ROAuth* as an extension to OAuth 2.0 and built a Firefox browser extension to allow Facebook users to configure (and possibly limit) requested permissions in SSO login with Facebook.

## 2.5 Security of OAuth and OpenID Connect Systems

Much of the research literature on OAuth gives focus to security issues and the development of automated tools to test the security of RP and IdP implementations. OAuth-based web SSO security has been studied extensively both in empirical studies of real-world OAuth and OIDC systems [45, 62, 66, 100, 106], and in formal analyses [41, 42, 55] of OAuth protocols. Using a variety of security approaches, these studies have identified many vulnerable RP and IdP deployments.

### 2.5.1 Security Testing

Zhou and Evans [114] built *SSOScan*, a SSO security testing tool to automatically scan sites with Facebook SSO login by simulating user actions using browser automation. Using a similar methodology, Drakonakis et al. [24] built *XDriver*, a security tester that scans web login implementations (including RPs that offer login with Facebook and Google) for vulnerabilities related to exposed authentication cookies. Ghasemisharif et al. [45] reviewed access revocation in RPs following an IdP account compromise and presented the resulting security implications. Westers et al. [109] built *SSO-Monitor* for analyzing the security of SSO login workflows with three IdPs. They developed a visual-based approach to identify SSO login buttons. Similar to the OAuthScope tool (Section 3.2) and the Comparative mode of the SSOPrivateEye Chrome extension (Chapter 4), these other tools rely on custom heuristics to search RP HTML pages for SSO login options and related UI components.

### 2.5.2 Security Measurements

Sun et al. [100] evaluated OAuth 2.0 implementations by three major IdPs (Facebook, Microsoft, and Google) and 96 RPs supporting Facebook SSO, and found several implementation decisions that led to security issues such as access token thefts. Yang et al. [113] proposed an OAuth 2.0 security testing framework and evaluated four IdPs (Facebook, Sina, Renren, and Tencent Weibo) and the 500 top-ranked web apps in US and China. Chen et al. [18] evaluated the use of OAuth in mobile apps and revealed many vulnerabilities across several apps due to implementation flaws. In a

2012 field study of popular web SSO systems, Wang et al. [106] analyzed SSO web traffic through the browser and identified flaws in popular RPs and IdPs, allowing attackers to impersonate victim users. Li et al. [62] developed *OAuthGuard*, a Chrome extension for security analysis of HTTP traffic in RPs with Google SSO login to detect vulnerable OAuth 2.0 implementations. Using the tool, they found 69 of 137 RPs with at least one OAuth-related vulnerability.

These empirical studies reveal a lack adequate security protections in many OAuth 2.0 systems, raising security and privacy concerns about users' online identity and personal information. Privacy implications discussed in our work (Section 3.3) complement prior work on OAuth security implications from identified vulnerabilities.

### 2.5.3   Security Analysis Approaches

Mainka et al. [67] categorized SSO testing approaches used in previous research, and built *PrOfESSOS*, a fully-automated security tool for testing OpenID Connect implementations. They also discussed the benefits and limitations of each analysis approach. Assuming malicious IdPs, Mainka et al. [66] developed a penetration testing tool to check IdP implementations for security issues. Using this tool, they also developed new attacks on 11 of 16 OpenID implementations.

Fett et al. [41] pursued formal analysis of the OAuth 2.0 standard and proofs of security properties for all OAuth 2.0 flow types. Helmschmidt et al. [55] performed formal analysis of the currently in-development GNAP [86] standard (a potential OAuth 2.0 successor). Using formal verification techniques, Wang et al. [107] modelled security assumptions made by OAuth app developers in the use of IdP SDKs. Their findings highlight RP implementation flaws due to a disconnect between implicit security assumptions in the SDKs and those made by RP app developers.

Rahat et al. [84] built *Cerberus* to test web OAuth 2.0 implementations for compliance with security best practices, and *OAuthLint* [83] for similar testing of OAuth 2.0 apps in mobile applications. Yang et al. [113] built *OAuthTester*, a fuzz testing tool to identify security issues in OAuth RP and IdP implementations by modifying security-related protocol parameters such as OAuth 2.0 `state` and `redirect URI`. Bai et al. [9] developed *AuthScan* to extract security properties from authentication

protocols—including SSO protocols—and verify the security of their implementations.

## 2.6 Web SSO Studies on Usable Security and Privacy

In this section, we discuss prior work related to SSO user perceptions and user privacy. Existing research in this area has focused mainly on individual IdP platforms. Many studies (e.g., [10, 26, 101]) have identified several user concerns about privacy and security in the use of web SSO systems. Studies also show that SSO login prompts are generally ineffective in informing users about the privacy [87] and security [101] of SSO login systems.

### 2.6.1 Properties of Web SSO Schemes

Bonneau et al. [13] 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. [1] proposed a web SSO evaluation framework and compared 14 web SSO schemes, including password managers and OAuth protocols, and identified defining characteristics of each scheme based on various security, privacy, usability, and deployability properties. While OAuth is dominant among SSO schemes, their comparison highlights that although OAuth offers various benefits to RPs and IdPs, it can be unfavourable to users because of its lack of focus on privacy.

### 2.6.2 User Perceptions

Balash et al. [10] investigated user perceptions of third-party sites that support Google login and access personal information from Google accounts. In their findings, participants raised concerns about granting RP sites access to sensitive data such as personal emails and contacts. However, they also found considerable variations in participants' understanding of the necessity of particular permissions, including participants who perceived more sensitive permissions (such as access to email messages) as more necessary than other basic profile information. They make suggestions on IdP design changes to improve transparency of SSO permissions.

Bauer et al. [11] considered three IdPs (Google, Facebook, and Google+) to study the effectiveness of the consent dialogs used by these IdPs. Robinson and Bonneau [87] evaluated participants' understanding of Facebook Connect's SSO permissions, specifically about their interpretation of read and write permissions. They noticed that participants did not realize the extent to which an RP app with access could post to their Facebook profiles. Participants also incorrectly assumed that regardless of the permissions they grant, their privacy settings on Facebook account would prevent sites from reading data marked as private.

These studies show that SSO login prompts used by major IdP platforms are often ineffective in adequately informing users about the privacy implications of choosing SSO and granting third parties access to their data.

### 2.6.3  User Attitudes

To understand user concerns in OpenID SSO login systems, Sun et al. [101] conducted a study to identify factors that influence users to avoid OpenID SSO login options. They observed that participants were hesitant to adopt (OpenID-based) SSO due to several concerns including incorrect assumptions about the SSO system and a lack of motivation to replace existing password management processes with SSO logins. Many users held the misconception that their IdP credentials were shared with the RP. Users also indicated privacy concerns when they were prompted to grant access to personal information (e.g., contact lists, status updates) from their IdP profiles.

Egelman [26] studied user login decisions in sites with Facebook Connect, an earlier version of the current Facebook Login SSO platform, focusing on the privacy and convenience aspects related to the SSO platform. We find similar privacy-usability tradeoffs in our study (Section 5.4.3), but within a larger context involving other IdP platforms and additional privacy choices.

### 2.7  Generic Web Privacy and User Awareness

Due to relatively limited existing work focused on privacy of real-world OAuth deployments, we now look at privacy research in the broader web and mobile ecosystems to better inform our work on SSO privacy.

### 2.7.1 Web Privacy and Deceptive Patterns

The Selenium-based *OpenWPM* platform by Englehardt et al. [27] is designed for large-scale analysis of user tracking on the web. It examines sites to identify privacy leaks involving unique user identifiers and PII (Personally Identifiable Information). In a survey of one million sites, the authors found abuse of newer browser APIs (e.g., Battery Status API, Canvas API) by sites to fingerprint and track users. Narayanan et al. [74] discussed deceptive design patterns in online services used to influence less-informed users into making choices not in their best interest. Mathur et al. [68] investigated deceptive patterns in 11K shopping sites and found 1,818 instances of designs intended to increase user purchases. Weinshel et al. [108] built a browser extension to inform users about web trackers using privacy dashboards that help visualize web tracking activity.

### 2.7.2 Privacy in Mobile Apps

Many researchers have studied permissions and user privacy in mobile apps. The *AppCensus* tool, by Reardon et al. [85], uses dynamic analysis to automatically instrument the privacy behaviour of Android apps to generate privacy labels. Kelley et al. [60] introduced "privacy nutrition labels" to inform web users about privacy policies, and evaluated its usability. More recently, Apple introduced *privacy labels* [7] to inform its users of privacy practises in iOS apps. Felt et al. [38] conducted user studies to evaluate the effectiveness of Android permissions and they found that the majority of participants were unaware or did not look at permission warnings. Unlike mobile apps where users are given only one set of permissions, web SSO users often have the choice (although hidden) to login to a given website with a less privacy-intrusive alternative [70], so user awareness could significantly influence decisions.

Cao et al. [17] collected data on users' privacy behaviour in Android app permissions, finding that users were more likely to grant permissions when informed about its use. Wijesekera et al. [110] built an Android privacy classifier to infer privacy preferences of users based on previous app permission requests. Felt et al. [36] compared different approaches in Android apps to request permissions and obtain consent from users. Liu et al. [65] proposed a privacy tool to manage permissions in Android apps

according to users' privacy preferences.

## 2.8   Summary

OAuth-based SSO deployments offer convenience to users, but many RPs and IdPs are found with vulnerabilities that compromise security and privacy. Separate from unintended vulnerabilities, RPs could also access privacy-compromising personal information of users through OAuth platforms. While there exists several works involving security analysis of OAuth protocols and deployments, there is little research available on the privacy of OAuth systems. We begin to address this gap through our work presented in the following chapters.

# Chapter 3

# Empirical Privacy Analysis of OAuth SSO Systems

Questionable third-party privacy practices observed on certain SSO platforms include IdPs that allow RPs to override users' privacy preferences [105], and RPs that access personal information beyond what is necessary to offer the stated functionality [37]. Over the past decade, there has been an increase in the number of RPs that offer multiple SSO choices [58]; this motivates further research on the privacy issues in current SSO systems. In this chapter, we present empirical analysis of privacy practices in popular RP and IdP systems involving multiple SSO platforms and SSO login choices.

Three primary factors affect users' privacy when making login decisions in web SSO systems. First, each IdP exposes APIs giving access to user data that are relevant to their own app ecosystem, e.g., Facebook's SSO platform [32] offers APIs for RP apps to access a user's photos and videos. Second, user privacy depends on the actual user data requested by an RP for access through the SSO login option selected by a user. Third, API access is granted for broadly specified user data types and APIs do not restrict access to data from a specific time range. This means that the amount of user data an API releases to a site depends on user factors such as how long ago the user's account was created and how much user activity is associated with the account. This chapter focuses on the first two factors, and makes the following contributions:

- A categorization of user data attributes made available by four popular IdPs for access by third-party RPs (Section 3.1).

- *OAuthScope*, a tool to extract OAuth protocol request parameters from RP sites supporting SSO login options (Section 3.2).

- An empirical study of privacy in OAuth-based logins in the Alexa Top 500 sites for five countries (Section 3.3).

## 3.1 API Analysis of Identity Providers

User data available to third-party RP services through OAuth-backed APIs vary with each IdP. The diverse sets of native services provided by IdPs lead to differences in the types of user data made 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 four major IdPs (Google, Facebook, Apple, and LinkedIn) and categorized them based on how the information could be used (or misused). To this end, we identified five data categories: *basic* (online identity), (real world) *identity*, *personal*, *interests*, and *other sensitive*, as listed in Table 3.1. We note that Google and Facebook make available other OAuth 2.0 attributes (e.g., for use by advertising customers [50]) but these are not related to personal information of users; thus these APIs are not discussed in the work presented herein.

### 3.1.1 Overview of API Categories

**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 requesting access. 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 attributes 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 RP user's videos). We include data attributes such as Facebook Likes (i.e., 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 profile users based on their online behaviour. Finally, we group other user data such as email contents, contact lists, and documents in personal cloud storage under

Table 3.1: Categorization of data attributes in IdP APIs related to personal information of users. Default fields (applicable to Google and Facebook) are italicized. For further details explaining the entries, see API documentation of Google [51], Facebook [33], Apple [4], and LinkedIn [63]. Fields denoted with * indicate permissions not requested (but available) by any RP site in our study presented in Section 3.3.

| Category | Google | Facebook | Apple | LinkedIn |
|---|---|---|---|---|
| **Basic** (online identifier) | email (address) *profile* openid | email (address) *public_profile* | email (address) name (customizable) | 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 |

the *other sensitive* category as such data could include sensitive personal information.

**Categorization approach.** We note that some user data attributes can be associated with multiple categories. For example, a passport document 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 categorization of the APIs. Table 3.1 groups relevant attributes from each provider into one of 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 popular websites considered in our privacy study (presented in Section 3.3).

**Related categorization.** In a recent study, Dimova et al. [23] built on our categorization to compare OAuth 2.0 user data attributes made available by 33 IdPs. Besides our categories, they separated data attributes based on *read* and *write* privileges. They also introduced a *functional* category to group attributes that help RPs manage access to the IdP services; as noted above, we excluded these attributes as they are not related to user personal data.

Next, we provide further details on the user data attributes and OAuth protocol features supported by each of the four IdPs.

### 3.1.2    Google OAuth API

Google's OAuth 2.0 Scopes document [51] categorizes data attributes into APIs based on the service where each API is typically 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 that externally manage users' emails in their Gmail inboxes through OAuth 2.0, including Mozilla's Thunderbird email client [73]. Google also uses OAuth 2.0 to make available several other APIs for use by developers and customers across different apps, including popular products such as Google Cloud Platform (GCP).[1] Google's OAuth 2.0 APIs classify a subset of their attributes as belonging to *sensitive* (different from the *sensitive* category defined in this chapter) and *restricted* scopes. Most RP applications requesting access to any of these attributes must go through a verification process reviewed by Google [49].

Google's OAuth 2.0 platform includes two types of profile-related attributes. The `profile` [48] attribute under *basic* category shown in Table 3.1 includes only the user's name, email, and public profile image. These attributes are included by default in response to RP requests, and prompted to users in the IdP consent dialog. The other type (`userinfo.profile`) includes all publicly available information from the user's Google profile (which might include their birthday or home address), and must be explicitly requested by the RPs.

---

[1]`https://developers.google.com/identity/protocols/oauth2/scopes#cloudasset`

**Supported Flows**

Google's SSO platform implements both OAuth 2.0 and OpenID Connect specifications and supports standard protocol flows, including the implicit and authorization code flows. RPs use the standard `response_type` parameter to specify the flow type, as described in Section 2.2. Additionally, the value "permission" can be used for the `scope` parameter to indicate use of implicit flow [48]. The platform also supports the OIDC attribute, `openid` [90], that allows the RP to obtain from Google an ID token containing fields that assert the user's identity.

### 3.1.3   Facebook OAuth API

Facebook's Graph API [32] is the platform for third-party RP apps and websites to interact with a user's social data on Facebook. RPs implementing SSO login with Facebook use the *Facebook Login* interface to identify users and access user data. This platform provides unique social data (such as Facebook Likes and social media posts created by the user) not available through other IdPs. In our empirical study of top 500 websites (presented in Section 3.3), we find that almost all of the Facebook Login attributes (denoted in Table 3.1) are requested for access by RPs in the top 500 websites. Similar to Google SSO above, Facebook Login also offers several administrative APIs beyond attributes related to personal user data. For example, a subset of APIs enable RPs to view, create, edit or delete content on Facebook Pages administered by a user. For the work presented in this chapter, we exclude these attributes and limit our analysis to the user-data attributes listed in Table 3.1.

All SSO login requests from RPs (including those that do not explicitly request the `public_profile` attribute) to *Facebook Login* require users to grant the RPs access to default fields (i.e., Facebook name and profile picture) [33]. These mandatory fields are useful for RP websites to display basic information about an authenticated user, e.g., as a confirmation of login through the IdP account. RPs requesting permissions beyond the `public_profile`, `email` and `pages_show_list` (list of Facebook Pages managed by the user) attributes are required to undergo an approval process by Facebook [31]; Section 5.1.3 gives further details on these RP reviews conducted by Facebook and other IdPs.

**Supported Flows**

*Facebook Login* supports the standard OAuth 2.0 authorization code flow and the implicit flow. Beyond the standard protocol parameters, 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 OpenID Connect specification is not explicitly supported by *Facebook Login*, authentication can be performed using the `signed_request` parameter which returns a signed base64url encoded JSON object containing a user ID issued by Facebook.

### 3.1.4 Apple OAuth API

The *Sign in with Apple* SSO platform was introduced in 2019 as a privacy-friendly alternative for Apple users wanting to use third-party web and mobile apps without disclosing their personal information. It enables RPs to authenticate SSO users and to optionally request access to the user's `name` and `email` through the standard OAuth 2.0 `scope` parameter. Apps on Apple's App Store using a third-party SSO service (e.g., Facebook) are required to also offer Apple as an additional SSO login option [8].

Sign in with Apple offers privacy features where users can login to an RP site without revealing their real name or email address. When an RP site requests the user's name, Apple's SSO login prompt shows an option to choose a custom name, thus enabling the user to hide their real name from the RP. When access to email address is requested, Apple allows its users to either share their original email address or create a pseudonymous email address enabled by Apple's Private Email Relay Service [95]. Using this privacy feature, email messages between a user and an RP are routed through Apple without revealing the user's actual email address to the RP. This enables the RP to communicate with the user through their email inbox while the IdP restricts the RP's ability to track or sell the user's email address.

Sign in with Apple also helps RPs to distinguish real users from bots through a real user indicator [4]. The platform's documentation [3] suggests that it determines an account status based on its history and usage patterns on Apple devices. There are three possible values ("LikelyReal", "Unknown", or "Unsupported") that indicate whether the user signing in to the RP is likely a real user. Depending on the precision,

RPs might find this feature more useful over CAPTCHAs that involve manual user steps or forced delays.

**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 [4] lacks explicit mention of the OpenID Connect standard, it closely follows OpenID Connect conventions discussed in Section 2.3.1. *Sign in with Apple* does not support the implicit flow.

### 3.1.5   LinkedIn OAuth API

The *Sign In with LinkedIn* [63] SSO platform allows users to authenticate and authorize LinkedIn profile access to third-party RPs. LinkedIn categorizes users' profile data into three groups of permissions (i.e., values that RPs can specify in OAuth login requests) for RPs to access. First, the `r_emailaddress` scope grants minimal access to a user's email address. Second, the `r_liteprofile` (or `r_basicprofile` in some earlier versions of the LinkedIn SSO platform) scope is for RPs requesting access to the user's full name, profile picture (including image metadata), and profile headline. Third, the `r_fullprofile`[2] scope covers further access by additionally including all the other fields (listed in Table 3.1) appearing on the user's LinkedIn profile. While grouping attributes might simplify managing permissions for RPs, privacy-conscious users might prefer more fine-grained permissions to better control personal information disclosed to RPs.

**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* [53] for systems requiring machine-to-machine authorization, without user involvement, and

---

[2]Changes to LinkedIn SSO in Aug 2023 suggest the deprecation of the `r_fullprofile` scope [64].

is outside the scope of this thesis. The Version 1 of *Sign In with LinkedIn* (2018) discussed herein does not explicitly support the OpenID Connect authentication protocol, though it provides a parallel mechanism using OAuth 2.0. Instead of the standard ID token, LinkedIn Version 1 enables RPs to identify users by including *member IDs* (unique identifiers specific to RP-user pairs) in OAuth 2.0 access tokens. However, Version 2 (updated in August 2023) [64] introduced support for the OpenID Connect protocol, and replaces member IDs with the standard ID token for conveying user identity information (Section 2.3.1).

### 3.1.6  IdP Login Interface

Figure 3.1 (page 36) shows the IdP SSO login dialogs presented to users when they choose to login with popular IdPs. Information about permissions requested by an RP is presented to users at different stages of the login process by each IdP. If the user is not already logged in, some IdPs (e.g., Apple, Facebook, and LinkedIn) ask the user to login before showing the requested permissions (the consent dialogs). This approach means that, when RPs offer multiple SSO login options, users can only view the permissions requested with a specific IdP *after* signing in with that IdP.

To make an informed login decision, users would need to click on all listed SSO login options and complete authentication with each IdP (i.e., enter credentials if they are not already signed in with that 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 than intended. While some IdPs allow users to opt-out of certain permissions (as in the Facebook example shown in Figure 3.1b), others require users to grant all the requested permissions. In an RP site with multiple SSO login options, the RP could request different amounts of personal information through each login option. Thus, it is important to present the requested permissions for all login options to users prior to their decision to login with a specific login option.

### 3.2  The OAuthScope Tool

In this section, we present a web tool for extracting OAuth 2.0 and OpenID Connect protocol parameters from real-world RP and IdP implementations. We describe

(a) Google login prompt

(b) Facebook consent dialog



(c) LinkedIn consent dialog

(d) Apple consent dialog
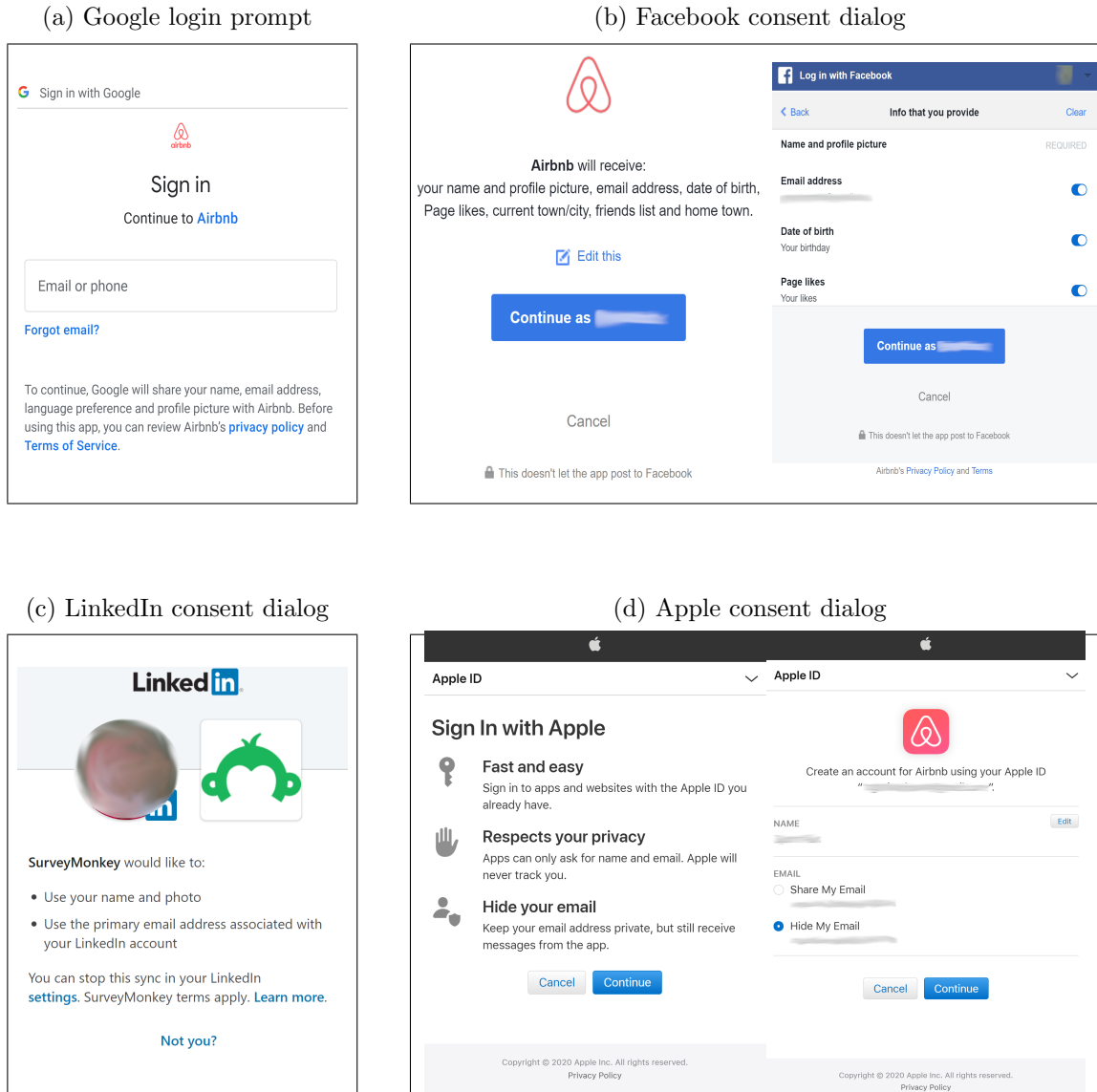


Figure 3.1: SSO login prompts and consent dialogs on IdP sites. (a) Google and (c) LinkedIn do not allow users to alter fine-grained permissions granted to RPs. (b) Facebook allows users to selectively opt-out of non-*default* permissions requested by an RP. (d) Apple allows users to use a substitute name and pseudonymous email with an RP. In cases of (b), (c), and (d), the IdP presents its default login prompt before showing the consent dialogs if the user is not already logged in, as discussed inline. Personal details are greyed out in these images.

the design and the implementation of our tool, which we used for collecting data for the empirical privacy study of SSO login options in RP websites (presented in Section 3.3). We compare our work with other related tools, highlighting the differences and similarities in approaches to scan and collect data on SSO login implementations.

### 3.2.1  Design Approach

We designed *OAuthScope*, a Java web tool based on the open-source Spring Framework [103] for Java web applications. The goal of OAuthScope is to scan websites, and to extract and store OAuth 2.0 protocol-related parameters from authentication and authorization requests made by relying parties to identity providers. Spring is a popular Java framework that provides packages to simplify the development of applications that rely on Java language features such as JPA (Java Persistence API) and JDBC (Java Database Connectivity) APIs.

When provided a list of webpage URLs as input, OAuthScope visits each URL in a fresh automated browser window on a local server to scan the website and locate OAuth-based SSO login requests to any of the four major IdPs (Google, Facebook, Apple, and LinkedIn). Then, OAuthScope triggers and intercepts SSO login requests for the identified SSO login options and extracts the OAuth 2.0 protocol parameters, including the user data permissions requested by the RPs. A fresh browser window for each website ensures that the scans are unaffected by previous browser sessions which might include site-specific data (e.g., cookies, cache) and browsing history.

OAuthScope stores the extracted protocol-related data for analysis in a MySQL Database, managed through Spring's Data module interface.[3] We designed OAuthScope to support multiple database tables to simplify analysis of individual datasets (e.g., our empirical study in Section 3.3 involved database tables for storing data on websites in different countries). OAuthScope also includes a web front-end shown in Figure 3.2 (page 38) for displaying and filtering the entries in a selected database. This interface parses and displays the protocol parameters after translating the RP authorization requests into structured data in JSON objects to simplify analysis.

---

[3]`https://spring.io/projects/spring-data-jpa`

Figure 3.2: OAuthScope web interface showing OAuth 2.0-protocol parameters collected from top US RP websites. The search textbox (left) can be used to display only data records containing specific keywords (e.g., to find RPs that request access to "user_birthday"). The dropdown field allows for retrieving data records from available backend database tables.

## 3.2.2 Implementation

OAuthScope uses *Selenium WebDriver* [92], 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 related to SSO login options on an RP site and simulating web page navigation to extract OAuth 2.0 protocol data from supported IdP authorization servers. For each SSO login option available on the RP site, OAuthScope simulates the user actions to trigger login requests to the corresponding IdP and captures the protocol parameters in the request, including the flow type and the `scope` parameter which specifies the user data resources for which the RP intends to obtain access. The primary steps performed by OAuthScope can be summarized as follows:

1. **Identify login-related HTML elements**: 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. The

search criteria consisted of a list of XPath (XML Path Language) expressions to specify HTML elements containing text related to login buttons (e.g., "Login", "Sign in", etc).[4] 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 Web-Driver *context* (target webpage component) as necessary. Our tool also captures a screenshot of the login page, for use in manual verification of completeness (e.g., to identify any login options missed during scans).

2. **Identify and trigger SSO login requests**: On an identified login page (or `iframe`), OAuthScope conducts a search for HTML elements that potentially lead to an SSO login page of one of the four IdPs (Google, Facebook, Apple, and LinkedIn) We defined XPath expressions for identifying each IdP login option based on common HTML tags and texts we found in RPs pointing to IdPs, similar to how we defined the expressions for finding login buttons. After obtaining potential SSO login elements, our tool triggers each target element and checks if the URL of the resulting page matches with a list of predefined endpoints for each IdP. We built this list to contain OAuth 2.0 and OIDC endpoints published on each IdP's developer documentation pages.

3. **Extract OAuth protocol parameters**: The OAuth 2.0 framework specifies that protocol parameters should be added as query components of the request URI initiated by an RP (Section 2.2.1). OAuthScope extracts these parameters encoded in the link as key-value pairs, and parses those parameters into structured JSON objects. Finally, these objects and the login screenshots are stored to a database for review and analysis.

OAuthScope identifies RP login and SSO login options based on predefined XPath expressions, as described above. Thus, for a given list of websites, OAuthScope cannot automatically ensure identification of all SSO login-related elements. However, to facilitate manual checks for completeness, OAuthScope captures screenshots of the login pages and of the landing page for any site where our tool did not find HTML

---

[4]Similar to other comparable tools (e.g., [114]), OAuthScope may miss some login buttons in real-world websites despite our attempts at thoroughness, given the unlimited possible variations across different implementations.

elements related to login options, and stores the screenshots to the database. For any SSO options missed by the automated scan, the SSO login links can be manually fed to OAuthScope for extraction of the protocol parameters. We iteratively improved the initial list of XPath expressions based on strings and tags found in RP sites in our empirical study (Section 3.3). These improvements gradually reduced the need for manual involvement in data collection. In total, OAuthScope's codebase consisted of 1,493 lines of code.

### 3.2.3   Comparison with Other Tools

With the focus on SSO *security*, previous tools automatically interact with sites to test for vulnerabilities. SSOScan [114] searches sites for Facebook SSO login options and attempts to log into each site using SSO. 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 login options available on each RP site. The authentication security tool by Drakonakis et al. [24] also uses the WebDriver framework to automate login tasks (similar to OAuthScope) and to test the security of cookie-based authentication systems, including SSOs with Facebook and Google. Instead of looking for security flaws, OAuthScope targets protocol parameters related to the privacy of different SSO login options on given websites where individual login options might request different amounts of data. Thus, OAuthScope enables us to study aspects of privacy for SSO login options in RP websites, as in the study presented in the next section.

### 3.3   Analysis of Privacy Practices in Web SSO Login Services

In this section, we present our empirical privacy analysis of web SSO login options in popular RP websites to understand the privacy implications for users opting to use OAuth-based SSO to log into RP sites. We compare the privacy and security of different SSO login options available on individual RP websites, and also compare RP website versions presented to users in different countries. Based on our findings, we discuss privacy and security implications for SSO users in OAuth 2.0 systems.

Using OAuthScope (Section 3.2), we collected OAuth-protocol parameters, including the user data permissions, specified by RP websites through different IdPs. By analyzing the protocol parameters, we identify privacy consequences of choosing SSO login options in popular RP websites. We pursue the following research questions in this study:

1. What categories of user data do RP sites request from IdPs? Additionally, do sites offer similar levels of privacy to users in countries with different privacy laws?

2. How prevalent is each SSO scheme (including weak OAuth 2.0 flows) in popular RPs across different countries?

3. If an RP supports multiple SSO login options, how do the options differ in terms of requested user data attributes?

Alexa's Top 500 lists [2] provide a snapshot of the most visited websites in a given country based on 1-month traffic analysis. In an initial exploration, we manually scanned the top 500 sites in Canada and found that three major IdPs—Google, Facebook, and Apple—are predominantly supported as SSO login options. In addition to these three IdPs—which primarily relate to personal information of users—we also included LinkedIn as a fourth IdP given its popularity as a platform for sharing professional information.

Similar to LinkedIn SSO, *Log in with Twitter* [102] is also offered as an SSO login option by a number of top 500 sites. However, unlike other IdPs, Twitter SSO does not use the standard OAuth 2.0 protocol (including the use of `scope` parameter for indicating the user data released to RPs); for this reason, we do not include it in our analysis. For each of the four IdPs in our study, 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.

### 3.3.1 Methodology and Dataset

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 recorded 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 2,500 sites) as OAuthScope does not guarantee identification of SSO login options in all websites (as discussed in Section 3.2.2). Our manual effort 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 [24, 114]. 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.

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 observe 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 during July-September 2020.

Figure 3.3 shows the popularity of SSO login options presented in the top 500 sites across the five countries in our study. Google and Facebook were the most popular SSO login options in RPs across the five countries. Apple was the third most popular option, although Apple's SSO platform was introduced relatively recently in 2019. Apple SSO was also observed to be less popular in India where the Apple brand is
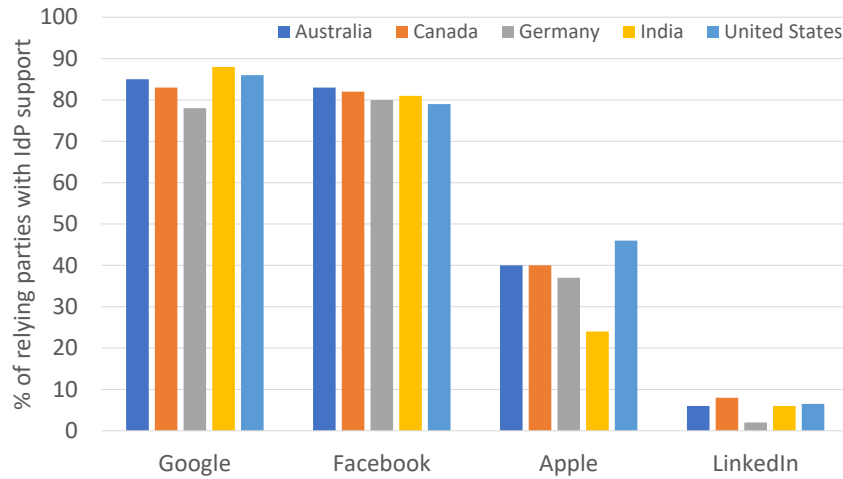
Figure 3.3: Percentage of RPs per country that offer SSO login with each IdP.

not as popular as in other countries [97]. We note that a recent mandate requiring RP apps on Apple's App Store to support Apple SSO (described in Section 3.1.4) could potentially lead to an increase in the use of Apple SSO.

### 3.3.2 Result 1: Variations in Privacy of SSO Site Options

To examine privacy differences in SSO login choices offered by RPs, we searched the US sites in our dataset and found 146 of 184 RPs with two or more available SSO login options. For each of these RPs, we categorized the requested data (using our classification shown in Table 3.1) and filtered out RPs that requested equivalent categories of user data attributes from each supported login option (i.e., where all of the RP's SSO options requested an equivalent amount of data). This resulted in a list of 43 RPs that requested different categories of data across two or more SSO login options (i.e., where an RP's SSO options each requested different amounts of data).

**Privacy choices for SSO users.** Among the above 43 RP sites, we found 42 RPs which support at least one SSO login option that requested 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 offered SSO login options 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, with Google or Apple, the site requested only *basic* attributes such as the user's *name* and *email address.*

In addition to the type of user 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 login 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.

**Deceptive design patterns.** User interfaces with deceptive designs trick users into decisions they would typically not be inclined to make. Previous studies of deceptive designs have found common UI strategies (e.g., in shopping websites) used to influence users into making choices not in their best interest [68, 74]. In our study, we found that most RPs show the most privacy-preserving choices as the last login option; for `Airbnb.com`, Facebook is shown to users first. Table 3.2 (page 45) highlights the issue: each RP is shown with its SSO login options listed in the order presented to users. In 30 of these 43 sites with two or more SSO logins, the first login option requests more data than others, suggesting the possibility of a deceptive design pattern where SSO users are subtly nudged towards more privacy-invasive SSOs.

It is possible that RPs simply list popular SSO login options earlier or encourage users to choose certain SSO options and request more data to offer a better user experience (e.g., to 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. Standard practice [81, §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.

Table 3.2: 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 RP's login page.

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

IdP: ⦿ Facebook , G Google , ⦿ Apple and in LinkedIn
Data: (**b**)asic, (**i**)dentity, (**p**)ersonal, i(**n**)terests, (**s**)ensitive

Table 3.3: 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 3.2 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 | ⌘ basic | in basic | in 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 ∗ | • |  |  |  |  |  | • |  |  |  |  |  |  | • | • |

IdP: f Facebook, G Google, ⌘ Apple, in LinkedIn
Use of implicit OAuth/OIDC flows are shown with ∗Facebook and †Google

**IdP with most user data released to RP sites.** Table 3.3 shows the individual data attributes requested by RPs with each offered SSO login option. This list includes the subset of all US RPs that requested access to two or more non-*basic* attributes from at least one SSO login option. We found that RPs requested considerably more user data with Facebook SSO compared to other IdPs (note the 10 columns in Table 3.3 for Facebook). This is unsurprising since the Facebook SSO platform [33] provides APIs (described in Section 3.1.3) for accessing a variety of user data.

Our analysis of privacy differences is limited to the data attributes *requested* by

RPs. However, 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 that requests Facebook's `user_likes` scope attribute 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 revealed to RPs could vary per user.

### 3.3.3   Result 2: SSO Differences Across Countries

We observed many websites that served different versions to users in different countries (or regions) offering varying SSO-related features in each version. For example, `Rakuten.com` (a popular e-commerce site) listed three SSO login options (Google, Facebook, and Apple) for US users and requested read access to the user's *email content* when signing in with Google. However, `Rakuten.ca` (for Canadian users) only provided two options (Facebook and Apple), while `Rakuten.de` (for German users) showed no SSO options, prompting users to create a site-specific account to shop on the site.

**Fewer login options for German users.** We considered each US site in our dataset and compared the user data revealed via SSO login options offered 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 redirected to either a country-specific sub-domain or a separate page within the same site. Other sites used scripts within the same webpage to serve different content to users in the two countries. To ensure that the compared website versions were owned by the same entities, we manually searched for each site's presence on the country-specific top-level domain and compared the site content in both versions for similarity.

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 RP sites (listed in Table 3.4) offered reduced SSO functionality to users in Germany, including fewer or no login options. Among the 80 RP sites with different

Table 3.4: 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  - | f - - - |
| houzz.com | f G  - | f G - - |
| yelp.com | f G  - | - G  - |
| aol.com | f G - - | No SSO (requires site-specific login) |
| businessinsider.com | f G - in | No SSO (requires site-specific login) |
| marriott.com | f - - - | No SSO (requires site-specific login) |
| rakuten.com | f G  - | No SSO (requires site-specific login) |
| buzzfeed.com | f G  - | No login |
| chicagotribune.com | f G  - | No login |
| cnet.com | f - - - | No login |
| foodnetwork.com | f G  - | No login |
| nvidia.com | f G  - | No login |
| allrecipes.com | f G - - | *"Because of the General Data Protection Regulation (GDPR)...can still browse the site, 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"* |
| usatoday.com | f G - - | Login page does not load |

IdP: f Facebook, G Google,  Apple, in LinkedIn

versions, we found zero instances where the RP's Germany site version requested more data than the US version. While some RPs presented a login button that failed to load (or returned an error) in Germany site version, others showed dialogs stating lack of account-requiring features for EU (European Union) users 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 RP from Germany. We were unable to collect data on 4 RP sites that blocked our access when attempting to visit using the VPN connection.

**Other country-specific differences.** Table 3.5 (page 49) provides a sorted list of the most requested user data attributes for each IdP per country. The majority of RPs requested 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 emerged across countries, we do note some variations on particular

Table 3.5: 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. Attributes denoted with * are included by the IdP by default.

| IdP | Data Attribute | AUS | CAN | DEU | IND | USA |
|---|---|---|---|---|---|---|
| G | profile* | 100 | 100 | 100 | 100 | 100 |
| | email (address) | 99 | 97 | 98 | 99 | 97 |
| | openid | 55 | 54 | 53 | 64 | 62 |
| | userinfo.profile | 11 | 11 | 17 | 17 | 10 |
| | user.birthday.read | 2 | 1 | 2 | 1 | 1 |
| | contacts | 1 | 1 | 1 | 2 | 1 |
| | gmail (email content) | 0 | 0 | 1 | 0 | 1 |
| | drive | 0 | 0 | 0 | 1 | 0 |
| f | public_profile* | 100 | 100 | 100 | 100 | 100 |
| | email (address) | 91 | 87 | 90 | 89 | 89 |
| | user_birthday | 9 | 12 | 14 | 12 | 8 |
| | user_friends | 7 | 10 | 9 | 6 | 13 |
| | user_location | 3 | 8 | 6 | 6 | 3 |
| | user_hometown | 3 | 6 | 4 | 3 | 2 |
| | user_likes | 2 | 4 | 3 | 3 | 3 |
| | user_gender | 2 | 6 | 6 | 0 | 2 |
| | user_photos | 1 | 2 | 2 | 2 | 3 |
| | user_link | 1 | 2 | 2 | 2 | 0 |
| | user_posts | 1 | 1 | 1 | 1 | 1 |
| | user_age_range | 1 | 3 | 0 | 1 | 0 |
| | user_videos | 1 | 0 | 1 | 1 | 1 |
| ́ | email (address) | 100 | 100 | 94 | 100 | 99 |
| | name (as given by user) | 94 | 95 | 87 | 95 | 92 |
| in | 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 |

IdP: f Facebook, G Google, ́ Apple, in LinkedIn

attributes. For example, Google's `userinfo.profile` was requested more frequently in Germany and India, while LinkedIn's `r_basicprofile` and `r_fullprofile` were requested most frequently in India, followed by Australia, and not requested at all in Canada and Germany.

### 3.3.4 Result 3: Use of Weak OAuth Flows

SSO protocol data collected in our study included information about which OAuth 2.0 flows were implemented by each RP. Table 3.6 (page 50) lists the individual OAuth 2.0 and OpenID Connect flows supported by the IdPs, and shows the number of RPs using these flows per country. Of particular concern is that a significant number of RPs used the less secure implicit flow. For example, among the RPs with Facebook

Table 3.6: OAuth 2.0 and related OpenID Connect (OIDC) flows used by RPs per country. Hybrid flows represent multi-valued response types described in Sec. 2.3.1.

| IdP | OAuth 2.0 / OIDC Flow | Response Type | Australia | | | Canada | | | Germany | | | India | | | US | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | N | n | % | N | n | % | N | n | % | N | n | % | N | n | % |
| G | 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 |
| f | 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 |
| in | Authorization code | code | 9 | 9 | 100 | 11 | 11 | 100 | 3 | 3 | 100 | 10 | 10 | 100 | 11 | 11 | 100 |

G Google, f Facebook, Ⓐ Apple, in LinkedIn
% = percentage of RPs using a given flow (i.e., % = $n/N * 100$)
n = number of RPs using the given flow
N = total number of sites (among top 500) offering the IdP per country (number of RPs)
↓ denotes that the N value above applies until otherwise indicated

SSOs, between 29% (Germany) and 43% (US; India) of RPs used implicit flows. Similarly, implicit flows with Google SSO were used by between 17% (Germany) and 38% (India) of RPs. Apple and LinkedIn SSOs do not support implicit flows, thus forcing all RPs to use more secure designs. We found some cross-country differences, notably fewer RPs in Germany using less secure flows compared to RPs in other countries. This could be due to stricter data protection laws in the EU.

Use of the implicit flow by RPs is a security concern (Section 2.2.2 explains its weaknesses) since the flow involves returning access tokens in the URLs which are retained in users' browsing history and it increases the attack surface for credential leaks [46]. Leaked access tokens can be used by any bearer to access user data within the granted scopes. Users may not be aware of the risks associated with access token leaks, especially when RPs use implicit flows. The potential damage to user privacy increases when an access token's scope allows excessive access to sensitive user data. RPs (especially those marked in Table 3.3 requesting multiple non-*basic* permissions) can reduce the attack surface by requesting minimum access and using secure flows.

### 3.3.5   Discussion

We now extend our analysis of user privacy in SSO login systems beyond the top 500 sites and discuss privacy issues in the broader OAuth-based SSO ecosystem.

**Misuse of user data access.** OAuth-based SSO enables an RP to improve usability for users through customization based on user data attributes from an IdP. If access is granted, the RP is able to download and persist user data for further processing in the future. Some IdPs conduct reviews of RPs that request access to certain sensitive user data, providing some assurance on the RP's stated use of user data (Section 5.1.3 describes the security reviews performed by three IdPs). IdPs also provide an interface for users to revoke previously granted access to an RP, invalidating all access tokens issued to the RP. However, these measures do not prevent a rogue RP from misusing any user data already obtained. Since user data is processed on RP apps (not controlled by IdP), it is not possible for the user or the IdPs to be aware of any misuse of downloaded user data.

**Risks from vulnerable RP and IdP implementations.** Without proper security measures, even a well-intentioned RP can be vulnerable to data breaches that increase the attack surface for users and their personal information. Although using OAuth ensures that the user's IdP account passwords are safe from an attack on RP, leaked access tokens can equally cause damage by allowing attackers to access user data [76]. For example, attackers have exploited a vulnerable RP site through an SQL injection attack to steal the OAuth access tokens of GitHub users [19]. Attackers have also exploited vulnerabilities in IdP SSO platforms; e.g., in 2018, attackers exploited bugs related to Facebook's "View as" profile option to obtain OAuth access tokens and compromise over 50 million Facebook user accounts [44]. It can be challenging for users to keep track of such attacks on RPs and IdPs, and understanding the implications requires a mental model of the SSO system that many users lack [101].

**Challenges related to managing SSO accounts.** SSO login systems enable RPs to identify users through their IdP accounts. This design involves inherent challenge

for SSO users if SSO accounts are decommissioned/deactivated by RPs or IdPs (e.g., after a period of account inactivity). If a user loses access to their IdP account or has de-linked their IdP account from an RP, it may not be possible for the user to later identify themselves to, and correspond with, the RP (e.g., to demand deletion of personal data).

On a given RP site, users might not remember existing accounts and might create multiple accounts (with different SSO or non-SSO login options). If the RP identifies users based on common user identifiers such as name or email address, then the RP can indicate the existing account to the user and offer to merge duplicate accounts. However, if user identification is based on OAuth access tokens or OIDC ID tokens from different IdPs (i.e., involving unique user identifiers), then the RP might create multiple accounts per user, including those who might not remember having previous accounts. This could lead to users granting RPs access to multiple IdP accounts and revealing more personal information compared to a single account. As a potential solution, IdPs could offer UIs that help users to view and manage linked RP accounts, and also to revoke previously issued access tokens.

To the best of our knowledge, our work in this section offers the first in-depth analysis of OAuth-based SSO with a primary focus on privacy as opposed to security. The privacy issues identified during our analysis motivated us to explore possible solutions to improve user privacy in SSO login systems. In Chapter 4, we present a browser extension tool that aims to inform users about the privacy consequences of choosing individual SSO login options across different RP sites and countries.

## 3.4   Analysis of Client-Side SSO Implementation Patterns

In this section, we empirically analyze real-world RP implementations of OAuth and OIDC protocols. RP implementations of OAuth and OIDC protocols differ significantly across RP sites, and can include variations within an RP site for the implementation of individual IdP login options. For example, some RPs use an IdP-provided SDK to exchange protocol messages with the IdP while others use custom code to manage the OAuth flows (either JavaScript code or through backend RP server code

such as Java or PHP). These variations complicate automated security and privacy analysis of OAuth systems. Understanding these variations is important for developing tools that address the privacy issues discussed in Section 3.3.

We examined the implementations of RP authorization (in OAuth 2.0) and authentication (in OIDC) requests[5] in the top 500 sites of the Tranco [61] list. We focused our empirical analysis on the RPs that implemented login with Facebook, Google, and Apple, these being the most common IdPs [70]. We found 153 RPs that used SSO services of these IdPs, and identified four client-side code patterns for RP implementations of the SSO requests: 36 HTML-based; 56 JavaScript-based; 44 IdP SDK-based; and 17 using a combination of the three patterns. These results are based on our analysis of the 500 websites in May 2023.

Authorization requests are triggered by the user, typically by clicking a button or link from the RP site. OAuth 2.0 authorization requests (Section 2.2.1) redirect users to the IdP with several protocol parameters relevant to security and privacy analysis. Identifying recurring RP code patterns can be used to guide the design of automated tools (such as the one we present in Chapter 4) as they help build coverage requirements. Next we describe the four patterns identified through our empirical analysis. We also discuss approaches for automatically extracting protocol data from sites in each pattern.

### 3.4.1   Pattern 1: HTML-Based SSO

In this code pattern, the SSO requests are embedded directly into the SSO-related HTML elements of the RP webpage. An example of this pattern is shown in Listing 2. When the user selects an SSO login option, an SSO request is sent to the link in the element's `href` attribute. In most RP sites, we observed that this link leads to the RP server code which responded with an HTTP 302 code to redirect the user agent (i.e., the user's browser) to the IdP endpoint. A small number of RPs hardcoded the IdP link and the OAuth 2.0 parameters directly in HTML, reusing the parameter values across multiple requests; to avoid vulnerabilities, the OAuth 2.0 `state` parameter must not be reused across requests (it should be a non-guessable nonce, to protect

---

[5]For simplicity, we refer to requests in OAuth 2.0/OIDC as SSO requests.

```
<!-- HTML code pattern -->
<div class="sso-logins">
  <a id="sso-google" href="https://example.com/sso-google">
    <div>Sign in with Google</div>
  </a>
</div>
```

Listing 2: Client-side implementation of an SSO request in HTML code.

against CSRF attacks [53]).

In our dataset of 153 RP implementations, we found 36 RPs with SSO requests in the HTML code. One RP, `ok.ru`, directly included the IdP endpoint in HTML; all other RPs implemented SSO requests through redirects from their backend servers. These requests can be extracted by finding and sending GET requests to the RP. For each valid request, the RP server returns a response to redirect the user agent to the IdP endpoint. The OAuth 2.0 protocol parameters are included in the redirection URL as query strings. We distinguished requests to endpoints of each targeted IdP using a set of URLs associated with each IdP.

We identified RP implementations with this code pattern by scanning the HTML of RP login pages for `href` and `form` tags, including endpoints to known RP and IdP servers.

### 3.4.2 Pattern 2: JavaScript-Based SSO

In this code pattern, DOM events such as button clicks on an RP login page trigger RP JavaScript code and generate SSO requests. Listing 3 gives an example. We found that most RPs send the SSO requests to their backend servers before redirecting to the associated IdP endpoints. This design is useful when RPs want to initialize and maintain per-user state information (separate from the OAuth `state` parameter) in their backend servers. If RP functionality is fully front-end implemented, this implies that the authorization request triggered by the RP is also implemented entirely in

```
<!-- JavaScript code pattern -->
<div class="sso-logins">
  <button id="sso-fb" value="Login with Facebook"
  onclick="sso()"></button>
</div>
<script> function sso() {
  req = new XMLHttpRequest();
  req.open("POST", "https://example.com/sso")
  req.send('ssoWith=facebook');
 }
</script>
```

Listing 3: Client-side code sample of SSO implementation in JavaScript.

JavaScript with no backend server communications. The OAuth 2.0 `state` parameter (returned by the IdP after login) allows RPs' callback code to link the returned IdP responses to the corresponding SSO requests. Scanning implementations of this pattern by analyzing (but not executing) loaded HTML page elements is challenging as RP JavaScript code often includes parameters such as `scope` evaluated only during runtime.

We found that 56 of 153 RPs sent the SSO requests from JavaScript code. Although we did not exhaustively search every RP script (from code visible at the client-end), we found many sites that did not include authorization parameters in JavaScript code. Instead the parameters are located in RP server responses to dynamically constructed HTTP requests. In these implementations, extracting information (including protocol parameters) related to authorization requests may not be possible solely by searching RP code visible at the client (e.g., searching for pattern matches to static strings); rather it might require dynamically executing RP code to simulate user clicks. Although this might be acceptable in tools targeted for researchers, dynamically triggering RP code is not suitable in tools for end users as it may involve opening new browser windows and increase latency.

To identify RPs with this code pattern, we monitored the network traffic after manually clicking each SSO login button on the RP's login page. If a request was sent to the RP server (and if the link was not embedded in HTML), we searched the JavaScript in RP HTML pages for traces of the request URL. This pattern can also be identified by searching for listener functions (e.g., specified in `onclick` attributes) that are triggered when an SSO-related element is clicked.

### 3.4.3   Pattern 3: IdP's SDK-Based SSO

Some IdPs provide software development kits (SDKs) for RPs to integrate their SSO services. RP sites use these SDKs by importing the SDK library to manage SSO requests and responses with the IdP. An example that uses the Facebook Login SDK is shown in Listing 4. Although these libraries are designed to simplify integration with IdP services, the SDKs often make implicit security assumptions that might not be well understood by RP developers, leading to security bugs in RP implementations [107]. These libraries also contain functions that allow the IdP to provide a consistent user experience across different RP and IdP sites. For example, the Google Sign-In library for JavaScript apps [48] offers the "One Tap" feature which allows RP landing pages to include Google's sign-in prompt for asking the user to log in using Google.

We found that 44 of 153 RPs used IdP SDKs to manage SSO requests. Unlike the other code patterns, all these SDK-using sites send their requests directly to the IdP. Therefore, the SSO request parameters (specified as arguments to the SDK functions) are available in RP's JavaScript code. Analyzing the RP's client-side code could help to identify SDK function calls. Searching and extracting these parameters might be simple if the arguments are included directly in the function calls. However, if the arguments are formed by combining other variables, extracting them might require more advanced methods such as data-flow analysis.

These implementations can be identified by searching for the presence of IdP libraries which are typically imported into HTML using `script` tags.

**IdP SDK privacy concerns.** When a user visits an RP site that imports multiple

```html
<!-- SDK code pattern -->
<div class="sso-logins">
  <button id="sso-fb" value="Login with Facebook"
  onclick="sso()"></button>
  <script> function ssoFB() {
      FB.login(function(response) {
        // handler function for IdP response
      }, {scope: 'user_friends, user_likes'});
    }
  </script>
</div>
<script src="https://connect.facebook.net/en_US/sdk.js">
</script>
```

Listing 4: Client-side code sample of SSO implementation using Facebook SDK.

IdP SDK scripts (typically embedded in the landing page), a request is made to each IdP to download its SDK. This allows the IdPs to track a user's RP visits, including users who might hope to avoid IdP tracking of their browsing activity by choosing non-SSO login options. In the other code patterns (where IdP scripts are absent), only the user's chosen IdP learns about their RP visits, instead of multiple IdPs. Section 4.4 provides further discussion on metadata exposure to IdPs through third-party scripts.

### 3.4.4   Pattern 4: Mixed SSO

We found 17 RPs that implemented the SSO requests using more than one pattern. We observed that etsy.com implemented SSO requests to Google and Facebook using the respective IdP SDKs. Although Apple offers its own SDK [4], the site had implemented SSO requests to Apple by including the IdP URL directly in its HTML code.We are not sure why sites implemented certain SSO options using different

patterns. Perhaps some developers (adding a new IdP option) simply prefer a different code pattern than colleagues who implemented previous options.

## 3.5  Summary

In this chapter, we presented empirical studies related to the privacy of SSO login systems. Our analysis of SSO platforms offered by four popular identity providers focused on the user data attributes and OAuth 2.0 flows they make available for use by relying party implementations. We presented a categorization of the IdP data attributes across the four IdPs to help with privacy analysis of real-world RPs that support SSO login with these IdPs. In addition, we presented the OAuthScope web tool to automatically scan websites and extract OAuth-related protocol parameters from RP and IdP web implementations.

Using our categorization of IdP attributes and OAuthScope, we presented an empirical analysis of the user data attributes requested by RPs in the top 500 websites across five countries. Our results reveal that RPs request different categories and amounts of personal data from individual SSO login options, often with at least one choice undeniably more privacy-intrusive. In addition, we find that users in Germany—where stricter privacy laws apply—are offered fewer or no SSO login options compared to other countries including US and Canada. We also find that privacy-friendly login choices tend to be listed last, suggesting a possible deceptive pattern favoring options that release more user data to RP sites. These privacy choices (and their privacy implications) remain highly invisible to users. Based on our analysis, we discuss privacy and security implications for users of web SSO systems.

We presented an analysis of RP implementations of OAuth-based SSO login in the top 500 websites. We identify and describe four code patterns used by RP client-side software to implement SSO login using OAuth protocols. Our findings guide and inform possible end user tools that could address the privacy issues raised in this section. In the following chapter, informed by the client-side code patterns, we present a browser extension tool to extract and display to users information about the privacy of SSO login options.

# Chapter 4

# Improving Permission Transparency of SSO Systems

To increase the transparency of data collection practices in web SSO systems, we built SSOPrivateEye (SPEye), a browser extension tool that informs users about the privacy consequences of using SSO to log into RP sites. Informed by the analysis of RP client-side implementations (Section 3.4), we designed SPEye to extract and display comparative information about the user data accessed by RP sites through SSO login options. This enables users to compare the privacy of available SSO choices and make informed login decisions.

In this chapter, we explain the privacy issues that motivate SPEye (Section 4.1), we present SPEye's design (Section 4.2.1), and we discuss the implementation of its two workflow modes (Section 4.2.3) to display permission-related information on RP and IdP login pages. We also discuss privacy issues not directly addressed by SPEye, limitations, and future research directions (Section 4.4).

## 4.1  Motivation

Our work aims to address SSO design-related privacy issues associated with RP requests to access IdP user data. In the discussion below, we first describe the typical UI design used in SSO login procedures to obtain consent from users about permission requests. We then highlight motivating privacy issues in this design.

As observed in recent studies (e.g., [58, 70, 109]), many RPs offer multiple SSO options to support users across popular IdPs, including Facebook, Google, and Apple. User data accessed through OAuth APIs is useful for RPs wanting to give a more personalized user experience and offer extended functionality. For example, an RP site might autofill forms using personal information from an IdP or offer tools to edit photos in a user's IdP photo library. In some cases, the requested user data might be optional, meaning that the RP site can function as designed when a user authenticates

using SSO login but denies access to a subset of requested data. In other sites, RP services such as sending email messages on behalf of the user require access to the user's IdP data (e.g., email account). This distinction between the user's IdP data that is essential and data that is optional for provisioning RP services is often unclear based on information given to inform users. We illustrate this with the sample SSO UI below.

A typical RP login form lists one or more SSO login choices along with traditional username and password fields for non-SSO (i.e., site-specific) accounts. If a user picks an SSO login, their browser agent redirects to the IdP where they are prompted to enter their IdP account credentials (if they are not already signed in with that IdP). Then, the IdP informs the user of the data access requested by the RP. Figure 4.1 (page 61) highlights the UI design in a typical SSO login flow from the RP to the individual IdPs. For the SSO choices in Figure 4.1a, the screenshots illustrate the UI prompts (see Figs. 4.1b and 4.1c) where information on permission requests is displayed. This design raises two main questions about the transparency of permission requests:

- *Is the access necessary to use the service?* Both the RP and IdP UIs lack information on whether the requested access is essential for the RP to offer its services. For example, the Google login dialog in Figure 4.1b informs users that `Rakuten.com` wants access to the user's email messages but it is not clear whether this data is required for the site to function properly. Note that it is also possible to use the site with the Facebook (or the Apple) login option which only require the user to disclose their name and email address (Figure 4.1c). The secondary prompt in the Facebook login UI (in Figure 4.1b) suggests an opt-out option for the email address. However, we find that an attempt to login without revealing the email address raises an error (Figure 4.1d) on the RP site asking the user to retry using their email address.

- *Which login option is more privacy friendly?* The second issue is the lack of visibility across the available SSO choices. The OAuth user data attributes exposed by an IdP depend on the type of user data it stores and the IdP's privacy policy for sharing user data with RPs. Users may not be aware of data requested

(a) Rakuten.com (RP) login

(b) Google (IdP) authorization form



(c) Facebook (IdP) authorization form

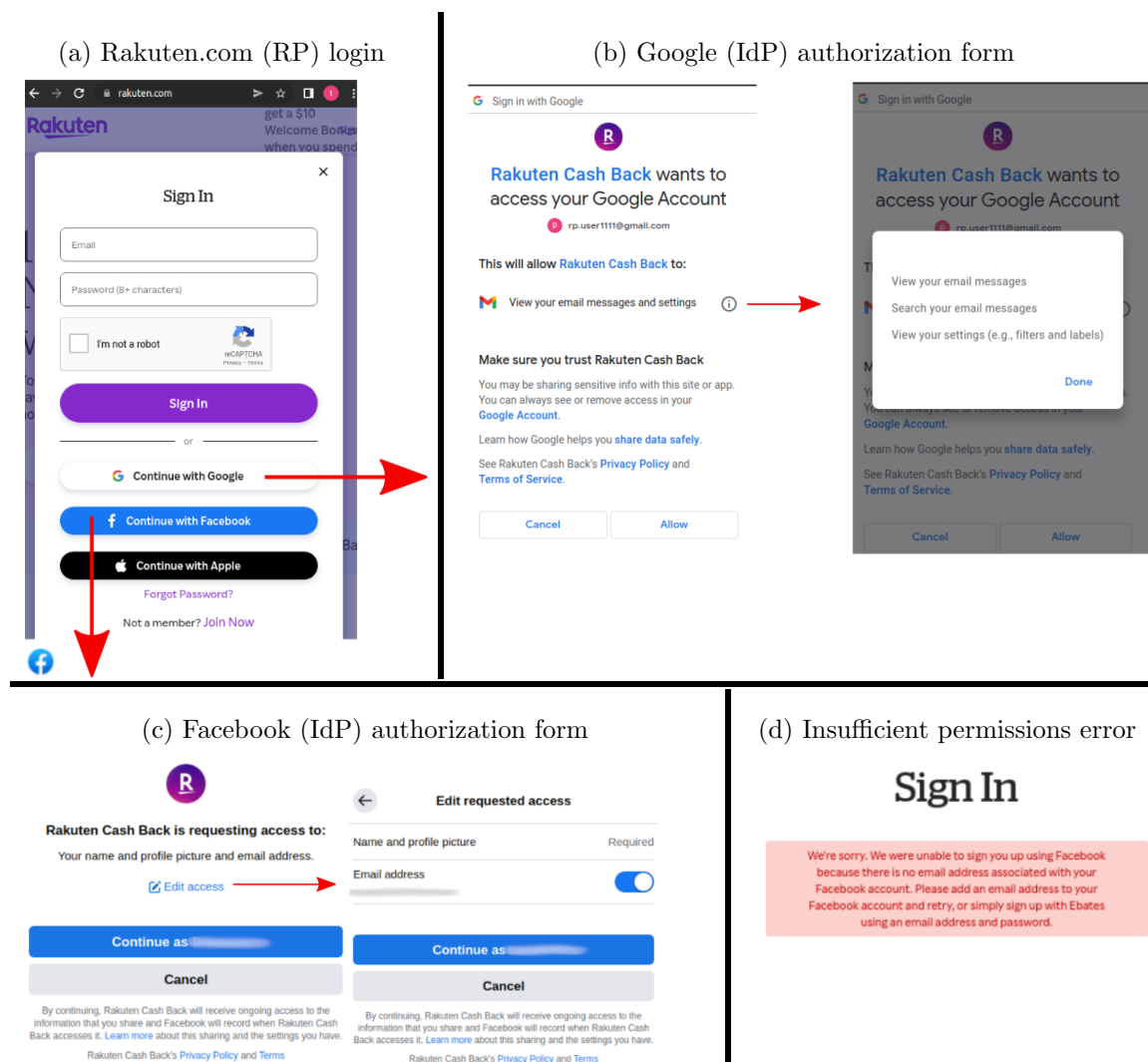(d) Insufficient permissions error



Figure 4.1: Lack of transparent information at time of (a) user login prompt. When signing into Rakuten.com with (b) Google or (c) Facebook, the user is informed about permission requests only for the selected choice (typically after the user has committed to using that SSO). In (d), an attempt to login using Facebook without revealing the email address raises an insufficient permissions error on the RP site. Note the lack of justification on why data access is needed; to access this information, a user would need to navigate to and search a secondary page such as the RP's privacy policy.

by other SSO choices as they are only informed (after completing authentication) about the permissions requested with the SSO option they select. If a user is aware that an alternate SSO choice reveals less personal data compared to other options, they might choose differently. In current SSO UI design, the user would need to login with each SSO choice and manually compare the permissions to be fully informed about the privacy choices. This is time consuming as it could require entering credentials (and completing two-factor verification where enabled) with each IdP login.

We highlight that at the key decision point of selecting an SSO option and entering their IdP credentials (which occurs in Figure 4.1a), the user lacks the knowledge necessary to make an informed comparison of options. These privacy issues can lead to users making privacy decisions without full information. Our work aims to address this through a browser extension that automatically extracts SSO login requests and reveals the permissions that would be requested by the RP after login. It provides this information without the user completing login at each IdP; and for a subset of RPs, the extension generates a comparison of IdP permissions earlier on the RP login page, thus further reducing user effort required to compare available privacy choices and make informed login decisions. Our tool addresses the second issue above by enabling users to identify which options are better aligned with their privacy preferences, thus offering better control over their privacy when using SSO to login.

Addressing the first issue is more challenging because it is difficult to predict at which point in its service workflow an RP might request extra permissions (Section 4.4 discusses these challenges in more detail). While our tool does not directly address this issue, we hope that our comparison before the login prompt increases the user's ability to make an informed choice.

## 4.2 SSO-Private-Eye (SPEye) Browser Extension Tool

SSOPrivateEye (SPEye) is our new prototype extension for the Chrome browser. It aims to inform users about the privacy implications of using SSO login on RP sites by enabling comparisons of available SSO privacy choices. The current version

of SPEye supports SSO login options with three major IdPs: Facebook, Google, and Apple. We structured SPEye such that further IdPs (that use standard SSO protocols) can be added without major structural changes. We present SPEye's design and implementation below.

### 4.2.1 Approach

We designed SPEye to scan RP/IdP login pages (after page loading) for SSO services and generate permission information for available SSO login options. We implemented SPEye as a browser extension due to three main design goals:

**G1) Real-time comparison.** RPs often update the SSO services they offer. For example, after the "Sign in with Apple" SSO platform was introduced in 2019, Apple quickly overtook Twitter as the third most popular IdP [58]. Tools that aim to inform users about SSO privacy practices should ideally use up-to-date data to take into account recent RP changes. By extracting permissions at the time of login prompt, SPEye gives users up-to-the-minute privacy data of the available SSO choices.

**G2) User-location-specific comparison.** Some RPs present different versions of their sites, e.g., to users within different countries, and request varying amounts of user information through the individual SSO login options (Section 3.3). This practice might be a result of stricter privacy laws in some regions such as the EU's GDPR and the CCPA in the US. To account for RP site differences, user tools need to consider the current location of the user. SPEye uses local scripts within the user's browser application, and therefore extracts information from the RP site relative to the user's current location.

**G3) End-user focus.** We designed SPEye to run its analysis without navigating the user away from the current RP or IdP login page to minimize disruption to their workflow. Tools that are not targeted for end-users might not be subject to this constraint. Several previous tools targeted for researchers (e.g., [45, 58, 70, 109, 113]) have instead extracted the protocol data using browser automation tools (such as

Selenium [92]) to open a scripted browser window and simulate user actions in the SSO workflow. SPEye's constraints differ for two reasons: (a) our tool is meant for end-users, so disrupting the user's view away from the current login page (e.g., by opening new windows) would limit usability, and (b) our tool does not require executing the entire SSO workflow as it only needs to extract the authorization request to a specified IdP.

**Centralized Design Alternative**

A potential alternative proposal that we did not follow due to its limitations is to design the browser extension to query results data from a central database server populated by a backend tool (e.g., a crawler that visits and analyzes a list of RP sites). As described in G1 and G2 above, RP requested permissions are frequently updated. Even with frequent crawls to provide up-to-date data: (a) changes across locations would require the crawler to rely on VPN services, which can be identified/blocked by RPs/IdPs, and (b) some RPs present different SSO features based on user data (e.g., GDPR-compliant site version for EU citizens), as described in Section 3.3.3. Therefore, privacy tools that collect and analyze RP data from a central location would need user-specific data (e.g., citizenship as above) unavailable to a crawler. SPEye's client-side design resolves these issues.

### 4.2.2 Design of Workflow Modes

Privacy information on RP permissions could be presented to users at two points in a typical SSO login workflow: (M1) on a given IdP login page *before* the user enters their IdP credentials; (M2) on the RP login page where all available login options might be listed. We consider both options:

**M1) Focused mode (IdP login page).** As the user navigates from an RP to an IdP login page (after selecting an SSO login option), SPEye extracts and displays permission information about the data access the RP intends to request from this IdP.

**M2) Comparative mode (RP login page).** For a subset of RPs, a second mode additionally offers a comparison of permissions requested by the RP for each

SSO login option while the user is still on the RP login page (before selecting any login option). In Section 3.4, we identified four RP client-side software patterns; SPEye's Comparative mode is currently available for the HTML-based software pattern to extract comparative information about privacy (extendable to security) implications of SSO services for display to users on the RP login page.

### 4.2.3 Design Architecture

The Focused mode of SPEye involves IdP login pages. These IdP login pages and their endpoints remain more consistent compared to RP login pages, whose SSO characteristics vary considerably from one site to another (as shown by the four software patterns in Section 3.4), thus leading to a more robust Focused mode implementation. Implementation of the Comparative mode (on RP login pages) is more complex as it relies on heuristics to automatically identify SSO login options and extract permission information, based on earlier code analysis from different RP implementations. For SPEye's Comparative mode implementation, we used the 36 HTML-based RP sites (Section 3.4.1) to design and test our approach. We tested SPEye on a VM running Ubuntu 22.04.2 and Chrome version 113.0.5672.

**Focused Mode Implementation**

SPEye's Focused mode extracts permission data on each of the three supported IdPs' login pages. The permissions are displayed on a per IdP basis as the user visits each given IdP login page, and before entering IdP account credentials. Compared to workflows without SPEye, this avoids the need to login with each IdP to compare permissions, and thus may also reduce the amount of information shared with RPs.

However, this introduces a usability penalty in that users need to click on an individual IdP login button to access details about that IdP. Given the challenges with implementing Comparative mode, discussed below, we viewed this as a reasonable compromise, given also that this task need only be completed when a user is debating login options for an RP (as opposed to during every login).

## Comparative Mode Implementation

To identify RP web pages with SSO login options, SPEye's Comparative mode searches the DOM (Document Object Model) for SSO elements using a list of CSS Selectors (Table B.1 in Appendix B lists these selectors). A similar heuristics-based approach is used by other tools (e.g., [114]) to identify SSO login options. We divided the 36 HTML-based RP sites into a *training set* of 21 sites and a *testing set* of 15 sites. We used the RP sites in the training set to build the list of CSS Selectors for identifying SSO login options, and to determine how to extract authorization requests for the identified login options.

**Overview of evaluation.** After implementation using the training set as target RP sites, Comparative mode identified SSO login options and extracted authorization requests in 19 of 21 RPs (90%). We then tested the Comparative mode implementation using the RPs in the testing set, and found extraction of comparative information in 12 of 15 RPs (80%). Looking across both the training and testing sets totalling 36 RP sites, all 5 misses were due to custom RP implementations that prevent automated tools such as SPEye from extracting protocol data from RP login page. We note that the Focused mode achieved 100% coverage in all RPs that follow standard OAuth (i.e., whether HTML, JS, SDK, or Mixed pattern SSO implementations described in Section 3.4). We provide further details in Section 4.3.1.

**Alternative approach to identify SSO login options.** On the RP login page (i.e., in Comparative mode), a more generalized search criterion (e.g., identifying all DOM elements with one or more attributes containing the string "facebook") could lead to wider coverage but also create false matches and unintended activity on the user's IdP profile. For instance, many RPs include non-SSO related buttons (with attributes similar to SSO buttons) such as an option to share a post to the user's IdP profile (e.g., share on Facebook). To prevent SPEye from making unintended requests, we restrict matches to known SSO login buttons. In our testing, we did not find any false matches that led to SPEye making non-SSO related requests.

**Alternative approach to extract protocol data.** Extracting protocol data from RP login pages (including the custom RP implementations described above) might be possible using dynamic approaches involving browser automation (e.g., by
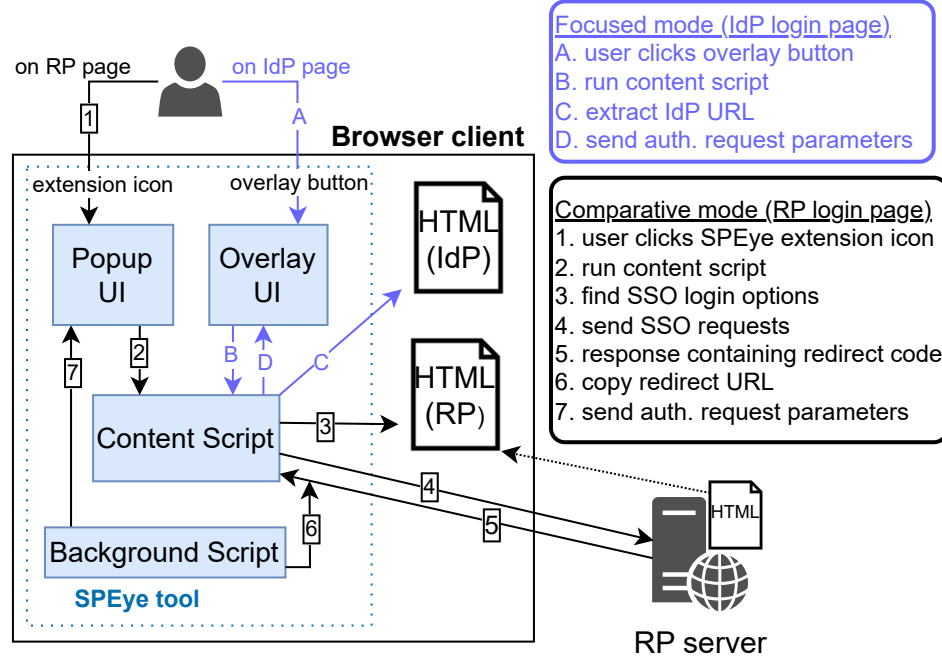
Figure 4.2: Architecture of SPEye and its two workflow modes. The Focused mode overlay button is displayed on every IdP login page, and the Comparative mode extension icon is available on a subset of RP login pages.

simulating user clicks, as in the OAuthScope tool presented in Section 3.2), but such tools introduce other challenges including undesired impact on usability, incomplete RP coverage, and increased performance overhead. With our Comparative mode, we chose to focus on the subset of RP sites where we can avoid such undesirable effects and provide reliable comparisons.

## 4.3    SPEye: Implementation and Evaluation

Figure 4.2 shows the high-level interactions between SPEye components and the RP and IdP website stack. SPEye has four main components: a) a *popup script* with *popup UI* triggered on an RP login page when a user clicks on the *SPEye extension icon* on the browser toolbar to open it; b) an *overlay UI* on the IdP login page, opened through an *overlay button* displayed by SPEye; c) a *content script*, which runs on the visited webpage when the interface is opened; and d) a *background script* monitoring browser traffic for HTTP redirection requests to IdP endpoints.

The SPEye codebase adapts portions of OAuthScope (described in Section 3.2.2) code for common tasks such as extracting OAuth protocol parameters from authorization requests, and hence the details regarding these steps are not repeated in this chapter. In total, SPEye's software components for the Focused and Comparative modes comprised 1,198 lines of code (JavaScript, CSS, and HTML).

**Overview of workflow.** When the user triggers SPEye (through the overlay button in the Focused mode or through the extension icon in the Comparative mode), the content script uses predefined search criteria (listed in Appendix B) to scan the current target webpage and identify IdP login prompts (Focused mode) and RP webpages with SSO login options (Comparative mode). For each identified SSO login option (in the Comparative mode), SPEye further scans the target webpage to isolate and trigger SSO login requests to the individual IdPs. After each login request is completed, SPEye reads the IdP responses to extract and display the requested data permissions to the user. The details of these steps are described below.

SPEye relies on the Chrome Extension APIs[1] to access target webpages and to intercept RP login requests and IdP server responses. SPEye scans the current webpage only when the extension interface is opened. This prevents SPEye from interfering with user-triggered requests during which the interface remains closed. It also eliminates unnecessary background processing of all the webpages the user visits (while the extension is active) and limits performance impact. SPEye interface triggers the following tasks (Step **A** or Step 1 in Fig. 4.2).

**1) Search RP/IdP webpage.** Based on whether the user is on a login page of an IdP (Focused mode) or an RP (Comparative mode), the following steps are executed:

**1a) Focused mode.** When the current (in-focus webpage) URL matches a known IdP authorization server, SPEye displays the overlay button (see top right in Fig. 4.3a showing SPEye's UI on page 70). The user can click (Step **A**) to view the permissions, and opt-out of certain permissions requested by the RP. Using the Chrome runtime API,[2] the interface script sends a message to the content script (Step **B**) to process

---

[1]`https://developer.chrome.com/docs/extensions/reference`
[2]`https://developer.chrome.com/docs/extensions/mv3/messaging/`

the current webpage, extracting permission information from the RP's authorization request to the IdP.

**1b) Comparative mode.** When the extension icon (see Figure 4.3b) is clicked, the content script uses CSS Selector strings to identify RP login pages based on matching SSO buttons, login forms and login URLs (Step 3 ). When a matching element is found, the script searches its attributes (e.g., `href`, `onclick`) for URLs to RP and IdP servers. In the common match case (as observed in our training dataset), the script extracts the URL from an `href` attribute and makes an `XMLHttpRequest` for each matching element (Step 4 ).

If a link is not found, the matching element could be part of an HTML form which can be identified by the presence of attribute types such as "input" and "submit". For these matches, the script searches the DOM to identify its parent form element and extracts the path to which the form will be submitted along with other form data linked to a specific SSO choice. As an example of form data observed in our dataset, selecting "Sign in with Facebook" on `fandom.com` involved a custom form attribute `value="facebook"`. SPEye's content script makes an `XMLHttpRequest` to submit the form with these custom attributes (sequentially, for each SSO login) along with other internal parameters (e.g., nonce values) in the form (Step 4 ).

`ebay.com` is an example we use to illustrate the type of typical differences we found in RP implementations. When an RP link is found in the `onclick` attribute (which specifies a trigger to a handler function within JavaScript code), SPEye's content script extracts the link and attempts to make a GET request. However, this may or may not succeed if the link is passed to a JavaScript function where it is modified before making requests. For example, choosing SSO on `ebay.com` invoked a function with the SSO link as an argument. The function then added an additional argument before sending the login request to its backend server. Since SPEye does not scan JavaScript, it is limited to sites that include all necessary parameters in the HTML.

**2) Extract authorization request parameters.** SSO design requires RPs to redirect the user to the IdP login page with authorization request parameters in the URL.

(a) Focused mode

Overlay button



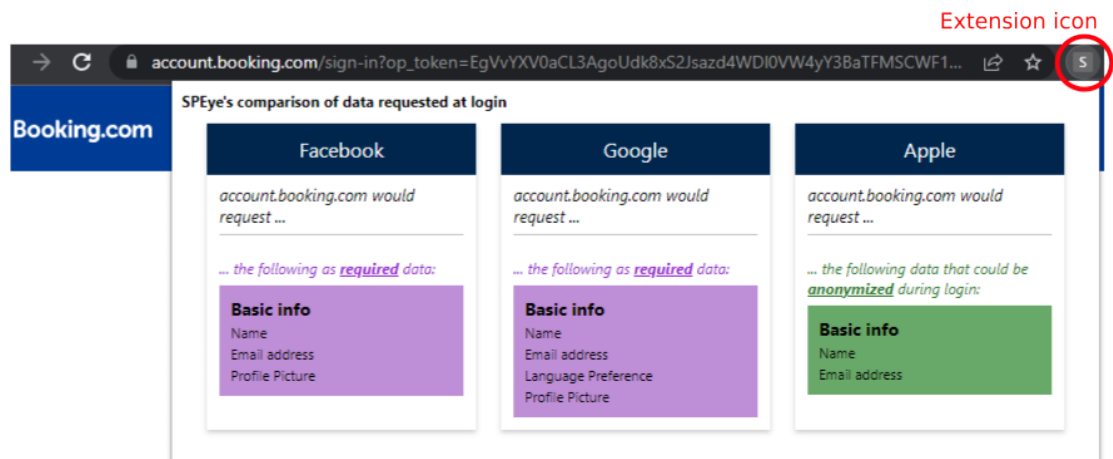(b) Comparative mode

Extension icon
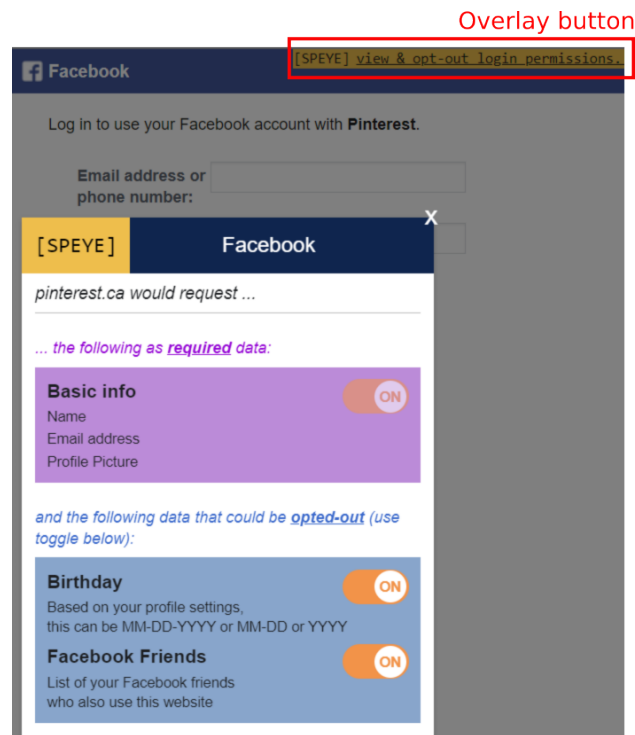


Figure 4.3: SPEye's UI for (a) Focused and (b) Comparative workflow modes showing permission information on example sites. In the case of Apple SSO, SPEye also indicates Apple's privacy feature (not shown in this image but available on Apple's IdP UI) that allows the user to anonymize their information released to the RP. To get this view of permission data without SPEye, the user must login with each SSO login option to manually collect, record, and compare the information requested.

SPEye compares each redirection URL against a set of regular expressions (included in Listing 5 in Appendix B) derived from known IdP authorization server URLs.

**2a) Focused mode.** If the user is on the IdP login page, SPEye extracts the parameters from the current IdP page without making any additional requests. SPEye's content script compares the URL of the current page (Step **C**) to match with a known IdP server. Then, the script extracts the RP domain name and the list of permissions requested by the RP from the protocol parameters in the IdP URL. The extracted results are sent to the overlay UI (Step **D**) for displaying to the user.

**2b) Comparative mode.** If the user is on the RP page, SPEye's content script automatically triggers the SSO requests identified in the RP client-side code. When the RP server accepts the request, it responds with an HTTP redirect code (Step 5). SPEye's background script uses the Chrome webRequest API[3] to observe HTTP redirects (Step 6). If the redirection URL is identified as a known IdP server, the background script copies and sends the URL to the popup script for further processing (Step 7). Although the extension's background scripts are always active (i.e., listening to registered events in the background), redirect URLs are only received by the popup script when the SPEye user interface is opened by the user. Other redirects such as user triggered SSO logins, during which the SPEye interface remains closed, are unaffected by the background script.

`theguardian.com` is an example that illustrates the type of typical variations we found across RP implementations. We observed RPs that redirect through multiple intermediate RP endpoints before sending the user agent to the IdP server. In this example, the site's backend server responded to SPEye's SSO requests by returning a redirect to an intermediate (RP) endpoint which then returned with the IdP redirection URL. In such instances, SPEye follows the sequence of redirects to observe the eventual IdP endpoint.

`nytimes.com` is another example where the RP's backend server responded to SPEye as expected for Facebook and Google, but provided a non-redirect response

---

[3]`https://developer.chrome.com/docs/extensions/reference/webRequest/`

for Apple. Although the server responses were different, the three SSO options were implemented identically by the site's frontend. As a result, SPEye's comparison on this site (and on another site with similar implementation) is limited to two of the three SSO login options (only in the Comparative mode).

**3) Display permission information.** SPEye aims to present the extracted permission information to the user in a consistent interface across different RP sites and IdP SSO platforms. We reviewed IdP documentation to map each OAuth 2.0 `scope` parameter value to descriptive text explaining the associated permissions. Where scope values from different IdPs refer to the same category of user information (e.g., user's IdP profile info), we modify the text provided by the IdPs to provide a consistent description.

**3a) Focused mode.** On the IdP login page, SPEye users can view permissions requested by the RP as illustrated in Figure 4.3a. This option is available before users enter their credentials, thus removing the effort of having to complete login with the IdP (along with multi-factor authentication, if applicable) before being able to view the RP requested permissions with that IdP. This overlay UI also contains toggles to opt-out of certain permissions deemed optional by the IdP platform. When an opt-out is requested by the user, SPEye sends the IdP a new authorization request without the opted-out permissions. The IdP might override the modified request and instead display a pre-registered set of permissions. The user may still be able to opt-out of some of these permissions through the IdP UI.

**3b) Comparative mode.** On the RP login page, Comparative mode (Fig. 4.3b) shows a privacy comparison of SSO login options. The background script extracts the permissions from authorization requests (similar to the Focused mode above). SPEye generates and displays a comparative summary of permission descriptions using the identified requests.

Independent of the Comparative mode, SPEye users can visit an IdP login page to view the Focused mode output (in RPs that follow standard OAuth 2.0).

### 4.3.1 SPEye's Comparative Mode Misses

In the Comparative mode, SPEye failed to extract comparative data in five of 36 RPs, as mentioned in Section 4.2.3. (SPEye's Focused mode still covers these cases as expected.) We provide details on two different RPs to illustrate the Comparative mode misses:

`unsplash.com`. Although SPEye's Comparative mode extracted and sent the SSO request to the correct URL, the RP server returned a non-redirect response. Closer inspection of the site's HTML page reveals `<meta>` tags containing custom CSRF token parameters [78]. Extracting these is beyond scope in the current version of SPEye as they might be added to the login requests from the RP's JavaScript code. However, to confirm our findings, we modified SPEye to resubmit the request that included these parameters, and the RP server responded with a redirect request for the selected choice.

`themeforest.net`. SPEye's requests to this RP's server were initially blocked. To identify why, we compared a request sent by SPEye with a similar request triggered through the SSO button on the RP page. The requests were identical except for two HTTP request headers (`sec-fetch-dest` and `sec-fetch-mode`) typically added automatically by the browser. The values in these headers are used to indicate to the server that the request was initiated by a user clicking a link. SPEye cannot modify these parameters as they are read-only,[4] and managed by the browser.

### 4.3.2 IdP Deviation from Standard Protocol

SPEye extracts RP requested permissions by inspecting the RP's authorization requests, as per standard OAuth 2.0 protocol. In February 2023, we observed that after manually signing in using Facebook to certain popular RP sites (e.g., `airbnb.com`, `cbc.ca`), the IdP prompt changed to show only the basic permissions (i.e., name and email address) and, contrary to expectations from standard OAuth 2.0, non-basic permissions such as photos were not displayed or granted.

To measure the extent of such anomalies, we first identified 133 RPs with Facebook SSO in the Tranco top 1,000 sites and compared permissions in the authorization

---

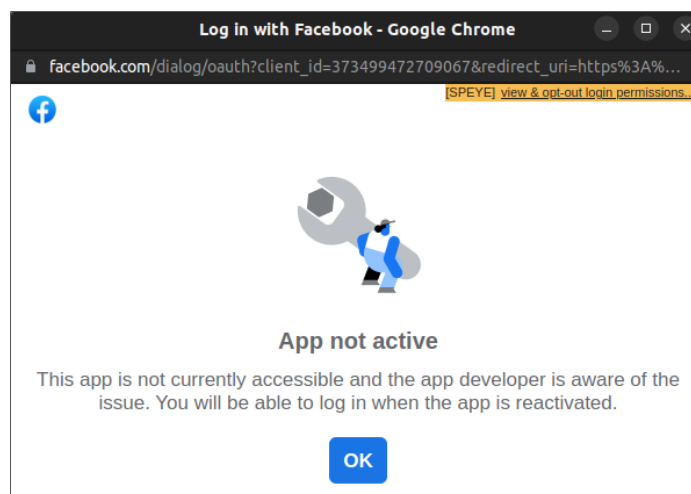[4]`https://developer.mozilla.org/en-US/docs/Web/API/Request/mode`

Figure 4.4: A temporarily blocked message displayed on Facebook prompt when attempting to log in to `myspace.com` using Facebook.

requests made by each RP to the permissions displayed in the IdP prompt (shown after login). For 113 sites (85%), we found that the permissions in the RP authorization requests were consistent with the permissions displayed on the IdP login prompt. For 14 sites (10%), Facebook did not show a login prompt, instead displaying an error stating that users can "log in when the app is reactivated"; the screenshot in Figure 4.4 provides an example. For the remaining 6 sites (5%), the non-basic permissions included in the RP's request were pruned by the IdP, and hence not displayed or granted. These sites included `tripadvisor.com`, `theatlantic.com`, `coursera.org`, `airbnb.com`, `cbc.ca`, and `aliexpress.com`.

These deviations could be due to recent changes to Facebook's SSO policy [20] that restrict certain sites from access to non-basic data (i.e., permissions other than name and email address). It is possible that the protocol deviations we observed in these RPs are temporary until the RP apps are reviewed by Facebook under its new SSO permissions policy.

The aim of SPEye is to extract and display permissions as requested by the RP, assuming that the OAuth protocol is followed. If an IdP deviates from the protocol (as in Facebook above) by modifying the set of permissions requested by the RP, the information displayed by SPEye reflects what a user should see at the IdP interface in a compliant login flow. However, as explained above, this may differ from what a

non-compliant IdP actually shows.

### 4.3.3   User Interface

SPEye's output (illustrated in Fig. 4.3, page 70) enables users to compare permissions before they commit to a login choice. For each SSO login, the UI lists the information requested by the RP through the IdP's user data APIs. For a subset of attributes, SPEye also describes what data (amount or type) will be released to the RP if the associated SSO choice is selected. For example, if the user's IdP profile privacy settings is set to only reveal their age, the birthday API might only reveal the user's year of birth to the RP. By enabling comparison across SSO choices, SPEye provides users with information to make informed login decisions aligned with their privacy preferences. This design can also inform users about specific privacy features. For example, Apple offers its users the option to hide their email address (as indicated for the Apple SSO option in Fig. 4.3b, page 70) by generating a unique per-RP email address which relays emails between the RP and the user's email inbox (Section 3.1.4).

### 4.4   Discussion

Here we discuss other privacy considerations relating to SSO and how these complement SPEye's goals.

**Metadata exposure to IdPs.** SSO protocols involve RP requests to the IdP to get access tokens and SDK scripts (in the SDK pattern RPs, described in Section 3.4.3). Consequently, the IdP can see the user's visit to the RP site (through OAuth redirect URI or HTTP referrer), including metadata such as timestamps for the visit, which raises privacy concerns relating to IdP tracking of the user's browsing activity [112].

To inform users about SSO permissions with each SSO login option, SPEye's Comparative mode makes a request to each IdP (described in Section 4.3). This raises a privacy concern as these requests reveal site visit metadata to each of three IdPs, as opposed to the one IdP the user might select to login. Users may view this privacy trade-off as acceptable since it is likely a one-time disclosure as we expect SPEye to be primarily used during signup when selecting a login on a new RP site,

not for each subsequent RP visit (which would be a bigger concern). Alternatively, users can limit exposure to selected IdPs by using the Focused mode.

We note that the site visit metadata can also be acquired by IdPs through embedded third-party services. Websites (including RP sites) typically embed third-party scripts for advertising, analytics, or social network plugins [69]. A study across the top 1-million sites found that the most popular third-party services were Google and Facebook services [27]. For example, `google-analytics.com` itself was found in about 65% of the sites, not counting other Google-owned services. The third-party prevalence of Google and Facebook, along with RP requests (in the SDK pattern) to get SDKs from IdPs including Apple, mean that the above privacy issue relating to tracking of browsing activity by third parties exists regardless of SPEye's IdP requests.

**Permissions requested after first login.** SPEye enables comparison of permissions requested by an RP at the initial IdP login prompt where permissions are not usually visible to users, but of course cannot include data the RP might request at a later time (e.g., after first login). For example, an RP that requests only the user's name from the IdP with whom the user has completed login might at a later point (in an RP UI, as Fig. 4.1d on page 61) ask the user to enter their email address. A privacy-unfriendly RP might request additional permissions gradually at various points in its workflow (escaping SPEye), perhaps across many different interactions with the RP, leading to a form of 'permission creep' (a progressive increase in permissions) as the user performs different tasks. The UI workflow for these extra requests explicitly prompts the user to grant or deny the additional permissions 'just-in-time', without having to complete login or make a new login choice.

IdP guidelines (e.g., Facebook's [33]) suggest RPs to request additional permissions in the context where the data is needed. Although this design aims to give users better control over what data is requested and how it might be used, it withholds from users overall context about how much other data is already being collected or how this new permission fits into this overall picture. In a type of deceptive pattern [74], an RP could request minimal information during the initial login (and thus appear privacy-friendly in SPEye) and then gradually request additional permissions, hoping

that the user will not realize the extent of data collected, or will feel compelled to stick with their decision (e.g., due to difficulty in transferring an existing RP user-account to a different IdP) having invested time and effort into it.

SPEye provides comparative information necessary to make an informed initial login choice and offers a model of the type of information that helps users make informed decisions. Future research may explore how to keep users informed throughout their relationship with an RP.

**RP handling of opted-out permissions.** Except for basic information such as the user's name and profile picture, major IdPs (e.g., Facebook) classify other permissions as 'optional' and allow users to decline such permissions requested by an RP. The apparent intent by the IdPs is to give users control over their data. However, as described in Section 4.1 and Figure 4.1d, some RPs use tactics to persuade users to grant permissions despite being labelled as optional on the IdP's login prompts (if users want to use the given IdP). Although RPs may have legitimate uses for the data, such workarounds suggest the potential presence of a misleading or deceptive pattern (cf. above) in RP implementations, bypassing any intended IdP privacy controls. SPEye is unable to detect such RP workarounds because they occur outside of the SSO login interaction. Conflicting cues in existing SSO UIs, such as whether access to specific user data is mandatory (or not) for successful login at the RP site, are visible in Figs. 4.1c and 4.1d. These conflicting cues may confuse users and consequently reduce user trust in the privacy of SSO systems. Tools like SPEye may help users better manage access control to data items by offering opt-out options at the start of IdP login workflows.

## 4.5   Limitations and Future Work

Here we report limitations of the current version of SPEye and provide ideas for future improvements. We also discuss a potential security extension to our work.

**User data usage by RP.** SPEye's extraction shows what data an RP requests, but cannot determine how the requested data will be used by the RP. Users might want

to weigh any asserted purposes before granting access to personal data. While an RP's privacy policy might offer information on intended use, it is difficult to ascertain actual usage (or even intent) without access to RP systems or developers.

**Expanding IdP support in SPEye.** SPEye currently covers the top three IdPs supported by RPs. RPs might offer other IdP login options that request different data. During our analysis of the 153 SSO-supporting RPs (in the top 500 sites; Section 3.4), we found 25 unique IdPs. At least 81 OAuth providers exist according to a Wikipedia IdP list [111]. SPEye's code is modular such that expanding the list of IdPs supported can be done with relatively little redesign, e.g., adding to a list of known IdPs by including data relevant to the new IdP such as authorization endpoints and permission information lists.

**SSO privacy in mobile apps.** The OAuth 2.0 framework [53] originally targeted web apps and is less suited for RP implementation in mobile apps. The mobile ecosystem is sufficiently different that it has its own issues separate from web RP implementations. Platform differences in mobile OSs make it challenging to securely implement OAuth 2.0 in mobile apps. A study of mobile app RPs [18] found a number of vulnerable apps primarily caused by custom solutions for storing and delivering protocol secrets. For example, due to an absence of secure redirection mechanism (in iOS and Android) from an IdP site on a mobile browser to an RP mobile app, access tokens could not be delivered safely. Moreover, UI constraints in mobile apps require a novel approach to inform mobile users about SSO privacy consequences. Our work analyzed web SSO implementations and built SPEye to automatically extract protocol data from RPs in desktop web browsers; addressing the parallel but unique problem in the mobile ecosystem and with mobile apps as RPs is equally valuable but beyond our current scope.

**SPEye-like tool to inform security decisions.** SPEye demonstrates the feasibility of extracting SSO protocol data from RPs; it could be expanded to convey information about potential security weaknesses in addition to privacy information, and

warn users about security risks prior to committing to login decisions. Security-related information extracted using methods from previous security tools can be presented to users in an SPEye-like workflow (but now showing security rather than privacy-related info). For example, a security scan could use the data available through SPEye's existing framework to scan RP client-side code, e.g., for use of weak OAuth 2.0 implicit flows (Section 2.2.2 describes its weakness) or lack of CSRF protection (as applied through the OAuth 2.0 `state` parameter). Such an augmented version of SPEye could warn users about vulnerable RP implementations before they commit to SSO login decisions.

## 4.6 Summary

We designed and built the SPEye browser extension which extracts and displays real-time privacy information about SSO permission requests. The extension enables privacy comparisons across individual SSO login options in RP sites. SPEye has two workflow modes (Focused and Comparative) that display privacy information to users at different points of a typical SSO login workflow. The Focused mode covers 100% of web SSO systems that use standard OAuth 2.0, though it involves a usability cost of visiting individual IdP login pages. For RP sites with SSO implementations following the HTML pattern, the Comparative mode removes this usability cost by showing comparative information from the different IdP options with one click.

SPEye focused on privacy; tools such as ours that investigate and report on the privacy practices of service providers help raise public awareness of privacy issues, encouraging more transparent privacy practices. Our work on SPEye takes a step towards increasing transparency in web SSO systems by enabling users to be informed about permissions and to "Sign in with (more) privacy". In the next chapter, we explore how users make login decisions in web SSO systems, and evaluate the impact of displaying comparative information extracted by SPEye on users' login decisions.

# Chapter 5

# User Motivations in SSO Login Decisions

Web SSO systems offer users various advantages and disadvantages related to usability, privacy, and security properties [1]. In OAuth-based RP services, where users are asked to reveal personal information in exchange for using related features, users might prioritize certain factors when making login decisions. In this chapter, we explore how users choose from a list of login options, and the factors that influence their login decisions in web SSO systems. We also examine the impact of displaying comparative information about the IdP permissions on login decisions.

As sites offer numerous login options, including both SSO and password-based options [58, 70], users are asked to make decisions where each login choice might involve different user benefits and risks. For example, an RP that wants to access a user's location to offer suggestions on nearby restaurants shows a login prompt with different SSO login (e.g., with Facebook) and non-SSO login (e.g., with username and password) options, thereby expecting users to choose between privacy and usability. With little or no information provided about the available choices, informed users might prioritize based on different factors including tradeoffs between privacy and usability [26], and uninformed users might make login decisions that are against their personal preferences or interests.

Descriptions of the permissions in IdP consent dialogs tend to be vague and insufficient to adequately inform users about the access granted to RP sites [87]. The lack of clear information can lead to users making incorrect assumptions about the necessity of permissions [10], e.g., by assuming that sensitive information such as personal email messages are necessary for app functionality. Even if IdPs improved their consent dialogs, the necessity of permissions might not be fully visible because users of sites with multiple SSO login options are only shown the permissions requested with the chosen login option. This design could lead to SSO users of certain RP sites

(e.g., as in illustrated example in Fig. 4.1, page 61) to incorrectly assume that the requested permissions are mandatory for the RP site to provide its services, unless they are fully informed about the alternate privacy choices (i.e., other login options). These issues motivate us to study the factors that influence login decisions and the effects of presenting comparative information about the IdP permissions.

We conducted a study examining login decisions in sites that offer multiple SSO and non-SSO login options. Specifically, we explored the factors that influenced participants' decisions, and the impact of displaying comparative information about IdP permissions on these decisions. In an online survey, we asked 200 participants to choose their preferred login options among a list of different SSO and non-SSO login choices on a given RP site. Participants were shown screenshots of SPEye's output containing the permissions requested by the RP for each IdP login option on a given RP site. We prompted participants for their login decisions before (pre-SPEye) and after (post-SPEye) viewing these screenshots. Along with each login decision, participants were also asked to provide reasons for their decisions. We then compared the pre- and post-SPEye login decisions to test whether SPEye's output influenced participants to make privacy-informed decisions. While usability preferences and inertia (i.e., habituation) were among the dominant factors before the screenshots, many participants prioritized privacy over other factors after viewing the comparative permissions.

The contributions of this chapter can be summarized as follows.

- We present a quantitative analysis of login preferences in sites with SSO and non-SSO login options, and compare the privacy of pre- and post-SPEye login decisions.

- We present a qualitative analysis of the self-reported reasons for login decisions, identifying factors that influenced pre- and post-SPEye login decisions.

- We discuss privacy and security implications based on our analysis of participants' login decisions and mental models of the web SSO login systems.

## 5.1 Overview of IdP Platforms

In this section, we provide an overview of three major IdP platforms (Google, Facebook, and Apple), and discuss platform features and differences related to privacy and usability design. A main difference pertains to the collection of user data each IdP makes available through its APIs (see Section 3.1). In many instances, the user data attributes are based on other services offered by the IdP, for example, a calendar API for Google Calendar, or a public profile API for Facebook Profile. These APIs are designed for parsing by RP developers to implement OAuth permissions in their applications.

Here, we instead consider IdP platform differences from the viewpoint of a user on a typical RP site. We extend the earlier analysis in Section 3.1 to consider platform features beyond OAuth-specific data attributes and discuss characteristics of each platform that could impact security, privacy, and usability for users. These characteristics also help inform the design of our study to include each available login choice when asking participants to make login decisions in the RP sites used in the study.

### 5.1.1 IdP Platform Differences

The granularity and customization options of permissions vary considerably across different IdP platforms. For instance, the Facebook Login platform [30] offers the option to opt-out of a subset of permissions requested by RPs. When users login using Facebook, they are prompted with a consent dialog that allows the user to deselect a subset of RP requested permissions and proceed with the login—although the opt-out option is nested in a secondary UI triggered from the consent dialog. Some basic permissions such as Facebook profile picture and profile name cannot be opted-out (regardless of whether the RP has requested those permissions). Google's IdP UI [96] instead uses an all-or-nothing permissions model where users are forced to either approve all or reject all of the RP requested permissions.

For a small subset of permissions, Facebook Login [30] offers fine-grained access controls on the data released to RPs. For example, if an RP requests permission to manage user-administered Facebook pages on the user's behalf, the user can choose to grant the RP access to only a certain subset of all the pages they administer. But such

fine-grained controls are not offered for many sensitive Facebook permissions (such as personal photos), and are completely absent in other IdP platforms including Google.

### 5.1.2   IdP Pseudonymization of Personal Information

Apple's SSO platform [95] offers its users the option to login to an RP site using a custom name and a pseudonymous email address. As explained in Section 3.1.4, this feature allows the user to exchange email messages with an RP site without revealing their real email address to the RP. This feature might be preferable to privacy-conscious users with an active Apple account, but requires trusting Apple with any personal information that might be revealed in emails from/to the RP. Google and Facebook do not support similar privacy services.

### 5.1.3   RP App Reviews by IdPs

Many IdPs categorize their user data attributes into a basic set for user profile data (e.g., name, email address) that might be less private and/or necessary to provision RP services, and a separate privacy-sensitive set for more personal user data such as user photos and email messages. RP sites that depend on these privacy-sensitive APIs are typically required to go through the IdPs' app review process before requesting access to user data APIs.

**Google SSO.** Google's RP verification [47] is divided into three levels of assessments depending on the user data attributes a given RP wants to access. All RPs undergo a basic "brand verification" process which involves IdP checks of the information provided by the RP for inclusion in the IdP consent dialog (e.g., RP domain name, RP privacy policy URL). RPs that request privacy-sensitive data APIs (e.g., Google Calendar API) must accept Google's User Data Policy which requires the RP to agree to certain privacy practices such as requesting the minimum set of permissions necessary for services. RPs that request further restricted APIs (e.g., Gmail APIs) must additionally go through a *security assessment* certification based on Google's App Defense Alliance[1] testing program. This program was launched in

---

[1]https://appdefensealliance.dev/casa

2019 to detect security threats in Google Play Store apps, and has since been expanded to cover third-party services including RP sites that access certain restricted user data APIs [21].

***Facebook SSO.*** If the RP requests permissions beyond the basic public profile or email address APIs, Facebook's App review [34] requires RP developers to get approval before accessing the APIs. This review involves a basic brand verification check for RP's privacy policy URL (similar to Google above). The verification is also supposed to check whether the RP site uses the requested user data APIs to provision one or more of its services. During the review process, developers are asked to explain why their app requires access to the requested APIs, and to submit video demos of app features that utilise the requested data. This aims to limit RPs from accessing user data without any apparent functionality, but RPs with the primary aim to track users might introduce superficial functions exclusively designed to pass the IdP review processes.

***Apple SSO.*** Requirements specified by *Sign in with Apple* focus on user experience, branding, and Apple's guidelines for data collection [6]. In addition, as mentioned in Section 3.1.4, Apple mandates RPs on its App Store (e.g., apps targeted for iOS or macOS devices) that offer an SSO login option with an IdP other than Apple to also offer users an option to login using Apple [5]. Beyond these requirements, RPs are also subject to Apple's App Reviews[2] and security checks for general apps (i.e., not involving SSO) on the Apple App Store. The lack of other extensive SSO-specific reviews might be because Apple does not offer RPs API access to user data beyond the user's name and email address (Section 3.1.4).

The IdP app reviews (described above) aim to prevent misuse of user data by RPs for purposes other than providing RP services, for example, by checking if a requested API is necessary for any relevant RP app functionality. When evaluating the necessity of the requested APIs, the scope of the IdP reviews are limited to only that IdP's login option and RP functionality directly related to its APIs. But this evaluation may not be effective as RPs can request varying amounts of user data across different

---

[2]https://developer.apple.com/app-store/review/

SSO login options [70], meaning that the user data APIs claimed as necessary with one IdP login choice might differ from those marked as necessary with other IdP login options on the same RP.

## 5.2 Study Methodology

We conducted an anonymous survey using a between-subjects design to investigate how users choose from a list of different SSO and non-SSO login options. This study also aims to understand the impact of enabling users to compare IdP permissions on login decisions. We implemented the survey using the Qualtrics[3] online survey platform and collected responses in January – February 2023.

### 5.2.1 Survey Design

The survey was built around the login options in four popular RP sites (`Airbnb.ca`, `CBC.ca`, `NYTimes.com`, and `Rakuten.com`). Our intention was not to assess these specific RPs but to represent a variety of requested login permissions and a range of privacy choices. For example, `Rakuten.com` requested more privacy-intrusive permissions with Google SSO compared to other options, whereas `CBC.ca` requested most permissions with Facebook SSO. All four RPs offered exactly three SSO login options (Google, Facebook, and Apple) and a non-SSO login option using an email address; `Airbnb.ca` users could also log in using their phone number.

To inform participants about the privacy of each IdP login option, we used screenshots of SPEye's output on each listed IdP login option which includes the individual permissions requested by the RP. Figure 5.1 shows SPEye's UI (presented in Section 4.3.3) displaying the permissions requested by an example RP site with Facebook login. The permissions are highlighted in two color-coded blocks to differentiate between (i) mandatory permissions required for login with a given IdP and (ii) permissions that can be opted-out (as designated by the individual IdPs). We presented a screenshot of this interface for each IdP login option on a given RP site to enable a comparison of the SSO permissions across the supported IdPs.

---

[3]`https://www.qualtrics.com`
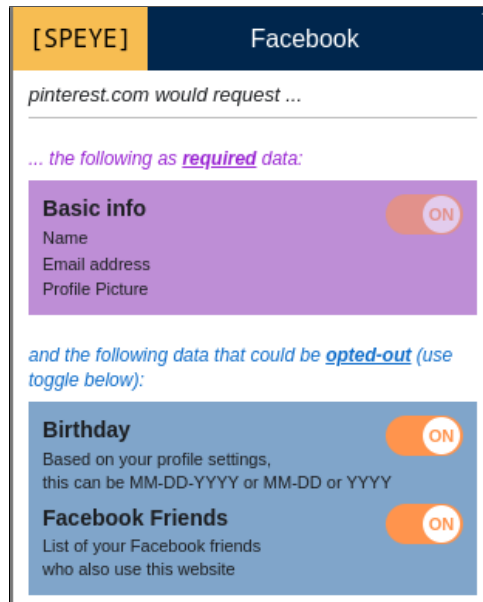
Figure 5.1: SPEye interface showing the permissions requested by an example RP site through Facebook SSO login.

We designed the survey to show participants the usual login user interface of an RP site and prompt for an initial login choice (Pre-SPEye) along with an explanation for their choice. Next, participants saw our SPEye interface facilitating comparison of permissions and they again entered a login choice (Post-SPEye) and explanation. To preserve the validity of responses, participants could not navigate back to earlier survey pages.

To avoid priming effects, each participant was shown only one of the four RP sites. We configured our survey recruitment to allocate exactly 50 participants per RP, for a total of 200 participants. The average completion time for the survey was 6 minutes and 23 seconds, and participants were paid £1.50.

The survey (shown in Appendix A) was structured as follows:

1. Basic demographic questions, and basic questions on web login experience.

2. Pre-SPEye login: we showed each participant a screenshot of the usual login prompt of their assigned RP and prompted for their preferred login option (among both SSO and non-SSO) using a multiple choice question, followed by an optional open-ended question for the rationale behind their choice. By

showing the usual login prompt of the RP, we prompt participants to choose based on the information usually available to users at the time of login.

3. Post-SPEye login: we presented screenshots of SPEye's output on the individual IdP login pages, and again prompted the participant for a login choice, with an optional open-ended question for their rationale. This time the login options in the question included additional privacy choices made apparent by SPEye's output (e.g., Apple's email relay service described in Section 5.1.2). In essence, we explore what happens when additional privacy-related information is made visible and easily comparable.

4. We asked 5-point Likert scale questions with statements related to their login choice and privacy preferences.

5. We asked participants to indicate if they had an account with any of the popular IdPs (including Apple, Facebook, and Google), and if so, how often they used applications and/or services (related to both SSO and non-SSO usage) connected to their IdP accounts.

### 5.2.2 Participants

We recruited 202 participants in the US and Canada through the Prolific[4] recruitment platform. Responses from two of the participants were excluded from the analysis because they completed the survey in less than two minutes and selected the same option for each Likert scale question, leaving us with 200 valid responses. There were 97 participants who self-identified as men, 97 as women, 3 as non-binary, one fluid, one demigirl, and one preferred not to answer. The self-reported ages (excluding a participant who preferred not to answer) ranged from 19 to 80 ($M = 37.2$, $SD = 13.9$). The majority of participants were university graduates or students currently enrolled in a university degree; 43% with a Bachelor's degree, and 21% with Master's or PhD degrees.

---

[4]https://www.prolific.co/

Table 5.1: Participant demographics and web SSO usage

| Demographics | | |
| --- | --- | --- |
| Gender ($N = 199$) | Man | 49% |
| | Woman | 49% |
| | Other | 2% |
| Age ($N = 199$) | 18-19 | 2% |
| | 20-29 | 33% |
| | 30-39 | 28% |
| | 40-49 | 18% |
| | 50-59 | 11% |
| | 60+ | 8% |
| Education ($N = 197$) | High school | 18% |
| | College | 18% |
| | Bachelors | 43% |
| | Advanced degree | 21% |
| **Web login and SSO usage** | | |
| Daily web logins ($N = 198$) | 0-2 | 4% |
| | 3-5 | 48% |
| | 6-8 | 30% |
| | 9-11 | 10% |
| | 12+ | 8% |
| Used SSO ($N = 199$) | Yes | 90% |
| | No | 7% |
| | Unsure | 3% |
| Non-work accounts linked with SSO ($N = 197$) | 0 | 15% |
| | 1-5 | 59% |
| | 6-10 | 15% |
| | 11-15 | 6% |
| | 16-20 | 1% |
| | 21+ | 4% |

**Web Login and SSO Familiarity**

Among the 198 participants who responded, 48% said they log into 3-5 different sites (not pertaining to any specific login method) on a typical day and another 30% reported logging into 6-8 sites per day. The majority of participants (90% of 199) reported using web SSO in the past to log into at least one site, either to a work account (e.g., office email) or to a personal web account. When asked about web SSO use outside work (197 responses), 59% reported using SSO on 1-5 sites outside work and 15% reported not using SSO outside work. We report additional details on demographics, web login, and web SSO accounts used by participants in Table 5.1.

Table 5.2: Participants' level of interaction with applications and/or services related to non-work accounts, based on a sample size of 198 participants who responded to the demographics questions.

| Frequency of account use | Google | Facebook | Apple |
|---|---|---|---|
| at least once a day | 78% | 35% | 26% |
| at least once a week | 12% | 19% | 11% |
| at least once a month | 4% | 13% | 14% |
| at least once a year | 1% | 10% | 10% |
| never | 1% | 3% | 1% |
| no account | 4% | 20% | 38% |

**IdP Account Usage**

Table 5.2 presents a summary of participants' use of apps and services offered by Google, Facebook, and Apple, which are IdPs widely supported by RPs for web SSO login [70]. Of the 198 responses, 96% had a Google account, 80% had a Facebook account, and 62% had an Apple account. Most participants reported using Google apps/services regularly, with 78% who used a Google service at least once a day, and 12% who used once per week. In contrast, less than half of participants reported using their Facebook account (35%) or their Apple account (26%) on a daily basis. The remaining participants reported using Facebook or Apple infrequently; 26% of the participants for Facebook and 25% for Apple said they used their accounts once a month or less. We note that some of these self-reported values may be underestimated because they might involve instances where a participant reported using their account (such as Apple) infrequently but in practice uses their accounts regularly by using a device (e.g., iOS smartphone) associated with the IdP account.

### 5.2.3  Ethical Considerations

This low-risk study was reviewed and cleared by Carleton University's Research Ethics Board. To protect privacy, we collected no information about participants' login credentials and we did not ask participants to directly interact with our test sites. Identifiable information was limited to Prolific's generated participant identifier, which we stripped from the data set before starting analysis.

### 5.2.4  Limitations

We note that our data is collected from participants' self-reported responses on login decisions and related reasons. These responses might not fully align with actual user behaviour in practice. Participants reported their login decisions based on one of the four RP sites used in this study. These responses may not reflect participants' preferences on other RP sites, e.g., on a different category of site, or on RPs with a different set of login options. Our sample of RPs represents a small variety of sites that support web SSO systems. We recruited participants in the US and Canada using the Prolific platform; these participants, by virtue of having signed up for online survey platforms, may differ (e.g., by possibly being more tech-savvy) from the general population.

### 5.3  Result: Login Preferences

Figure 5.2 (page 91) summarizes the distribution of 200 participants' pre- and post-SPEye login decisions. The pre-SPEye decisions are distributed into aggregates of SSO and non-SSO login choices. The post-SPEye decisions are split into choices that remained the same as pre-SPEye ("no change"), and choices that differed from pre-SPEye ("changed decision"). For example, among the 89 participants who chose Google login option as their pre-SPEye choice, 55 stayed with Google after SPEye and 34 changed to a different login option. Among these 34 participants, 19 chose Apple, 14 chose non-SSO, and *one* chose Facebook login options.

We now discuss participants' pre- and post-SPEye login preferences. As explained next, where participants changed their login decisions, we observed a tendency to move towards more privacy-friendly login options.

### 5.3.1  Pre-SPEye Login Decisions

55% of participants ($n = 110$) preferred an SSO login option as their pre-SPEye login choice (i.e., asked on a separate page before we showed SPEye's screenshots). Among the SSO users, Google login was by far more popular ($81\%; n = 89$) than Facebook ($10\%; n = 11$) or Apple ($9\%; n = 10$) login. Common reasons given for
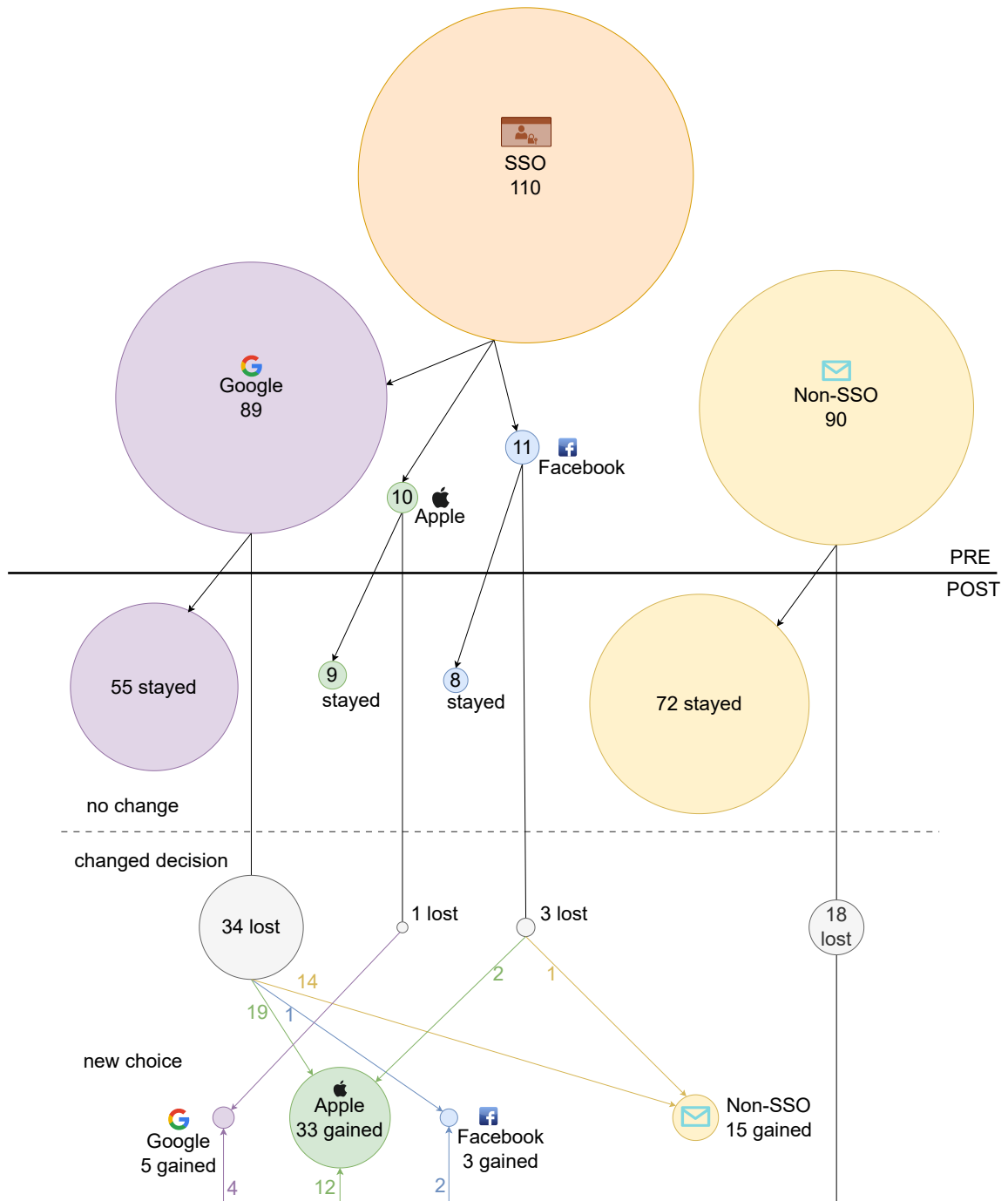
Figure 5.2: Distribution of the pre- and post-SPEye login decisions. Decisions that changed are further broken out to show the new login choices. For example, among the 10 participants who chose Apple SSO, 9 stayed with Apple post-SPEye and *one* changed to Google login. Bubble sizes are proportional to the numbers they represent. Non-SSO includes password-based login with email address or phone number.

Table 5.3: Number of participants choosing each SSO login before and after viewing comparative information about IdP permissions. Choices for each RP are ranked from least (1) to most (6) privacy-friendly depending on the user data revealed to the RP (e.g., the 🅕† option on Airbnb is ranked higher than 🅕 as it opts out of the birthday permission). Cells with 2 icons mean those options are privacy-equivalent, and the corresponding count is the sum for both options. The Wilcoxon test results show the statistical differences between the pre- and post-SPEye privacy ranks.

| Rank | Airbnb login | pre | post | CBC login | pre | post | NYTimes login | pre | post | Rakuten login | pre | post | Total pre | post |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 🅕 | 4 | 2 | 🅕 | 1 | 0 | - | - | - | G | 25 | 10 | 30 | 12 |
| 2 | - | - | - | G | 19 | 16 | - | - | - | - | - | - | 19 | 16 |
| 3 | G, 🅕† | 24 | 23 | 🅕† | - | 2 | G, 🅕 | 24 | 15 | 🅕 | 3 | 3 | 51 | 43 |
| 4 | 🍎 | 3 | 3 | 🍎 | 2 | 5 | 🍎 | 3 | 1 | 🍎 | 2 | 3 | 10 | 12 |
| 5 | ✉ | 19 | 18 | ✉ | 28 | 24 | ✉ | 23 | 20 | ✉ | 20 | 25 | 90 | 87 |
| 6 | 🍎‡ | - | 4 | 🍎‡ | - | 3 | 🍎‡ | - | 14 | 🍎‡ | - | 9 | - | 30 |

| $p = 0.0593$ | $p = 0.1919$ | $\boldsymbol{p = 0.0004^\star}$ | $\boldsymbol{p = 0.0006^\star}$ |

p-values computed using one-sided Wilcoxon signed-rank test
★statistically significant with Bonferroni multiple-test correction, i.e., $p < 0.0125$

dash (-) means option unavailable, rather than that 0 selected it.
✉: Non-SSO login option (email address or phone number)
🍎‡: Apple login option with **pseudonymized data**
🅕†: Facebook login option with **permissions opted out**

selecting Google included convenience because participants were already using other related services (such as Gmail) that results in them remaining logged in with Google, and the convenience of linking RP accounts to their primary web account. This is consistent with the large number of participants (76% of 200) who indicated using Google services daily; versus 31% for Facebook, and 21% for Apple (Section 5.2.2).

Non-SSO login options (in our study, password-based) were selected as an initial choice by 90 participants (45%), and common reasons given were privacy concerns related to SSO linking of multiple RP accounts with the IdP. Additionally, some participants perceived security benefits from having separate accounts to reduce the risk of one compromised account impacting another. Besides releasing personal data to RPs, some non-SSO users noted concerns about IdP tracking of RP site visits. In Section 5.4.2, we discuss IdP tracking privacy issues in more detail.

### 5.3.2 Post-SPEye Login Decisions

Overall, 28% of participants ($n = 56$) decided to change their login choice after viewing the comparative IdP permissions. Of these participants, 23 changed from one SSO login option to another SSO option, 18 moved from non-SSO to SSO login options, and 15 changed from SSO to a non-SSO login option. Figure 5.2 (page 91) illustrates these changes.

Among the 89 participants who initially chose Google login (the most popular pre-SPEye SSO login choice), 34 changed their preference: 19 changed to Apple login and 14 changed to password-based login. The primary reason given for this change was the availability of a more privacy-friendly login option. Apple login was more privacy-friendly because it offered a privacy option to pseudonymize the user's name and email address (Section 5.1.2) and to login using SSO without revealing real (personal) information to the RP. In total, 42 of the 200 participants preferred to login with Apple after viewing SPEye's screenshots; and 30 of these participants said they would use Apple's pseudonymization privacy feature.

For our analysis, we ranked the login choices per RP using a 6-point scale based on our subjective view of privacy-friendliness as explained next, based on the type and number of data permissions requested by each RP[5]. We ranked login choices that included non-basic personal data (as defined in Section 3.1.1) as more privacy-intrusive, and ranked the login choices that included only basic permissions (*name*, *email address*, and *profile photo*) as more privacy-friendly. For example, the Google login option on `Rakuten.com` was ranked as most privacy-intrusive (1) as it included access to personal email messages; among the privacy-friendly choices, we ranked the Apple login option that allows users to pseudonymize their data as the most privacy-friendly (6), followed by the non-SSO login option which reveals the user's email address (5), and the Apple login option (non-pseudonymized) which reveals the user's name and email address (4). Within an RP, we assigned the same rank to login options with comparable privacy—e.g., the Google and Facebook login options on `NYTimes.com` which requested basic permissions are both ranked as 3 (privacy-neutral). This is also a relative ranking with unequal differences between adjacent

---

[5]This ranking was not visible to participants taking the survey

ranks, e.g., the differences in permissions between ranks 1 and 2 may be much more significant than the differences between ranks 5 and 6. Table 5.3 (page 92) shows the privacy ranking for each RP and the distribution of participants' pre- and post-SPEye login decisions.

For each of the four RPs, we conducted a Wilcoxon signed-rank test with Bonferroni correction (i.e., adjusted $p < 0.0125$ to account for the four tests) comparing the pre- and post-SPEye login decisions, and found statistically significant differences for two RPs (`NYTimes.com` and `Rakuten.com`), where the post-SPEye decisions were more privacy-friendly, as reported in Table 5.3. We found no statistically significant differences for the `Airbnb.ca` and `CBC.ca` RPs. However, among the participants who changed (`CBC.ca`: $n = 12$, `Airbnb.ca`: $n = 13$), most of them moved to a more privacy-friendly option (`CBC.ca`: $n = 8$ and `Airbnb.ca`: $n = 10$). Furthermore, the majority of the pre-SPEye login choices for `Airbnb.ca` and `CBC.ca` RPs were already privacy-friendly (i.e., the least privacy-friendly choice, Facebook login, was initially chosen by only four and one participants respectively). Many of the participants who selected Google login for `CBC.ca` (ranked as second least privacy-friendly choice) said in their open-ended responses that they were comfortable with their pre-SPEye decision as they considered the requested permissions (i.e., *name, email address, language preference, profile picture*, and *public profile*) acceptable. Section 5.4.3 provides further details on participants' rationale for login decision changes.

**Summary of Effects of Comparative Information**

Figure 5.3 shows the privacy of pre-SPEye login decisions and highlights the shift in participants' post-SPEye decisions towards more privacy-friendly login options. Overall, we found 48 of 56 participants who changed their login decisions (i.e., 86% of those changing) moved to a more privacy-friendly login option after viewing the IdP permissions. Among the remaining 72% of participants ($n = 144$) who did not change, many of them had chosen login options that were privacy-neutral (ranked 3-4) or privacy-friendly (ranked 5-6) already; e.g., 81 of these 144 participants chose a non-SSO login option (ranked 5, privacy-friendly) or the Apple login option (ranked 4, privacy-neutral), as implied by Figure 5.2.
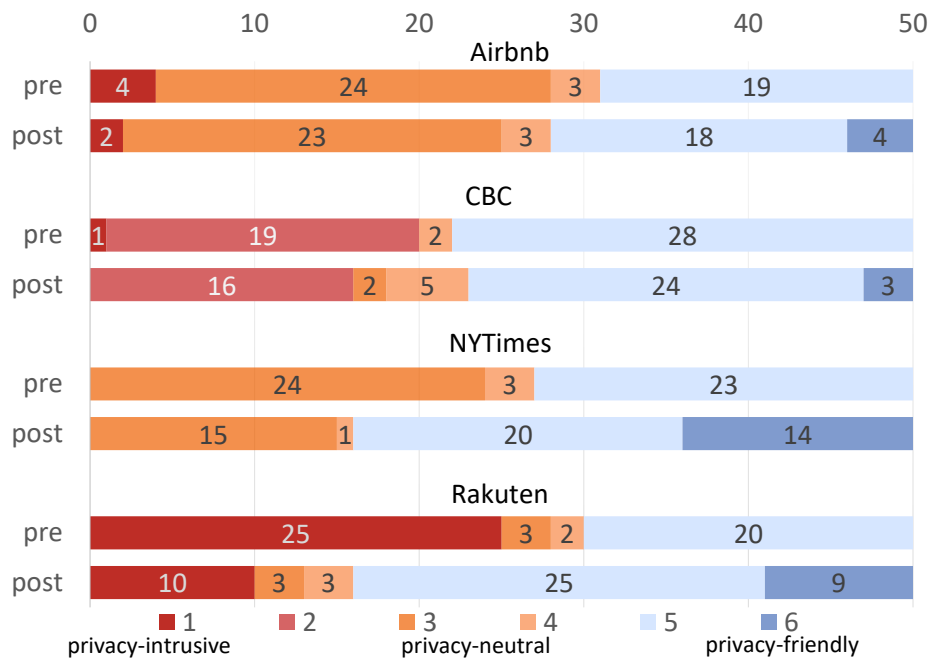
Figure 5.3: Number of participants making each pre- and post-SPEye login decision shown per RP. Login choices for each RP are ordered from privacy-intrusive (1) to privacy-friendly (6) to show shifts in the privacy of login decisions after viewing IdP permissions.

These results suggest to us that many users will make more privacy-informed decisions when provided with comparative information about the differences between IdPs with respect to requested IdP permissions.

## 5.4 Result: Factors Influencing Login Preferences

We now summarise the qualitative analysis of the reasons given by participants for their login decisions. We outline the factors that influenced their pre- and post-SPEye decisions, and highlight the effects of showing comparative information about IdP permissions.

### 5.4.1 Coding Methodology

We conducted an initial data cleaning pass to remove non-meaningful responses (e.g., 'No response', 'N/A', or non-legible responses). After this filtering, we had 333 valid open-ended responses relating to participants' reasons for their pre-SPEye ($n = 176$)

and post-SPEye ($n = 157$) login decisions. These responses were categorized through inductive coding. Inductive coding is a process where the responses are carefully reviewed and conceptualized into an initial set of codes (codebook) which are iteratively revised as new codes are recognized.

The lead researcher first reviewed the 176 pre-SPEye responses to develop a set of 14 codes, then added 7 codes from reviewing the 157 post-SPEye responses, forming an initial codebook containing 21 codes in total. The research team met several times to iteratively review and revise the codebook and discuss the classification of individual excerpts, resulting in a final set of 17 codes including the pre- ($n = 13$) and post-SPEye ($n = 4$) codes. All three members of the research team agreed with the final categorization of excerpts into codes. We further grouped the 17 codes into 7 themes (6 pre- and one post-SPEye) based on common topics that emerged during our analysis. Table 5.4 lists the individual codes and themes.

The results discussed next are split into pre- and post-SPEye themes along with sample excerpts for the individual codes. For each theme, we include the number of login decisions that were influenced by the codes to differentiate more popular themes. When a participant's response indicated multiple reasons for their login decision (e.g., privacy and security), we counted the response in each of the themes. Thus, a single participant's response might be split and included in the counts for multiple factors. In addition, responding to these open-ended questions was optional so our analysis is based on participants who provided valid responses indicating reasons for their login decisions.

### 5.4.2 Pre-SPEye Factors

The pre-SPEye login decisions were primarily based on *usability* preferences and *inertia* as indicated by the number of excerpts identified as related to these themes. *Security* and *privacy*-related reasons were the third and fourth most mentioned, respectively, among the excerpts provided before SPEye. And the remaining two themes (*linked account concerns* and *trust perceptions*) appeared to influence comparatively fewer pre-SPEye login decisions.

Table 5.4: Themes that emerged during our analysis of the participant responses for their pre- and post-SPEye login decisions. The right two columns represent codes relating to participants who chose SSO and non-SSO login options. Codes marked with a dagger † denote those derived from responses after participants viewed the comparative information about IdP permissions (i.e., Post-SPEye).

| Theme | SSO login codes | Non-SSO login codes |
|---|---|---|
| 1. Usability benefits and obstacles | SSO usability better | Non-SSO usability better |
| 2. Inertia/habits | Used primary account | Use of existing non-SSO account |
| | Used existing RP/IdP account | |
| 3. Perceived security | Perceived SSO security benefits | Perceived non-SSO security benefits |
| 4. Privacy-related motivations | Self-managed privacy strategy | Non-SSO privacy better |
| | Privacy is not a major concern | |
| | † Provides better privacy | |
| 5. Linked account concerns | | Concerns related to linking accounts |
| 6. Trust perceptions | Perceived trust/reputation in the IdP | Concerns related to trusting SSO entities and the overall system |
| 7. Privacy-usability tradeoffs | † Good compromise between privacy and usability | † Privacy concerns but lack existing privacy-friendly IdP account |
| | † Usability-privacy conflict but chose usability | |

**1. Usability benefits and obstacles.** We define usability in the web SSO context as relating to convenience in storing/retrieving passwords, and to the ease of accessing services and devices related to SSO accounts. Where login also involves release of personal information to RP sites, usability also relates to convenience in managing the personal information in the IdP account.

Usability was a dominant factor in pre-SPEye login decisions, especially among participants ($n = 53$) who chose SSO login options. These participants' responses indicated that SSO was convenient because they did not have to create or remember new passwords on individual RP sites. SSO was also preferred for fast login because participants reported remaining signed in to their IdP account on a related website or device which they used regularly.

> "I'm usually logged into Gmail so it's easier to use my Google account [to login]." (P186)

Conversely, a smaller group of participants ($n = 15$) found non-SSO login options simple and easy to use. These participants mentioned using tools such as password managers and browser autofills which offer users comparable login experiences as SSO login options (i.e., no need to remember/type individual account passwords), sometimes with perceived added benefits of having separate accounts. For example, P21 felt they had better control over their personal information with non-SSO accounts. Using non-SSO accounts, users can choose what personal information they share with the sites they log into, as opposed to SSO accounts where the information might be linked from the IdP account.

> *"I prefer having a separate account for sites as it allows me to better customize my privacy and profile settings [...]" (P21)*

Other benefits mentioned by participants included easier recollection of previously chosen logins, e.g., by always choosing login with email and password for web logins across different websites.

The perceived usability of SSO and non-SSO login options depended on which features were most valued by individuals, and on participants' use of services and devices related to IdPs. Participants primarily associated SSO with better usability as it removed the need to create/remember extra passwords and enabled fast login using IdP accounts they were already logged into. Comparatively fewer participants associated non-SSO login with better usability, though participants felt better control over their data which justified their non-SSO decision.

**2. Inertia/habits.** Inertia in software systems is defined as the user tendency to persist with the use of an existing system in the presence of alternate options [82]. We regard inertia in the context of web login as the tendency of a user to persist with an existing login method/habit when other alternative login options are available.

Inertia was the second most dominant theme as participants preferred login options with which they were already habituated from having used them across different sites. Similar to the usability theme, most participants in this theme chose SSO login

options ($n = 48$) compared to non-SSO options ($n = 13$). Participants preferred login options associated with their primary IdP accounts because they were most familiar with these accounts and were often signed in with other related IdP apps (e.g., Gmail, Safari browser).

> "*I have used Google for many years so its [sic] my go to when logging onto different sites.*" *(P42)*

Participants who did not have a strong preference for how to login were influenced to choose SSO login because they preferred to use their primary email address, with or without SSO. For example, P167 chose SSO login because their primary email address was associated with an IdP:

> "*[Google login is] faster and even if I was to sign up for an account, [I] would use my gmail for it anyways.*" *(P167)*

Some participants mentioned other benefits with primary accounts such as account longevity (i.e., continued use of primary account in future), and easier SSO account management from connecting different accounts to a central primary account.

Existing login habits influenced many participants' login decisions, especially among those who preferred SSO login options using primary accounts. This preference aligns with web SSO design which connects different web login accounts to a primary IdP account. We found that a significantly larger number of participants were habituated with SSO login options (78% of those observed to be choosing based on inertia) compared to non-SSO login options. This result contrasts findings in a 2011 SSO study [101] where users generally preferred non-SSO login and were habituated to login using email address and password. Our results suggest that many users have become more habituated/familiar with web SSO login over the last decade, possibly due to the increasing presence of SSO login options on RP sites, and popular IdPs such as Google and Apple.

**3. Perceived security.** In our context, perceived security involves perceptions about the security of web login options including the security of login credentials,

and in the case of SSO, the security of any personal information requested through IdP login accounts.

Security perceptions influenced pre-SPEye login decisions among 25 participants. Like usability, participants articulated security arguments for and against both SSO and non-SSO login options. However, unlike themes 1 and 2 above, the majority of participants prioritizing security (19 of 25) in this theme chose a non-SSO login option. Participants felt that using unique passwords offered better security than SSO options, possibly because they used password generators to create and manage passwords that are difficult to guess, or possibly due to misconceptions about the security of SSO accounts and access tokens. Another common perception among participants was that non-SSO accounts offered better security because a security breach of one account would not put at risk their other accounts.

> *"Separate logins are more secure in the event of data breach. You don't need someone to have access to all of your accounts because they have access to your google [IdP] account." (P11)*

> *"[...] It is more secure to use email and password combination for sign in, if a unique password is used for each website." (P199)*

Although SSO protocols aim to improve security by replacing user-chosen passwords with unique access tokens, concerns about a single point of failure motivated these participants to choose non-SSO login options. These concerns are not baseless, though it is unclear whether participants had knowledge of specific risks or whether they were generally uneasy about using SSO. A vulnerability in IdP code could be exploited by attackers to compromise access tokens; for example, in 2018 attackers exploited a vulnerability in Facebook's SSO software to gain access to 50 million user accounts, including access tokens linked to every compromised account [88].

**4. Privacy-related motivations.** Privacy in the web login context relates to personal information entered or linked to login accounts, including data released from an IdP to an RP site during SSO login.

Privacy-related reasons motivated the pre-SPEye login decisions for 23 participants. Most of these participants (14 of 23) chose non-SSO login options to reveal

less personal information with RP sites, though participants had not viewed the permissions requested with the alternate SSO login options yet. Separately, some participants avoided SSO due to concerns about the IdP's ability to gain knowledge about users' website visits (i.e., cross-site tracking) through SSO login events.

> "I am not comfortable with giving Google, Facebook, or Apple even more
> of my personal information than what they already have." (P125)

Other privacy-conscious participants ($n = 6$) who selected SSO took steps towards privacy using IdP tools (e.g., permission opt-out options offered by some IdPs such as Facebook) or through strategies such as using throwaway accounts that contained less personal information or that they used infrequently. The remaining participants ($n = 3$) with comments relating to privacy explicitly mentioned that they didn't have major privacy concerns about using SSO on the presented RP website, which could be due to an indifference towards privacy, or due to other factors including personal opinions about the RP sites and/or IdP login options we showed them.

These findings show that privacy concerns (besides security perceptions as noted previously) are common reasons motivating non-SSO login choices. While better privacy controls (e.g., opt-out mechanisms for permissions requested by RPs) might incentivise a subset of privacy-conscious users to use SSO, other concerns including cross-site tracking by IdPs influence users to avoid SSO altogether.

**5. Linked account concerns.** Besides security and privacy, a group of participants ($n = 18$) had other concerns about linking accounts which motivated them to choose non-SSO login options. These participants avoided SSO because they were generally uneasy about linking different accounts, though they did not elaborate on these concerns. A specific concern mentioned by P29 related to the freshness of the personal information in their older accounts:

> "I get nervous about connecting my other accounts to websites because a
> lot of the options are accounts that I made when I was 10 [years old] and
> that data might not reflect me anymore." (P29)

Although participants did not explicitly mention privacy or security, it is possible that some of the concerns about linking accounts were related to privacy and/or security, similar to many of the participants in the previous two themes.

**6. Trust perceptions.** Trust perceptions in the web SSO context relate to the confidence users might have in entities involved in the SSO login system including RPs, IdPs, and other components in the SSO ecosystem (e.g., the protocols, and software components in browsers and servers). This confidence may include perceptions of an entity's intentions (perhaps malicious) and their competency or ability to protect user information; our study is unable to distinguish between these aspects of confidence (or "trust").

Trust perceptions influenced pre-SPEye login decisions among 13 participants, and these were mainly based on the perceived reputation of IdPs. One common perception among the participants who chose SSO login ($n = 6$) was that they trusted Google more than Facebook or Apple:

> *"Google probably shares my personal information less often that the other [login] options." (P89)*

This selective trust of Google did not extend to the remaining participants, all of whom chose non-SSO login options ($n = 7$). These participants did not want the IdPs to know about their RP site visits as they did not trust any of the IdPs (Google, Facebook, or Apple).

Compared to the previous themes, trust perceptions had the smallest influence on pre-SPEye login decisions, as implied by the number of participants providing comments falling in this theme. Where trust did influence login decisions, it was primarily based on participants' perception of the IdPs, suggesting that major IdPs (Google, Facebook, and Apple) are more influential than RPs in users' perception about web SSO systems.

### 5.4.3   Post-SPEye Factors

After explaining the reasoning for their pre-SPEye login decision, participants viewed SPEye's screenshots displaying comparative information relating to the information requested of each IdP. They then had the chance to reconsider their decision in light of this new information. We now compare the pre- and post-SPEye login rationales, and discuss how participants' decisions were affected (or not affected) by the comparative information.

Pre-SPEye login decisions were mainly based on usability preferences and inertia, with privacy reportedly impacting relatively fewer pre-SPEye decisions. However, privacy-related motivations were among the dominant factors influencing many of the post-SPEye login decisions. Usability preferences and inertia were noted in fewer excerpts (compared to pre-SPEye), though these were the second and third most popular post-SPEye factors, respectively. We also identified a new theme that emerged among participants who weighed both usability and privacy factors when making their choices. We observed fewer post-SPEye excerpts associated with concerns relating to linked account (8 from the previous 18). Many of the original 18 participants articulated concerns about linked accounts (including privacy- and security-related) more clearly in the post-SPEye responses after viewing the requested permissions.

Security and trust perceptions were two factors that remained largely unchanged as participants expressed similar views on these factors, and the number of participants influenced by these factors was similar. The lack of change in security perceptions is expected as we did not show any information about the security of login options. Trust perceptions observed in participants' responses were based on the perceived reputation of IdPs, thus we do not expect these to change post SPEye. Additional SSO participants mentioned trusting Apple login (besides others who mentioned trusting Google login) in their post-SPEye responses, possibly a result of an increase in the number of participants choosing Apple login as their post-SPEye choice (Section 5.3.2).

We describe the new codes (denoted with a dagger † in Table 5.4) that emerged in Themes 4 and 7 below.

**4. Privacy-related motivations.** After viewing the permissions requested by the RP, participants expressed privacy concerns about disclosing personal information requested by RPs, including both basic (*email address* and *profile picture*) and sensitive personal data (e.g., *email messages*). The availability of another login option with better privacy influenced many participants ($n = 39$) to change their login decision to a more privacy-friendly option, with Apple SSO (pseudonymized data) being the most popular choice selected by 64% of those changing. For example, P169 who first chose Google SSO was concerned about revealing their email address to the RP, and changed their preference to Apple SSO with the pseudonymization option:

> *"[...] I don't want [...] my email address being sold to other providers. I already get enough junk mail."* (P169)

Among another group of participants ($n = 27$), showing the requested IdP permissions reinforced their pre-SPEye login decisions which they considered privacy-friendly. These participants did not change their login decisions, though they expressed privacy concerns similar to the concerns raised by the pre-SPEye group of participants motivated by privacy. This reinforcing effect was mostly observed among participants who chose non-SSO login options ($n = 24$). This aligns with our earlier finding among pre-SPEye decisions that most participants motivated by privacy chose non-SSO login (Section 5.4.2), suggesting that privacy-conscious users tend to choose non-SSO login options.

**7. Privacy-usability tradeoffs.** A new theme emerged among a subset of participants ($n = 17$) who considered both usability (the most dominant pre-SPEye factor) and privacy preferences (the most dominant post-SPEye factor) when making their post-SPEye decisions.

Many of these participants ($n = 9$) weighed both the privacy and usability of their login choices after being informed about the permissions requested by the RP. While these participants expressed concern about their privacy, each chose an SSO login option and reported being comfortable with the personal information that would be released to the RP through their login choice. For example, P10 changed their login

decision from login using email and password to Google login because they considered the personal information being requested by `CBC.ca` through Google (*name, email address, language preference, profile picture*, and *public profile*) as already being publicly available. Participants also mentioned approving the privacy-usability tradeoff for their specific assigned RP sites, suggesting that they might weigh these factors differently based on individual sites.

> *"I don't mind [CBC.ca] having access to already fairly accessible information about me [...]" (P10)*

Some of the participants ($n = 5$) valued the usability benefits of an SSO login option despite the option involved revealing extra personal information to the RP. For these participants, although their preferred login choice did not align with their privacy preferences for the RP site, they favoured the usability benefits of their chosen SSO login option. A small number of privacy-conscious participants ($n = 3$) appreciated the privacy of certain alternate IdP options, but they did not have existing accounts with those privacy-friendly IdP options. These participants felt compelled to use non-SSO login due to their privacy concerns and lack of alternate IdP accounts.

IdPs might retain these segments of users by offering privacy controls such as permission opt-outs and data customization options, and get users to adopt their SSO platforms across different RP sites.

## 5.5 Related Work

In this section, we focus on prior work related to web SSO user perceptions and user privacy. We compare the findings in our study with existing research, and discuss differences and similarities.

Prior work on user perceptions related to SSO login systems has focused mainly on individual IdP platforms. Balash et al. [10] investigated user perceptions of Google users on granting third-parties access to their personal information through Google SSO login. Our study differs as it includes two additional major IdPs besides Google and explores user perceptions of both SSO and non-SSO login options. Bauer et al. [11] explored the effectiveness of the consent dialogs displayed by three IdPs

(Google, Facebook, and Google+) by prompting participants to login using a given IdP option. Some of the participants were exposed to basic permissions, while others were shown privacy-invasive permissions. They found that participants had preconceived views of the IdP permissions, which was not impacted by the IdP consent dialogs. Here, we focus on RP login prompts which are displayed before users see IdP consent dialogs. Robinson and Bonneau [87] studied the consent dialog used by Facebook Connect to evaluate participants' understanding of IdP permissions. Their findings suggest that users tend to misinterpret the extent to which requested permissions might grant access, e.g., many of their participants did not understand that the requested permissions would grant RPs the ability to post to their Facebook profiles.

Other studies that examined login decisions focused on individual factors including privacy and security. Egelman [26] focused on privacy tradeoffs in RPs with Facebook Connect login, where users reveal personal information to RPs in exchange for SSO login convenience. We find similar privacy-usability tradeoffs, but in a wider context involving other IdP platforms and a variety of privacy choices. In 2011, Sun et al. [101] explored factors influencing users to avoid OpenID SSO systems, finding single point of failure concerns and lack of trust in RP sites. Our results show a shift in SSO adoption over the last 12 years, and show that security and trust perceptions not only influence users to avoid SSO, but also influence users to adopt certain SSO login options. Besides login preferences, our study also investigated the effects of comparative privacy information on login decisions.

Increasing concerns about privacy issues have motivated researchers to build privacy-enhancing tools. Weinshel et al. [108] designed a browser extension that extracts information on third-party web trackers and displays tracking activity to users on privacy dashboards. Farooqi et al. [35] designed *CanaryTrap*, a measurement tool that detects misuse of Facebook Login user data APIs by third-party RP apps on Facebook through honeypot email accounts. Using the tool, they monitored 1,024 RP apps on Facebook, and identified numerous instances of user information misuse, including selling of user information to malware creators and advertisers.

Empirical studies related to SSO privacy practices have focused on the personal information released to third-parties through SSO. Wang et al. [105] analyzed the

privacy notices of 1,800 RP apps on Facebook to identify privacy issues in Facebook's SSO data sharing. They found that the SSO login permissions could override users' Facebook profile privacy settings, which may not align with what users might expect of their privacy settings. Felt and Evans [37] studied 150 popular third-party apps to determine the necessity of requested permissions. They identified that many apps did not require access to all the requested permissions, and proposed using data anonymization techniques to improve privacy.

Beyond the SSO domain, many researchers have studied permissions and user privacy in the mobile app ecosystem. Cao et al. [17] performed an in-situ study with 1,719 participants on their behaviours relating to Android app permissions. Their findings showed that users were twice as likely to approve permissions that included an explanation as ones without any explanation of their necessity. Wijesekera et al. [110] explored the possibility of automatically selecting responses to Android app permission requests based on users' privacy preferences inferred from past behaviour. They reported that their approach correctly predicted user decisions in 96.8% of requests. Felt et al. [36] explored various consent dialog design approaches for third-party data sharing, and provided usability guidelines for designing effective consent dialogs with a focus on mobile UI systems. Liu et al. [65] designed an Android privacy assistant tool to recommend and nudge privacy settings based on the user's privacy preferences. These proposals for improving privacy in Android devices highlight the usefulness and effectiveness of making it easier for users to make privacy decisions.

## 5.6   Discussion

Our study provided results showing participants' reported login decisions before and after viewing the IdP permissions requested by RP sites. Inductive coding of open-ended responses providing rationale for their choices uncovered factors that influenced these decisions, reflecting potential effects of showing the comparative IdP login permissions.

In this section, we provide further insights on user motivations identified in our study, and discuss challenges to informing users about different privacy and security issues in web SSO systems.

**Informed login decisions.** After we showed the permissions requested by RPs, there was a shift in priorities (cf. Figure 5.3) towards more privacy-friendly login options; many participants expressed that this was due to concerns about releasing personal information. In some cases, we observed the reverse effect where users informed about the permissions made a privacy-usability tradeoff to share more information with RPs in exchange for the convenience of SSO. We attribute these changes to enabling users to compare differences in IdP permissions across login options, and more visibly displaying privacy implications of different login options.

In practice, informing users about these differences is challenging as current SSO UI designs focus on individual login options without providing sufficient context to compare the implications of different login choices. Nonetheless, we suggest that RPs and IdPs interested in being transparent about the privacy implications of login choices adopt UIs that enable comparison of permissions (such as elements of the SPEye UI used in our study), thus enabling privacy-informed user decisions.

**IdP tracking issues.** Regarding factors influencing login decisions, we found that users motivated by privacy had a tendency to believe that choosing non-SSO login would offer privacy benefits, including less information shared with RPs and avoiding cross-site tracking of their RP visits by IdPs. However, the privacy mental models of users who chose non-SSO login were misguided with respect to avoiding cross-site tracking, because major IdPs such as Google and Facebook can still track non-SSO users, e.g., through tracking scripts such as Google's Doubleclick and Facebook's Pixel—which appear in a large number of websites [27]. We note that such scripts enable IdPs to track users independent of their use of SSO login. We encourage future work exploring ways to increase awareness of broader tracking issues in the web SSO context where non-SSO users might not anticipate such tracking.[6]

**Misunderstand release of personal information.** Our study found that users who made login decisions based on trust factors (Section 5.4.2) were influenced by

---

[6]Ad blockers such as Ublock Origin, which can stop tracking by Doubleclick and Pixel, are not designed to stop tracking via IdP scripts related to web SSO toolkits [71].

their perceptions of IdP reputation. This is despite the personal information requested depending primarily on the individual RPs, and being released to these RPs (which users may not have the same trust for). This IdP-to-RP trust transference (i.e., users perceiving an RP as trustworthy based on the user's trust in the IdP, and the RP-IdP relationship) was more frequently observed with Google than Facebook or Apple (pre-SPEye), perhaps due to Google's popularity among participants. This independent observation is consistent with findings of IdP-to-RP trust transference in a recent study specifically on Google SSO login [10]. However, such trust transference contrasts earlier SSO studies (e.g., [26, 101]) that found that users based trust perceptions on the RPs' reputation rather than that of IdPs. This contrast could be due to a number of factors including users being (intentionally or unintentionally) misguided by SSO UI designs about how SSO permissions work. It might also be that IdPs that wish to keep users locked to their platforms have less incentive to improve transparency in permissions UIs. Besides ineffective UIs, it is also possible that some users trust the IdPs (and therefore the RPs that offer these IdP options) that release less information to RPs, e.g., Apple SSO APIs currently only reveal name and email address.

**Security of SSO login options.** Across different participants in our study, both security and privacy of personal information were significant factors for choosing and avoiding both SSO and non-SSO login options. Beyond our study, SSO protocols such as OAuth simplify security for RPs by outsourcing password management, but also complicate security because implementing OAuth 2.0 securely is challenging [79], e.g., requiring developers to understand various OAuth 2.0 extended features (in separate RFCs for a multitude of use cases and security mechanisms) and to decide which ones match their specific web services (see Section 2.2.5). Other research (e.g., [45, 56, 84, 100, 114]) has found many vulnerable RP implementations, raising security and privacy concerns about personal information in SSO user accounts.

We encourage future work that explores the effects of security-related information on SSO and non-SSO login choices and how to help users make better-informed decisions with respect to (not only privacy-related factors as in our study but also)

use of sites with security weaknesses, to complement our work herein focused on how displaying privacy-related information affects login decisions.

## 5.7 Concluding Remarks

Web SSO login systems have expanded from authentication systems to data-release authorization systems, whereby users end up authorizing IdPs to release user personal information to RPs. The volume of personal information released to RPs increases with increased use of web SSO (and as users generate more such information over time); for this reason, we believe it has become more important to understand how the factors studied herein, including user perceptions and the information displayed to users, influence their web SSO login decisions. Our work is an early step in this direction.

# Chapter 6

# Discussion and Conclusion

Challenges to data privacy in OAuth-based web SSO systems relate to user data collected directly by IdPs—including major technology companies such as Google and Facebook—and made available for access to third parties.[1] This thesis was centered around whether users would make more informed choices when provided with information about web privacy. In this regard, this work focused on three research goals (outlined in Chapter 1) related to third-party data sharing in web SSO systems and the impact on user privacy. We now revisit these goals and summarize the main findings.

G1: *Analyzing what types of user data RP websites request in OAuth-based web SSO systems.*

Our empirical study on the privacy practices of popular RP websites (Section 3.3) revealed significant differences in the categories and amounts of user data requested by RPs across individual IdPs, often with privacy-friendly choices not obvious to users. Our findings also showed that users in EU countries with strict privacy regulations tend to be offered fewer (although more privacy-friendly) SSO choices than users in countries such as US and Canada.

G2: *Developing tools to inform users about the privacy issues in web SSO systems.*

In Chapter 4, we presented the SPEye browser extension tool which informs users by enabling comparison of the permissions requested by RP websites through different SSO login options. We discussed our motivations for SPEye's design and highlighted differences with alternative approaches for designing a tool targeted to inform users in web SSO systems.

---

[1]We note from Chapter 1 that third party refers to entities (e.g., RPs) other than users and IdPs.

G3: *Understanding how users make login decisions in websites that provide more than one SSO login option, and studying the effect of displaying privacy-related login information on RP websites prior to selecting an option.*

In Chapter 5, we presented quantitative and qualitative results on how participants in our user study made login decisions in RP websites, and the effect of displaying information about comparative IdP permissions. We identified usability preferences and inertia as the top factors influencing decisions before permissions were displayed. However, after we displayed the permissions, many participants prioritized privacy over other factors and selected privacy-friendly choices. These findings show the effect of displaying the information extracted by SPEye to help users make privacy-informed login decisions.

We also met other intermediate objectives that helped pursue these goals. In Section 3.1, we described a cross-IdP categorization of user data made available by four IdPs through their SSO platforms. This categorization helped with the comparative analysis of user data permissions in our empirical study on RP privacy practices. In addition, the OAuthScope tool (presented in Section 3.2) enabled the automated collection of OAuth protocol parameters for this empirical study. Our analysis on the software implementations of OAuth protocols and interfaces by RPs (in Section 3.4) identified four software patterns which can inform work on tools that inspect RP SSO login implementations, including our SPEye browser extension.

## 6.1 Further Web SSO Privacy Considerations

This section provides further discussion on web SSO privacy beyond our findings in the previous chapters.

### 6.1.1 Misaligned Stakeholder Interests

Web SSO systems involve multiple stakeholders including users, RPs, and IdPs. In Section 3.3, we highlighted the lack of transparency in current web SSO systems on the privacy consequences of SSO choices. Addressing these concerns is challenging because of the misaligned interests of different stakeholders, as discussed next.

**Web SSO implementers.** It is difficult to incentivize RPs to promote privacy-friendly choices (e.g., to request less user data) as it may not be in the RP organization's business interest [12]. On the other hand, the IdP might also have its own agenda (e.g., encouraging adoption of its services, or prioritizing ease-of-use over privacy) 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 when an RP implements this feature (and the user needs to be signed into Google on the same browsing window), the UI offers useful partial information for a user signing in using SSO. Given the sometimes misaligned goals of stakeholders, a user tool by a third-party (such as SPEye, Chapter 4) might offer a better solution in providing relevant information for users to make informed decisions.

**User preferences.** Another factor complicating privacy improvements is that there is no one-choice-fits-all solution because of variations in privacy preferences across users. Our study in Chapter 5 identified factors (listed in Table 5.4) that influenced the login decisions of our participants, such as *privacy-usability tradeoffs* and *self-managed privacy strategies*. These findings suggest that some users may prefer to share personal data (e.g., in exchange for using a 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 accommodate preferences, compared to security solutions where the secure option can be pre-configured (no choice required from user). We believe that SSO systems should follow a privacy-by-default approach [98] and allow users to opt-in to sharing sensitive personal information instead of the current opt-out design of permissions. In addition, reducing the effort required to obtain relevant privacy and security information could help users make login decisions that align better with their personal preferences.

Prior work on *privacy-enhancing identity management* (PIM) has explored economic incentives for stakeholders [12] and technical possibilities of implementing privacy solutions (e.g., [15, 59]); many of these studies relate to initiatives focused on

improving privacy in online identity management within European frameworks.

### 6.1.2   Risks of Large-Scale Data Harvesting

Besides personal information of users, some IdPs (e.g., Facebook, Google) offer user data APIs that grant RPs access to a user's network (friends/contacts on the IdP platform) and personal information of others in that network; Table 3.1 (page 30) includes these user data attributes. In this case, when a user approves an RP's request to access their network, the IdP reveals to the RP personal information about others in the user's network; this access is explicitly requested from (and granted by) only the user at the RP site without informing others in the network.

In the 2018 Cambridge Analytica-Facebook scandal [14, 104], a third-party RP app used the Facebook Login SSO platform to harvest personal information from users' Facebook profiles, including those users who logged into the app (∼270,000 users) *and* others who were among the networks of those users logging in, leading to a dataset on around 50 million user profiles. The dataset harvested by the RP was made available [104] to Cambridge Analytica without user approval (outside the Facebook SSO platform) which used it to build psychological profiles of the users based on their locations and interests to influence their votes (e.g., by targeting these users with customized links and news articles) in elections around the world.

Following the scandal, Facebook updated its SSO platform [115] to restrict the amount of data revealed about a user's friends (the current `user_friends` API[2] reveals a list of the user's friends who are also using the same RP app). However, this scandal illustrates how personal information disclosed to rogue RP third parties could result in misuse of the personal data of SSO users, and illustrates by one example the problematic disclosure of personal information of users in a given network without explicit approval. Our SPEye browser extension can help protect privacy by making it more apparent for users—before they login—what personal information (about the user and their friends) is requested by RPs and compare it to other available choices.

---

[2]`https://developers.facebook.com/docs/permissions/#u`

## 6.2 Toward More Privacy-Friendly OAuth SSO

This thesis identified privacy issues in OAuth-based web SSO systems, and our SPEye tool tackled the lack of transparency of permissions in these systems. In this section, we discuss directions for researchers and stakeholders to further explore and improve privacy in SSO systems.

### 6.2.1 Preliminary Changes for Stakeholders

Based on our findings, we suggest protocol and interface changes to improve privacy, and the transparency and accountability of RPs.

C1: *When registering with an IdP, an RP could be required to provide descriptions committing to the intended use of each OAuth user-data attribute they plan to request from users.*

This might be accompanied by *value labels* (cf. *privacy nutrition labels*[3]) that convey intended uses and benefits (if any) to users, possibly with cooperation from the browser vendors. The IdP/browser app could show these labels in a standardized UI before asking users to authorize release of RP-requested data attributes. In addition, browser vendors could promote privacy-informed choices by adopting SPEye-like UIs to enable comparisons of different SSO privacy choices.

C2: *IdPs could enforce exclusive use of server-side flows (i.e., disallow the client-side implicit flow) for any RP request involving access to non-basic user data.*

Here "non-basic" is as defined in Table 3.1. Our motivation is that widely scoped access tokens create greater risks, and client-side flows increase the attack surface (see Section 2.2.2). The OAuth spec could mandate the above, as well as the following.

C3: *The OAuth spec could allow optional scope parameters distinct from those denoted mandatory for RP operation.*

---

[3]`https://cups.cs.cmu.edu/privacyLabel/`

Although it is currently already possible for IdPs to allow users to opt-out of scope parameters, IdPs currently cannot distinguish which parameters are mandatory for RP operation. RPs and IdPs that favor transparency could use an *optional* scope parameter to denote user data attributes that are not required for an RP to provision its core features. RPs could then provide the value labels (as mentioned above) to convey the benefits for users opting-in to the optional permissions.

The above changes could allow audits or privacy compliance checks (by IdP or third parties), and support informed choices by privacy-conscious users. Over time, this could result in RP-IdP pairs following the privacy best practice of requesting only *need-to-know* data, and privacy-friendly IdPs gaining a "preferred-IdP" status. We offer a fourth suggestion—complementing our SPEye tool—that may help privacy-aware users make informed decisions.

C4: *Reputation-based or data-driven community efforts could provide privacy ratings for SSO options at popular RPs.*

Such privacy ratings may help inform the community about RP privacy practices and may motivate RP organizations to limit the amount of personal data they collect. Separate from SSO, Mozilla has published similar privacy ratings[4] of several popular consumer devices and apps to highlight concerning privacy and security practices.

Many of these suggestions may of course face numerous hurdles, one being that the agendas of commercial RPs, IdPs, and browser vendors are often not aligned with those of privacy-conscious users. However, individual organizations within a stakeholder group might have their own business models and interests, e.g., compared to Facebook or Google, Apple's SSO platform offers significantly fewer user data attributes to RPs (as illustrated in Table 3.1) and does not rely as heavily on advertising revenue. Thus, some privacy-oriented IdPs and browser vendors (e.g., DuckDuckGo, Firefox) may be inclined to adopt these changes to improve privacy, e.g., to increase their market share among privacy-conscious users.

We also argue that awareness and discussion of technical possibilities are important steps towards supporting user privacy, and shedding light on any RP deceptive patterns [74].

---

[4]https://foundation.mozilla.org/en/privacynotincluded/

### 6.2.2 Future Research Directions

Future work directly related to our work is discussed in the relevant chapters. Here, we briefly discuss ideas for future exploration of other aspects of SSO privacy.

**Hiding Metadata from IdPs.** Section 4.4 described privacy issues related to metadata about RP site visits (including by both SSO users and non-users) being exposed to IdPs. Metadata leaks in SSO systems introduce privacy challenges that affect users in a variety of application domains including web consumer, enterprise, and government organizations [59]. Existing privacy research (listed in Sec. 2.4) on limiting metadata exposure has focused on OpenID Connect-based (authentication) SSO systems. Extending these proposals to OAuth-based SSO systems is challenging as OAuth is designed to support RP-to-IdP user-specific data retrieval requests which reveal metadata on users' browsing activity to the IdPs. Future research could explore possible improvements to limit such metadata exposure in OAuth systems.

**Browser-based IdP services.** Many browser applications offer built-in identity services linked to user accounts including password managers and private email forwarding services. For example, Firefox [43] and DuckDuckGo [25] offer users an option to create unique pseudonymous email addresses (similar to Apple SSO's Private Email described in Sec. 3.1.4) and create accounts at websites without revealing their real email addresses. Browser vendors and OAuth specification designers could jointly explore future OAuth-like web SSO alternatives that offer more privacy benefits than available in existing systems. For example, after a user has created an account at an RP site that offers login through the IdP's SSO service, a local browser-IdP application could authenticate the user without redirecting to the IdP endpoint, thus limiting the exposure of users' browsing history. Reducing the number of redirects might also benefit RPs and IdPs through reduced load on IdP authorization servers and improved login experience for users at RP sites.

**OAuth privacy in constrained devices.** Recent growth of IoT deployments has led to new OAuth-based SSO schemes (e.g., ACE-OAuth [91]) designed for constrained devices with limited resources and UI capabilities. In addition, OAuth standards currently in development (including GNAP, described in Sec. 2.2.5) pursue support for access control in UI-constrained IoT devices. Typical applications of

these schemes might involve several sensor devices that collect information about the physical environment and share it with authorized third-party applications.[5] Future research could explore the types of sensitive or personal information produced and disclosed in IoT deployments, and explore protocol and interface designs that facilitate privacy-friendly OAuth.

## 6.3 Final Remarks

IdPs have grown from primarily providing identity information to additionally enabling third parties to access personal information of users. Over the past decade, certain IdPs have become increasingly utilized by RP websites to obtain users' personal information in addition to their identity information. This trend results in greater privacy concerns for web SSO users, and thus referring to "IdP" could mislead users into believing that only their identity information may be revealed. Alternatively, we suggest referring to these entities as *identity and data providers* (IDPs) to avoid misleading users. This change from IdP to IDP would also align better with their actual function in current SSO systems.

As countries around the world—including Canada—implement privacy laws that mandate web services to obtain informed consent before collecting or sharing personal information about users, privacy studies and tools such as those included in this thesis are important steps towards achieving privacy-informed web societies.

---

[5]https://www.ericsson.com/en/blog/2023/7/ace-oauth-standard-for-lightweight-authorization

# Bibliography

[1] Furkan Alaca and Paul C. van Oorschot. Comparative Analysis and Framework Evaluating Web Single Sign-on Systems. *ACM Computing Surveys*, 53(5):112:1–112:34, 2020.

[2] Alexa. The top 500 sites on the web. `https://www.alexa.com/topsites`, Accessed: January 2021.

[3] Apple. Authenticating users with Sign in with Apple. `https://developer.apple.com/documentation/sign_in_with_apple/sign_in_with_apple_rest_api/authenticating_users_with_sign_in_with_apple`, Accessed: October 2023.

[4] Apple. Sign in with Apple. `https://developer.apple.com/documentation/sign_in_with_apple`, Accessed: October 2023.

[5] Apple. App Store Review Guidelines - Sign in with Apple. `https://developer.apple.com/app-store/review/guidelines/#4.8`, Accessed: September 2023.

[6] Apple. Sign in with Apple - Human Interface Guidelines. `https://developer.apple.com/design/human-interface-guidelines/sign-in-with-apple`, Accessed: September 2023.

[7] Apple. App privacy labels now live on the App Store. `https://developer.apple.com/news/?id=3wann9gh`, December 14, 2020.

[8] Apple. New Guidelines for Sign in with Apple. `https://developer.apple.com/news/?id=09122019b`, September 12, 2019.

[9] Guangdong Bai, Jike Lei, Guozhu Meng, Sai Sathyanarayan Venkatraman, Prateek Saxena, Jun Sun, Yang Liu, and Jin Song Dong. AuthScan: Automatic Extraction of Web Authentication Protocols from Implementations. In *NDSS*, 2013.

[10] David G. Balash, Xiaoyuan Wu, Miles Grant, Irwin Reyes, and Adam J. Aviv. Security and Privacy Perceptions of Third-Party Application Access for Google Accounts. In *USENIX Security*, 2022.

[11] Lujo Bauer, Cristian Bravo-Lillo, Elli Fragkaki, and William Melicher. A Comparison of Users' Perceptions of and Willingness to Use Google, Facebook, and Google+ Single-Sign-On Functionality. In *ACM Workshop on Digital Identity Management*, pages 25–36, 2013.

119

[12] Rainer Böhme and Sven Koble. On the Viability of Privacy-Enhancing Technologies in a Self-Regulated Business-to-Consumer Market: Will Privacy Remain a Luxury Good? In *Workshop on the Economics of Information Security (WEIS)*, 2007.

[13] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *IEEE Symp. Security and Privacy*, 2012.

[14] Carole Cadwalladr and Emma Graham-Harrison. Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. `https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election`, March 17, 2018.

[15] Jan Camenisch, Ronald Leenes, Marit Hansen, and Jan Schallaböck. An Introduction to Privacy-Enhancing Identity Management. *Digital Privacy: PRIME - Privacy and Identity Management for Europe*, pages 3–21, 2011. `doi:10.1007/978-3-642-19050-6_1`.

[16] Brian Campbell, John Bradley, Nat Sakimura, and Torsten Lodderstedt. RFC 8705: OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens. `https://datatracker.ietf.org/doc/html/rfc8705`, 2020.

[17] Weicheng Cao, Chunqiu Xia, Sai Teja Peddinti, David Lie, Nina Taft, and Lisa M Austin. A Large Scale Study of User Behavior, Expectations and Engagement with Android Permissions. In *USENIX Security*, pages 803–820, 2021.

[18] Eric Y. Chen, Yutong Pei, Shuo Chen, Yuan Tian, Robert Kotcher, and Patrick Tague. OAuth Demystified for Mobile Application Developers. In *ACM CCS*, 2014.

[19] Catalin 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/`, July 27, 2020.

[20] Stephanie Curran. Developer Platform will now require Business Verification for Advanced Access. `https://developers.facebook.com/blog/post/2023/02/01/developer-platform-requiring-business-verification-for-advanced-access/`, February 1, 2023.

[21] Brooke Davis. Expanding the App Defense Alliance. `https://security.googleblog.com/2022/12/app-defense-alliance-expansion.html`, December 15, 2022.

[22] Arkajit Dey and Stephen Weis. PseudoID: Enhancing Privacy in Federated Login. In *Hot Topics in Privacy Enhancing Technologies*, 2010.

[23] Yana Dimova, Tom Van Goethem, and Wouter Joosen. Everybody's Looking for SSOmething: A Large-scale Evaluation on the Privacy of OAuth Authentication on the Web. *Proceedings on Privacy Enhancing Technologies*, 2023.

[24] Kostas Drakonakis, Sotiris Ioannidis, and Jason Polakis. The Cookie Hunter: Automated Black-box Auditing for Web Authentication and Authorization Flaws. In *ACM CCS*, 2020.

[25] DuckDuckGo. Email Protection. `https://duckduckgo.com/duckduckgo-h elp-pages/email-protection/what-is-duckduckgo-email-protection/`, Accessed: October 2023.

[26] Serge Egelman. My Profile is My Password, Verify Me! The Privacy/Convenience Tradeoff of Facebook Connect. In *CHI*, page 2369–2378, 2013.

[27] Steven Englehardt and Arvind Narayanan. Online Tracking: A 1-million-site Measurement and Analysis. In *ACM CCS*, 2016.

[28] Nick Ragouzis et al. Security Assertion Markup Language (SAML) V2.0 Technical Overview. *OASIS SSTC*, March 2008. URL: `http://docs.oasis-open. org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html`.

[29] Scott Cantor et al. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. *OASIS SSTC*, March 2005. URL: `http: //docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`.

[30] Facebook. Facebook Login. `https://developers.facebook.com/docs/fac ebook-login/guides/permissions`, Accessed: April 2023.

[31] Facebook. App Review. `https://developers.facebook.com/docs/app-rev iew`, Accessed: August 2023.

[32] Facebook. Graph API. `https://developers.facebook.com/docs/graph-a pi/`, Accessed: August 2023.

[33] Facebook. Permissions Reference. `https://developers.facebook.com/docs /permissions/reference/`, Accessed: October 2023.

[34] Facebook Login. Introduction - App Review. `https://developers.faceboo k.com/docs/app-review/introduction`, Accessed: April 2023.

[35] Shehroze Farooqi, Maaz Musa, Zubair Shafiq, and Fareed Zaffar. Canarytrap: Detecting Data Misuse by Third-Party Apps on Online Social Networks. *Proceedings on Privacy Enhancing Technologies*, 2020(4):336–354, 2020.

[36] Adrienne Porter Felt, Serge Egelman, Matthew Finifter, Devdatta Akhawe, and David Wagner. How to Ask for Permission. In *Proceedings of the 7th USENIX Conference on Hot Topics in Security*, HotSec'12, page 7, 2012.

[37] Adrienne Porter Felt and David Evans. Privacy Protection for Social Networking APIs. *Web 2.0 Security and Privacy (W2SP)*, 2008.

[38] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android Permissions: User Attention, Comprehension, and Behavior. In *SOUPS*, 2012.

[39] Daniel Fett, John Bradley, Brian Campbell, Torsten Lodderstedt, and Michael B. 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.

[40] Daniel Fett, Ralf Küsters, and Guido Schmitz. SPRESSO: A Secure, Privacy-Respecting Single Sign-On System for the Web. In *ACM CCS*, 2015.

[41] Daniel Fett, Ralf Küsters, and Guido Schmitz. A Comprehensive Formal Security Analysis of OAuth 2.0. In *ACM CCS*, 2016.

[42] Daniel Fett, Ralf Küsters, and Guido Schmitz. The Web SSO Standard OpenID Connect: In-depth Formal Security Analysis and Security Guidelines. In *IEEE Computer Security Foundations Symposium (CSF)*, 2017.

[43] Firefox. Relay Email Mask. `https://relay.firefox.com/`, Accessed: October 2023.

[44] Sean Gallagher. 50 million Facebook accounts breached by access-token-harvesting attack. `https://arstechnica.com/information-technology/2018/09/50-million-facebook-accounts-breached-by-an-access-token-harvesting-attack/`, September 28, 2018.

[45] Mohammad Ghasemisharif, Amrutha Ramesh, Stephen Checkoway, Chris Kanich, and Jason 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.

[46] Robert Gilbert. Information Exposure Through Query Strings in URL. `https://owasp.org/www-community/vulnerabilities/Information_exposure_through_query_strings_in_url`, Accessed: June 2021.

[47] Google. OAuth API verification FAQs. `https://support.google.com/cloud/answer/9110914`, Accessed: April 2023.

[48] Google. Google API for Authentication. `https://developers.google.com/identity/sign-in/web/reference`, Accessed: August 2023.

[49] Google. OAuth API verification FAQs. `https://support.google.com/cloud/answer/9110914`, Accessed: January 2021.

[50] Google. AdSense Host API Reference. `https://developers.google.com/adsense/host/v4.1`, Accessed: October 2023.

[51] Google. OAuth 2.0 Scopes for Google APIs. `https://developers.google.com/identity/protocols/oauth2/scopes`, Accessed: October 2023.

[52] Sven Hammann, Ralf Sasse, and David Basin. Privacy-Preserving OpenID Connect. In *AsiaCCS*, 2020.

[53] Dick Hardt. RFC 6749: The OAuth 2.0 Authorization Framework. `https://tools.ietf.org/html/rfc6749`, 2012.

[54] Dick Hardt, Aaron Parecki, and Torsten Lodderstedt. Internet Draft: The OAuth 2.1 Authorization Framework. `https://datatracker.ietf.org/doc/html/draft-ietf-oauth-v2-1-08`, 2023.

[55] Florian Helmschmidt, Pedram Hosseyni, Ralf Kuesters, Klaas Pruiksma, Clara Waldmann, and Tim Würtele. The Grant Negotiation and Authorization Protocol: Attacking, Fixing, and Verifying an Emerging Standard. Cryptology ePrint Archive, Paper 2023/1325, 2023. `https://eprint.iacr.org/2023/1325`.

[56] Louis Jannett, Vladislav Mladenov, Christian Mainka, and Jörg Schwenk. DISTINCT: Identity Theft using In-Browser Communications in Dual-Window Single Sign-On. In *ACM CCS*, 2022.

[57] Michael B. Jones, John Bradley, and Nat Sakimura. RFC 7519: JSON Web Token (JWT). `https://tools.ietf.org/html/rfc7519`, 2015.

[58] Oscar Järpehult, Fredrik Josefsson Ågren, Madeleine Bäckström, Linn Hallonqvist, and Niklas Carlsson. A Longitudinal Characterization of the Third-Party Authentication Landscape. In *International Federation for Information Processing (IFIP) Networking*, 2022.

[59] Farzaneh Karegar, Christoph Striecks, Stephan Krenn, Felix Hörandner, Thomas Lorünser, and Simone Fischer-Hübner. Opportunities and Challenges of CREDENTIAL: Towards a Metadata-Privacy Respecting Identity Provider. *Privacy and Identity. IFIP Advances in Information and Communication Technology*, pages 76–91, 2016.

[60] Patrick Gage Kelley, Lucian Cesca, Joanna Bresee, and Lorrie Faith Cranor. Standardizing Privacy Notices: An Online Study of the Nutrition Label Approach. In *CHI*, 2010.

[61] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *NDSS*, 2019.

[62] Wanpeng Li, Chris J. Mitchell, and Thomas Chen. OAuthGuard: Protecting User Security and Privacy with OAuth 2.0 and OpenID Connect. In *Workshop on Security Standardisation Research*, 2019.

[63] LinkedIn. Sign In with LinkedIn. `https://docs.microsoft.com/en-us/linkedin/consumer/integrations/self-serve/sign-in-with-linkedin`, Accessed: January 2021.

[64] LinkedIn. Sign In with LinkedIn using OpenID Connect. `https://learn.microsoft.com/en-us/linkedin/consumer/integrations/self-serve/sign-in-with-linkedin-v2`, Accessed: October 2023.

[65] Bin Liu, Mads Schaarup Andersen, Florian Schaub, Hazim Almuhimedi, Shikun (Aerin) Zhang, Norman Sadeh, Yuvraj Agarwal, and Alessandro Acquisti. Follow My Recommendations: A Personalized Privacy Assistant for Mobile App Permissions. In *SOUPS*, pages 27–41, June 2016.

[66] Christian Mainka, Vladislav Mladenov, and Jörg Schwenk. Do Not Trust Me: Using Malicious IdPs for Analyzing and Attacking Single Sign-on. In *IEEE EuroS&P*, 2016.

[67] Christian Mainka, Vladislav Mladenov, Jörg Schwenk, and Tobias Wich. SoK: Single Sign-On Security — An Evaluation of OpenID Connect. In *IEEE EuroS&P*, 2017.

[68] Arunesh Mathur, Gunes Acar, Michael J. Friedman, Elena Lucherini, Jonathan Mayer, Marshini Chetty, and Arvind Narayanan. Dark Patterns at Scale: Findings from a Crawl of 11K Shopping Websites. *ACM Human-Computer Interaction*, 3(CSCW), 2019.

[69] Jonathan R. Mayer and John C. Mitchell. Third-Party Web Tracking: Policy and Technology. In *IEEE Symp. Security and Privacy*, 2012.

[70] Srivathsan G. Morkonda, Sonia Chiasson, and Paul C. van Oorschot. Empirical Analysis and Privacy Implications in OAuth-based Single Sign-On Systems. In *Workshop on Privacy in the Electronic Society*, 2021.

[71] Srivathsan G. Morkonda, Sonia Chiasson, and Paul C. van Oorschot. "Sign in with ... *Privacy*": Timely Disclosure of Privacy Differences among Web SSO Login Options. *Under submission*, 2023. A preliminary version is at: `https://arxiv.org/abs/2209.04490`.

[72] Srivathsan G. Morkonda, Sonia Chiasson, and Paul C. van Oorschot. Influences of Displaying Permission-related Information on Web Single Sign-On Login Decisions. *Computers & Security*, 139:103666, 2024. 15 pages. `doi:https://doi.org/10.1016/j.cose.2023.103666`.

[73] Mozilla Thunderbird. Automatic Conversion of Google mail accounts to OAuth 2.0 Authentication. `https://support.mozilla.org/en-US/kb/automatic-conversion-google-mail-accounts-oauth20`, Accessed: October 2023.

[74] Arvind Narayanan, Arunesh Mathur, Marshini Chetty, and Mihir Kshirsagar. Dark Patterns: Past, Present, and Future. *ACM Queue*, 18(2), 2020.

[75] Clifford Neuman, Tom Yu, Sam Hartman, and Kenneth Raeburn. RFC 4120: The Kerberos Network Authentication Service (V5). `https://datatracker.ietf.org/doc/html/rfc4120`, 2005.

[76] Lily Hay Newman. Think Twice Before Using Facebook, Google, or Apple to Sign In Everywhere. `https://www.wired.com/story/single-sign-on-facebook-google-apple/`, September 21, 2020.

[77] OASIS. OASIS Standards. `https://www.oasis-open.org/standards/#samlv2.0`, Accessed: September 2023.

[78] OWASP. Cross-Site Request Forgery Prevention Cheat Sheet: Storing the CSRF Token Value in the DOM. `https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#storing-the-csrf-token-value-in-the-dom`, Accessed: June 2022.

[79] Aaron Parecki. It's Time for OAuth 2.1. `https://aaronparecki.com/2019/12/12/21/its-time-for-oauth-2-dot-1`, December 12, 2019.

[80] Aaron Parecki. Is the OAuth 2.0 Implicit Flow Dead? `https://developer.okta.com/blog/2019/05/01/is-the-oauth-implicit-flow-dead`, May 1, 2019.

[81] Paul C. van Oorschot. *Computer Security and the Internet: Tools and Jewels from Malware to Bitcoin*. Springer Nature, 2nd edition, 2021.

[82] Greta L. Polites and Elena Karahanna. Shackled to the Status Quo: The Inhibiting Effects of Incumbent System Habit, Switching Costs, and Inertia on New System Acceptance. *MIS quarterly*, pages 21–42, 2012.

[83] Tamjid Al Rahat, Yu Feng, and Yuan Tian. OAuthLint: An Empirical Study on OAuth Bugs in Android Applications. In *International Conference on Automated Software Engineering (ASE)*, 2019.

[84] Tamjid Al Rahat, Yu Feng, and Yuan Tian. Cerberus: Query-driven Scalable Security Checking for OAuth Service Provider Implementations. In *ACM CCS*, 2022.

[85] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. 50 Ways to Leak Your Data: An Exploration of Apps' Circumvention of the Android Permissions System. In *USENIX Security*, 2019.

[86] Justin Richer and Fabien Imbault. Grant Negotiation and Authorization Protocol. `https://datatracker.ietf.org/doc/html/draft-ietf-gnap-core-protocol`, 26 June 2023. Intended Status: Standard track, currently Internet Draft.

[87] Nicky Robinson and Joseph Bonneau. Cognitive Disconnect: Understanding Facebook Connect Login Permissions. In *ACM COSN*, 2014.

[88] Guy Rosen. Facebook Security Update - Security issue affecting almost 50 million accounts. `https://about.fb.com/news/2018/09/security-update/`, September 28, 2018.

[89] Nat Sakimura, John Bradley, and Naveen Agarwal. RFC 7636: Proof Key for Code Exchange by OAuth Public Clients. `https://tools.ietf.org/html/rfc7636`, 2015.

[90] Nat Sakimura, John Bradley, Michael B. Jones, Breno de Medeiros, and Chuck Mortimore. OpenID Connect Core 1.0. `https://openid.net/specs/openid-connect-core-1_0.html`, 2014.

[91] Ludwig Seitz, Göran Selander, Erik Wahlstroem, Samuel Erdtman, and Hannes Tschofenig. RFC 9200: Authentication and Authorization for Constrained Environments Using the OAuth 2.0 Framework (ACE-OAuth). `https://datatracker.ietf.org/doc/rfc9200/`, 2022.

[92] Selenium. Selenium WebDriver. `https://www.selenium.dev/documentation/en/webdriver/`, Accessed: January 2021.

[93] Mohamed Shehab, Said Marouf, and Christopher Hudel. ROAuth: Recommendation Based Open Authorization. In *SOUPS*, 2011.

[94] Shibboleth Consortium. Shibboleth Single Sign-On Software. `https://www.shibboleth.net/`, Accessed: September 2023.

[95] Sign in with Apple. Communicating using the Private Email Relay Service. `https://developer.apple.com/documentation/sign_in_with_apple/sign_in_with_apple_js/communicating_using_the_private_email_relay_service`, Accessed: April 2023.

[96] Sign in with Google. Setup - Configure your OAuth Consent Screen. `https://developers.google.com/identity/gsi/web/guides/get-google-api-clientid`, Accessed: April 2023.

[97] Manish 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`, January 29, 2018.

[98] Sarah Spiekermann and Lorrie Faith Cranor. Engineering Privacy. *IEEE Transactions on Software Engineering*, 35(1):67–82, 2008.

[99] Elizabeth Stobert and Robert Biddle. The Password Life Cycle: User Behaviour in Managing Passwords. In *SOUPS*, 2014.

[100] San-Tsai Sun and Konstantin Beznosov. The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems. In *ACM CCS*, 2012.

[101] San-Tsai Sun, Eric Pospisil, Ildar Muslukhov, Nuray Dindar, Kirstie Hawkey, and Konstantin Beznosov. What Makes Users Refuse Web Single Sign-On? An Empirical Investigation of OpenID. In *SOUPS*, 2011.

[102] Twitter. Log in with Twitter. `https://developer.twitter.com/en/docs/authentication/guides/log-in-with-twitter`, Accessed: October 2023.

[103] VMware. Spring Framework. `https://spring.io/projects/spring-framework`, Accessed: October 2023.

[104] Kurt Wagner. Here's how Facebook allowed Cambridge Analytica to get data for 50 million users. `https://www.vox.com/2018/3/17/17134072/facebook-cambridge-analytica-trump-explained-user-data`, March 17, 2018.

[105] Na Wang, Heng Xu, and Jens Grossklags. Third-Party Apps on Facebook: Privacy and the Illusion of Control. In *ACM Symposium on Computer Human Interaction for Management of Information Technology*, 2011.

[106] Rui Wang, Shuo Chen, and XiaoFeng 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.

[107] Rui Wang, Yuchen Zhou, Shuo Chen, Shaz Qadeer, David Evans, and Yuri Gurevich. Explicating SDKs: Uncovering Assumptions Underlying Secure Authentication and Authorization. In *USENIX Security*, 2013.

[108] Ben Weinshel, Miranda Wei, Mainack Mondal, Euirim Choi, Shawn Shan, Claire Dolin, Michelle L Mazurek, and Blase Ur. Oh, the Places You've Been! User Reactions to Longitudinal Transparency About Third-Party Web Tracking and Inferencing. In *ACM CCS*, pages 149–166, 2019.

[109] Maximilian Westers, Tobias Wich, Louis Jannett, Vladislav Mladenov, Christian Mainka, and Andreas Mayer. SSO-Monitor: Fully-Automatic Large-Scale Landscape, Security, and Privacy Analyses of Single Sign-On in the Wild. *arXiv:2302.01024 [cs]*, Feb 2023. URL: `https://arxiv.org/abs/2302.01024`.

[110] Primal Wijesekera, Arjun Baokar, Lynn Tsai, Joel Reardon, Serge Egelman, David Wagner, and Konstantin Beznosov. The Feasibility of Dynamically Granted Permissions: Aligning Mobile Privacy with User Preferences. In *IEEE Symp. Security and Privacy*, pages 1077–1093, 2017.

[111] Wikipedia. List of OAuth providers. `https://en.wikipedia.org/wiki/List_of_OAuth_providers`, Accessed: June 2022.

[112] Rongwu Xu, Sen Yang, Fan Zhang, and Zhixuan Fang. MISO: Legacy-compatible Privacy-preserving Single Sign-on using Trusted Execution Environments. *IEEE EuroS&P*, 2023.

[113] Ronghai Yang, Guanchen Li, Wing Cheong Lau, Kehuan Zhang, and Pili Hu. Model-based Security Testing: An Empirical Study on OAuth 2.0 Implementations. In *AsiaCCS*, 2016.

[114] Yuchen Zhou and David Evans. SSOScan: Automated Testing of Web Applications for Single Sign-On Vulnerabilities. In *USENIX Security*, 2014.

[115] Paolo Zialcita. Facebook Pays $643,000 Fine For Role In Cambridge Analytica Scandal. `https://www.npr.org/2019/10/30/774749376/facebook-pays-643-000-fine-for-role-in-cambridge-analytica-scandal`, October 30, 2019.

# Appendix A

## Study Questionnaire

**Basic demographic and SSO usage questions**

1. What is your Prolific ID? _____

2. What is your gender?

   - Man

   - Woman

   - Non-binary

   - Not listed above [please specify]: _____

   - Prefer not to answer

3. Please enter your age in years. If you prefer not to answer, enter 100 _____

4. What is your highest level of education (completed or currently enrolled)?

   - High school diploma or lower

   - College or associate degree

   - Bachelor's degree

   - Advanced degree (e.g., Master's, PhD)

   - Prefer not to answer

   - Other [please specify]: _____

5. How many different websites do you log into on a typical day? [0-2], [3-5], [6-8], [9-11], [12+], [prefer not to answer]

6. Single Sign-On (SSO) includes services that allow you to login to a website using another account (e.g., "Sign in with Facebook", "Login with Google", "Continue with Apple", etc.).

   Have you ever used single sign-on (SSO) to login to a website? [Yes], [No], [Unsure], [Prefer not to answer]

   *page break – participants could not change earlier responses*

   *Participants were shown one of Group A, B, C, or D*

## Group A - Airbnb.ca

1. If you were to login to Airbnb.ca, which login option would you choose?



- Continue with Phone number

- Continue with Facebook

- Continue with Google

- Continue with Apple

- Continue with email

- Prefer not to answer

- I would take some other action [please specify]: _____

2. Why did you choose this option? If you prefer not to answer, enter N/A (No Answer) _____

*page break – participants could not change earlier responses*

3. If the following information was available when logging in to Airbnb.ca, which login option would you choose?
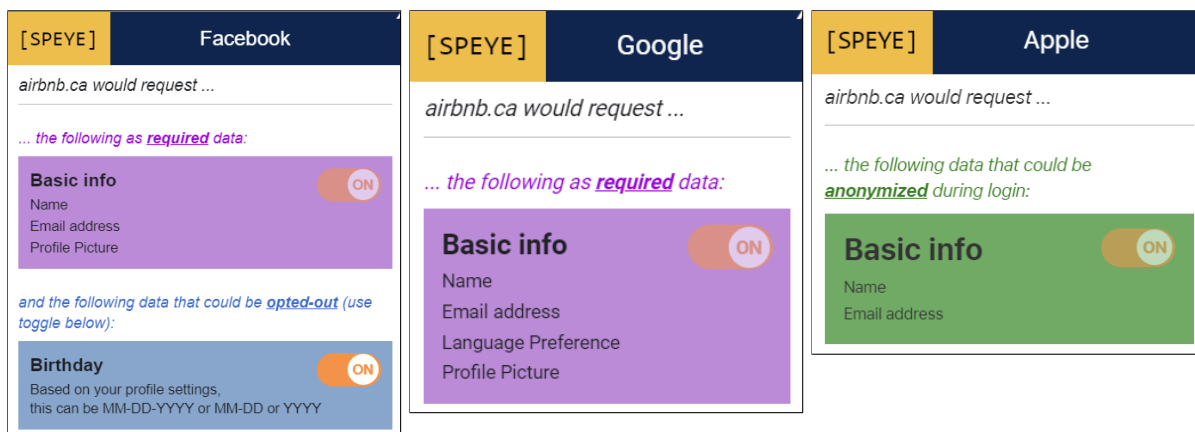


*Airbnb.ca login prompt from Q1 displayed again here*

- Continue with Phone number

- Continue with Facebook (with all requested data included)

- Continue with Facebook (with birthday data opted out)

- Continue with Google

- Continue with Apple (with real name and email address)

- Continue with Apple (with anonymized name and email address)

- Continue with email

- Prefer not to answer

- I would take some other action [please specify]: _____

4. Why did you choose this option? If you prefer not to answer, enter N/A (No Answer) _____

*page break – participants could not change earlier responses*

5. Indicate how much each of the following impacted your chosen login option in the previous question.

   *Reminder: You selected <copy Q3 response> in the previous question*

   *[Displayed only if "Prefer not to answer" was not selected in Q3]*

   *[Likert scale responses - 1 (strongly disagree) to 5 (strongly agree)]*

   a. I chose this login option because I have an existing account with the SSO provider. †

   b. I chose this login option because I trust the SSO provider. †

   c. I chose this login option because it was listed first in the login prompt.

   d. I chose this login option because it requested less data than other options.

   e. I chose this login option because it lets me opt-out of requested data. †

   f. I chose this login option because it lets me anonymize my data. †

   g. I don't want to use SSO on Airbnb.ca

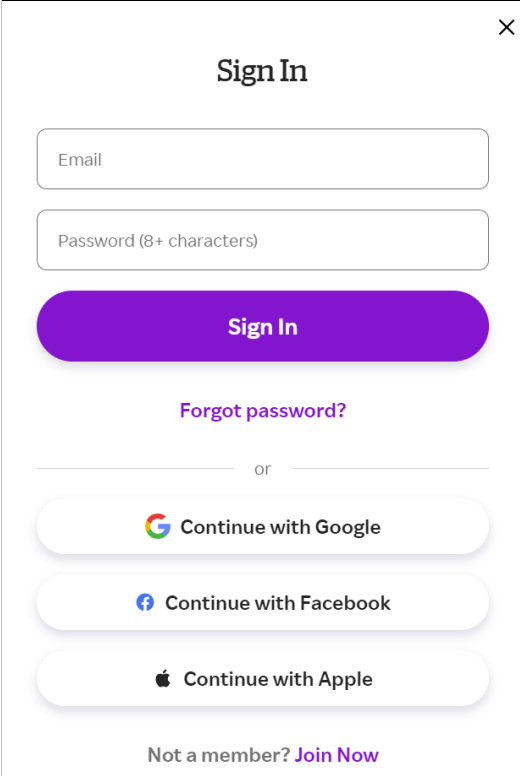[† *Displayed only if an SSO login option was selected in Q3*]

6. Do you have an existing account with Airbnb?

   [Yes] [No] [I don't know] [Prefer not to answer]

*page break – participants could not change earlier responses*

**Group B - Rakuten.com**

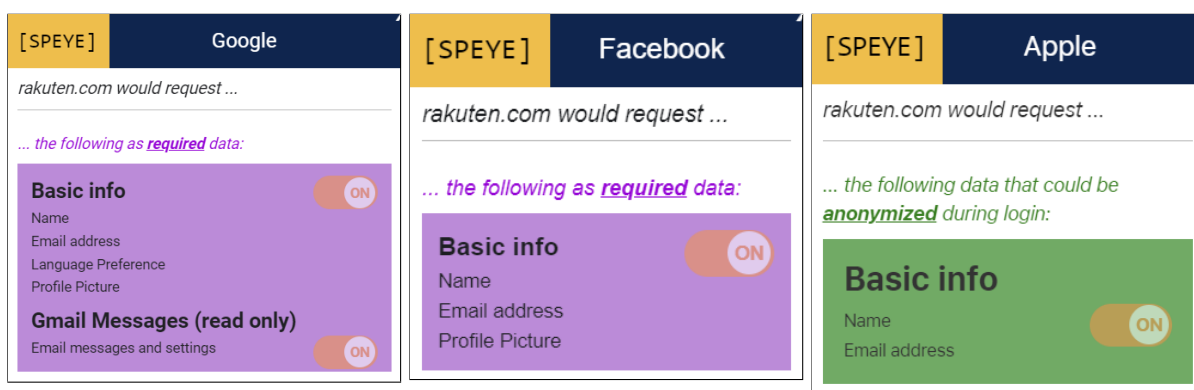1. If you were to login to Rakuten.com, which login option would you choose?



- Sign in using Email and Password

- Continue with Google

- Continue with Facebook

- Continue with Apple

- Prefer not to answer

- I would take some other action [please specify]: _____

2. Why did you choose this option? If you prefer not to answer, enter N/A (No Answer) _____

*page break – participants could not change earlier responses*

3. If the following information was available when logging in to Rakuten.com, which login option would you choose?



*Rakuten.com login prompt from Q1 displayed again here*

- Sign in using Email and Password
- Continue with Google
- Continue with Facebook
- Continue with Apple (with real name and email address)
- Continue with Apple (with anonymized name and email address)
- Prefer not to answer
- I would take some other action [please specify]: _____

4. Why did you choose this option? If you prefer not to answer, enter N/A (No Answer) _____

*page break – participants could not change earlier responses*

5. Indicate how much each of the following impacted your chosen login option in the previous question.

   *Reminder: You selected <copy Q3 response> in the previous question*

   [*Displayed only if "Prefer not to answer" was not selected in Q3*]

   [*Likert scale responses - 1 (strongly disagree) to 5 (strongly agree)*]

   a. I chose this login option because I have an existing account with the SSO provider. †

   b. I chose this login option because I trust the SSO provider. †

   c. I chose this login option because it was listed first in the login prompt.

   d. I chose this login option because it requested less data than other options.

   e. I chose this login option because it lets me opt-out of requested data. †

   f. I chose this login option because it lets me anonymize my data. †

   g. I don't want to use SSO on Rakuten.com

   [† *Displayed only if an SSO login option was selected in Q3*]

6. Do you have an existing account with Rakuten?
   [Yes] [No] [I don't know] [Prefer not to answer]

*page break – participants could not change earlier responses*

**Group C - NYTimes.com**

1. If you were to login to NYTimes.com, which login option would you choose?



- Log in with Email Address

- Continue with Google

- Continue with Facebook

- Continue with Apple

- Prefer not to answer

- I would take some other action [please specify]: _____

2. Why did you choose this option? If you prefer not to answer, enter N/A (No Answer) _____

*page break – participants could not change earlier responses*

3. If the following information was available when logging in to NYTimes.com, which login option would you choose?
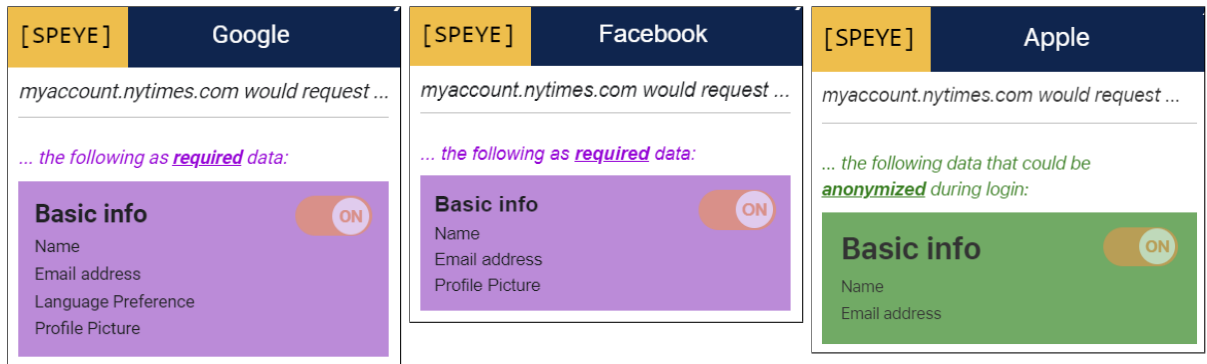


*NYTimes.com login prompt from Q1 displayed again here*

- Log in with Email Address

- Continue with Google

- Continue with Facebook

- Continue with Apple (with real name and email address)

- Continue with Apple (with anonymized name and email address)

- Prefer not to answer

- I would take some other action [please specify]: _____

4. Why did you choose this option? If you prefer not to answer, enter N/A (No Answer) _____

*page break – participants could not change earlier responses*

5. Indicate how much each of the following impacted your chosen login option in the previous question.

   *Reminder: You selected <copy Q3 response> in the previous question*

   *[Displayed only if "Prefer not to answer" was not selected in Q3]*

   *[Likert scale responses - 1 (strongly disagree) to 5 (strongly agree)]*

   a. I chose this login option because I have an existing account with the SSO provider. †

   b. I chose this login option because I trust the SSO provider. †

   c. I chose this login option because it was listed first in the login prompt.

   d. I chose this login option because it requested less data than other options.

   e. I chose this login option because it lets me opt-out of requested data. †

   f. I chose this login option because it lets me anonymize my data. †

   g. I don't want to use SSO on NYTimes.com

   *[† Displayed only if an SSO login option was selected in Q3]*

6. Do you have an existing account with NYTimes?
   [Yes] [No] [I don't know] [Prefer not to answer]

   *page break – participants could not change earlier responses*

## Group D - CBC.ca

1. If you were to login to CBC.ca, which login option would you choose?

- Log in with Facebook

- Log in with Google

- Log in with Apple

- Log in with Email and Password

- Prefer not to answer

- I would take some other action [please specify]: _____

2. Why did you choose this option? If you prefer not to answer, enter N/A (No Answer) _____

*page break – participants could not change earlier responses*

3. If the following information was available when logging in to CBC.ca, which login option would you choose?

*CBC.ca login prompt from Q1 displayed again here*

- Log in with Facebook (with all requested data included)

- Log in with Facebook (with 1+ permissions opted-out)

- Log in with Google

- Log in with Apple (with real name and email address)

- Log in with Apple (with anonymized name and email address)

- Log in with Email and Password

- Prefer not to answer

- I would take some other action [please specify]: _____

4. Why did you choose this option? If you prefer not to answer, enter N/A (No Answer) _____

*page break – participants could not change earlier responses*

5. Indicate how much each of the following impacted your chosen login option in the previous question.

   *Reminder: You selected <copy Q3 response> in the previous question*

   *[Displayed only if "Prefer not to answer" was not selected in Q3]*

   *[Likert scale responses - 1 (strongly disagree) to 5 (strongly agree)]*

   a. I chose this login option because I have an existing account with the SSO provider. †

   b. I chose this login option because I trust the SSO provider. †

   c. I chose this login option because it was listed first in the login prompt.

   d. I chose this login option because it requested less data than other options.

   e. I chose this login option because it lets me opt-out of requested data. †

   f. I chose this login option because it lets me anonymize my data. †

   g. I don't want to use SSO on CBC.ca

   *[† Displayed only if an SSO login option was selected in Q3]*

6. Do you have an existing account with CBC?
   [Yes] [No] [I don't know] [Prefer not to answer]

*page break – participants could not change earlier responses*

**Single Sign-On Use**

1. Outside of work, on how many different websites do you use single sign-on (SSO) login?

- I don't use SSO

- 1-5 websites

- 6-10 websites

- 11-15 websites

- 16-20 websites

- 21+ websites

- Prefer not to answer

2. Answer the following questions about using single sign-on (SSO) login outside of work.

   [*Likert scale responses - 1 (strongly disagree) to 5 (strongly agree)*]

   a. SSO is my preferred login method

   b. I find it easy to use SSO login

   c. Using SSO login helps reduce the number of passwords I need to manage.

   d. I pay close attention to the types of data requested during SSO login.

   e. The ability to opt-out of data requested during SSO login is important to me.

3. Outside of work, with which of following do you have an account? [Select all that apply]

   ☐ Apple

   ☐ Facebook

   ☐ GitHub

   ☐ Google

   ☐ LinkedIn

   ☐ Microsoft

   ☐ Twitter

   ☐ Vk

   ☐ Yahoo!

   ☐ Prefer not to answer

   ☐ None of these

4. Other than for SSO login, how often do you use applications and/or services related to accounts from these providers (selected in the previous question) outside of work?

*[Displayed if "Prefer not to answer" or "None of these" was not selected in Q3]*

*[Likert scale rows for each choice selected by the participant in Q3]*

| At least once a day | At least once a week | At least once a month | At least once a year | Never |
|:---:|:---:|:---:|:---:|:---:|
| ○ | ○ | ○ | ○ | ○ |

5. Which of the following accounts have you used at least once for SSO login outside of work?

☐ I don't use SSO      ☐ Microsoft

☐ Apple      ☐ Twitter

☐ Facebook      ☐ Vk

☐ GitHub      ☐ Yahoo!

☐ Google      ☐ Prefer not to answer

☐ LinkedIn      ☐ Other [please specify]: _____

# Appendix B

# SPEye Implementation Details

This section provides details on SPEye's implementation discussed in Section 4.3.

SPEye identifies SSO login options in RP sites using CSS Selectors to search the DOM for strings such as "Sign in with" and related elements. We built these Selectors based on strings (converted to lowercase for comparison) and associated HTML elements found in 21 RP sites in our training set, as described in Section 4.2.3. Table B.1 (page 145) shows the distribution of the SSO strings we found in the 36 HTML-based (including RPs in both the training and testing sets) SSO implementations.

To identify IdP endpoints related to SSO login options (i.e., IdP authorization servers), we used regular expressions (shown in Listing 5, page 145) derived from IdP documentation.

Table B.1: Number of RP sites that display SSO login options using the specific strings within HTML elements.

| SSO string | DOM element | # of RP sites (N=36) |
|---|---|---|
| sign in with | span/text() | 5 |
| sign in with | div/text() | 3 |
| sign in with | a/text() | 2 |
| sign in with | small/text() | 1 |
| sign in | button/text() | 2 |
| continue with | span/text() | 3 |
| continue with | div/text() | 3 |
| continue with | a/text() | 3 |
| continue with | button/text() | 1 |
| log in with | span/text() | 3 |
| log in with | div/text() | 1 |
| log in with | p/text() | 1 |
| login with | a/text() | 2 |
| login with | p/text() | 1 |
| login via | p/text() | 1 |
| connect using | span/@data-text | 1 |
| or use | span/text() | 1 |
| idp link | a/@title | 1 |
| idp link | iframe/@src | 1 |

```
https://(.*)\\.facebook\\.com/login(.*)

https://(.*)\\.facebook\\.com/oauth(.*)

https://graph\\.facebook\\.com/(.*)

https://(.*)\\.facebook\\.com/(.*)/oauth(.*)

https://(.*)\\.google\\.com/(.*)/oauth(.*)

https://oauth2\\.googleapis\\.com/(.*)

https://openidconnect\\.googleapis\\.com/(.*)

https://googleapis\\.com/oauth(.*)

https://(.*)\\.apple\\.com/auth(.*)
```

Listing 5: Regular expressions for identifying IdP endpoints.