Security APIs for Online Applications

Jonathan Anderson
Computer Laboratory
University of Cambridge
jonathan.anderson@ieee.org

Joseph Bonneau
Computer Laboratory
University of Cambridge
joseph.bonneau@cl.cam.ac.uk

Frank Stajano
Computer Laboratory
University of Cambridge
frank.stajano@cl.cam.ac.uk

I. Introduction

Online social networks, in their current form, require users to place a vast amount of trust in the operators of both the core network and the third-party applications they use. Since both of these actors have shown themselves to be untrustworthy in the past [1], [2], [3], [4], [5], we have proposed a model for social networks in which client software runs on the user's computer, encrypted blocks are stored on a "dumb" server and third-party applications are sandboxed to avoid the leakage of personal information [6].

In this scheme, the interface between applications and the core client software resembles a system call API in which a kernel offers applications the means to perform privileged operations. We have begun exploring this API to determine its functional requirements and desired security properties, but we welcome comments from and engagement with the security API community in order to provide the users of social networks with meaningful promises of personal privacy.

II. Functional Requirements

An API for social applications must allow users to adapt the technology to their own needs [7]; we cannot predict what applications users will find compelling. We must, however, build this API so that it can support the functionality that we already know users will expect. This functionality includes finding other users, especially real-life friends, communicating and sharing personal information with them and adding value to other users' content by tagging and commenting on it.

A. Finding Friends

In order to establish connections, a member of a social network must advertise themselves to other users. By default, users of a site should be invisible to all other users. They should be able to advertise themselves in three ways:

1) By making some personal information (e.g. name, photo) available on the Web;
2) By sending social network information via existing communication channels (e.g. IM, e-mail);
3) Via social relationships in the network itself.

The first method requires that applications be able to retrieve or at least parse Web content, the second requires interoperating with arbitrary communications protocols and the third requires some sharing of personal information with other users.

B. Identity Verification

Another feature which privacy-preserving social network should provide – and which current social networks do not – is a means for users to verify the online identities of their real-world friends. Today's social networks are already used to impersonate important politicians for comedic purposes [8], but the potential to use social networks for highly targeted phishing attacks is tremendous. Users should therefore be provided with an easy-to-use yet cryptographically strong means of verifying identity using limited out-of-band signalling (in person, over the phone, etc.).

C. Messaging

Users must be able to send each other messages, either in real-time (*a la* IM or Twitter) or a delay-tolerant fashion (*a la* e-mail or Facebook messages). Applications should be able to send such messages and cause them to be signed in desired.

D. Storage

Some applications – e.g. photo sharing applications – will require long-term storage within a pseudo-filesystem. Posting personal content in existing models is as straightforward as putting the content on the server and not restricting other users' access to it. In a client-centric system, however, access control must be done by client software.

Our system should provide applications with a general-purpose filesystem, and applications could share content, where permitted, by passing capabilities via the kernel.

E. Joint Content

One of the most interesting cases in the debate over online privacy is the case of *joint content*: content that was created and posted by one person, but which involves another person's name, likeness, opinions or comments. For instance, if a user is "tagged" in a Facebook photograph, they have the right to remove the tag – which recognises the stake that they hold in the content – but tagging another user's photo also requires the permission of the person who posted the photo.

We must provide the ability for applications to generate and share such content, and the joint content must be carefully tied by the kernel to its original context.

III. Privacy Mechanisms

Our application API must provide privileged operations that applications can execute, but its primary purpose is to safeguard the privacy of users' personal information. The

API, then, must hide personal information from applications unless it is truly required and user-authorised. The API should provide applications with the means to operate successfully without private information and, should they truly need it, privileged but limited operations which they can perform.

A. Access Control

Access control in existing social APIs is extraordinarily simplistic and permissive. In the case of Facebook, applications have some access to the personal information of almost all users, even those those who do use the application in question [9]. Access to other users' personal information depends on whether or not users have declared themselves to be "friends", which we have argued is an incorrect approach for a system that is concerned with privacy [10]. We propose that a privacy-preserving social network should not share information because users are in a list of friends, but that access control should be inferred from the normal course of user action [11], and that if friend lists exist, they should be derived from sharing policy, not the other way around.

In some cases, privacy policy should be simple: many applications will need either personal information or external interaction, but not both. Other applications, however, will have more sophisticated requirements, and our system must be capable of servicing their requests.

Application behaviour will be governed by a kernel-maintained policy. That policy should be restrictive by default but make it easy for capabilities to be granted by the course of normal user actions [11], i.e. without "security prompts" that distract users from their purposes for the network [12].

B. Placeholders

In many instances, applications do not actually require social information, it is placing them in a social context that adds value. Application code could use placeholders and pseudonyms in user interaction which the kernel could replace with actual social information, as suggested by Felt and Evans' *privacy-by-proxy* concept [13].

For instance, a chess application does not need to know my name or my opponent's, but it could generate messages such as "${opponent_name} has offered to resign" or tell the UI to "place the opponent's profile picture in the rectangle [50,0,100,100]" without seeing bitmap data.

C. Privileges

In our implementation, applications will run as plugins to a Java-based framework with no permission to perform system operations such as open network sockets. All external or inter-application interaction, then, must be accomplished via privileged operations provided by our API.

The atomic operations which it provides should be performed in such a way that their combination does not introduce unforeseen vulnerabilities.

Java's security policy does not allow us to restrict access to the current system time; if it did, we might even shut down covert schannels of information flow among malicious applications [14]. However, changing the flow of private information from unrestricted to covert-channel-only would be a significant improvement over current practice.

IV. Related Work

While Felt and Evans' concept of placeholders for social information [13] could be very useful, the utility of anonymized social networks is suspect given the ease with which these networks can be de-anonymised [15].

May, Gunter and Lee have previously proposed *Privacy APIs* [16], by which they mean the formalisation and analysis of legal policies in which access control requirements are supplemented by notification and logging requirements.

V. Conclusion

By recognising social application APIs as security APIs, we could provide users with a much safer social networking experience, giving them control over and visibility of what applications do with their personal information. We welcome the security API community's feedback on and involvement with these efforts.

References

[1] J. Bonneau, J. Anderson, R. Anderson, and F. Stajano, "Eight Friends Are Enough: Social Graph Approximation via Public Listings," in *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems (SNS '09)*, 2009.

[2] J. C. Perez, "Facebook's Beacon More Intrusive Than Previously Thought." http://www.pcworld.com/printable/article/id,140182/printable.html, Nov 2007. PCWorld.

[3] B. Stone, "Facebook Aims to Extend Its Reach Across the Web," *The New York Times*, vol. 12, no. 1, 2008.

[4] T. S. Schmidt, "Inside the Backlash Against Facebook." http://www.time.com/time/nation/article/0,8599,1532225,00.html, 2006. Time Magazine.

[5] E. Mills, "Facebook suspends app that permitted peephole." http://news.cnet.com/8301-10784_3-9977762-7.html, 2008. CNET News.

[6] J. Anderson, C. Diaz, J. Bonneau, and F. Stajano, "Privacy Preserving Social Networking Using Untrusted Servers," in *Proceedings of the Second ACM SIGCOMM Workshop on Online Social Networks (WOSN '09)*, 2009.

[7] D. M. Boyd, *Taken Out of Context – American Teen Sociality in Networked Publics*. PhD thesis, University of California, Berkeley, 2008.

[8] B. McGonigle, "Some profiles on MySpace.com not what they seem." http://www.boston.com/news/nation/washington/articles/2006/10/16/some_profiles_on_myspacecom_not_what_they_seem/, 2006. The Boston Globe.

[9] J. Bonneau, J. Anderson, and G. Danezis, "Prying Data Out of a Social Network," in *Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining*, 2009.

[10] J. Anderson and F. Stajano, "Not That Kind of Friend: Misleading Divergences Between Online Social Networks and Real-World Social Protocols," in *Proceedings of the Seventeenth International Workshop on Security Protocols (SPW '09)*, 2009.

[11] K.-P. Yee, "Aligning security and usability," *IEEE Security and Privacy Magazine*, vol. 2, no. 5, pp. 48 – 55, 2004.

[12] A. Whitten, *Making Security Usable*. PhD thesis, Carnegie Mellon University, 2004.

[13] A. Felt and D. Evans, "Privacy Protection for Social Networking Platforms," in *Proceedings of Web 2.0 Security and Privacy 2008*, 2008.

[14] B. W. Lampson, "A Note on the Confinement Problem," *Communications of the ACM*, vol. 16, no. 10, pp. 613 – 615, 1973.

[15] L. Backstrom, C. Dwork, and J. Kleinberg, "Wherefore art thou r3579x?: anonymized social networks, hidden patterns and structural steganography," in *Proceedings of the 16th International Conference on the World Wide Web (WWW '07)*, pp. 181–190, 2007.

[16] M. J. May, C. A. Gunter, and I. Lee, "Privacy APIs: Access Control Techniques to Analyze and Verify Legal Privacy Policies," *Computer Security Foundations Workshop*, 2006.