COMP116 Computer Security 2015
Final Project

Security and Privacy Concern for Single Sign-on Protocols

Name: Fangyuan Xu
Login: fxu01

Mentor: Ming Chow
Due Date: 12/15/2015

Abstract

Nowadays, Single Sign-on (SSO) service is becoming a new trend and there is a wide range of implementation, from mobile applications to browser-based protocols and even the cloud, which connects everything. It relieves users from the burden of dealing with multiple credentials but at the same time presents new security challenges. To address the problem, the paper first examines the basic model of SSO and then discusses the exiting flaws in the main emerging SSO protocols, namely SMAL SSO and OpenID. The discussion involves technical concerns, such as the vulnerabilities and the possible attacks in SSO protocols, and also the related privacy issue. Based on that, it proposes possible directions to deal with the problem.

1. Introduction

In the age that witnesses an exceedingly increase of web applications on the Internet, almost everyone has to create and remember a huge amount of accounts and corresponding credentials. A 2007 large-scale study of password habits reveals that a typical web user has about twenty-five accounts that require passwords and types eight passwords per day, which leads to "password fatigue" (Sun, Boshmaf, Hawkey, and Beznosov, 2010). As a result, users tend to use repeated or similar account password combinations across multiple websites, which imposes severe security vulnerability to the protected information.

To address the problem, there has been a wide implementation of *single sign-on* (SSO) systems. As its name implies, a Single Sign-on protocol allows the user to be authenticated and therefore have access to many different web services with one single identity. To put it another way, Single Sign-on protocol is the mechanism behind the "sign in with your Google account" button on a website like StackOverflow.

1.1 Concepts

Generally speaking, there are three parties involved in an instance of a SSO protocol: a *user* (U), an *identity provider* (IdP) and a *relying party* (RP).

The identity provider serves as a centralized identification service for the users, who establishes his identity. Some examples of such identity providers include Facebook Connect and OpenID. When the user wants to get access to services provided by a relying party, which is a website like Stack Overflow in the above case, the replying party uses the services provided by an IdP to authenticate the user. From the aspect of user, Single Sign-on enables the experience of logging in Stack

Overflow with one's Facebook account. Alternative services might include but not limit to automatic connection to Facebook friends who also log into the website with their Facebook accounts.

1.2 How does it work?

There are mainly two Single Sign-on solutions used by most of the Relying Parties, the OpenID protocol and *Security Assertion Markup Lnaguage*(SAML) 2.0. OpenID is an open and user-centric protocol for Web SSO. According to the OpenID Foundation, more than one billion OpenID enbaled user accounts were provided by major content-hosting and service providers such as Google, Yahoo and etc. To allow an OpenID user to login in, the following steps are invovled:

1. User **U** enters his OpenID ***i***, which is a URL, via a login form presented by a relying party **RP**.
2. **RP** makes an HTTP request on ***i*** to fectch a document that contains the OpenID **IdP** endpoint and redirects **U** to **IdP**. The GET request contains the RP's App ID, which is identified by the **IdP** and a redirection URL.
3. **U** enters his user name and password of the account which he sets up with **IdP**.
4. **IdP** verifies the credential and redirects **U** back to **RP** with an authentication token that **RP** can verify.

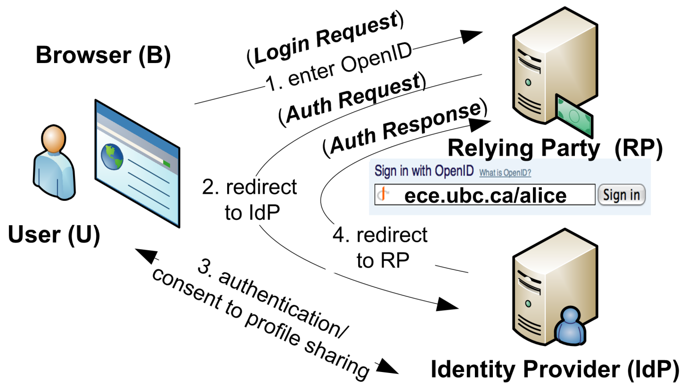

Figure 1: How OpenID works

SAML, based on which Google has developed the Single Sign-on service for the web applications including Gmail, Google Calendar, Google Docs and etc., has a different set of procedures for user authentication, which is listed as follows.

1. User **U** asks Service Provider **SP** to provide a resource. **SP** then sends **U** an HTTP redirect response to Identity Provider **IdP**, containing an authentication request AuthReq(ID, SP), where ID is a randomly generated identifier for the request. In a lot of cases, the original URI requested by **U** is placed in the *RelayState* field.
2. If **U** doesn't have an existing security context with **IdP**. For example, if the user is not currently log-in with his facebook account in the same browser, **IdP** will ask **U** to provide valid credentials.
3. If the authentication succeeds, **IdP** builds an authentication assertion and place it in a response message *Resp*. **IdP** then places *Resp* together with *RelayState* into an HTML form and sends the result back to C in an HTTP reponse, which includes some script that will post the form to **SP**.
4. **SP** then delivers the resuqested URI to **U**.

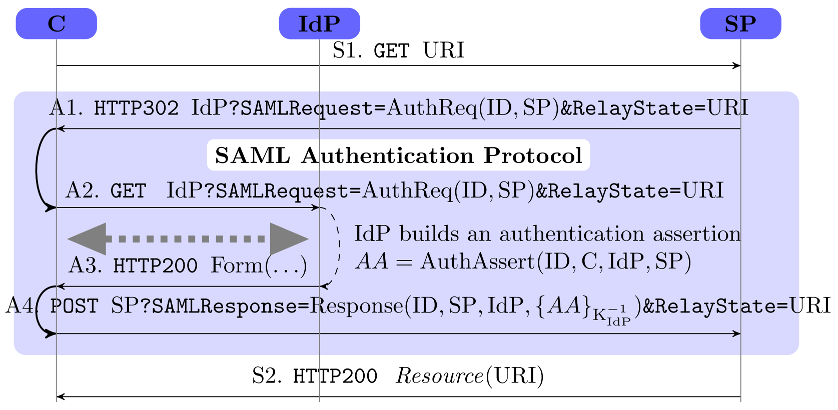
C
IdP
SP
S1. GET URI
A1. HTTP302 IdP?SAMLRequest=AuthReq(ID, SP)&RelayState=URI
SAML Authentication Protocol
A2. GET IdP?SAMLRequest=AuthReq(ID, SP)&RelayState=URI
IdP builds an authentication assertion
AA = AuthAssert(ID, C, IdP, SP)
A3. HTTP200 Form(...)
A4. POST SP?SAMLResponse=Response(ID, SP, IdP, $\{AA\}_{K^{-1}_{IdP}}$)&RelayState=URI
S2. HTTP200 *Resource*(URI)

Figure 2: How SAML 2.0 Works

2. To the Community

With Web 2.0, there has been prolieferation of web applications that permate into almost every aspect of our life. While more and more available services on the Internet bring us increasingly convenience, the management of our cridentials and personal informations have become a chanllenge in the current ecosystem of the Internet.

Single Sign-on protocol, as the current solution of the problem, is a scheme with exploitable vulnerabilities and the consequences can be severe. The attacks are possible due to some essential features of Web programming, such as the execution of script language, and human factor, as the authentication process requires human interaction.

Besides, the model of using a single internet identity on a large number of web applications created by the idea of Single Sign-on has also posed threat to personal information management and protection.

To ensure that introduction of more web application would do us good , more attention should be drawn to the design of credential management and Single Sign-on is a good point of penetration.

3. Vulnerabilities

With three different parties participating in the authentication processes, both SSO solutions involve different layers of communication and exchange of credentials that are enabled by using HTTP redirection and Javascript, which creates several vulnerabilities for attacks to exploit and makes SSO a launch pad for typical attacks such as Cross Site Request Forgery and Cross Site Scripting.

3.1 Malicious Identity Provider

With different layers transitied by redirections in the process of SSO authentication, attackers are provided opportunity to steal the identity of the trusted party and therefore launch the attack. As the **User** is redirected to the **IdP** by Service Provider, it is easy for a malicious **SP** to redirect **U** to a fake **IdP** and therefore steal the user's credentials. Since a lot of users might not be careful enough to identify the fake web page, the same old enemy, also known as "phishing" is still an unresolved and well-known attack against SSO.

3.2 Malicious Relying Party/Service Provider

Compared to a fake web page, it is even harder to identify the integrity of a URI request. Based on that, the attack can be conducted on a website that supports SAML-based Single Sign-on, and it invovles four pourties: a user(U), a malicious service provider(M), a honest service provider(SP) and a honest Identity Provider(IdP). At the beginning, **U** requests a resource URI^1 from **M**. **M** then pretends to be U and request a different URI^2 at **SP**, which will react according to SAML standard by generating an Authentication Request to IdP. **M** then redirects **U** to **IdP** by an HTTP response containing the Authentication request and URI^2 . And thus **U** is forced to consume a different resource from **SP**, which is the URI^2 created by **M** while **U** thought he asked URI[1] from **M**. This vulnerability is possible based on the fact that the user has no means, and they usually have no incentive, to verify whether the authentication request **SP** generates and the authentication assertion from **IdP** are related to the same initial one.

Up to this point, the vulnerability seems nothing more harmful than being served a cup of coffee when asking for hot chocolate. However, it can be a launching pad for Cross Site Scripting, which might result in comprimised user's credentials. Google Apps, implemented in the SAML-based SSO, was once vulnerable to the attack. In the service offered by Google up to 2009, when **SP** recieves a request for a resource URI from **U**, if the request is accompanied by a valid session cookie, such as Google Calender, Gmail and etc., then the resouce is returned right away. Therefore, in the above scenario, if **M** injects some malicious code such as http://i/collect.php?cookies='+document.cookie in the *RelyState* field, it will be executed in the client's browser, which leads to the theft of the Google Calender, Gmail cookies of the User. Then **M** can use to the cookie to gain access to the Google Apps using **U**'s account. Though the issue has been fixed by properly sanitizing the *ReplyState*, it still poses concerns regarding to the flaw existed in the SSO model.

3.3 Replay attack

The last redirection involved in the SSO also tempts the attackers to evil. In an SSO authentication, the User is redirected by the **IdP** back to the **SP** through a URL, which looks something like:

http://www.somewebsite.com/finish_auth.php?openid.assoc_handle=%7BHMACHA1%7D%7B47bb..&openid.identity=http%3A%2F%2user.name%2F&openid.mode=id_res&openid.return_to=http%3A%2F%2www.somewebsite.com&openid.sig=vbUyND6n39Ss8IkpKl19RT83O%2F4%3D&openid.signed=mode%2Cidentity%2Creturn_to& nonce=wVso75KH

However, this URL can be obtained by anyone through methods like sniffing the network and thus the attacker can get logged into the **SP** as the **U**. An existing solution to the problem is to use a nonces, i.e. number used once, to allow a user to log into the site once and fail all consecutive attempts. However, the attacker can still achieve his goal by obtain it before the user and reset the user's TCP connection to conduct the replay attack.

The fact that the **SP** doesn't dictates the user login policy also paves way for Cross Site Request Forgery. Assuming that the user has already logged in with the **IdP** when they log in on some **SP**^{1}, the attacker can silently make the user log into a different **SP**^{2} by another account associated with the same **IdP** account and exploit the service provided by the **SP**^{2} by inserting command in the URI.

4. Other concerns

Apart from the above vulnerability created by the flaw in SSO protocols, there are other concerns when users are using the same identity on different websites. Issues brought by the Single Sign-on mechanism includes the vulnerabilities in further communication between the Service Provider and the Identity Provider, which further relates to the privacy concern of users' personal information.

4.1 You are under surveillance

As the Identity Provider acts as a center for authentication of user identity, it can easily spy on a user's activity on the Internet, i.e. what websites the user browses and therefore collect enormous amount of user information.

4.2 Link-ability and a key to a million locks

With the scheme of Single Sign-on, what is shared goes beyond the user's identity. It is a common practice for the Service Provider, after the authentication, to query the Identity Provider, such as Facebook, for the user's information including his list of contact, his personal information such as birthday, age and etc. This action is performed with permission gained through a bottom like “Let’s see who is also on Yelp” and it does provide to a certain degree of convenience and a richer experience of social connection. However, the personal information exposed in the background and the great link-ability created still pose concern regarding a person’s identity disclosure on the Internet.

While releasing the user from the burden of creating and memorizing different sets of usernames and passwords, the Single Sign-on scheme is a key to a million locks. That being said, if the identity the user establishes with the Identity Provider is compromised, the aftermath will be severe.

4.3 More vulnerability

As mentioned above, there can be more communication between **SP** and **IdP** in exchange of the user’s personal information, which is often supported by API provided by the **IdP**. However, vulnerabilities are found in the process, which might lead to the leak of users’ personal information. For example, Facebook provides a JavaScript library for sites implementing Facebook Connect, which creates two hidden iframes for communication. The first iframe communicates directly with the **SP** while the second one serves as a proxy between the **SP** and Facebook. However, it is found in a study that it is possible to launch a man-in-the-middle attack due to the fact that **SP** doesn’t validate the sender of messages and the parameter indicating target origin is set to * occasionally. This is an implication of the real world situation where developers do not use the properties provided by the API correctly, which is another question in terms of the ecosystem of Web programming (Hanna, Shin, Akhawe, Boehm, Saxena and Song, 2010).

5. Mitigations

As discussed above, the lack of identity authentication in each step of communication largely accounts for the vulnerability of SSO protocols. However, the use of a single SSL connection for the exchange of different messages cannot be guaranteed, as an SSL server could not resume a previously established session. Also, how to design a user-friendly login UI and login interaction flow for web users is still an open problem for ongoing SSO research. As an attempt to address the problem, several possible directions are listed in this section.

5.1 Verifying the Service Provider

To avoid a man-in-the-middle attack between **IdP** and **SP**, the **IdP** can ask the user to verify if the authentication is coming from the same **SP** that the user has asked for service before issuing the authentication assertion to **SP**.

5.2 Verifying the User

An alternative approach suggests that **SP** should verify the identity of the user to make sure it is always interacting with the same user.

Some researchers suggest the use of cookie to realize the verification. By setting the target URI (the resource requested by the user from the **SP**) as the value of cookie, the value of cookie can be verified when the user is redirected back from **IdP** to **SP**. Self-signed certificate of the client has also been brought up as a solution (Armando, Carbone, Compagna, Cuéllar, Pellegrino, and Sorniotti, 2013).

6. Conclusion

This paper focuses on the security issues regarding to the Single Sign-on protocols. By examining the two existing Single Sign-on solutions, OpenID and SAML 2.0, it analyzes the vulnerability posed by the use of SSO and lists some possible attacks. Besides, it discusses the security concerns brought by SSO beyond the authentication process, which includes concerns for personal information protection. Finally, it proposes possible directions to address the problem.

Reference

Armando, A., Carbone, R., Compagna, L., Cuéllar, J., Pellegrino, G. and Sorniotti, A., 2013. An authentication flaw in browser-based single sign-on protocols: Impact and remediations. *Computers & Security*, *33*, pp.41-58.

Cao, Y., Shoshitaishvili, Y., Borgolte, K., Kruegel, C., Vigna, G. and Chen, Y., 2014. Protecting Web-based Single Sign-on Protocols against Relying Party Impersonation Attacks through a Dedicated Bi-directional Authenticated Secure Channel. In *Research in Attacks, Intrusions and Defenses* (pp. 276-298). Springer International Publishing.

Hanna, S., Shin, R., Akhawe, D., Boehm, A., Saxena, P. and Song, D., 2010, May. The emperor's new APIs: On the (in) secure usage of new client-side primitives. In *Proceedings of the Web* (Vol. 2).

Sun, S.T., Boshmaf, Y., Hawkey, K. and Beznosov, K., 2010, September. A billion keys, but few locks: the crisis of web single sign-on. In *Proceedings of the 2010 workshop on New security paradigms* (pp. 61-72). ACM.

Tsyrklevich, E. and Tsyrklevich, V., 2007. Single sign-on for the Internet: a security story. *BlackHat USA*.