Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 7, July 2014, pg.148 – 154

RESEARCH ARTICLE

PRIVACY PRESERVING IN MULTI USERS INFORMATION SHARING FOR SOCIAL COMMUNITY NETWORK MODEL

P.N.V. Mani Kumar[1], **K.V.N. Rajesh** (Assistant Professor)[2], **R. Pradeep**[3]

[1,2,3] Department of Information Technology, Vignan's Institute of Information Technology, India

[1] manikumarp@hotmail.com; [2] kvn.rajesh@gmail.com; [3] pradeep.deepu1990@gmail.com

Abstract - Online social networks [1] (OSNs) have experienced tremendous growth in recent years and become a de facto portal for hundreds of millions of Internet users. These OSNs offer attractive means for digital social interactions and information sharing, but also raise a number of security and privacy issues. While OSNs allow users to restrict access to shared data [2], they currently do not provide any mechanism to enforce privacy concerns over data associated with multiple users. To this end, we propose an approach to enable the protection of shared data associated with multiple users in OSNs. We formulate an access control model to capture the essence of multiparty authorization requirements, along with a multiparty policy specification scheme and a policy enforcement mechanism. Besides, we present a logical representation of our access control model that allows us to leverage the features of existing logic solvers [3][4] to perform various analysis tasks on our model. We also discuss a proof-of-concept prototype of our approach as part of an application in Facebook and provide usability study and system evaluation of our method.

Keywords- Social network [13], multiparty access control [5][6][7], security model[8], policy specification and management[9]

I. INTRODUCTION

Online social networks (OSNs) such as Facebook, Google+, and Twitter are inherently designed to enable people to share personal and public information and make social connections with friends, co-workers, colleagues, family and even with strangers. In recent years, we have seen unprecedented growth in the application of OSN's. For example, Facebook, one of representative social network sites, claims that it has more than 800 million active users and over 30 billion pieces of content (web links, news stories, blog posts, notes, photo albums, etc.) shared each month. To protect user data, access control has become a central feature of OSNs. For the protection of user data [10], current OSNs indirectly require users to be system and policy administrators for regulating their data, where users can restrict data sharing to a specific set of trusted users. OSNs often use user relationship and group membership to distinguish between trusted and untrusted users. For example, in Facebook, users can allow friends, friends of friends, groups or public to access their data, depending on their personal authorization and privacy requirements.

Advantages of the Proposed Approach

1. Flexible access control mechanism in a multi-user [11] environment like OSNs should allow multiple controllers, who are associated with the shared data.
2. Regulate access over shared data, representing authorization[12] requirements from multiple associated
3. A group of users could collude with one another so as to manipulate the final access control decision.
4. Cost Effective and Simple Implementation.

II. THE PROPOSED APPROACH

In this paper, we pursue a systematic solution to facilitate collaborative management of shared data in OSNs. We begin by examining how the lack of multiparty access control (MPAC) for data sharing in OSNs can undermine the protection of user data. Some typical data sharing patterns with respect to multiparty authorization in OSNs are also identified. Based on these sharing patterns, an MPAC model is formulated to capture the core features of multiparty authorization requirements that have not been accommodated so far by existing access control systems and models for OSNs, after careful analysis the system has been identified to have the following modules

Fig. 1 Overall Architecture of proposed system

Owner Module:
In Owner module let d is a data item in the space m of a user u in the social network. The user u is called the owner of d. The user u is called the contributor [12] of d. We specifically analyse three scenarios—profile sharing, relationship sharing and content sharing—to understand the risks posted by the lack of collaborative control in OSNs. In this the owner and the disseminator can specify access control policies to restrict the sharing of profile attributes. Thus, it enables the owner to discover potential malicious activities in collaborative control. The detection of collusion behaviours in collaborative systems has been addressed by the recent work.

Contributor Module:
In Contributor module let d be a data item published by a user u in someone else's space in the social network. The contributor publishes content to other's space and the content may also have multiple stakeholders (e.g., tagged users). The memory space for the user will be allotted according to user request for content sharing. A shared content is published by a contributor

Stakeholder Module:
In Stakeholder module let d is a data item in the space of a user in the social network. Let T be the set of tagged users associated with d. A user u is called a stakeholder of d, if u 2 T who has a relationship with another user called stakeholder, shares the relationship with an accessory. In this scenario, authorization requirements from both the owner and the stakeholder should be considered. Otherwise, the stakeholder's privacy concern may be violated. A shared content has multiple stakeholders.

Disseminator Module:
In Disseminator module let d be a data item shared by a user u from someone else's space to his/her space in the social network. The user u is called a disseminator of d. A content sharing pattern where the sharing starts with an originator (owner or contributor who uploads the content) publishing the content, and then a disseminator views and shares the content. All access control policies defined by associated users should be enforced to regulate access of the content in disseminator's space. For a more complicated case, the disseminated content may be further re-disseminated by disseminator's friends, where effective access control mechanisms should be applied in each procedure to regulate sharing behaviours. Especially, regardless of how many steps the content has been re-disseminated, the original access control policies should be always enforced to protect further dissemination of the content.

MPAC Module:
MPAC is used to prove if our proposed access control model is valid. To enable a collaborative authorization management of data sharing in OSNs, it is essential for multiparty access control policies to be in place to regulate access over shared data, representing authorization requirements from multiple associated users. Our policy specification scheme is built upon the proposed MPAC model. Accessory Specification: Accessory is a set of users who are granted to access the shared data. Accessory can be represented with a set of user names, asset of relationship names or a set of group names in OSNs.

III. METHODOLOGY

Flow chart:

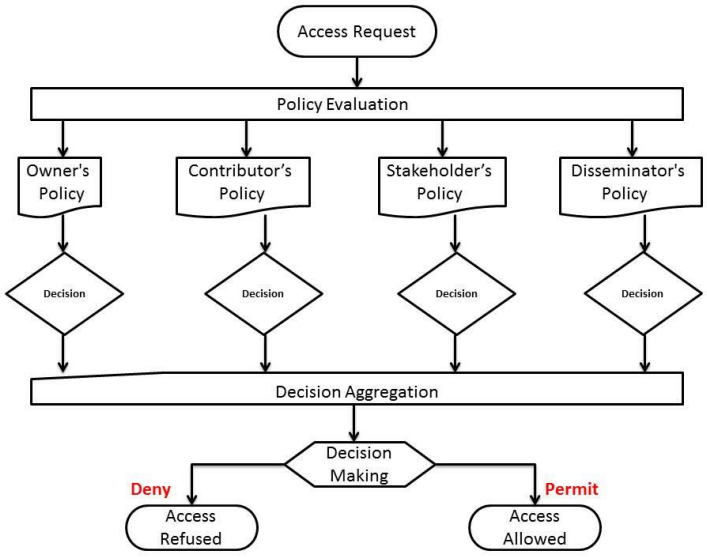

Fig. 2 Procedural Flowchart

Algorithm:

Step 1: Request for processing of request for new friend
Step 2: To check their policy evaluation
Step 3:
Owner's policy:
If it has direct user only sees his updates
Else if
Contributor's policy: undirected users can't see his updates to restrict them.
Else if
Stockholder's policy: share their information to unknown users to restrict using controller methodology
Else
Disseminator's policy: to restrict the users to access their walls from unwanted users
Step 4: To take the decision
Step 5: Permission only allow to direct users, to restrict from unwanted users
Step 6: To access only direct users using "MCONTROLLOR" methodology

Working Procedure:

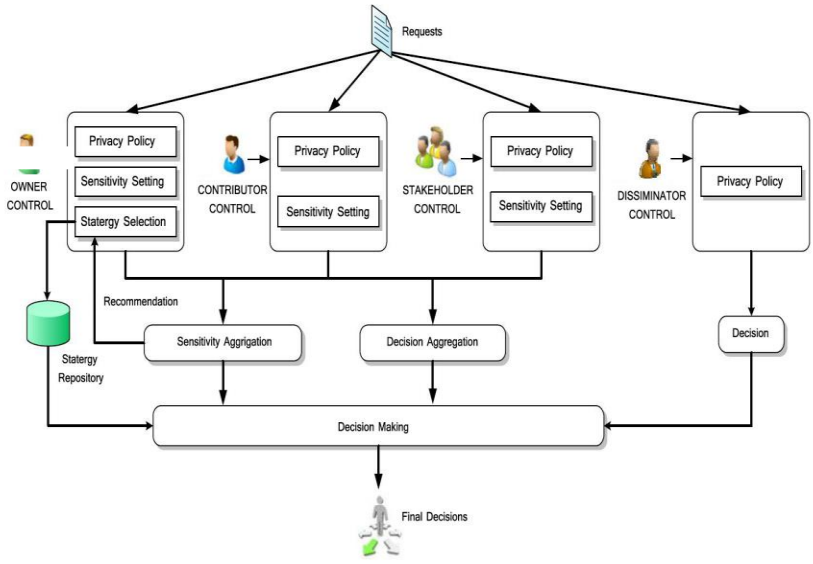

Fig. 3 Overall Architecture

Prototype Implementation:

We implemented a proof-of-concept Facebook application for the collaborative management of shared data, called MController (http://apps.facebook.com/MController). Our prototype application enables multiple associated users to specify their authorization policies and privacy preferences to control a shared data item. It is

worth noting that our current implementation was restricted [14] to handle photo sharing in OSNs. Obverse, our approach can be generalized to deal with other kinds of data sharing, such as videos and comments, in OSNs as long as the stakeholder of shared data is identified with effective methods like tagging or searching. The architecture of MController, which is divided into two major pieces: Facebook server and application server. The Facebook server provides an entry point via the Facebook application page, and provides references to photos, friendships, and feed data through API calls. A Facebook server accepts inputs from users, and then forwards them to the application server. The application server is responsible for the input processing and collaborative management of shared data. Information related to user data such as user identifiers, friend lists, user groups, and user contents are stored in the application database. Users can access the MController application through Facebook, which serves the application in an I Frame. When access requests are made to the decision-making portion in the application server, results are returned in the form of access to photos or proper information about access to photos. In addition, when privacy changes are made, the decision making portion returns change-impact information to the interface to alert the user. Moreover, analysis services in Controller application are provided by implementing an ASP translator, which communicates with an ASP reasoned. Users can leverage the analysis services to perform complicated authorization queries. Controller is developed as a third-party Facebook application, which is hosted in an Apache Tomcat application server supporting PHP and MySQL database. MController application is based on the I Frame external application approach. Using the JavaScript and PHP SDK, it accesses users' Facebook data through the graph API and Facebook query language. Once a user installs MController in her/his Facebook space and accepts the necessary permissions [15]. MController can access a user's basic information and contents. Especially, MController can retrieve and list all photos, which are owned or uploaded by the user, or where the user was tagged. Once information is imported, the user accesses MController through its application page on Facebook, where she/he can query access information, set privacy for photos that she/he is a controller, or view photos she/he is allowed to access. A core component of MController is the decision-making module, which processes access requests and returns responses (either permit or deny) for the requests depicts a system architecture of the decision-making module in MController. Decision as the response of the request. Multiparty privacy conflicts are resolved based on the configured conflict resolution mechanism when aggregating the decisions of controllers. If the owner of the content chooses automatic conflict resolution, the aggregated sensitivity value is utilized as a threshold for decision making. Otherwise, multiparty privacy conflicts are resolved by applying the strategy selected by the owner, and the aggregated SC is considered as are commendation for strategy selection. Regarding the access requests to disseminated content, the final decision is made by combining the disseminator's decision and original controllers' decision adopting corresponding combination strategy discussed previously. A snapshot of main interface of MController is shown in All photos are loaded into a gallery-style interface. To control photo sharing, the user clicks the "Owned, "Tagged," "Contributed," or "Disseminated" tabs, then selects any photo to define her/his privacy preference by clicking the lock below the gallery. If the user is not the owner of selected photo, she/he can only edit the privacy setting and sensitivity setting of the photo. If the user is the owner of the photo, she/he has the option of clicking "Show Advanced Controls" to assign weight values to different types of controllers configure the conflict resolution mechanism for the shared photo. By default, the conflict resolution is set to automatic. However, if the owner chooses to set a manual conflict resolution, she/he is informed of an SC of shared photo and receives a recommendation for choosing an appropriate conflict resolution strategy. Once a controller saves her/his\privacy setting, a corresponding feedback is provided to indicate the potential authorization impact of her/his choice. The controller can immediately determine how many users can see the photo and should be denied, and how many users cannot see the photo and should be allowed. MController can also display the details of all users who violate against the controller's privacy setting the purpose of such feedback information is to guide the controller to evaluate the impact of collaborative authorization. If the controller is not satisfied with the current privacy control, she/he may adjust her/his privacy setting, contact the owner of the photo to ask her/him to change the conflict resolution strategies, or even report a privacy violation to OSN administrators who can delete the photo. A controller can also perform authorization analysis by advanced queries. Both over sharing and under sharing can be examined by using such an analysis service in MController.

IV. SOFTWARE IN DETAIL

About the Java Technology

The Java technology lets developers, designers, and business partners develop and deliver a consistent user experience, with one environment for applications on mobile and embedded devices. Java meshes the power of a rich stack with the ability to deliver customized experiences across such devices. Java APIs are libraries of compiled code that you can use in your programs. They let you add ready-made and customizable functionality to save you programming time. Java programs are run (or interpreted) by another program called the Java Virtual Machine. Rather than running directly on the native operating system, the program is interpreted by the Java VM for the native operating system. This means that any computer system with the Java VM installed can run Java programs regardless of the computer system on which the applications were originally developed. In

the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains byte codes — the machine language of the Java Virtual Machine (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine. Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris TM Operating System (Solaris OS), Linux, or Mac OS. Java technology is both a programming language and a platform.

The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Object oriented
- Distributed
- Multithreaded
- Dynamic
- Architecture neutral
- Portable
- High performance
- Robust
- Secure

Each of the preceding buzzwords is explained in *The Java Language Environment* , a white paper written by James Gosling and Henry McGilton.

In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains *byte codes* — the machine language of the Java Virtual Machine[1] (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine.

Fig. 4 Java Machine

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris™ Operating System (Solaris OS), Linux, or Mac OS. Some virtual machines, such as the Java Hotspot virtual machine, perform additional steps at runtime to give your application a performance boost. This includes various tasks such as finding performance bottlenecks and recompiling (to native code) frequently used sections of code.

V. RESULTS

Fig. 5 Setting Privacy

Fig. 6 Private Images-Shown Separately

VI. CONCLUSIONS

We have proposed a novel solution for collaborative management of shared data in OSNs. A multiparty access control model was formulated, along with a multiparty policy specification scheme and corresponding policy evaluation mechanism. In addition, we have introduced an approach for representing and reasoning about our proposed model. A proof-of-concept implementation of our solution called MController has been discussed as well, followed by the usability study and system evaluation of our method.

ACKNOWLEDGMENT

I consider it as a privilege to thank all those people who helped me a lot for successful completion of this paper. First of all I would like to thank our beloved project guide Mr K.V.N. Rajesh who has given me a lot of support and freedom during my work. I would like to thank Principal of Vignan's Institute of Information Technology Dr. K. Alice Mary, for her encouragement to me during the work. I would like to thank our ever-inspiring Head of the Department of Information Technology, Mr B. Prasad, for his spontaneous response to every request though he was busy with his hectic schedule of administration and teaching. I would like to express my deep sense of thanks to all my Teaching Staff for enlightening me with constructive suggestions for solving my problems patiently and helping me to improve the quality of work.

REFERENCES

[1] Facebook Developers. http://developers.facebook.com/.
[2] Facebook Privacy Policy. http://www.facebook.com/policy.php/.
[3] Facebook Statistics. http://www.facebook.com/press/info.php? statistics.
[4] G. Ahn and H. Hu. Towards realizing a formal rbac model in real systems. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 215–224. ACM, 2007.
[5] G. Ahn, H. Hu, J. Lee, and Y. Meng. Representing and reasoning about web access control policies. In *Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual*, pages 137–146. IEEE, 2010.
[6] A. Besmer and H. Richter Lipford. Moving beyond untagging: Photo privacy in a tagged world. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 1563– 1572. ACM, 2010.
[7] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the 18th international conference on World wide web*, pages 551–560. ACM, 2009.
[8] B. Carminati and E. Ferrari. Collaborative access control in online social networks. In *Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 231–240. IEEE, 2011.
Transactions on Information and System Security (TISSEC), 13(1):1–38, 2009.
[9] E. Carrie. Access Control Requirements for Web 2.0 Security and Privacy. In *Proc. of Workshop on Web 2.0 Security & Privacy (W2SP)*. Citeseer, 2007.

[10] J. Choi, W. De Neve, K. Plataniotis, and Y. Ro. Collaborative face recognition for improved face annotation in personal photo collections shared on online social networks. *Multimedia, IEEE Transactions on*, 13(1):14–28, 2011.
Facebook Beacon, 2007.
[11] T. Zeller, "AOL Executive Quits After Posting of Search Data," The New York Times, no. 22, http://www.nytimes.com/2006/08/22/ technology/22iht-aol.2558731.html?pagewanted=all&_r=0, Aug. 2006.
[12] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava, "Anonymizing Social Networks," Technical Report 07-19, Univ. of Massachusetts Amherst, 2007.
[13] K. Liu and E. Terzi, "Towards Identity Anonymization on Graphs," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '08), pp. 93-106, 2008.
[14] J. He, W. Chu, and V. Liu, "Inferring Privacy Information from Social Networks," Proc. Intelligence and Security Informatics, 2006.
[15] E. Zheleva and L. Getoor, "Preserving the Privacy of Sensitive Relationships in Graph Data," Proc. First ACM SIGKDD Int'l Conf. Privacy, Security, and Trust in KDD, pp. 153-171, 2008. [10] R. Gross, A. Acquisti, and J.H. Heinz, "Information Revelation and Privacy in Online Social Networks," Proc. ACM Workshop Privacy in the Electronic Soc. (WPES '05), pp. 71-80, http:// dx.doi.org/10.1145/1102199.1102214, 2005.

