Creating a unified online identity to provide a single seamless presence on the internet

Apurva Patkeshwar [*], Kushan Kunal Prasad[*], Suril Dhruv[*], Dr. Radha Shankarmani [**]

[*] Information Technology, Sardar Patel Institute of Technology, Mumbai, India
[**] Head of Department, Information Technology, Sardar Patel Institute of Technology, Mumbai, India

Abstract- Online presence has become an important part of everyone's life. This paper provides an overview of the technologies used in creating a portal to achieve a unified online identity via integrating user identities across different networks. It mainly focuses on the MVC architecture of the portal, the bcrypt algorithm for a secure log in, OpenID and OAuth implementations for authentication and Oliver's text similarity algorithm for an efficient search. It also describes the scope for further improvements and possible future developments in achieving a seamless unique online presence across the internet.

Index Terms- unique identity, online presense, MVC, bcrypt, OpenID, OAuth, text similarity, UID, UOI

I. Introduction

Internet identity (sometimes also referred as "internet persona") is a social identity that an Internet user establishes in web portals and online communities. It can also be considered as an actively constructed presentation of oneself. Although some people prefer to use their real names online, some Internet users prefer to be anonymous, identifying themselves by means of pseudonyms, which reveal varying amounts of personally identifiable information. An online identity may even be determined by a user's relationship to a certain online social group they are a part of. Some may even prefer to be deceptive about their identity.

In some online contexts, including internet forums, chat rooms, and massively multiplayer online role-playing games (MMORPGs), users can represent themselves visually, by choosing an avatar, an icon-sized graphic image. These avatars are one of ways the users express their online identity. As other users interact with an established online identity, it acquires a reputation, which enables them to decide whether the identity is worthy of trust. [1] Some websites also use the user's IP address to track their online identities using methods such as tracking cookies.

So this project aims at providing a central web portal for maintaining every users list on online identities hence aiding searching of known people on new networks. This helps in a multitude of tasks. It assists users by providing a facility to search for people with similar interests. It also caters to emerging or lesser known websites, not just the famous ones.

II. Need of a Unified Online Identity

Almost everyone has access to the internet in today's rapidly advancing age. And most people who use the internet are sure to have an account on more than one social/professional networking sites or other networks. Even though the person accessing these accounts many be one, he is essentially a different entity on every site. This gives way to many problems.

Primarily, from a layman's point of view, it is increasingly difficult to find the person you want on a website primarily due to numerous repeated names. For instance, from a list of John Doe's it becomes impossible to figure out who is the one being searched for.

Also, if a person is not present on a specific network, it opens an opportunity for a malicious person to impersonate him on the same. So it becomes imperative to know which accounts are the real accounts of one's friends on different networks. In this way one can protect oneself from falling prey to identity thefts.

Furthermore, in the current scenario, a few sites like Google and Microsoft provide integration with other networks. For example, Google integrates the accounts for YouTube, Gmail, Google+, Google Drive etc. into one. But the scope of this is very limited. Like in the case of Google, the integration is limited to Google-owned networks. In the case of Microsoft, a little wider range is allowed as it allows integration with a few other networks as well. But it caters only to big, well known networks.

So the need arises for a web portal to do the work of a central repository where one can just list all his/her online identities. This facilitates one's searches for one's friends. It also lets him/her get searched and found by people with common interests or even organizations with specific needs.

III. System Architecture

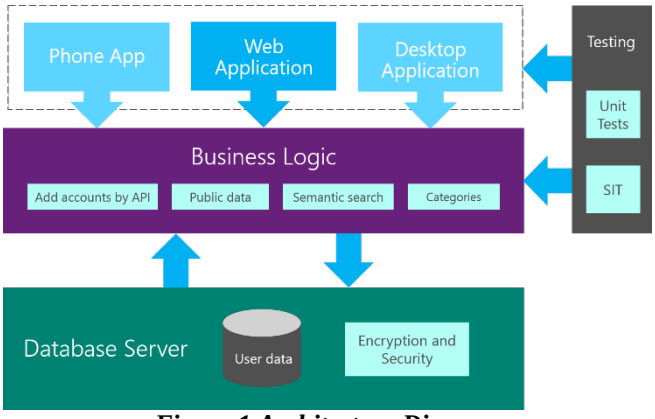

Figure 1.Architecture Diagram

MVC

The basic structure of MVC is as follows:

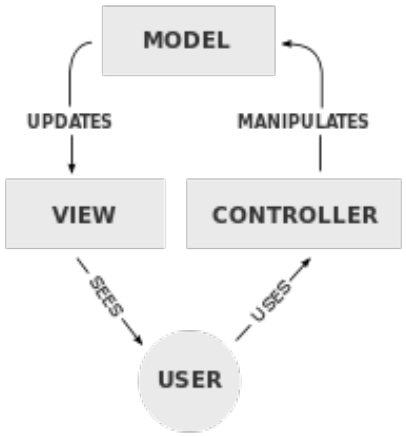

Figure 2.Model-View-Controller

The framework divides the application into three components. It also defines the interactions that take place between them.

- A controller can send commands to the model to update the model's state (e.g., changing contents of a document). It can also send commands to its associated view(s) to change the view's presentation of the model (e.g., by scrolling the document).
- A model can notify its associated view(s) and controller(s) its state changes. This notification allows the views to update the output, and the controllers to change the set of commands. A passive implementation of MVC may omit these notifications, because the application may not require them and/or the software platform may not support them.
- A view requests information from the model that it needs for representing the output data to the user.

Codeigniter framework

The framework being used for the implementation of the MVC architecture is the PHP-based framework, CodeIgniter.

CodeIgniter, an open source rapid development web application framework, is a comprehensive toolkit for building dynamic websites with PHP.

It is an extremely light weight MVC-based system which features Active Record Database Support, Form and Data Validation, Security and XSS Filtering, Session Management, Email Sending Helper Class, Image Manipulation Libraries, FTP Helper Class, Localization, Pagination, Data Encryption, Benchmarking, Full Page Caching, Error Logging, Application Profiling, XML-RPC Library, Unit Testing, Flexible URI Routing and support for Hooks and Class Extensions.
The portal is built on CodeIgniter version 2.1.4.

IV. Security

Bcrypt

"Bcrypt", in simple terms, is a key derivation function for passwords. Based on the Blowfish cipher, it was designed by Niels Provos and David Mazières, and presented at USENIX in 1999. [2] Besides incorporating a salt to protect against rainbow table attacks, bcrypt is also an adaptive function: the iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks even with the increasing computation power over time.

Blowfish is notable among block ciphers for its expensive key setup phase. It starts off with subkeys in a standard state, then uses this state to perform a block encryption using part of the key, and uses the result of that encryption (which is a hashing) to replace some of the subkeys. Then it uses this modified state to encrypt another part of the key, and uses the result to replace more of the subkeys. It proceeds in this manner, using a progressively modified state to hash the key and replace bits of state, until all subkeys have been set.

Provos and Mazières took advantage of this approach, and took it further. They developed a new key setup algorithm for Blowfish, and dubbed the resulting cipher "Eksblowfish" ("Expensive Key Schedule Blowfish"). The key setup begins with a modified form of the standard Blowfish key setup, in which both the salt and password are used to set all subkeys. There are then a number of rounds. In each round, the standard Blowfish keying algorithm is applied, using alternately the salt and the password as the key. Each round starts with the subkey state from the previous round. Cryptotheoretically, this is no stronger than the standard Blowfish key schedule. But, as the number of rekeying rounds is configurable; this process can therefore be made arbitrarily slow, which helps deter brute-force attacks upon the hash or the salt.

The iteration count is a 2^n, where 'n' is the power of two, which is an input to the algorithm. The number is encoded in the textual result.

The bcrypt algorithm depends heavily on its "Eksblowfish" key setup algorithm, which runs as follows:

EksBlowfishSetup(*cost*, *salt*, *key*)
state ← InitState()
state ← ExpandKey(*state*, *salt*, *key*)
repeat (2^{cost})
state ← ExpandKey(state, 0, key)
state ← ExpandKey(state, 0, salt)
return *state*

The full bcrypt algorithm utilizes these functions to compute a hash as follows:
bcrypt(*cost*, *salt*, *input*)
state⟵ EksBlowfishSetup(*cost*, *salt*, *input*)
ctext⟵ "OrpheanBeholderScryDoubt" //three 64-bit blocks
repeat (64)
ctext⟵ EncryptECB(*state*, *ctext*) //encrypt using standard Blowfish in ECB mode
return Concatenate(*cost*, *salt*, *ctext*)

Authentication and Authorization

Majority of the API's provided by different networks to authenticate the user and authorize third-parties to access user details use two mechanisms: OpenID and OAuth. These standards are explained in detail below:

OpenID

OpenID is an open standard that allows users to be authenticated by certain co-operating sites (known as Relying Parties) using a third party service. This eliminates the need for webmasters to provide their own ad hoc systems and allows users to consolidate their digital identities together. [3]

OAuth

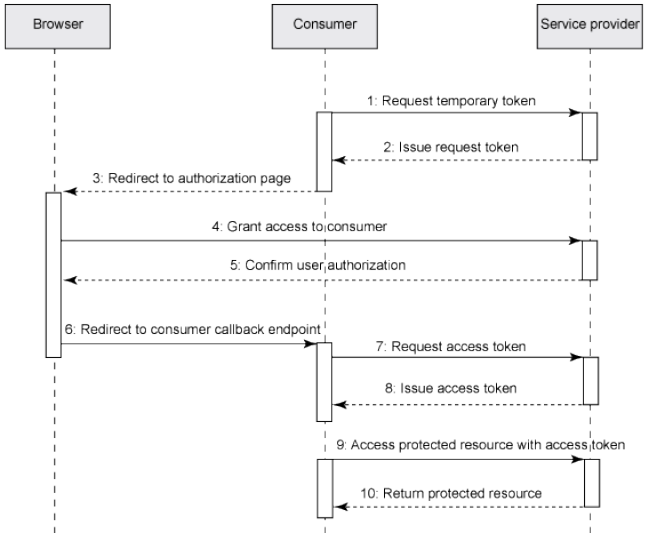

Figure 3.Three-Legged OAuth Dance

OAuth is an open standard for authorization. OAuth provides a method for clients to access server resources on behalf of a resource owner (such as a different client or an end-user). Also, it provides a process for end-users to authorize third-party access to their server resources without sharing their credentials, typically a username and password pair, using user-agent redirections.

OAuth is a service that is complementary to, and therefore distinct from, OpenID.

OpenID vs. pseudo-authentication using OAuth

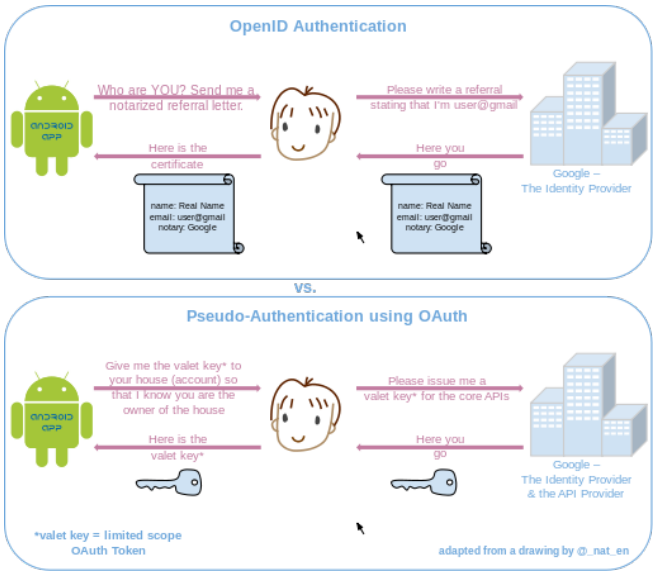

Figure 4.OpenID vs. Pseudo-Authentication using OAuth

The following figure highlights the differences between using OpenID vs. OAuth for authentication. Note that with OpenID, the process starts with the application asking the user for their identity (usually an OpenID URI), whereas in the case of OAuth, the application directly requests a limited access OAuth Token (a valet key) to access the APIs on the user's behalf. If the user can grant that access, the application can retrieve the unique identifier for establishing the identity using the APIs.

V. SEARCH

Introduction

Search is an important functionality provided to the user, which facilitates him to search his friends on the web.

Text Similarity
Introduction

Text similarity has two basic stages. Firstly, feature strings (called "fingerprints") are extracted from each text in different ways. Secondly, the similarity of a text pair is measured by comparing the fingerprints of the texts, using different models.

The five similarity models[5] are:

1. Ferret model [5, 6, 7], which looks for lexical matches. Each text is converted to a set of 3 word trigrams. Then, the matching trigrams are found. Though there are usually some matches, above a certain threshold the texts are similar. The comparison is based on the Jaccard coefficient value.
2. String matching model, which counts those words/word sequences that occur in both texts (omitting stop words) and calculates the Jaccard coefficient value to determine the similarity between two texts.
3. Meaning matching model, which is developed from the string matching model. The difference is that meanings rather than lexical strings are matched between texts.

The meanings of a string include the synonyms/hypernyms of the whole string as well as that of each word in the string. WordNet is used to look up word meanings.

4. Semantic sequence kin model (SSK), which extracts semantic sequences from a text as its feature strings, and then takes into account both word and word position when 2 semantic sequences are compared.
5. Common semantic sequence model (CSSM), which is similar to semantic sequence kin model, but uses another formula to calculate similarity of semantic sequences. In this model, the word position is not considered.

Out of these five similarity models, we have used the "string-matching model". This model has been implemnted by an in-built PHP function similar_text(). This function for text similarity implements Oliver[1993] algorithm[4]. It compares the two strings, one of them is entered by user and the other is database entry. This function returns a value known as the "similarity ratio" based on number of characters common in both the strings. If the similarity ratio is above the set threshold, then the link to the profile is stored in an array and after completition of search is shown as an output to the user. This implementation does not use a stack as in Oliver's pseudo code, but recursive calls which may speed up the whole process. Also, note that the complexity of this algorithm is O(N**3) where N is the length of the longest string. The user can view the public profile of searched individual and can find his links on other website. Hence the objective is met.

Calculation of similarity_ratio:

Similarity ratio = sim * 200.0 / (len_t1 + len_t2)

where, sim-Number of similar characters, len_t1 and len_t2 are the lengths of the strings to be compared.

Future Development

There is a scope of improvement in the search functionality. The queries can first check the cache and if a hit is found there then return the link to the profile from there itself. It saves time as the need to scan the entire database is not required. The caching functionality is available in PHP and the decision can be made to cache the type of queries. This reduces the burden on system and reduces the search time.

VI. Future Development

Semantic search and Natural Language Processing

There are three methods to implement semantic search as follows:

- String matching method

The first step identifies locally frequent words; the second step extracts phrases consisting of adjacent frequent words and the third step looks for matches. A semantic sequence can be a phrase or a single word. The method is implemented by the Java class StringMatchBasedSimilar with the class SemSeqBasedTopicStrings and the class SameWord SS.

- Semantic Sequence Kin method (SSK)

Like string matching method for semantic sequences, the first step is to identify locally frequent words. These frequent words are compared for the two texts. The metric used also takes into account the distance between frequent words. The method is implemented by the class SSK along with the same class SameWord SS.

- Common Semantic Sequence Model (CSSM)

Similar to the Semantic Sequence Kin method (SSK), but employing a different formula to measure the similarity score. The method is implemented by the class CSSM with the class SameWord SS.

Web service

The project develops a centralized repository of all the online identities of a specific user. This data is very useful for many purposes. It can be used by search engines to improve the validity of the results. It can also be used by existing networks to identify if an account is genuine or not. So we can write a web service that makes this data available to any consumer. This API could return the online identities of a user to authorized applications.

Use Crawlers

Crawlers can be used to authenticate a users profile on networks that do not use OpenID or OAuth. The crawler can go to the network and verify from the web page whether it belongs to the same user by matching details.

They can be also be used to suggest a user his identities on other networks. This can be done by searching other networks with the known usernames.

VII. Conclusion

The portal helps unify all the internet identities into a single online presence. The web portal does the work of a central repository where one can just list all his online identities. It makes it easier for his/her friends and acquaintances to find him/her on any website. It also lets him/her get searched and found by people with common interests or even organizations with specific needs. It also prevents malicious people to impersonate users who are not present on a particular network. In this way, the portal helps protect users from falling prey to identity thefts.

Furthermore, it provides a complete solution to integrate a multitude of networks unlike the current integration options provided by tech giants like Google and Microsoft, which mostly focus on integration of their own services.

References

[1] Nabeth, Thierry (26 May 2006). Understanding the Identity Concept in the Context of Digital Social Environments (PDF). "D2.2: Set of use cases and scenarios". FIDIS Deliverables(FIDIS) 2 (2): 74–91.

[2] Provos, Niels; Talan Jason Sutton 2012 (1999). "A Future-Adaptable Password Scheme". Proceedings of 1999 USENIX Annual Technical Conference: 81–92.

[3] Eldon, Eric (2009-04-14). "Single sign-on service OpenID getting more usage". venturebeat.com. Retrieved 2009-04-25.

[4] Ian Oliver; "Programming Classics: Implementing the World's Best Algorithms" (01 April 1994) (Pearson Education – ISBN 13: 9780131004139)

[5] JunPeng Bao; "Comparing Different Text Similarity Methods" http://homepages.stca.herts.ac.uk/~comqcml//TR2-v6.pdf

AUTHORS

First Author – Apurva Patkeshwar, Information Technology, Sardar Patel Institute of Technology, Mumbai, India, E-mail: apurva.p@hotmail.com

Second Author – Kushan Kunal Prasad, Information Technology, Sardar Patel Institute of Technology, Mumbai, India E-mail: kushankunal@hotmail.com

Third Author – Suril Dhruv, Information Technology, Sardar Patel Institute of Technology, Mumbai, India, E-mail: surildhruv21@gmail.com

Fourth Author – Dr. Radha Shankarmani, Head of Department, Information Technology, Sardar Patel Institute of Technology Mumbai, India, E-mail: radha_shankarmani@spit.ac.in