PESAP: a Privacy Enhanced Social Application Platform

Tom Reynaert*, Willem De Groef†, Dominique Devriese†, Lieven Desmet† and Frank Piessens†
IBBT–DistriNet Research Group
KU Leuven
Leuven, Belgium
**tom.reynaert@student.kuleuven.be*, †*firstname.lastname@cs.kuleuven.be*

***Abstract*—Nowadays, social networking sites provide third party application developers with means to access their social graph, by providing a social application platform. Through their users, these developers acquire a significant set of personal information from the social graph. The current protection mechanisms, such as privacy policies and access control mechanisms fall short on protecting the privacy of the users. In this paper we present a framework for a privacy enhanced social application platform, called PESAP, that technically enforces the protection of the personal information of a user, when interacting with social applications. The framework is based on two pillars: anonymization of the social graph and secure information flow inside the browser. PESAP is targeted to be as compatible as possible with the current state-of-the-art design of social application platforms, while technically enforcing the protection of user privacy. We evaluate this compliance, based on a classification of applications in different categories.**

***Keywords*-privacy; online social network; social application platform;**

I. Introduction

Today social networking sites are ubiquitous. They host an important part of the on-line communication and contain the majority of personal information that is available on the web. Ever since Facebook launched their application development platform in May 2007, social applications have been an important evolution in the world of social networking sites. Almost every major social networking site provides means to access data in their social graph. Third-party applications spread through the on-line communities and the popularity of these social applications keeps increasing. Although they might be hard to configure and adjust to one's wishes, users usually trust the social networking sites to respect their privacy settings. Trusting Facebook, Google, and other big social network providers to keep to their policies and to respect your privacy, is hard to avoid when using social networking sites. Trusting each third-party application developer to keep to the policies and to respect your privacy is more difficult to justify. A 2010 Wall Street Journal article illustrates the problem by showing that many of the most popular Facebook applications were (perhaps unknowingly) transmitting identifying information to advertising and tracking companies [1].

The first hurdle between application providers and the content of the social graph is an access control mechanism. All social application platforms, require an explicit or sometimes implicit authorization of the user before granting an application access to her data. The extent and protection range of this authorization procedure strongly depends on the social application platform. Facebook, for example, provides a fine-grained permission model in which a user can clearly control which parts of her information are accessed by which applications. In addition, accessing personal information is rigidly regulated with short-lived access tokens. In contrast, the OpenSocial standards do not provide any granularity in their permission model, and access tokens can be kept alive for a long time. These access control mechanisms provide a first shielding of the personal information of a user from application developers. However, once given consent, this shield is broken and application developers can harvest and possibly misuse the user's personal information.

The rules by which an application developer must abide, while using personal information are often stated in so called developer policies. All major social networking sites prohibit application developers from misusing personal information or forwarding it to other parties, such as advertising companies. However, it is difficult to verify the compliance of application developers with these rules, which is reflected in the weak guarantees given by social networking sites towards the user:

> Remember that these games, applications and websites are created and maintained by other businesses and developers who are not part of Facebook, so you should always make sure to read their terms of service and privacy policies.
> — *Facebook Developer Policy*

With these two protection mechanisms in place, the social networking site shifts the responsibility of protecting personal information to the user: she is responsible for (i) assessing the trustworthiness of the application provider before granting permissions and (ii) detecting privacy intrusions and holding the application provider accountable. The privacy features that the social networking site provides are only technical and legal supporting measures that allow the user to perform these assessments herself. It is already a difficult and complex task for social networking sites to verify trustworthiness and well-behavior of application

providers in this framework, for individual users this is a mission impossible.

This paper presents a social application platform that technically enforces the protection of personal information. To make it as compliant as possible with the state-of-the-art of social application platforms and applications, Section II studies the general architecture of current social application platforms. In Section III we present our proposal for a privacy enhanced social application platform, which is evaluated in Section IV. Finally, Section V shortly examines relevant related work and suggests some future research.

II. State-of-the-art Social Application Platform

Originally social applications were only built using the so called *gadget approach*. A gadget is a document specifying the behavior of the application. It strongly resembles a normal web page but is combined with a small set of metadata. A gadget can only be accessed inside a wrapping social networking site page. Upon request of the gadget page, the social networking site fetches the specification of the gadget from the developer and embeds it into its own page. Because in this strategy, the social application provider operates as a proxy between the application and the user, gadgets can use special HTML tags, e.g. FBML, which can be translated by the social networking site. As the application code is directly embedded inside a social networking site page, the use of JavaScript is typically limited to a safe subset, such as CAJA or FBJS. In this setting access to the social graph is acquired through special HTML tags or a designated JavaScript API.

Recently the architecture of social application platforms has shifted towards a more *distributed approach* in which the social networking site no longer operates as a regulating proxy between the application and the user but only as the partner who supplies the personal information. In the distributed approach, applications can run under their own domains and use regular HTML and JavaScript. This allows for more dynamic applications to be developed. The applications can query personal information by issuing HTTP requests to a designated REST API hosted by the social application platform. In this new distributed design, the social application platform loses a lot of the regulating power it had as a proxy between the gadget and the user. In return, application developers are not limited anymore by the use of a safe subset of JavaScript and by running inside the social networking site's pages. They can create far more complex social applications and even socialize content on existing websites. An application can still be accessed inside the social networking site's pages, where it is loaded inside an iframe. Although OpenSocial still supports the gadget approach together with the distributed approach, Facebook deprecated their implementation of the gadget approach in favor of the distributed approach, mainly because the shift eliminated the technical differences between developing social applications and regular web applications. [1]

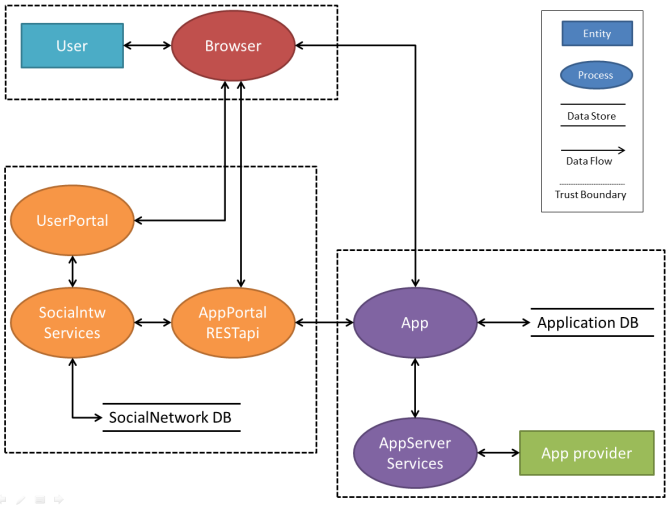

Figure 1: Data flows in the distributed architecture of social application platforms

The main responsibility of the social application platform in the distributed approach is twofold. It has to provide an authentication and authorization procedure and a REST API. When accessing a social application, the user's browser is redirected through a procedure, mostly OAuth2.0, in which she has to log in to the social application platform and grant permissions to the application. In most platforms, the permissions for each application are only explicitly requested once. Subsequent times the same set of permissions is granted implicitly. At the end of the procedure the browser is redirected to the application with an access token appended to the redirect URL. With this access token, the application developer can authenticate HTTP requests to the REST API on behalf of the user. Responses only contain the information that is in scope of the set of permissions implied by the access token. Fig. 1 illustrates the data flows in the common distributed architecture of social application platforms.

Fig. 1 reveals an important flaw in the distributed design: social application providers can store personal information on their own server, beyond the control boundaries of both the user and the social application platform. As a consequence, neither the social application platform nor the user can technically protect this personal information. The sole remaining mechanism to protect the user's privacy is the complex task of verifying whether an application provider complies with the developer policy and does not leak, sell or misuse this personal information.

[1] *Deprecation of Facebook Markup Language (FBML)*, Facebook developers references, September 2011,http://developers.facebook.com/docs/reference/fbml/

III. Design and Implementation of PESAP

From analyzing the state-of-the-art in social applications platforms, we conclude that there is an important shift towards the distributed design. This new design features the storing of personal data on the application-side server, which is undesirable for the user as it implies a total loss of control over this data. However, users should be able to view personal information of their friends in their browser, as this is the key ingredient that makes social applications popular. These three conclusions form the main goal of our research: designing a privacy enhanced social application platform, PESAP, which stays as close as possible to the distributed architecture in order to minimize impact on social applications and social application platforms, while technically preventing personal information to leak from the browser to application providers or other third parties.

We define the *social view* of a user as the set of entities and the information about these entities that a user can view by browsing the social networking site.

A. Anonymization of the Social Graph

Almost any social application needs to store user depending state, such as high scores or progress, in between different uses of the application. In addition, a lot of them make use of the relations between users and other entities of the social graph. However, only a small minority of the social applications seems to depend heavily on the actual content of personal information. Section IV provides a more detailed analysis of different types of applications. This inspires us to provide the application developers with at most the stripped-down, anonymized framework of the social graph, illustrated in Fig. 2. The anonymization is done by encrypting the ids of the entities of the social graph with a per-application symmetric key. PESAP provides the applications with a REST API to query – via HTTP requests – information about the structure of the social graph. These requests need to be authenticated with an access token which can be retrieved by following an authentication and authorization procedure similar to the one described in the previous section. The content of the responses is limited to the social view of the user that provided the access token.

The strength and popularity of social applications strongly depend on the interactions with friends. In order to provide these interactions, users have to be able to see the personal information of their friends in the context of social applications. Therefore PESAP provides a re-identification endpoint. Querying this endpoint can only be done by sending XMLHTTPRequests from a browser where a PESAP user is authenticated. Besides generating an access token, PESAP also stores a per user secure token as a cookie in the browser, during the authentication and authorization procedure. Based on the presence of this cookie in a re-identification request, PESAP identifies the originating browser. It then responds with the requested personal information as long as it is

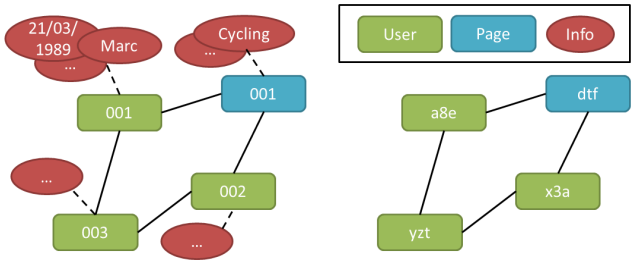

Figure 2: Anonymization of the social graph: left the full sized social graph, right the anonymized framework provided to the application

part of the social view of the user requesting it. The response containing the personal information is sent with the necessary CORS headers, to circumvent the Same-Origin-Policy and allow cross resource sharing. As we do not want re-identified information to leak away, we need a mechanism to keep the information in the browser.

B. Secure Information Flow in the Browser

The second pillar of PESAP is the protection of private information in the browser of the users: PESAP should not block all outgoing communication from the browser, as this strategy would severely harm the functionality of the application. We want to keep track of the personal information inside the application's code in the browser, and prevent outgoing communication to be influenced directly or indirectly by this personal information. Secure information flow techniques offer this functionality.

The objective of secure information flow techniques, also called information flow analysis techniques, is to analyze information flows inside a computer program or process and ensure that those flows comply with a certain policy. This policy usually labels inputs and outputs with a certain security label. These labels come from a partially ordered lattice. Secure information flow analysis techniques enforce that information only flows upwards in this lattice. The property that lower labeled outputs are not influenced by higher leveled inputs is called non-interference. In the most simple scenario there are only two different labels in the lattice: L for low, or public information and H for high, or secret information. In this case, the goal of secure information flow is to ensure that the L labeled outputs do not depend on the H labeled inputs.

The literature contains a vast amount of different secure information flow techniques. Static techniques analyze source code or byte code upfront, in order to verify the compliance with the policy. Dynamic techniques usually monitor and regulate programs during execution. An introduction to the former set of techniques is written by Smith [2] and a survey by Sabelfeld et al. [3], the latter are summarized by Le Guernic [4]. For PESAP we choose a dynamic secure

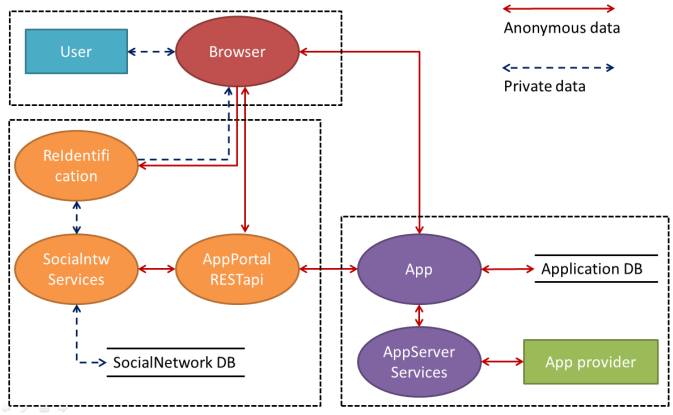

Figure 3: Data flows in PESAP, only the anonymized ids are allowed to be stored on the application server

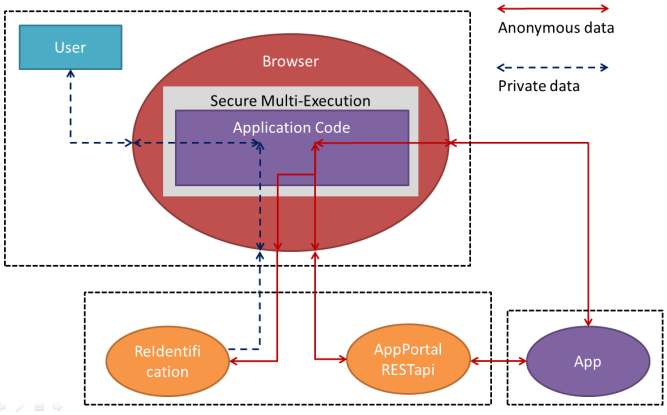

Figure 4: A PESAP aware browser, using secure multi-execution to prevent personal information to leak away

information flow technique called secure multi-execution, a technique developed by Devriese and Piessens [5] and further refined to a reactive system by Bielova et al. [6]. The idea is to label inputs and outputs of a system with security labels and to run a separate sub-execution of the program for each security label. A sub-execution at a given security level has only access to the information inputs that are labeled with the same or a lower security label. When an input is related to a side-effect, the input is only generated in the sub-execution with the same security level as the label of the input. All other, higher sub-executions wait for the input to be generated and re-use the input from the lowest sub-execution. All inputs that have a higher security label than the security level of a sub-execution are replaced with default values. At the same time, a sub-execution at a given security level, can only do outputs that have exactly the same security level. It is intuitively clear that this technique provides non-interference as outputs are only done in their own security level sub-execution and that execution does not have access to inputs that have a higher security label.

By labeling the re-identified personal information of users with a *H* security label, and all communication towards third parties, including the application provider, as *L* output, running secure multi-execution in the browser should be able to prevent the information from leaking out of the browser. The final design of PESAP is summarized in Fig. 3 and Fig. 4. Defining an adequate policy that is both secure and precise, is a challenging task.

C. Implementation of a Prototype

To test the feasibility of our strategy, we have implemented a prototype based on the design of PESAP. The prototype consists of a social networking site accompanied by a social application platform, which offers an authentication and authorization procedure, a REST endpoint to query the anonymized framework of the social graph and a re-identification endpoint.

We have also defined a first information flow policy for FLOWFOX, an implementation of secure multi-execution for JavaScript in the browser [7]. In FLOWFOX a policy is defined on the calls of the JavaScript API. Labeling each JavaScript call in such a way that they reflect a general information flow policy that is both secure and precise, is a complex task. Although we tailored a customized flow policy for PESAP, in practice FLOWFOX will use a general policy which can be adapted and extended by content providers. We can imagine the use of an additional HTTP response header that indicates that the content may be shared with other domains, but cannot leave the browser. An instance of FLOWFOX together with the adequate flow policy is considered a PESAP-aware browser. Running an application in a regular browser can lead to the leaking of the personal information contained in the social view of the user.

Finally, we have created two representative social applications. In the first one, users can throw dice and compare their high scores with those of their friends. In the second application, users can play games of rock-paper-scissors with each other.

IV. ANALYSIS

To prove that PESAP achieves its privacy and compatibility goals, we investigate the impact of de-anonymization attacks on our design and study the compliance of different types of applications with PESAP.

A. De-Anonymization Attacks

The choice of providing the application developer with the anonymized framework lowers in a way the overall privacy guarantees of the system. Indeed, the combination of the anonymized graph with some extra knowledge of the real graph, is the key ingredient for a broad set of graph de-anonymization techniques [8]–[11]. These de-anonymization techniques aim to map the entities of the anonymized graph back to entities of the real graph. Although these techniques

are a serious threat for all sorts of applications that make use of an anonymized version of a graph, we will argue that these techniques do not cause a major harm in the case of PESAP.

First of all, a lot of personal information such as the profile data that is linked to the entities, indicated in red in Fig. 2, is never released to the application provider. De-anonymization of the framework will never result in compromising this kind of personal information – so called content disclosure [10]. The only relevant information that can be mined from a de-anonymized framework of the social graph are the identities associated with nodes (identity disclosure) and the relations between entities (link disclosure).

The research in anonymization and de-anonymization techniques is mainly focused on scenarios where an entity performs a set of actions while being anonymized. These actions are thus linked to the anonymized identifier and by de-anonymizing the graph, they can be linked to the real entity. This threat is not very relevant for social application platforms as a user usually does not perform compromising actions while using a social application.

The only relevant threat, related to the de-anonymization of the framework of the social graph is the fact that an adversary might discover relationships or links between entities, that are not supposed to be visible [12]. De-anonymization techniques typically require some additional information about the real elements of the graph in order to deliver satisfying results. This information is used to decrease the anonymity set of the target. Typical examples of such information are the degree of a node, or the relation of the node to a known clique. One possible mitigation could be to extend the framework of the anonymized social graph with fake ids which have fake relationships to real entities or include a systematic framework for identity anonymization for graphs [10]. This could make it a lot harder to identify entities based on the numbers of relationships they have.

B. Influence on Applications

Analyzing how many real-life applications would remain functional in PESAP can give an important indication on how close PESAP stays to the current distributed design of social application platforms. The most precise evidence would be the fact that real-life applications are available in PESAP. Alas, PESAP has not been published on-line yet, nor did real-life application providers built applications for it. The only applications that are available for PESAP are Roll The Dice and Rock-Paper-Scissors, which are described in the previous section. The fact that these two applications function as expected is already an indication that a lot of similar real-life applications also do.

PESAP contains a lot of similarities to the privacy-by-proxy approach suggested by Felt et al. [13]. Both social application platforms use a protection strategy based on the encryption of entity ids and re-identification. The major difference is that Felt et al. built a solution for the gadget design. The fact that PESAP is built for the distributed design adds some complexity. Despite this difference, the view of an application on the social graph is quite similar in both cases. It only has access to an anonymized framework of the graph and never to the real data fields containing personal information. Although, their study focused on the most popular applications in 2007, we expect that the spectrum and field of social applications have not changed dramatically the last five years.

Felt et al. describe their analysis strategy in the following way:

> To conduct the survey we installed each application on a user account with the minimum amount of information filled out. If an application requested more data, broke, or required the interaction of multiple users, we installed it on a fully filled-out second account to observe the difference. We explored the features of each application to look for the appearance of data or use of the social graph. Although it is not possible to determine exactly how an application uses information without access to its source code, the simplicity and limited interactivity of Facebook applications gives us reasonably high confidence that this method captures application functionality well enough to understand their data use.

From their survey they conclude that 9 out of the 150 most popular Facebook applications would not function in their framework. Amongst others they include applications that require the actual date of birth of its users to generate a horoscope on their server. Although outdated, the survey done by Felt et al. indicates that the consequences of the privacy protection mechanism implemented in PESAP will probably not harm the majority of applications, provided they slightly change the user representation part of their implementation.

Because the analysis of Felt et al. is very time consuming and does not provide hard scientific evidence, a more general strategy is followed to analyze the compatibility of PESAP with the current real-life applications. First of all six different types of applications are identified in Table I.

Only applications of type 5 and 6 are not deemed compatible with PESAP. Applications of type 5 generate non-trivial content based on the data of certain fields of personal information. As explained before, horoscopes need to be based on the actual date of birth of a user. Each day, the application will query a horoscope database server with the specific date of birth. This implies that personal information has to leave the browser and those applications will no longer function within PESAP as this is in fact what we want to avoid. Applications of type 6 are complex applications that do not run in the browser, but as a separate stand alone application. Because these applications do not run

Table I: Types of Applications

App Type	Description	Examples
Type 1	Individual applications with limited shared data	Roll The Dice, Angry Birds, Geo Challenge
Type 2	Real-time interactive applications with shared dynamic data	Rock Paper Scissors, Poker, Chess
Type 3	Content providers that offer social sharing of the content	The Guardian, Yahoo Reader
Type 4	Individual applications with a big amount of non real-time shared data	Farm Ville, Mafia Wars
Type 5	Applications that do extensive processing of the content of private data	Daily Horoscope, Map Visualizations
Type 6	Non-browser applications that share personal preferences	Spotify, Microsoft Live

inside a browser, they have no possibility to issue valid re-identification requests and thus they can only use the anonymized version of the social graph. This will break the functionality of these applications. Again this is due to the general objective of PESAP.

To get an idea of how many popular applications belong to each category, the top 15 most popular Facebook applications[2] is given in Table II together with their application type. 3 out of 15 correspond to type 5 or 6 and would no longer function in PESAP.

Table II: Application type of most popular applications

Rank	Application Name	Nb of Monthly Users	Type
1	Static html Iframe Tabs	72.800.000	1
2	CityVille	44.600.000	4
3	Static iframe tab by woobox	37.400.000	1
4	Texas Holdem Poker	35.500.000	2
5	MyCalendar	30.600.000	5
6	Draw Something by OMGPOP	30.500.000	1
7	Hidden Chronicles	29.200.000	4
8	Farmville	28.100.000	4
9	Castleville	26.600.000	4
10	Bing	25.500.000	1
11	Yahoo reader	25.500.000	3
12	Microsoft Live	23.000.000	6
13	Yahoo!	21.500.000	6
14	Words With Friends	20.300.000	2
15	Angry Birds	19.900.000	1

V. Related Work and Future Research

As mentioned in the previous section, our work shows a lot of similarities to an older attempt to create a more private social application platform by Felt et al. [13]. However, their design was built for the gadget approach, where the social application platform was a proxy hosting the application inside its own pages. This role allowed the social application platform to analyze the application code and control the leaking of re-identified information. In practice, they put constraints on the usage of the already limited subset of JavaScript, FBJS, in the gadget approach, especially in combination with the re-identification HTML tags. Although Felt et al. performed pioneering work in both analyzing privacy issues for social applications and proposing a privacy friendly alternative, the social application platforms have evolved a lot since the publication. The solution of Felt et al. does not work for the distributed approach.

[2]The top 15 as found on http://www.appdata.com/ on March 31, 2012

Another interesting suggestion for a privacy friendly social application platform is an extensive framework, called xBook [14], in which the social application platform can again play its former role as regulating and monitoring proxy between the application and its users. The idea is to provide both a server-side container in which applications can be deployed and a client-side environment to display the application to the user. Prior to installing the application, the user is presented with a manifest stating in which way the application will process and handle her personal information. By regulating and monitoring all information streams, xBook then enforces that the personal information is processed in compliance with the manifest.

A third promising solution for the privacy problems in social application platforms is called PoX [15]. This solution is surprisingly simple and maps excellent to the current distributed design. PoX consists of installing a client-side-proxy that makes sure that the access token never leaves the browser. Whenever the application needs to access personal information, the request passes through the client-side-proxy. This proxy allows or disallows this request based on an access-control-list set by the user. Hence, each request for personal information is made explicit to the user, which can define a very fine-grained privacy policy. However, for xBook and PoX assessing the trustworthiness of an application is still the responsibility of the user, and one misjudgment can lead to the disclosure of an important part of the personal information.

All work in the research field of secure information flow [2], [4] is strongly related to the design of PESAP. In a sense, PESAP provides a practical application of secure information flow. To our knowledge, it is the first real-world application that makes use of secure multi-execution, a dynamic secure information flow technique [5]. Although Bielova et al. claim that secure multi-execution in the browser can in time replace the same origin policy [6], future research is necessary to allow a simple way of defining a flow policy for FLOWFOX and to provide means for different parties to add rules to this policy for the data they provide. This could for example be done by adding a CORS-like header to responses injecting data in third party code.

VI. Conclusions

This paper first explains the privacy issues involved by granting third party applications access to the social graph

of a social networking site. Next, the distributed design of current social application platforms is discussed. The main goal of the paper is to propose a privacy enhanced social application platform, called PESAP, that protects the privacy of its users when interacting with social applications and maps as close as possible to the current distributed architecture. To the best of our knowledge this is the first successful attempt to design and implement a prototype of such a platform. In the literature, we can find previous proposals for privacy friendly social application platforms, but either they are outdated and are relatively far away from the distributed architecture (Privacy-by-proxy, xBook), or they do not prevent the leaking of personal information of misled users (xBook, PoX). PESAP is supported by two pillars: anonymization of the social graph and secure information flow in a designated browser. Based on the analysis of our prototype we conclude that relatively hard privacy guarantees can be provided in PESAP, without compromising the logical functionality of the majority of the current social applications. Application developers will need to change the way they present personal information to be compliant with PESAP, but this will not harm the functionality of these applications.

ACKNOWLEDGMENT

This research is partially funded by the Research Fund KU Leuven, the EU-funded FP7 projects NESSoS and WebSand and by the IWT-SBO project SPION. Dominique Devriese holds a Ph.D. fellowship of the Research Foundation - Flanders (FWO).

With the financial support from the Prevention of and Fight against Crime Programme of the European Union European Commission – Directorate-General Home Affairs. This publication reflects the views only of the author, and the European Commission cannot be held responsible for any use which may be made of the information contained therein.

REFERENCES

[1] E. Steel and G. A. Fowler, "Facebook in privacy breach," *The Wall Street Journal*, 2010, uRL: http://online.wsj.com/article/SB10001424052702304772804575558484O-75236968.html, last verified on 2012-03-05.

[2] G. Smith, "Principles of Secure Information Flow Analysis," *Malware Detection*, vol. 27, pp. 291–307, 2007.

[3] A. Sabelfeld and A. C. Myers, "Language-Based Information-Flow Security," *IEEE Journal on Selected Areas of Communications (JSAC)*, vol. 21, no. 1, pp. 5–19, January 2003.

[4] G. Le Guernic, "Confidentiality Enforcement Using Dynamic Information Flow Analyses," Ph.D. dissertation, Kansas State University, 2007.

[5] D. Devriese and F. Piessens, "Noninterference through secure multi-execution," in *Proceedings of the IEEE Symposium on Security and Privacy*, May 2010, pp. 109–124.

[6] N. Bielova, D. Devriese, F. Massacci, and F. Piessens, "Reactive non-interference for a browser model," in *Proceedings of the International Conference on Network and System Security (NSS)*, September 2011, pp. 97–104.

[7] W. De Groef, D. Devriese, N. Nikiforakis, and F. Piessens, "FlowFox: a Web Browser with Flexible and Precise Information Flow Control," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2012.

[8] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava, "Anonymizing Social Networks," Computer Science Department, University of Massachusetts Amherst, Tech. Rep. 07-19, March 2007.

[9] A. Narayanan and V. Shmatikov, "De-anonymizing Social Networks," in *Proceedings of the IEEE Symposium on Security and Privacy (IEEESP)*, 2009.

[10] K. Liu and E. Terzi, "Towards Identity Anonymization on Graphs," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2008, pp. 93–106.

[11] L. Backstrom, C. Dwork, and J. Kleinberg, "Wherefore Art Thou R3579X? Anonymized Social Networks, Hidden Patterns, and Structural Steganography," in *Proceedings of the International World Wide Web Conference (WWW)*, 2007, pp. 181–190.

[12] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu, "Link privacy in Social Networks," in *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, 2008, pp. 289–298.

[13] A. Felt and D. Evans, "Privacy protection for social networking platforms," in *Proceedings of the Workshop on Web 2.0 Security and Privacy (W2SP)*, May 2008.

[14] K. Singh, S. Bhola, and W. Lee, "xBook: Redesigning privacy control in social networking platforms," in *Proceedings of the USENIX Security Symposium*, 2009, pp. 249–266.

[15] M. Egele, A. Moser, C. Kruegel, and E. Kirda, "PoX: Protecting users from malicious Facebook applications," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM)*, 2011, pp. 288–294.