Privacy-Preserving Web Single Sign-On: Formal Security Analysis and Design

Guido Schmitz

Abstract: Single sign-on (SSO) systems, such as OpenID and OAuth, allow Web sites to delegate user authentication to third parties, such as Facebook or Google. These systems provide a convenient mechanism for users to log in and ease the burden of user authentication for Web sites. Conversely, by integrating such SSO systems, they become a crucial part of the security of the modern Web.

So far, it has been hard to prove if Web standards and protocols actually meet their security goals. SSO systems, in particular, need to satisfy strong security and privacy properties. In this thesis, we develop a new systematic approach to rigorously and formally analyze and verify such strong properties with the Web Infrastructure Model (WIM), the most comprehensive model of the Web infrastructure to date.

Our analyses reveal severe vulnerabilities in SSO systems that lead to critical attacks against their security and privacy. We propose fixes and formally verify that our proposals are sufficient to establish security. Our analyses, however, also show that even Mozilla's proposal for a privacy-preserving SSO system does not meet its unique privacy goal. To fill this gap, we use our novel approach to develop a new SSO system, SPRESSO, and formally prove that our system indeed enjoys strong security and privacy properties.

ACM CCS: Networks → Network protocols → Protocol correctness; Networks → Network protocols → Network protocol design; Security and privacy → Formal methods and theory of security; Security and privacy → Network security → Web protocol security

Keywords: formal analysis, single sign-on, authentication, privacy, web security

1 Introduction

Many important and security-critical applications in our modern society are based on the World Wide Web, for example, social networks, commercial services, or e-government portals. Users are enabled by this universal information and communication platform to share private data, enter contracts, and even make monetary transactions online.

When users perform such sensitive actions, one important aspect is to *authenticate* users in a secure way. This means that a user needs to prove her identity to the services she wants to use. Only after successful authentication, a Web service should grant access to its services and the user's resources stored at this service. For example, a social network needs to ensure that a user successfully authenticates herself before the user gets access to her personal messages and is allowed to post content.

Traditionally, each service authenticates its users on its own, typically by prompting the user for her username and password. To this end, the service needs to maintain its own password database to be able to verify the passwords provided by its users. Secure password storage and management are non-trivial and hard to implement securely. Best practice recommendations (see, e.g., [17]) are complex, extensive, and updated quite often. Furthermore, many Web services realize password management in their own proprietary way leading to many different implementations in the wild. The heterogeneity of implementations and also responsibilities spread across many parties make it hard to identify and fix (common) mistakes and security problems or to update authentication mechanisms, e.g., to support two-factor authentication. Also, all of these services need to individually provide user support regarding authentication, such as password reset procedures.

For users, password management is non-trivial as well. They have to memorize many different passwords and

are confronted with various different authentication dialogs. Hence, a user can easily provide one of her passwords to an incorrect, potentially malicious party by mistake leading to a compromised account.

2 Web Single Sign-On

The concept of *Web Single Sign-On (Web SSO)* promises to relieve users and Web services from the burden of handling passwords. Web SSO systems enable Web sites, so-called *relying parties (RPs)*, to outsource user authentication to other entities, so-called *identity providers (IdPs)*. The task of an IdP in Web SSO is to authenticate users and attest their identities to RPs. Typically, some services that users use on a daily basis, such as email providers or social networks, serve as IdPs.

Many different SSO protocols have been developed so far and are widely deployed. For the Web, the most important protocols include *OAuth 1.0a* [15] (used, for example, by Twitter), *OAuth 2.0* [16] (used, e.g., by Facebook), *OpenID 2.0* [14] (used, e.g., by Yahoo), *OpenID Connect 1.0* [18] (used, e.g., by Google and Microsoft), and *Security Assertion Markup Language 2.0 (SAML)* [19] (used, e.g., by Amazon AWS).

Web SSO provides many advantages for all parties. A user only needs to remember her credentials for one account (her account at the IdP) to log in at many different parties. If the user is already logged in at the IdP, the IdP might enable her to log in at many RPs without further interaction. Logging in at an RP where the user did not log in before also becomes easier as the user does not need to register any authentication data at this RP. Using the SSO system, the user experience is always the same in all cases: the user only needs to interact with a familiar user interface (also reducing the user's susceptibility for phishing attacks).

An SSO system allows the RP to outsource almost all authentication-related tasks to a (set of) IdP(s). Hence, RPs do not need to develop and set up user registration and management systems and also do not need to handle user support concerning authentication (e.g., reset lost passwords). Furthermore, RPs are relieved from storing and protecting user credentials, which, as already mentioned, is not a trivial task.

While an SSO system seems to only bring high responsibility and cost to IdPs at first, also entities acting as IdPs can benefit from providing an SSO service: The IdP service provides added value for their users and therefore makes their services more appealing to new users.

Fundamental to SSO is security: An SSO protocol must ensure correct *authentication*, i.e., that nobody except the user herself should be authenticated under her identity to an RP. This means, in particular, that an adversary should be unable to authenticate himself to an RP as some (different) user, i.e., the attacker cannot "break-in" into the user's account at that RP. If this property is not satisfied by the SSO protocol, an adversary can easily access other users' data or impersonate these users at RPs.

Another important but not so obvious aspect is the *integrity of the user's login session*. That is that an attacker should also not be able to force a user to log in at some RP or even to manipulate a login process such that the user is logged in under some different account. The outcome of such attacks would be similar to the attack class of session swapping: The user might then interact with the RP under the attacker's identity and, for example, store confidential data at the RP that is then accessible under the attacker's account at the RP.

While aiming at security, most SSO systems, however, neglect *privacy*: IdPs can typically track their users as they (by design) learn at which RP a user logs in. This lack of privacy allows IdPs to create extensive user profiles and might cause some users not to use SSO at all. Moreover, IdPs are enabled to decide ad-hoc whether they allow a user to log in at a specific RP. Therefore, we need privacy-preserving SSO systems, which do not reveal to IdPs to which RP a user would like to log in or has logged in. The design of such systems, however, is very challenging as privacy can easily be compromised. Only one SSO system has been proposed with this kind of privacy in mind: Mozilla's BrowserID (a.k.a. Mozilla Persona) [1].

3 Analysis of Web Standards

So far, there has not been a systematic and rigorous methodology to check whether a Web standard (or application) meets its security and privacy goals. The typical approach is to have groups of experts examine a protocol closely (often during the standardization process), e.g., by checking whether known attacks are applicable. These experts might also discuss the protocol in detail and try to find new attacks, which obviously depends on the creativity of the analysts. Often, such analyses lack clear attacker models and make many implicit assumptions. Clearly, the result of such an analysis does not prove the non-existence of unknown attacks. Further, while these experts have some intuition about the properties they want to prove, they cannot state these properties in a precise fashion as they only use natural language to describe a more or less vague concept.

In my thesis [20], we tackle this problem by using formal methods to specify and analyze the security and privacy of SSO protocols. Our analyses are based on the *Web Infrastructure Model (WIM)* [8]. The WIM is the most comprehensive formal model of the Web infrastructure to date and can be applied to a wide range of Web applications and standards. At its core, the WIM is based on the concept of symbolic analysis by Dolev and Yao [6], where messages are formal terms that can be assembled

and derived based on an equational theory. Such messages can be exchanged among principals over a public network, which can be controlled by an adversary.

The WIM includes many relevant aspects of the Web. Besides basic infrastructure, such as Web and DNS servers, the WIM includes a very detailed model of Web browsers. This browser model captures many Web features, such as the handling of DNS, HTTP, and HTTPS messages, a detailed structure of windows and documents, an abstract model of JavaScript, Web storage and cookies, Web messaging (postMessage) and asynchronous HTTP communication (XMLHTTPRequests), a rich set of HTTP headers, HTTP redirections as well as security policies for cross-window navigation and access.

The WIM provides a concise view on all of these aspects and allows the modeling of complex modern Web applications and protocols. We can even capture systems composed of multi-window browser-based interactions across several entities.

4 Web SSO in the WIM

We create a generic framework for analyses of Web-based SSO systems based on the WIM. This framework provides a generalized definition of Web SSO that simplifies and unifies analyses of such systems and hence, enables easy comparison of different Web SSO protocols. Further, we also extend the WIM to express and prove privacy properties.

To create the generalized definition of Web SSO, we identify general characteristics of such systems and capture these in a generic template. This template captures, for example, important user-driven events such as the start of a login flow or the user entering her password. When modeling a concrete SSO system, we refine the template to create a concrete definition for the SSO system. This template simplifies the modeling and analysis and is also reusable for others to analyze SSO systems.

Based on this generic definition, we identify and specify properties for the security of SSO systems, namely a property for authentication, which captures that a user's account can only be accessed by its owner, and a property for session integrity, which captures the integrity of a login flow.

To enable privacy analyses, we extend the WIM such that, roughly speaking, we can comparatively analyze dynamic runs of a system that are scheduled by an adversary and leave one specific decision to the user. Such a user decision can, for example, capture that a user at some point decides at which RP she wants to log in. Based on this (non-trivial) extension, we then formulate a definition for privacy, i.e., that an IdP (as the adversary) cannot see at which RP a user is logging in. In particular, we require that the IdP cannot distinguish between two login flows at different RPs. For this property to be fulfilled, all information that the user's IdP can possibly see in each login flow has to look exactly the same to the IdP. This foundation allows us to, for the first time, carry out a systematic and rigorous formal analysis of privacy for Web SSO systems.

5 Analysis of BrowserID

As mentioned above, BrowserID is a Web SSO proposed by Mozilla in 2012 [1]. BrowserID is the only Web SSO system so far that claims to provide privacy to its users. This makes BrowserID a very interesting system to analyze. BrowserID is also quite complex and makes use of many modern Web features, making it a good example to exercise the expressiveness of the WIM.

Using our approach, we analyze the security and privacy of BrowserID [8, 9]. As a result of this first rigorous analysis of an SSO system in the Web infrastructure, we find severe attacks. These attacks not only affect the security of BrowserID but also show that BrowserID's unique privacy claim does not hold. We propose fixes for BrowserID and prove that the fixed system provides security. Regarding privacy, we show that BrowserID, unfortunately, is broken beyond repair. We reported our findings to Mozilla, who acknowledged the vulnerabilities and awarded us several bug bounties.

Extraction of the BrowserID model. At first, BrowserID was envisioned to be integrated into Web browsers, but to ease adoption, the actual realization only builds upon native features of the modern Web, such as Web messaging and Web storage. To achieve this goal, the protocol implemented in practice deviates from the original specification. In particular, the envisaged browser integration is implemented as a Web application hosted by Mozilla. As a result, BrowserID has become a very complex SSO protocol that, for example, takes more than 80 steps for a typical login flow.

The BrowserID implementation also extends the original BrowserID specification with a second mode, in which Mozilla serves as an IdP that is tightly integrated with the core of the protocol's implementation. This so-called secondary or fallback mode actually constitutes a separate protocol.

As pointed out above, the specification of BrowserID only provides a high-level idea of its real-world counterpart. To get a comprehensive overview of all steps of the protocol, we manually analyzed the code of BrowserID (approx. 47k lines of code, written in JavaScript). Based on this analysis, we create separate models for BrowserID's primary mode and secondary mode using the WIM, which covers all features needed to describe BrowserID.

Each model is based on the WIM, i.e., the communication model as well as generic components, such as brow-

sers and attackers, and contains (1) a definition for Mozilla's server, which provides the Web application component of BrowserID mentioned above along with the scripts that cover its browser-side parts, (2) a definition for IdPs (server and scripts), and (3) a definition for RPs (again, server and scripts). For these models, we follow our generic definition for SSO systems mentioned above and refine all generic SSO events such that our generic properties for security and for privacy are usable in these models.

Security of BrowserID. During modeling and while trying to prove the security properties (authentication and session integrity) for BrowserID, we found several severe attacks that break the security properties.

In the *identity injection attack*, a malicious IdP is able to force the user to sign in at RPs using an identity that is not owned by the user. The *login injection attack* allows any malicious party to foist a different identity on a user. The *identity forgery attack* allows an attacker to exploit an error in the identity bridge feature of BrowserID, effectively enabling the attacker to authenticate himself to any RP as any Gmail or Yahoo user. In the *key cleanup failure attack*, an attacker can get hold of a user's private key, although the user thinks that her key is removed from a shared device. Similarly, in the *cookie cleanup failure attack*, the attacker is able to link his key pair to a user's account due to incorrect usage of cookies.

For all of the attacks on security mentioned above, we propose fixes and show that these fixes are indeed sufficient using a formal proof. To this end, we incorporate our fixes into the BrowserID models and show based on these models that both modes of BrowserID indeed satisfy the security properties. This security proof is the first security proof carried out using the WIM and the most comprehensive formal analysis of a Web application at the time of publication.

Privacy of BrowserID. While trying to analyze BrowserID's privacy, we discovered a severe attack (and several variants of this attack) that completely breaks privacy. This attack is based on a novel attack vector: A malicious IdP can check whether a user logs into a specific RP by analyzing the window structure in the user's browser. As it turns out, this problem is not easy to fix and would require a major redesign of BrowserID.

Further, as BrowserID relies on a central Web application provided by Mozilla, this application and its operator need to be trusted ultimately. Mozilla is able to track all login activities of BrowserID's users in all cases. This, in particular, applies to the secondary mode in which Mozilla itself serves as an IdP.

6 Design and Analysis of SPRESSO

Based on the lessons learned from BrowserID and inspired by BrowserID's goal, we design a new SSO system for the Web: The *Secure, Privacy-Respecting Single Sign-On System (SPRESSO)* [10]. This SSO system aims to provide security, privacy, and true decentralization. Furthermore, SPRESSO is designed to be very easy to use. The protocol is based on native Web features only and does not require any third-party addons or extensions, which might complicate adoption or might break compatibility in the future. In SPRESSO, email providers become IdPs, and users use their email addresses to log in. SPRESSO can be seamlessly integrated into RPs, who often already use email addresses as login names. As a side-effect of SPRESSO's privacy, RPs can even start an SPRESSO flow automatically without any negative privacy implications when a user enters her email address in a login form.

Design. To design SPRESSO, we take a unique approach: After the initial design draft, instead of coding a proof-of-concept prototype, we first opt for the creation of a formal model based on the WIM. For development, the WIM provides a concise view on the Web infrastructure and thus, eases design decisions. This approach also enables us to perform a rigorous security and privacy analysis right away (see below).

For SPRESSO we need to master several challenges: To create an SSO system, RP and IdP need to be able to exchange data in some way that provides integrity while hiding the identity of the RP from the IdP. Also, this SSO system should be completely decentralized without the need for a central authority. Moreover, the SSO system needs to be usable ad-hoc without any setup in the users' browsers. With SPRESSO, we achieve all of these goals and create an easy-to-integrate and easy-to-use SSO system.

Security Proof of SPRESSO. Based on our model of SPRESSO built using the WIM, we show that the SPRESSO protocol indeed satisfies our security properties. As we designed SPRESSO with lessons learned from the BrowserID analysis in mind, we designed a much simpler protocol. As a result, the security analysis is less complex, more straightforward, and easier to comprehend.

Privacy Proof of SPRESSO. We proved privacy, one of the main goals of SPRESSO. To this end, we formally show, based on our model, that a malicious IdP cannot learn any information about the RP from the authentication procedure. We show that the IdP's view is always the same, independent of the RP chosen by the user.

Our analysis shows that SPRESSO is indeed the first Web SSO system to provide privacy and also the first

Web SSO system to feature a formal proof for its privacy and security properties from the beginning.

Proof-of-Concept Implementation. After successful analysis of SPRESSO using the WIM, we create an entirely usable proof-of-concept implementation of all SPRESSO components. This implementation demonstrates that SPRESSO is indeed easy to use and simple to deploy at RPs and IdPs.

7 Further Analyses and Future Directions

Our work on the WIM also includes security analyses of other very popular SSO protocols, such as OAuth 2.0 [11] and OpenID Connect 1.0 [12]. Moreover, we have also applied the WIM to analyze the W3C Web Payment specification [5] and enabled an in-depth analysis of the Financial-Grade API specification [7] that is based on OAuth and OpenID Connect. All of these analyses uncovered several severe flaws in these standards. Our findings have sparked design changes of many of these protocols and also started the series of the *OAuth Security Workshop (OSW)*[1] to foster the exchange between academic researchers, specification bodies, and developers.

As part of an ongoing international research project,[2] we aim to mechanize the WIM to make such analyses even more accessible and reusable by others while also aiming to ease the manual proof burden. To this end, we are developing a new framework, *DY** [4, 2], that enables Dolev-Yao-style analyses based on *F**, a fully-fledged functional programming language that is complemented with a powerful theorem prover [21]. We have already illustrated the expressiveness and usability of our approach by analyzing complex protocols such as Signal [4] and ACME [3].

Acknowledgement

This work was partially supported by the Deutsche Forschungsgemeinschaft (DFG) through Grants KU 1434/10-1 and KU 1434/10-2.

Literature

[1] Ben Adida et al. BrowserID Specification. Specifications for Mozilla's Identity Effort. `https://github.com/mozilla/id-specs`.

[2] Karthikeyan Bhargavan, Abhishek Bichhawat, Quoc Huy Do, Pedram Hosseyni, Ralf Küsters, Guido Schmitz, and Tim Würtele. A Tutorial-Style Introduction to DY*. In *Protocols, Logic, and Strands: Essays Dedicated to Joshua Guttman on the Occasion of His 66.66 Birthday*, volume 13066 of *LNCS*, pages 77–97. Springer, 2021.

[3] Karthikeyan Bhargavan, Abhishek Bichhawat, Quoc Huy Do, Pedram Hosseyni, Ralf Küsters, Guido Schmitz, and Tim Würtele. An In-Depth Symbolic Security Analysis of the ACME Standard. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS 2021))*, pages 2601–2617. ACM, 2021.

[4] Karthikeyan Bhargavan, Abhishek Bichhawat, Quoc Huy Do, Pedram Hosseyni, Ralf Küsters, Guido Schmitz, and Tim Würtele. DY*: A Modular Symbolic Verification Framework for Executable Cryptographic Protocol Code. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P 2021)*, pages 523–542. IEEE Computer Society, 2021.

[5] Quoc Huy Do, Pedram Hosseyni, Ralf Küsters, Guido Schmitz, Nils Wenzler, and Tim Würtele. A Formal Security Analysis of the W3C Web Payment APIs: Attacks and Verification. In *43rd IEEE Symposium on Security and Privacy (S&P 2022)*. IEEE Computer Society, 2022. To appear.

[6] Danny Dolev and Andrew C. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[7] Daniel Fett, Pedram Hosseyni, and Ralf Küsters. An Extensive Formal Security Analysis of the OpenID Financial-grade API. In *2019 IEEE Symposium on Security and Privacy (S&P 2019)*, volume 1, pages 1054–1072. IEEE Computer Society, 2019.

[8] Daniel Fett, Ralf Küsters, and Guido Schmitz. An Expressive Model for the Web Infrastructure: Definition and Application to the BrowserID SSO System. In *35th IEEE Symposium on Security and Privacy (S&P 2014)*, pages 673–688. IEEE Computer Society, 2014.

[9] Daniel Fett, Ralf Küsters, and Guido Schmitz. Analyzing the BrowserID SSO System with Primary Identity Providers Using an Expressive Model of the Web. In *20th European Symposium on Research in Computer Security (ESORICS 2015), Proceedings, Part I*, pages 43–65. Springer, 2015.

[10] Daniel Fett, Ralf Küsters, and Guido Schmitz. SPRESSO: A Secure, Privacy-Respecting Single Sign-On System for the Web. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*, pages 1358–1369. ACM, 2015.

[11] Daniel Fett, Ralf Küsters, and Guido Schmitz. A Comprehensive Formal Security Analysis of OAuth 2.0. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS 2016)*, pages 1204–1215. ACM, 2016.

[12] Daniel Fett, Ralf Küsters, and Guido Schmitz. The Web SSO Standard OpenID Connect: In-Depth Formal Security Analysis and Security Guidelines. In *IEEE 30th Computer Security Foundations Symposium (CSF 2017)*, pages 189–202. IEEE Computer Society, 2017.

[13] Daniel Fett and Guido Schmitz. Pi and More - eine Veranstaltungsreihe rund um „kleine Computer". In *46. Jahrestagung der Gesellschaft für Informatik (Informatik 2016)*, volume P-259 of *LNI*, pages 1195–1196. GI, 2016.

[14] Brad Fitzpatrick, David Recordon, et al. OpenID Authentication 2.0. `http://openid.net/specs/openid-authentication-2_0.html`.

[15] E. Hammer-Lahav (Ed.). The OAuth 1.0 Protocol. RFC 5849 (Informational), 4 2010.

[16] D. Hardt (Ed.). The OAuth 2.0 Authorization Framework. RFC 6749 (Proposed Standard), 10 2012.

[17] Open Web Application Security Project (OWASP). Password storage cheat sheet. `https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet`.

[18] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore. OpenID Connect Core 1.0 in-

[1] `https://oauth.secworkshop.events/`

[2] `https://reprosec.org`

corporating errata set 1. http://openid.net/specs/openid-connect-core-1_0.html.

[19] SAML 2.0 Technical Overview. Committee Draft 02. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.html.

[20] Guido Schmitz. *Privacy-Preserving Web Single Sign-On: Formal Security Analysis and Design.* PhD thesis, University of Stuttgart, 2019.

[21] Nikhil Swamy, Catalin Hritcu, Chantal Keller, Aseem Rastogi, Antoine Delignat-Lavaud, Simon Forest, Karthikeyan Bhargavan, Cédric Fournet, Pierre-Yves Strub, Markulf Kohlweiss, Jean Karim Zinzindohoue, and Santiago Zanella Béguelin. Dependent types and multi-monadic effects in F*. In *ACM Symposium on Principles of Programming Languages (POPL 2016)*, pages 256–270, 2016.

Dr. Guido Schmitz is a Lecturer in the Information Security Group (ISG) at the Royal Holloway University of London. He graduated from the University of Trier with a Diplom degree in computer science in 2012, and received his Doctorate (summa cum laude) under the guidance of Prof. Dr. Ralf Küsters from the University of Stuttgart in 2019. Guido Schmitz has been a finalist for the CAST/GI Dissertation Award for IT-Security in 2021 as well as for the German IT-Security Award in 2016. Besides his research on formal methods, protocol security, and Web technologies, he also organizes events to inspire others for computer science [13].

Address: Royal Holloway University of London, Information Security Group, Egham, Surrey, United Kingdom, E-Mail: Guido.Schmitz@rhul.ac.uk