SpoofedMe - Intruding Accounts using Social Login Providers
A Social Login Impersonation Attack

Or Peles, Roee Hay
IBM Security Systems
{orpeles,roeeh}@il.ibm.com

December 3, 2014

Abstract

In recent years, millions of web users have employed their social network accounts (such as Facebook and Google+) to connect to third-party websites using a process called 'Social Login'.

Social login (or social sign-in) is a general name for authentication mechanisms that allow an end-user to conveniently use a single existing account of an identity provider (typically a social networking service) for signing into multiple third-party websites, instead of having to use separate credentials for each one.

In this paper, we present an implementation vulnerability found in some popular social login identity providers (including LinkedIn, Amazon and Mydigipass.com - see the vendors' mitigations following our private disclosure in section 6) and show how this vulnerability allowed us to impersonate users of third-party websites.

1 Introduction

Many of today's websites contain private user information, and maintain local user accounts that require authentication by username and password. Users have to remember different passwords for logging in to different sites.

According to research conducted in 2007 [1] the average web user has 6.5 unique passwords for a total of 25 password-protected accounts, meaning there is significant password reuse (same passwords for multiple sites). It follows that if a list of users and passwords of a single website is leaked, the possessor could potentially use it to log into multiple additional accounts on other websites of the compromised users. On the other hand trying to avoid password reuse requires the user to remember many different usernames and passwords, and may quickly lead to what has been referred to as "Password Hell" [2].

To cope with the above problem (and others), a new trend called 'Social Login' has emerged and has gained much popularity. Social login is an authentication concept that allows a user to use a single account in order to sign into many different cooperating websites. This single account functions as the user's identity, and the issuer of this account is called an identity provider. Typically, the identity provider is a social network provider. For example, when signing into a site that supports social login, instead of supplying local site credentials, the user might click a "Sign In with Facebook" button. For the sake of simplicity, and because we don't rely on a specific protocol, we use abstractions throughout this paper regarding the social login term, as a mechanism that allows web authentication using an external identity provider, and use simple intuitive names instead of any protocol's specific terms.

Social login is a concept with many possible implementations, based on a variety of different protocols. Common implementations are based on the OAuth 1.0a[1] and OAuth 2[2] authorization protocols[3]. In general, OAuth provides websites (and client applications) a 'secure delegated access' to server resources on behalf of an end-user (who owns the resources). When used for authentication purposes, the "server resources" are usually web endpoints that let the website retrieve details (such as full name and email) about the user's identity account, using an access token passed to it throughout the OAuth authentication process. Additionally, there is a standard called OpenID Connect [3], which is an authentication layer on top of OAuth 2.0, that allows the identity provider to return the identity details directly to the (third-party) website as a signed parameter in a specific format, with (or without) the access token. Some deprecated standards include OpenID 1.1 [4] and OpenID 2.0. These protocols provide a way to prove that an end-user owns an Identity URL (e.g. http://www.google.com/profiles/john123 for john123 user account in Google). It is done without transferring the password, email address, or anything else the user doesn't want it to.

[1] OAuth 1.0 Protocol - http://tools.ietf.org/html/rfc5849
[2] OAuth 2.0 Authorization Framework - http://tools.ietf.org/html/rfc6749
[3] Although they serve the same purpose, OAuth 2.0 is a totally different protocol than OAuth 1.0.

Unfortunately, there is no single industry standard that all popular providers are aligned with. Additionally, some identity providers have implemented their chosen protocol slightly differently from the specifications (such as renaming fields or using different token types). Some popular identity providers (and the protocol they're based on) include Facebook Connect [5] (OAuth 2.0), "Google+ Sign-In" [6] (OAuth 2.0 + OpenID Connect), "Sign-In with Twitter" [7] (OAuth 1.0a), "Sign-In with LinkedIn" [8] (OAuth 2.0 version or OAuth 1.0a version) and "Login with Amazon" [9] (OAuth 2.0).

Our contribution **-** In our paper, we present a logical attack that allows an attacker to access local website user accounts, using social login, under certain conditions. This attack is based on a basic implementation vulnerability found in certain identity providers, together with a design problem in third-party websites that was previously known. In addition to the theory, we empirically used this attack "in the wild", and list some real world examples of vulnerable identity-provider and website pairs. In contrast to similar attacks on social login specific protocols, such as presented in Wang et al. [10] (see Related Work Section 9), our attack is not based on any particular social login protocol (and specifically not on the deprecated OpenID protocol). We also came with simple practical recommendations for vulnerable identity providers to mitigate the attack.

2 Social Login Authentication Process

The protocols using for social login have more than one possible flow and implementation for the authentication process, but because our attack doesn't depend on the specific implementation and flow used, we only present here a simplified authentication process that demonstrates the general idea without getting into irrelevant details.

Three entities are involved in the social login authentication process:

- A third-party website ("**the website**") - a website that maintain local user accounts .
- An end-user ("**the user**") - the user who wants to log into his local account in the website.
- A social login identity provider ("**the identity provider**").

The simplified authentication process is explained below and illustrated in Figure 1.

2.1 Simplified Social Login Authentication Flow Stages

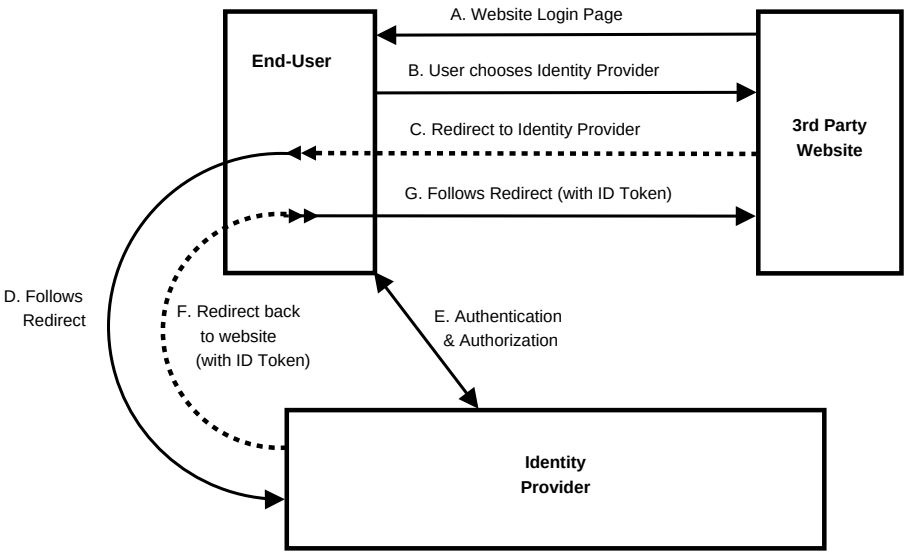

Figure 1: Simplified Social Login Authentication Flow

A. The user surfs (using a web-browser) to the website and is presented with a login page. The login page typically includes a login form for local account credentials, and identity providers' buttons for logging in using social login (e.g. "Login with FooProvider").

B. The user chooses to login with a specific identity provider, and clicks on its Login button.

C. The website responds with an HTTP Redirect code[4] that points the user's browser to the identity provider's social login authentication page, along with additional parameters that may include the website's ID (pre-registered with the identity provider) and whatever user identity field names the website needs.

D. The user's browser follows the HTTP Redirect (surfing to the identity provider's address with the parameters).

E. The identity provider presents an authentication form, into which the user enters his credentials (for the identity provider) and authenticates. After successfully authenticating to the identity provider, the user is requested to authorize the identity provider to supply a certain subset of the user's identity details (e.g. user-id within the identity provider, full name and email address) to the website. The user clicks a button to authorize the request.

F. The identity provider responds with an HTTP Redirect code that points the user's browser back to the website's page for returning from the social login identity provider, along with additional parameters that include an ID Token[5], a signed parameter that contains the user's identity details.[6]

G. The user's browser follows the HTTP Redirect (surfing to the website's address with the parameters), supplying the website with the user's identity information.

H. The website receives the signed user identity details, validates the signature and logs the user in based on the user's user-id in fooProvider (and its trust in fooProvider) that should be stored in the website's local user database.

3 Identity Provider Vulnerability

As we've seen in section 2, the identity provider transfers end-user identity fields to the third-party website. One of these fields may be the email address of the user. This is usually the email address of the account in the identity provider.

The vulnerability we encountered is that some identity providers agree to supply email addresses as part of the authentication process, even when the end-user's ownership of the email address had never been verified. This allows an end-user to create accounts with any email address he wants, as long as it isn't already in use by another identity account in that provider (assuming the identity provider doesn't allow the same email address for two separate accounts).

Intuitively, it is a good practice to only supply email addresses that have been verified [11][12], as most identity providers do. For this reason, in OpenID Connect specification, there is an optional identity field named "email_verified" that is "True if the End-User's e-mail address has been verified; otherwise false" [13].

In implementations where such a field is available, we won't consider the fact that an identity provider agrees to pass unverified email address to the third-party website as a vulnerability, but in this case, it's up to the third-party website to request and verify the "email_verified" field when needed.

4 Known Third-Party Website Design Problems

Both of the following design problems count on the third-party website's trust in the email address field supplied to it during the social login process. Both of these problems let a social login user log into his third-party website's local account based on proving ownership of his email address alone. These design problems were previously known (see Related Work - Section 9).

[4] Sometimes using a pop-up window

[5] Name influenced by OpenID Connect's ID Token - http://openid.net/specs/openid-connect-core-1_0.html#IDToken

[6] Instead of supplying the user's details directly, the identity provider may pass an "access token" (as a parameter), that can later be used by the website to fetch the user's identity details from the identity provider servers, or an "authorization code", to be exchanged by the website (from the identity provider's servers) for an ID Token and/or an access token.

4.1 A Third-Party Website Design Problem

A common third-party website design problem is the use of an email addresses as a unique identifier for its local user accounts, without verifying the specific identity provider(s) used with the account. This means that proving (through any identity provider) that we own an email address is enough to log us into the local account using that email address. This design problem may arise in cases where support for social login providers was added to an existing system without re-designing the user database in the migration process.

4.2 An Account Linking Implementation Problem

In some cases, even if the third-party website is aware of the provider used to create the account, we can count on a less common implementation vulnerability in the account linking feature. In order to ease the registration process for sites, there is a feature for account linking (or merging) that's used along with social login. Many websites allow users to log into their account in more than one way - for example: logging into an existing local account using a different identity provider than that used to create the account, or logging with social login into an account that was created and used with only regular local credentials. When logging in with a different identity provider for the first time, by providing an email address that is identical to that of an existing local user, an implementation problem could automatically (or by prompting the user) link the new account with the existing local account without asking for any additional credentials, because of the third-party website's trust in the end-user's e-mail address ownership.

5 Attack

As the attacker, we would like to be able to intrude into existing user accounts in third-party websites that support social login. To achieve this, we will impersonate the account owner with the help of the social login authentication process.

5.1 Attack Prerequisites

The attack requires the following:

- The third party website must support social login with a vulnerable identity provider (described in section 3), and request the email address field as part of the social login authentication process.
- The third party website must support at least one more login option - either using another identity provider, or the ability to use a local website account's credentials.
- The third party website must have one of the problems described in section 4.1 and section 4.2
- There exists a victim account whose email address is known, and who doesn't own an identity account with this address in the vulnerable identity provider, so that the identity provider will allow the attacker to register an identity account using the victim's email address.

5.2 The Attack Stages

1. In advance, the attacker registers a new account within the vulnerable identity provider, using the victim's local website account's email address. The new account will be created, but remain in an 'email-unverified' state. A verification mail may be sent to the victim's email address, but will not be clicked as the attacker doesn't have access to this email account.
2. The attacker surfs to the website, is presented with a login page (Step A., Figure 1) and chooses to login with the vulnerable identity provider (Step B., Figure 1). The attacker's browser now gets redirected to the authentication form in the identity provider website (Steps C & D, Figure 1).
3. At the identity provider site, the attacker enters the newly registered account credentials (from step 1) and agrees to pass the identity details (victim's email address included) to the website. (Step E, Figure 1). The attacker's browser gets redirected to the third-party website to continue login (Step F & G, Figure 1).
4. The attack is successful if one of the following occurs:

(a) In case of section 4.1 vulnerability: The website logs the attacker into the victim's local account, based on the matching email address passed to it via social login.

(b) In case of section 4.2 vulnerability: The website notices that the passed email address matches that of an existing account and automatically, or by providing the attacker with the account linking dialog, links the attacker's account with the existing (victim's) one, so the attacker is effectively logged into the victim's account.

6 Real World Cases

We checked popular social login providers for this vulnerability. To verify that an identity provider was vulnerable, we conducted the attack described above on a third-party website that relied on this provider. We created our own test account in advance, which was used as the victim account we tried to impersonate. Before announcing the attack, we privately disclosed the attack details to the ones found vulnerable and waited for them to apply their fixes.

We found the following providers vulnerable.

6.1 LinkedIn Identity Provider

LinkedIn as identity provider was vulnerable when using its deprecated OAuth 1.0a version. When using LinkedIn's new OAuth 2.0 version, the authentication process stopped with an error due to the unverified email address. From LinkedIn developers site [8] "... for those applications that still need to support it, we do continue to provide legacy support for OAuth 1.0a.". It's worth mentioning that at the time of this writing, the absolute majority of websites we encountered using LinkedIn as provider were using the "deprecated" (and vulnerable) version.

We successfully conducted the attack described above with the following third-party sites (partial list): Slashdot.org, Nasdaq.com, Crowdfunder.com, Spiceworks.com, Idealist.org, Scoop.it.

LinkedIn's security team followed our suggestion, and fixed the issue by not allowing social login requests that include the email field to continue, in case the email is un-verified.

6.2 Amazon Identity Provider

Amazon uses OAuth 2.0 protocol for "Sign In with Amazon" service, which was vulnerable. We found that in addition to allowing us to register an account with an email address we don't own, and pass it as part of the social login authentication process, it also allowed us to change the Amazon account email address to another unverified address, making it the new account's primary email address. This practice could allow other similar types of attack. Imagine the following scenario: An attacker logs in to a third-party site using Amazon identity provider account, with some irrelevant email address (for creating the account in the third-party website), logs out, changes the Amazon account email address to that of an existing (victim) account in the third-party site, and logs in with Amazon identity provider to that site again. The third-party website could have less restrictive login-time verifications when signing in with an existing social login account in contrast to signing in with social login account for the first time. If the third-party website re-retrieves the identity fields (specifically email address) from the identity provider each time a user logs in, this may open a window for attack.

Amazon's security team added a section to their developer documentation [14] that instructs their relying parties how to correctly link local accounts in their systems. In addition, they will be adding a "verified email" scope for Login with Amazon.

6.3 Mydigipass.com Identity Provider

VASCO's MYDIGIPASS.COM Secure Login service was also vulnerable to our attack, despite the fact it employs two-factor authentication – a mechanism that authenticates a user's identity by verifying that the user possesses a (pre-registered) physical device (possession factor), in addition to entering a password (knowledge factor). This mechanism didn't block our attack because as the attacker we used our own valid physical device (e.g. smart-phone) while registering a new Mydigipass account with fake email. Later, when signing into a website using MYDIGIPASS service, we again used our device for authenticating with MYDIGIPASS as any other valid member.

VASCO's security team fixed the issue by only supplying the email field to their relying websites when the user's email address is verified, and if the user actively chose to share it.

7 Mitigation

As our attack depends on vulnerabilities in both the identity provider and the third-party website, fixing the identity provider's vulnerabilities would be enough for the attack to be mitigated. In spite of that, it is important to stick with the best practice and also fix the vulnerabilities in the third-party websites, because other vulnerable identity providers could arise. Also, the third-party website vulnerabilities discussed may expose more accounts to compromise when one of the providers the website relies on gets compromised. Without fixing the third-party website vulnerability, if a single identity provider that the website trusts gets compromised, a similar attack would allow intrusion to all of the website's local accounts, even accounts that weren't linked to the compromised identity provider. With the mitigation in place, only accounts that were already linked with the compromised provider will be compromised, and the rest will stay safe.

7.1 Identity Provider Mitigation

There are two possible approaches for mitigating the attack on the identity provider side.

1. The recommended, and restrictive, mitigation would state that identity providers would not supply an unverified email address value as part of social login process. We encountered this method while checking Facebook Connect and Google+ Sign-In services behavior.
2. The less restrictive mitigation would be supplying an "email_verified" boolean field (similar to OpenID Connect optional field discussed above) along with the email address field, that indicates whether the email address has been verified. In this approach, the identity provider risks having the third-party website developers ignore this field, and still allowing the attack to happen.

7.2 Third-Party Website Mitigation

As discussed in OpenID Connect specifications (one of the social login protocols) [15], the only claims that should uniquely identify the user are the subject (the user's user-id in the identity provider) and issuer (the identity provider itself), used together. These are the only claims that a third-party website can rely upon as a stable identifier for the end user, since the subject claim must be locally unique and never reassigned within the identity provider for a particular end user.

This should be a best practice for the other social login protocols too (as long as the user-id is really locally unique and never reassigned within the identity provider).

As to the local account linking (or merging) feature, the linking of new account with an existing account should take place only after also proving ownership of the existing account, by logging into it either by password or by a social login identity provider used with it in the past. After making this change, our attack will fail because the attacker will have to login with the existing account's credentials (or by social login with an identity that was used in the past), which he doesn't possess.

8 Further Discussion and Conclusion

In contrast to most social login attacks that rely on phishing or the ability to intercept (through the browser) the traffic between the identity provider and the third-party website, our attack works within the expected protocol flow and is treated as a perfectly legitimate login attempt. Specifically, our attack works even when using OAuth 2.0 authorization code flow, which involves, after end-user authenticates to the identity provider, the forwarding of a secret code to the third-party website that is later used for grabbing an access token used for accessing the identity details from the identity provider's servers "behind the scenes" (without the end-user's browser involvement).

Our attack is logical, and would work on any new or proprietary social login protocol, regardless of signature mechanisms or flow, as long as the discussed vulnerabilities exist and the email address is provided to the third-party website. Thus, the OpenID 1.0 and 2.0 protocols by themselves are not vulnerable to our attack, which is based on passing the email address, but when used along with "OpenID Attribute Exchange" (AX) or "OpenID Simple Registration Extension" (SReg) - OpenID extensions that are used for passing additional fields, they could easily become vulnerable.

A possible side-effect of our attack process is the sending of an email notification to the victim's email address. The victim knows he hasn't registered an account recently and the notification may raise his suspicions. However, the notification won't reveal which of his third-party website accounts was targeted, and will not prevent the attacker from accessing his account.

9 Related Work

An extensive work from 2012 regarding social login protocols vulnerabilities published by Wang et al. [10] includes two attacks that also relied on passing to the third-party website an email address not owned by the attacker. Both these attacks were specific to the deprecated OpenID protocol, used together with OpenID Attribute Exchange extension. In section 4.1 of their paper, they present an attack that works only in cases in which the identity details coming from the identity provider back to the third-party website are transferred via the end-user's browser, and can be changed on the way. Their attack was based on their ability to cause an identity provider to sign on only partial identity fields. They excluded the email address field in the original request, and after the identity provider supplied (and signed) the other fields, they added an unsigned email address field to the response, hoping the third-party website will only validate the existing signature and ignore the absence of signature for the email address field. In section 4.5, they present an attack they call "Data Type Confusion" that exploited a flaw in Google OpenID identity provider's Attribute Exchange extension implementation, which allows them to supply any email address to login third-party accounts. They did it by tricking the third-party website into thinking that a value of a field of their choice and control (i.e. the user identity account's First Name field) is the end-user's email address. As a result of their paper, the third-party website design problem was discussed in a blog post [16] by Nat Sakimura, chairman of the OpenID Foundation and in Google's OpenID Best Practices page [11]. The third-party account linking design problem was also discussed into detail in a stackoverflow website thread [17].

Another work by Wang et al. [18] uncovered common vulnerabilities that were introduced while integrating popular social login SDKs into applications. Most of these vulnerabilities originated from false implicit assumptions taken by application developers that could easily have been avoided by clearer documentation.

A work by Sun et al. [19] focused on practical social login attacks on third-party website implementations using one of three specific identity providers: Facebook, Microsoft and Google. In section 4.3 they mention impersonation attacks that are based on specific third-party website implementation cases, such as not verifying signatures of the data returning from identity providers while using publicly accessible identifiers for their local accounts.

References

[1] D. Florencio and C. Herley, "A Large Scale Study of Web Password Habits," WWW 2007, Banff.

[2] How Facebook and Google Could Save Us from Password Hell -TIME (October 12, 2014) - http://time.com/84712/facebook-google-passwords/

[3] Wikipedia - OpenID Connect (October 12, 2014) - http://en.wikipedia.org/wiki/OpenID_Connect

[4] OpenID 1.1 (deprecated) - http://openid.net/specs/openid-authentication-1_1.html

[5] Facebook Developers - "Login Dialog" (October 12, 2014) - https://developers.facebook.com/docs/reference/dialogs/oauth

[6] Google Developers - "Using OAuth 2.0 for Login (OpenID Connect)" (October 12, 2014) - https://developers.google.com/accounts/docs/OAuth2Login

[7] Twitter Developers - "Sign in with Twitter" (October 12, 2014) - https://dev.twitter.com/docs/auth/sign-twitter

[8] LinkedIn Developers - "Authentication" (August 10, 2014) - https://developer.linkedin.com/documents/authentication

[9] Amazon Developers - "Login with Amazon" (August 11, 2014) - http://login.amazon.com/

[10] Rui Wang, Shuo Chen, XiaoFeng Wang: Signing Me onto Your Accounts through Facebook and Google: A Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services. IEEE Symposium on Security and Privacy 2012: 365-379

[11] Google Apps Developers - "Authentication Best Practices" (October 12, 2014) - Claimed identifiers vs. email addresses - https://developers.google.com/google-apps/marketplace/best_practices?csw=1#claimed

[12] A stackoverflow.com Q&A thread - "OAUTH - Request Verified Email Address from LinkedIn" (October 12, 2014) - http://stackoverflow.com/questions/19278201/oauth-request-verified-email-address-from-linkedin

[13] OpenID Connect Core 1.0 Specification - Standard Claims - http://openid.net/specs/openid-connect-core-1_0.html#StandardClaims

[14] Amazon Developers - "Integrate with Your Existing Account System" (November 11, 2014) - http://login.amazon.com/documentation/combining-user-accounts

[15] OpenID Connect Core 1.0 Specification - Claim Stability and Uniqueness - http://openid.net/specs/openid-connect-core-1_0.html#ClaimStability

[16] "Comments on Wang-Chen-Wang paper on OpenID Implementation Vulnerability", from personal blog of Nat Sakimura, Chairman of the OpenID Foundation - http://nat.sakimura.org/2012/04/27/comments-on-wang-chen-wang-paper/

[17] A stackoverflow.com Q&A thread - "Architecture for merging multiple user accounts together" (October 12, 2014) - http://stackoverflow.com/questions/6666267/architecture-for-merging-multiple-user-accounts-together

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

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