Addressing threats to real-world identity management systems

Wanpeng Li, Chris J Mitchell

Information Security Group, Royal Holloway, University of London
Egham TW20 0EX, UK
Wanpeng.li.2013@live.rhul.ac.uk me@chrismitchell.net

Abstract

Recent practical studies have revealed that, in practice, widely used identity management schemes such as OAuth 2.0 and OpenID Connect are often poorly implemented by relying parties, and as a result very serious vulnerabilities can result. In any event, any system relying on browser redirections, as is the case for OAuth 2.0 and OpenID Connect, is vulnerable to web-spoofing and phishing attacks. Many of these vulnerabilities would disappear if the user's browser (or other agent under user control) remained in charge of what credentials are divulged to whom, and when. We outline a system known as Uni-IdM, which has been successfully prototyped, which provides a generic service of this type. Through the installation of a simple JavaScript plugin, the user is provided with a unified means of managing and using all his or her credentials via a simple and intuitive interface, which will work with a multiplicity of identity management systems. This not only reduces the risk of credential and/or account compromise, but also greatly simplifies the work of the user in credential management as well as providing a much clearer view to the user of which end parties are being sent user information.

1 Introduction

There is clearly a pressing need for better ways of handling user credentials than expecting users to remembers tens of individual usernames and passwords. The end user demand for improved solutions to practical identity management can be seen from the rapid growth in adoption of services provided by Facebook, Google and others, primarily relying on OAuth 2.0 and its variants (in particular OpenID Connect). However, recent practical studies, as outlined in this paper, have revealed that in practice these schemes are often poorly implemented by relying parties, and as a result very serious vulnerabilities can result. In any event, any system relying on browser redirections, as is the case for OAuth, is vulnerable to web-spoofing and phishing attacks. Many of these vulnerabilities would disappear if the user's browser (or other agent under user control) remained in charge of what credentials are divulged to whom, and when.

In this paper we outline a system known as Uni-IdM, which has been successfully prototyped, which provides a generic service of this type. Through the installation of a simple JavaScript plugin, the user is provided with a unified means of managing and using all his or her credentials via a simple and intuitive interface, which will work with a multiplicity of identity management systems. This not only reduces the risk of credential and/or account compromise, but also greatly simplifies the work of the user in credential management as well as providing a much clearer view to the user of which end parties are being sent user information.

The remainder of the paper is organised as follows. In section 2, we first briefly describe how OAuth 2.0 works, and then go on to outline issues which have arisen in real-world implementations, and that have given rise to serious security weaknesses. We also consider equally serious problems arising in real-world OpenID Connect implementations. In section 3 we outline the rationale for, and operation of, Uni-IdM, and explain why it can help to address some of the problems we have observed in practice. We conclude in section 4 with a discussion of ways forward for practical identity management.

2 OAuth-based Identity Management

OAuth 2.0 had rapidly gained huge traction as a means of simplifying the user authentication process. It has been adopted by Facebook and many others as a means of providing identity management services, and these services have been very widely adopted. OpenID Connect, which builds additional functionality on top of OAuth 2.0, has also seen widespread adoption for the same purposes, not least involving Google as identity provider. We next describe briefly how these systems work, and at the same time consider some of the serious security threats arising from their use.

2.1 OAuth 2.0 – a Brief Introduction

Since OAuth 2.0 was published in 2012 [Hard12], it has been used by many websites worldwide to provide single sign-on (SSO) services. By using OAuth 2.0, websites can ease password management for their users, as well as saving them the inconvenience of re-typing attributes that are instead stored by identity providers and provided to relying parties as required.

OAuth 2.0 involves four roles. The *resource owner* is a host acting on behalf of an end user, which can grant access to protected resources. The *resource server* is a server which stores the protected resources and consumes access tokens provided by an *authorisation server*. The *client* is an application running on a server, which makes requests on behalf of the resource owner (the client is the Relying Party (RP) when OAuth 2.0 is used for identity management purposes). The authorisation server generates *access tokens* for the client, after authenticating the resource owner and obtaining its authorisation (the resource server and authorisation server together constitute the Identity Provider (IdP) when OAuth 2.0 is used for identity management).

Fig. 1 provides an overview of the operation of OAuth 2.0. The client initiates the process by sending (1) an authorisation request to the resource owner. In response, the resource owner generates an authorisation grant, and sends it (2) to the client. After receiving the authorisation grant, the client initiates an access token request by authenticating itself to the authorisation server and presenting the authorisation grant (3). The authorisation server issues (4) an access token to the client after successfully authenticating the client and validating the authorisation grant. The client makes a protected source request by presenting the access token to the resource server (5). Finally, the resource server sends (6) the protected resources to the client after validating the access token.

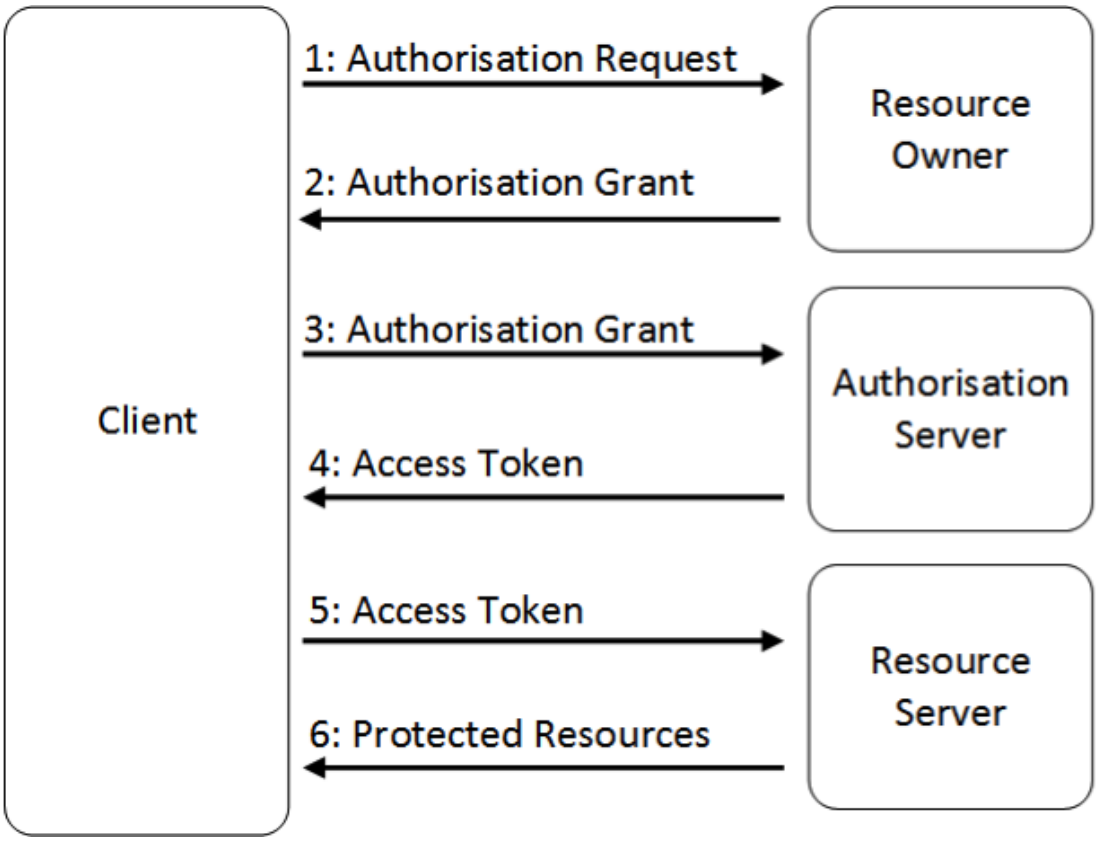

Fig. 1: OAuth 2.0 Protocol Flow

OAuth 2.0 was designed to provide a way of allowing controlled access by an application to resources protected by a resource server on behalf of the resource owner. The application is given the necessary rights in the form of an access token issued by the authorisation server, and this token is consumed by the resource server. The underlying goal is to allow the application to gain access to resources independently of the resource owner, after the resource owner has initially given consent, without being given the resource owner's credentials. That is, OAuth 2.0 is not a conventional identity management system, but is nevertheless used as one, as we describe below.

In order to use OAuth 2.0 for identity management, and in particular for SSO, the resource server and authorisation server together play the role of the IdP, the client plays the role of the RP, and the resource owner corresponds to the user. OAuth 2.0-based SSO systems build on web browser (or, more generally, user agent) redirections, where a user wishes to access services protected by the RP which consumes the access token generated by the IdP. The IdP provides ways to authenticate the user, asks the user to allow the RP to access the user's attributes, and generates an access token. The RP uses the access token to access the user's attributes using an API provided by the IdP.

OAuth 2.0 does not support identity federation as defined in identity management systems such as Shibboleth or SAML, although federation is necessary if SSO services are to be provided. In practice, a commonly used means of achieving identity federation involves the RP locally binding the user's RP-managed account with the user's IdP-managed account, using the unique identifier for the user generated by the IdP. After binding, a user is able to log in to the RP-managed account using his or her IdP-managed account.

Such a federation scheme typically operates as follows. After receiving the access token, the RP retrieves the user's IdP-managed account identifier and binds the user's RP-managed account identifier to the IdP-managed account identifier. When the user next tries to use his or

her IdP-managed account to log in to the RP, the RP looks in its account database for a mapping between the supplied IdP-managed identifier and an RP-issued identifier. If such a mapping exists, then the RP simply logs the user in to the corresponding RP-managed user account.

In real-world OAuth 2.0 SSO systems supporting federation, RPs typically use one of two ways to perform the binding. Firstly, suppose a user chooses to log using SSO. After finishing the authorisation process with the IdP, the user is asked either to bind the IdP-managed account to his or her RP-managed account or to log in to the RP directly. The user will need to provide his or her RP-managed account information (e.g. account name and password) to complete the binding. Alternatively, after a user has already logged into an RP, he or she can initiate a binding operation. After being authenticated by the IdP and granting permission to the RP, the user can bind his or her RP-managed account to the IdP-managed account. After binding, many RPs allow users to log in to their websites using an IdP-managed account.

2.2 Practical Issues with OAuth 2.0

A number of authors have analysed the operation of OAuth, and have identified issues in real world implementations. To understand the real-world security of OAuth 2.0, Wang, Chen and Wang [WaCW12] examined a number of deployed SSO systems, focussing on a logic flaw present in many such systems, including OpenID. In parallel, Sun and Beznosov [SuBe12] also studied deployed systems. Both these studies restricted their attention to systems using English. Indeed, until recently, very little research had been conducted on the security of OAuth 2.0 systems using other languages, some of which, like those in Chinese, have very large numbers of users. Indeed, OAuth 2.0 is very widely used on Chinese websites, and there is a correspondingly rich infrastructure of IdPs providing identity services using OAuth 2.0. For example, some RPs, such as the travel site Ctrip, support as many as eight different IdPs. At least ten major IdPs offer OAuth 2.0-based identity management services. RPs wishing to offer users identity management services from multiple IdPs must support the peculiarities of a range of different IdP implementations of OAuth 2.0. To try to redress this imbalance, Li and Mitchell [LiMi14] reported on an analysis of Chinese-language OAuth 2.0 systems.

This latter study identified a number of serious security issues in these implementations. The security of 60 implementations of OAuth 2.0 for federation-based SSO, as deployed by leading Chinese websites, was studied. Nearly half of these implementations were found to be vulnerable to cross-site request forgery (CSRF) attacks against the federation process, allowing serious compromises of user accounts. These attacks allow a malicious third party to bind its IdP-managed account to a user's IdP-managed account, without knowing the user's account name or password. Logic flaws were also discovered in real-world implementations of federation, which again allow binding of an attacker's IdP-managed account to a user's RP-managed account. These latter issues arise primarily because of the lack of a standardised federation process.

More generally, CSRF attacks are only possible because the operation of the identity management system is invisible to the user and to the browser. That is, all the actions of the user’s browser are controlled by remotely controlled mechanisms such as the use of HTTP redirects and JavaScript code downloaded from the RP. We return to this issue in section 3 below.

2.3 OpenID Connect

OpenID Connect 1.0 [SBJ+14] is built as an identity layer on top of the OAuth 2.0 protocol. The functionality that it adds enables RPs to verify the identity of an end user by relying on an authentication process performed by an OpenID Provider (OP), i.e. it adds identity management functionality to the OAuth 2.0 system. In order to enable an RP to verify the identity of an end user, OpenID Connect adds a new type of token to OAuth 2.0, namely the *id token*. This complements the access token, which is already part of OAuth 2.0. These two types of token are both issued by an OP, and have the following functions.

- An access token contains credentials used to authorise access to protected resources stored at a third party (e.g. the OP). Its value is an opaque string representing an authorisation issued to the RP. It encodes the right for the RP to access data held by a specified third party with a specific scope and duration, granted by the end user and enforced by the RP and the OP.
- An id token contains claims about the authentication of an end user by an OP together with any other claims requested by the RP. Possible claims within such a token include: the identity of the OP that issued it, the user's unique identifier at this OP, the identity of the intended recipient, the time at which it was issued, and its expiry time. It takes the form of a JSON Web Token [JoSB14] and is digitally signed by the OP.

Both access tokens and id tokens can be verified by making a call to the web API of the issuing OP.

OpenID Connect builds on user agent redirections. Suppose that an end user wishes to access services protected by the RP, which consumes tokens generated by the OP. The OP provides ways to authenticate the end user, asks the end user to grant permission for the RP to access the user attributes, and generates two types of token: an access token and an id tokens, where the latter contain claims about a user authentication event. After receiving an access token, the RP can use it to access end user's attributes using the API provided by the OP, and after receiving an id token the RP is informed about the authentication of the user.

Even though OpenID Connect was only finalised at the start of 2014, there are already more than half a billion OpenID Connect-based user accounts provided by Google, PayPal and Microsoft. This large user base has led very large numbers of RPs to integrate their services with OpenID Connect, and the Google service alone is being used routinely to protect many millions of user accounts, as well as sensitive information stored at both RPs and the Google OP server.

2.4 Practical Issues with OpenID Connect

Given the clear and growing practical significance of OpenID Connect, it is clearly important to understand its security in real-world deployments. A very recent study [LiMi15], conducted in early 2015, has sought to address this by conducting a large scale survey. The operation of all one thousand sites from the GTMetrix Top 1000 Sites providing services in English was examined. Of these sites, 103 were found to support the use of the Google's OpenID Connect service. All 103 of these websites were then further examined for potential vulnerabilities. In the study, all the RPs and the Google OP site were treated as black boxes, and the HTTP messages transmitted between the RP and OP via the browser were analysed to identify possible vulnerabilities. For every identified vulnerability, an exploit to evaluate the possible attack surface was implemented and tested.

This study revealed serious vulnerabilities of a number of types, which either allow an attacker to log in to the RP website as the victim user or enable the compromise of potentially sensi-

tive user information. Google has customised its implementation of OpenID Connect by combining SDKs, web APIs and sample code, and as a result the OpenID Connect specification only acts as a loose guideline to what RPs have actually implemented. Further examination suggests that the identified vulnerabilities are mainly caused by RP developers misunderstanding how to use the Google OpenID Connect service, and by making design decisions which sacrifice security for simplicity of implementation. Many of the attacks that were discovered use cross-site scripting (XSS) and CSRFs, well-established and widely exploited attack techniques.

As was the case for the vulnerabilities identified in OAuth 2.0 implementations, these problems would not have arisen if the system did not rely on a completely 'passive' browser, which simply executes remotely provided JavaScript and performs redirects as requested. Of course, in all cases so far identified, careful implementation of the schemes would also have avoided the problems, but the possibility of further flaws remains, and it seems inherently risky for users to rely on all the many RPs implementing identity management systems as carefully as is necessary.

3 Uni-IdM: a New Approach to ID Management

3.1 Rationale

Identity management systems are in many cases based on web browser redirections, as is the case for OpenID Connect and OAuth 2.0; as a result such systems are vulnerable to phishing attacks [Jaru03]. A means of mitigating such attacks is therefore needed. One general approach to mitigating such phishing attacks is to incorporate a client-based user agent into the identity management system, e.g. as is the case for the now-defunct CardSpace and Higgins. It is also possible to equip a redirection-based identity management system with a client-based user agent, which can help to reduce the threat of phishing attacks [AlMi12]. Recently a new scheme has been proposed [LiMi15b] which adopts this latter approach by integrating the OpenID Connect identity management system with client functionality both in order to reduce the risk of phishing attacks and to improve the usability of the system.

As noted above, since OpenID Connect is based on web browser redirections, and does not depend on any client-based components, it is therefore vulnerable to phishing attacks; e.g. a malicious relying party could redirect a user to a fake OpenID Connect provider which is under the control of the relying party. This would enable the relying party to collect sensitive user information, such as the account name and password of the user's OpenID Connect account at the impersonated provider. The effects of user credential theft could be very serious, potentially enabling unauthorised access to all the RP user accounts which the user has linked to the OP. Moreover, as we have observed above, many widely used practical implementations of both OAuth 2.0 and OpenID Connect possess vulnerabilities exacerbated by the lack of control on the user platform.

In some identity management systems, e.g. CardSpace and Higgins, a client-based user agent is used. Such an agent has a range of practical advantages including ease of use, greater user control, and resistance to certain classes of phishing attacks. However, it would appear that no systems of this general type have been widely adopted; indeed, Microsoft no longer supports CardSpace. Also these schemes typically require the use of specific protocols between the main parties, preventing their use with other identity management systems. Instead, identity management schemes based on web browser redirections have become widely used, not least because of their ease of deployment. Given the security advantages of client-based functionality, there is a potentially significant benefit to be gained from devising a way of adding

client-based functionality to these widely used redirect-based systems, particularly as it offers the possibility of combating phishing fraud and other frauds arising from RP implementation vulnerabilities (e.g. allowing CSRF attacks).

3.2 Uni-IdM – How it Works

Al-Sinani and Mitchell [AlMi12] proposed a client-based identity management tool which they called IDSpace. The idea underlying IDSpace is to provide a client-based environment which can operate with a wide variety of identity management protocols, and can also replace the CardSpace and/or Higgins agents. The primary goal of IDSpace is to provide a single, consistent and user-comprehensible interface to a wide range of identity management systems, and, through the deployment of trusted client functionality, to reduce the threats of phishing and other attacks.

For a variety of reasons the IDSpace architecture requires two separate software components: a browser extension and separate client software which executes independently of the browser. This complicates both installation and operation because of the need for the two components to intercommunicate. The newly proposed Uni-IdM scheme [LiMi15b], implements the same concept as IDSpace but follows a somewhat different architectural approach by implementing all the functionality within a browser extension. As a browser extension written in JavaScript, Uni-IDM is inherently portable, and could be implemented on a range of browsers, host operating systems and platform types with minimal modification. The Uni-IdM scheme has been prototyped for OpenID Connect [LiMi15b], but is designed to operate with a multiplicity of identity management systems.

Uni-IdM stores information about individual relationship between the end user and an IdP in a logical entity known as a *uCard*. A uCard specifies the type of identity management system with which the uCard can be used, and also the types of personal information held by the IdP on behalf of the end user. It does not contain potentially sensitive personal information, such as an account name or password. uCards are stored in the protected Uni-IdM *card store*. The *credential store* stores sensitive data associated with the uCards in the card store, such as personal information, user account names and passwords, and certificates. A variety of measures could be used to protect the card store and credential store, such as authenticated encryption, logical protection and/or physical protection.

The Uni-IdM *content scanner* searches the login page of an RP web-site in order to discover which identity management systems it supports. It sends the results of the search to the Uni-IdM *card selector*, which provides an interface enabling the user to interact with Uni-IDM. It displays the identity (address) of the RP website to the user, and if it is the first time that the user has visited the RP website (possibly indicating phishing) it enables the user to either terminate or continue. It further allows the user to manage his or her uCards, including creating, reviewing, modifying and deleting them, and it indicates which identity management systems are supported by the RP (in the case of the prototype, only OpenID Connect is supported). If the user has previously visited the RP website, it displays all the available uCards to the user; otherwise it requires the user to choose an identity management system and then create a uCard for the user-selected system.

The system automatically manages the submission of user credentials to the OP, and also the transfer of the id token from the OP to the RP. That is, HTTP redirections and the operations of any RP-provided JavaScript are controlled by the Uni-IdM plug-in, inherently protecting against certain classes of attack. These functions enable the user to achieve control over what information is sent to whom, and also prevents a range of attacks of the type discussed above.

3.3 Analysis

As the URL of the OpenID Connect OP is known to the Uni-IDM, once support for OpenID Connect is detected by Uni-IDM, it compares the known URL with the URL of the OP to which the browser is directing the user; only if the two URLs share the same domain will it submit the user's credential to the OP on behalf of the user. If a malicious RP website tries to redirect the user to a 'fake' OP under its control, Uni-IDM will terminate the process and will warn the user of a potential attack. This effectively mitigates the threat of phishing attack.

Many RPs put a specific OpenID Connect logo on their login page to indicate that they support OpenID Connect. In normal use the user must find the logo and click it in order to first initiate log-in to the RP using OpenID Connect; subsequently log-ins will take place automatically and, as a result, the user loses all control over the process. However, each RP designs its own website, and so the precise details of how a user initiates an OpenID Connect login will vary. This has the effect of downgrading the user experience because of the lack of a consistent login procedure, compounded by the subsequent lack of any control over continued use of the chosen OpenID Connect OP. However, Uni-IDM inherently provides a consistent user experience through the use of the card selector interface, and enables continued control over the login process. Whenever a user visits a RP which supports OpenID Connect, Uni-IDM will scan the DOMs of the login page and, assuming OpenID Connect support is identified, Uni-IDM will trigger its card selector; the user does not have to examine the website to find the OpenID Connect button, and always interacts with the card selector for the purposes of identity management. This improves the user experience for RP websites which support identity management systems, and also provides a consistent interface for the user. Moreover, it also gives the user a simple way of understanding exactly which authentication system is being used at all times.

As the Uni-IDM browser extension must scan every browser-rendered web page to find whether the page supports OpenID Connect, this might affect the performance of the web browser. However, tests using a Uni-IDM prototype have shown that this is not a big issue.

Perhaps the most serious possible limitation of Uni-IDM relates to the need for it to automatically detect whether or not RP websites support OpenID Connect. As websites implement support for the system in a range of different ways, e.g. using tags such as 'iframe' or 'img', the indicators used by the content scanner to search for support of OpenID Connect will also need to vary. As a result it is challenging for Uni-IDM to be designed to correctly detect all RPs which support OpenID Connect, requiring extensive testing on a site by site basis. Further details of experiments testing this are given in the paper describing the system [Li-Mi15b].

4 Conclusions

We have briefly reviewed the operation of the OAuth 2.0 and OpenID Connect systems, both of which are widely used today to support identity management services, notably those provided by Facebook and Google. We also summarised some of the recent findings regarding vulnerabilities in real-world implementations of the many websites which rely on these services. Whilst the problems we have discussed could be mitigated by more careful implementations, serious risks remain. Firstly, phishing attacks are not addressed by OAuth 2.0 and OpenID Connect, even when correctly implemented. Secondly, CRSF and XSS attacks can be performed in many ways, and vulnerabilities to such attacks are difficult to completely eradicate. As a result, some other method of improving the security of the operation of these systems is required, not least because of their very widespread use.

We have further described the operation of one system designed to mitigate these vulnerabilities, namely the Uni-IdM system [LiMi15b]. This is an evolution of the IDSpace system, proposed in 2012, [AlMi12]. Uni-IdM seeks to give users greater control over the identity management process through a simple browser plugin. This both enables the user to exert greater control over the identity management process, and mitigates a range of possible attacks, whilst remaining completely transparent to both RPs and IdPs.

This is, of course, just one possible approach to the long-term problem of improving operational security for identity management systems. As observed above, in practice these systems appear to be far from secure despite their widespread use, and hence this area is in urgent need of further research and development.

References

[AlMi12] Al-Sinani, Haitham S. and Mitchell, Chris J. A universal client-based identity management tool. In: Proc. EuroPKI 11. Editors: S. Petkova-Nikova, A. Pashalidis and G. Pernul, Springer-Verlag**,** 2012, p. 49-74.

[Hard12] Hardt, D. The OAuth 2.0 Authorization Framework. IETF RFC 6749, 2012.

[Jaru03] Jarupunphol, P. A critical analysis of 3-D Secure. In: Proceedings of the 3rd Electronic Commerce Research and Development (E-COM-3). 2003, p. 87-94.

[JoSB14] Jones, M., Sakimura, N., and Bradley, J. JSON Web Token (JWT). IETF RFC draft, 2014.

[LiMi14] Li, Wanpeng and Mitchell, Chris J. Security issues in OAuth 2.0 SSO implementations. In: Proc. ISC 2014. Editors: Chow, S. S. M., Camenisch, J., Hui, L. C. K., and Yiu, S.-M., Springer-Verlag, 2014, p. 529-541.

[LiMi15a] Li, Wanpeng and Mitchell, Chris J. Analysing the security of Google OpenID Connect. Preprint available from the authors, 2015.

[LiMi15b] Li, Wanpeng and Mitchell, Chris J. Enhancing user security for OpenID Connect. Preprint available from the authors, 2015.

[SBJ+14] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B. and Mortimore, C. OpenID Connect Core 1.0. The OpenID Foundation, 2014.

[SuBe12] Sun, S.T. and Beznosov, K. The devil is in the (implementation) details: An empirical analysis of OAuth SSO systems. In: Proc. CCS '12. Editors: Yu, T., Danezis, G., and Gligor, V. D., ACM, 2012, p. 378-390.

[WaCW12] Wang, R., Chen, S., and Wang, X. Signing me onto your accounts through Facebook and Google: A traffic-guided security study of commercially deployed single-sign-on web services. In: Proc. IEEE Symposium on Security and Privacy 2012. IEEE, 2012, p.365-379.

Index

authentication, cross-site request forgery, cross-site scripting, identity management, OAuth, OpenID, privacy, security