SINGLE SIGN ON (SSO) APPLICATION FOR WEBSITES

Udit Chitalia,

Maulik Sanghavi,

Sowmya Iyer,

Shrey Shah,

Varshapriya Jyotinagar

Department of Computer Engineering and Information Technology, Veermata Jijabai Technological Institute, H.R.Mahajani Marg, Matunga, Mumbai - 400019, Maharashtra, India.

ABSTRACT: *"Single sign-on (SSO) is a property of access control of multiple related, but independent software systems. With this property a user logs in once and gains access to all systems without being prompted to log in again at each of them"-Wikipedia. We use many websites like Facebook, Gmail, bank websites and many others on a day to day basis. Every time we have to individually enter the username and password for every website, which may be troublesome. It causes password fatigue. Web browsers like Google chrome allow saving the username and password, but the password is visible in the clear in the Google chrome settings to an outsider that is, another user of the same computer. Managing passwords of social sites, banks etc becomes risky and people end up saving them in a notepad file or in some way such that there is always a possibility of someone acquiring those passwords. This manual password management is obviously insecure. There are applications which act as password managers, which store the passwords in a centralised repository in a cryptographic form. Centralised and secure password management is the need of the day. We propose the idea of a "SINGLE SIGN ON APPLICATION" that has a master password. The user needs to login in this application which will further log him to all his websites, thus decreasing load on user and saving his time. This SSO application is secured and follows military grade security to user data.*

KEYWORDS: Single sign on, Password fatigue, Authentication.

INTRODUCTION

Many people today have multiple accounts on the Internet. For example, one may have an email account on www.yahoo.com, a travel account on www.travelocity.com, a credit card account on www.discovercard.com, a banking account on www.chase.com, an online stock trading account on www.delity.com, etc. Forrester Research reports that a typical web user manages an average of 15 passwords on a daily basis [10]. Most of these accounts are protected by passwords. As more services move to the Internet, the number of accounts a user needs to manage is expected only to grow. If one uses different and unrelated passwords for each account, then remembering all these unique passwords is a daunting task. This is known as **password fatigue**.

The average user has 6.5 passwords, each of which is shared across 3.9 different sites. Each user has about 25 accounts that require passwords. Users forget passwords a lot: we estimate that at least 1.5% of Yahoo users forget their passwords each month. We were able to measure that 0.4% of users type passwords (on an annualized basis) at verified phishing sites. In order to remember such an exorbitant number of passwords users end up having small passwords having only lowercase alphabets, unless forced to do otherwise or they have same passwords for multiple number of sites, to break such passwords is very easy for hackers.

So we aim to create an application which has a master password and logs in to all user's websites on his behalf. The application needs to implement highly secure protocols to protect user data.

What is Single Sign On

Most of current application architectures require the user to memorize and utilize a different set of credentials (e.g username/password or tokens) for each application he/she wants to access. However, this approach is inefficient and insecure with the exponential growth in the number of applications and services a user has to access both inside corporative environments and at the Internet. Mainly, it is difficult for a corporation to manage potentially multiple authentication solutions and databases individually used by each application. Furthermore, most users tend to rely on the same set of credentials for accessing all of their systems, posing a serious security threat since an attacker who discovers these credentials can easily access all of the user's applications.

In a single sign-on platform, the user performs a single initial (or primary) sign-on to an identity provider trusted by the applications he wants to access. Later on, each time he wants to access an application, it automatically verifies that he is properly authenticated by the identity provider without requiring any direct user interaction. Single sign-on solutions eliminate the need for users to repeatedly prove their identities to different applications and hold different credentials for each applications. Furthermore, a well designed and implemented single sign-on solution significantly reduces authentication infrastructure and identity management complexity, consequently decreasing costs while increasing security.

Types of SSO Systems

The different architectures into which the SSO system can be classified into:

Pseudo Sso Systems[1][2]

One obvious approach for the relieve of the user is to support him or her with the credential management. The user can store all passwords and authentication information into an encrypted password file or database which is secured with a master password. This password may be more complex than a normal password because the user only has to memorize this single password. It is also possible to secure the credential database with an fingerprint[3] or hardware based protection([4]). But as multiple different authentication processes are still taking place, even if this happens without necessary user intervention, such systems cannot be called true single sign-on systems.

1) *Implications* [5][2]: The credential database may be corrupted or lost, effectively locking the user out if he can't remember the passwords (which is likely if they are not used regularly). On the other hand the user has to type the master password very often, which may help him to memorize it. This also means that an attacker only needs to get the master password to access all services of the user. This solution also affects the users mobility because the credential database has to be available on every computer he uses. To solve this problem the credentials may also be stored online, but the user has to trust the provider of such a service.

Centralized SSO

The main characteristics of a centralized single sign-on systems are a centralized authentication site and a centralized user database. Such a site has a trust relationship with each of the Service Providers (SP) within a domain and is often called a Trusted Third Party (TTP). The user only has to authenticate himself to the TTP. Such systems differ in the way a user is authenticated and can be further categorized as follows:

1) *Token Based SSO*[6]: After a user is authenticated at the TTP he receives a software token which is stored (cached) on his client machine. The Service providers can validate such a token with cryptographic methods based on secret keys (symmetric cryptography). Those keys establish a trust relationship between the TTP and the SPs.

2) *PKI Based SSO*[6]: In PKI based systems the user first has to generate an asymmetric key pair. The public key of this pair has to be sent to the central authentication site which is called Certification Authority (CA) in this case. After the user has authenticated himself the CA signs his public key and sends the certificate generated this way back to the user. The Service Providers are now able to verify the users certificate by using the CA's public key.

Federated SSO[6]

Centralized single sign-on Systems are limited to a single Environment, Company or Domain. To establish trust between different Domains, Federations are needed. The goal is therefore a system in which a users credentials from his domain are accepted by a foreign domain without the user having to re-enter his credentials. The foreign Domains TTP (or CA) has to accept the local Domains credentials because of an existing contract or federation. No synchronization or copying of credentials is needed. For PKI based Systems this may be accomplished by CA hierarchies or cross certification.

SINGLE SIGN ON APPLICATION

Our goal is to create an application, in which a user can add his preferred websites (the ones he uses on a daily basis), the corresponding usernames and password for each of them, and by just signing into this application he automatically gets signed in to all his required websites. This acts as a Centralised SSO system.

Begging into a website

Logging in to a website when the website URL and credentials are known can be done programmatically using the following steps:

- Get the page source of the URL
- Find the unique identifiers of the html input elements corresponding to the username, password and sign in button fields.
- Using the identifiers, fill in the values of the username and password fields, which will initially have no values..
- Include a java script function such that when the body of the html page is loaded, the sign in button is clicked.
- Launch the webpage with the modified source code on a browser.
- The above approach is not actually a parser, but a logic that works for most websites including secured websites like Facebook. Here the job of creating the HTTP Post request is left to the browser and hence the implementation becomes simple and easy.

Ways to implement our SSO application

There are two possible ways to implement the SSO application:

1) The centralised database approach:

- The usernames and passwords of the websites will be stored centrally on a database server on a server machine
- The database will be protected and secured using cryptography and network security algorithms
- To retrieve the credentials, TCP/IP or UDP connections will have to be established between the client machine and the database server machine which will be slightly time consuming. This has to be done for every chosen website.
- The time delay is negligible, because the advantages like elimination of password fatigue, security of credentials become more prominent.
- The SSO application can be used from any machine to access websites as the data is stored centrally and hence there is ease of use.

2) The local database approach:

- The usernames and passwords of the websites will be stored locally on the client machine
- There is no need to use complex algorithms. Only a password or fingerprint to access the database is sufficient. But the risk of multiple users accessing the machine and security breach is possible.

- To retrieve the credentials, no TCP/IP or UDP connections are required. There is less delay as compared to centralised database approach.
- The SSO application can be used only from one machine as data is stored locally and this limits the use of the application.

Android SSO application

- Developing an SSO application for android smart phones and tablets may be a good research project. The credentials can be stored locally on the phone or tablet.
- But it may not appeal to the masses that still rely on desktop computers to access multiple websites and access the web.
- An android based application will be more successful when android appliances are used more than desktop computers and when the faster internet connections at affordable rates become available for phones and tablets.
- It will have a smaller target group in today's time compared to its desktop counterpart, but it is a good area for research and development.

User Centric User Identity Management (Our Secure Solution)

An authentication solution must take into consideration how the identifiers and credentials are to be handled by the user. If the usability is poor, then the authentication itself will be weak because users are unable to handle their credentials adequately.

In this regard, it is interesting to notice that service providers usually have automated systems to manage identities and authentication, whereas users normally manage credentials manually. From a user perspective, an increasing number of identifiers and credentials rapidly become totally unmanageable.

The idea is that if the user only needs to manage one set of identifier and credential, memorisation or other primitive methods for storing credentials are still acceptable. However, it is inconceivable that only one single federation domain will exist, and it is evident that there will never be a single identity domain for all service providers. Also, services with different levels of sensitivity and risk will require different types of credentials. As a rather optimistic scenario, it could be suggested that the number of identifier/credential sets a user needs to manage in case of widespread adoption of federated identity domains, would be 1 order of magnitude less than the number of service providers he accesses. Unfortunately, the user experience will still be poor when the number of online service providers is growing exponentially.

In our view, a totally new approach is needed. It seems natural to introduce automation and system support of the identity management at the user side. Expecting users to manage an unavoidably growing number of passwords and credentials by memorisation or other primitive methods is totally unrealistic.

A solution, which seems quite obvious, is simply to let users store identifiers and credentials from different service providers in a single tamper resistant hardware device which could be a smart card or some other portable personal device. This approach opens up a multitude of possibilities of

improving the user experience and of strengthening the mutual authentication between users and service providers.

Because its main purpose would be authentication, the device can be called a personal authentication device (PAD). This is illustrated in Fig below.

The user must authenticate himself to the PAD, e.g. with a PIN, before the PAD can be used for authentication purposes. Many different authentication and access models can be imagined with a PAD. In case the PAD has a keyboard and display, a simple solution could for example be to retrieve from PAD memory a static password, or let the PAD generate a dynamic password, that the user then can type into the login screen of the service provider. A morea dvanced solution could be to connect the PAD to the client platform via a communication channel such as bluetooth or wireless LAN, or to let the PAD communicate directly with the server through a secondary channel. This would allow the PAD to be fully integrated into the authentication process.

The functionality of a PAD could be integrated into other devices such as a mobile phone or personal digital assistant (PDA) which many people carry already. Using a mobile phone would also allow advanced solutions such as registration and challenge- response authentication through a mobile secondary channel. With a PAD connected to the client platform, virtual SSO solutions are possible. This could be implemented by letting the PAD automatically authenticate itself on behalf of the user as long as the PAD is connected to the client platform. The advantages of the user-centric user identity management architecture are that 1) the user only needs to remember one credential (e.g. the PAD PIN), 2) that virtual SSO is possible, Signs of this type of solution are already emerging. For example, the Mozilla browser provides virtual SSO capabilities for users so they do not have to remember their usernames or passwords for web sites. A master password protects the PKCS11 security device, which can be either a software or hardware device that stores sensitive information associated with their identity, such as usernames and passwords, keys and certificates. Recent releases of Mozilla have a software-based security device, and can also use external security devices, such as smart cards, if the user's computer is configured to use them. The master password for the browser's built-in software security device protects the user's master key, which is used to encrypt sensitive information such as email passwords, web site passwords, and other sensitive data

The PAD should be under the control of the user, and not under the control of the identifier providers, the credential issuers or the service providers. The latter would result in a proliferation of PADs which would defeat the purpose of having a single device for simplification identity management for the users. In order to gain full advantage of the PAD, it should be a general security device capable of handling any types of identities and credentials.

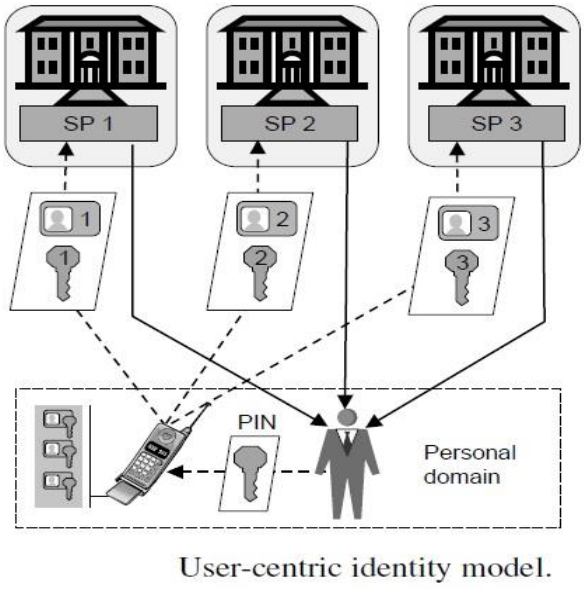

Figure 1. User Centric Identity Model

ADVANTAGES

Why Single Sign on when other applications for doing the same are present?

We take an example of most famous plugin **LastPass** which is used for Single Sign on

Last Pass	**Our application**
1. It's a plugin so its restricted to a specific browser. For eg: A Google Chrome plugin cannot work for IE or Mozzila FireFox.	1. It's a standalone application so it's not restricted to a browser any browser will work.
2. Just fills up the textboxes on a webpage does not log in directly	2. The application will directly login to a website
3. If a person steals the master password he gets access to all the sites.	3. ONE TIME PAD is used. A pin will be generated and used only for a particular session. So even if one steals the master password he will not be able to login.

Figure 2. Comparision between Last Pass and Our Application

The growing need for security and the even faster growing variety of applications, services and systems requires an appropriate authentication infrastructure. The diversity has led to an equally diverse range of authentication systems. Users need to manage different usernames, passwords or other forms of authentication to each of this systems. The wish for an single sign-on is therefore quite understandable

THREATS

Apart from the benefits of having only one password there could also be negative implications to the users security and privacy. Furthermore such a system would introduce a single point of failure, where one stolen password can give access to a lot of services. People may be hesitant about their passwords being stored in a centralized database.

CONCLUSION

A system with centralised storage of user credentials that allows access to multiple sites is, thus, possible. This paper described a possible solution to the problem of password fatigue along with its advantages and disadvantages. A description of the idea used to implement the solution has been presented. The solution proposed will thus provide a secure and portable means of multi site login with little difficulty to the user.

REFERENCES

[1] Pashalidis, A., Mitchell, C.J.: A taxonomy of single sign-on systems. LECTURE NOTES IN COMPUTER SCIENCE (2003).

[2] Ronagel, H., Zibuschka, J.: Single sign on mit signaturen. Datenschutz und Datensicherheit - DuD 30(12) (2006)

[3] Park, B., Hong, S., Oh, J., Lee, H., Won, Y.: One touch logon: Replacing multiple passwords with single ngerprint recognition. In: CIT '06: Proceedings of the Sixth IEEE International Conference on Computer and Information Technology, Washington, DC, USA, IEEE Computer Society (2006) 163.

[4] Jones, M.F., Zachai, A.: Encrypted data storage card including smartcard integrated circuit for storing an access password and encryption keys (April 1997).

[5] Adams, A., Sasse, M., Lunt, P.: Making Passwords Secure and Usable. PEOPLE AND COMPUTERS (1997) 1{20.

[6] De Clercq, J.: Single Sign-On Architectures. LECTURE NOTES IN COMPUTER SCIENCE (2002) 40{58.

[7] Delessy, N., Fernandez, E.B., Larrondo-Petrie, M.M.: A pattern language for identity management. Computing in the Global Information Technology, International Multi-Conference on 0 (2007) 31.

[8] Zwattendorfer, B.: Single sign-on unter verwendung der burgerkarte. Master's thesis, Graz University of Technology (May 2006).

[9] Stojceski, D.: Konzeption einer kerberos-basierten single sign-on losung fur ein ausgewahltes szenario im hochschulbereich. Master's thesis, University of Applied Sciences of Bonn-Rhein-Sieg (March 2006).

[10] R. Kanaley. Login error trouble keeping track of all your sign-ons? here's a place to keep your electronic keys, but you'd better remember the password. San Jose Mercury News (Feb. 4, 2001).

[11] Morgan, R.L., Cantor, S., Carmody, S., Hoehn, W., Klingenstein, K.: Federated security : The shibboleth approach. EDUCAUSE Quarterly 27(4) (2004).

[12] Geihs, K., Kalckl□osch, R., Grode, A.: Single sign-on in service-oriented computing. In: ICSOC. (2003) 384-394.

[13] Eian, M., Mjlsnes, S.: Large scale single sign-on scheme by digital certificates on-they. In: Norwegian Network Research Seminar. (2005)