PROTOSS: A run time tool for detecting Privacy violations in Online Social networks

Özgür Kafali, Akın Günay, and Pınar Yolum
Department of Computer Engineering
Bogazici University
TR-34342, Bebek, İstanbul, Turkey
Email: ozgurkafali@gmail.com, {akin.gunay, pinar.yolum}@boun.edu.tr

Abstract

As online social networks are becoming part of both social and work life, preserving privacy of their users is becoming tremendously difficult. While these social networks are promising privacy through privacy agreements, everyday new privacy leakages are emerging. Ideally, online social networks should be able to manage and maintain their agreements through well-founded methods. However, the dynamic nature of the online social networks is making it difficult to keep private information contained.

We have developed PROTOSS, a run time tool for detecting privacy leakages in online social networks. PROTOSS captures relations among users, their privacy agreements with an online social network operator, and domain-based inference rules. It then uses model checking to detect if an online social network will leak private information.

1. Introduction

Preserving privacy has long been an important mission of Web systems. The general process of preserving privacy is through privacy agreements. Web systems announce their policies through privacy agreements. Users are expected to use the system only if they are comfortable with the agreement. In settings, where a single system is responsible for managing information, carrying out privacy dealings through an agreement is reasonable. In online social networks, though, multiple parties manage information. That is, the system that provides the social network service (such as Facebook) allows users to see each others’ content, make comments, and even share the content with others. In such systems, it is difficult to maintain the privacy of users. Even if the system itself does not share the user information with other systems, other users on the social network can propagate a private content to others, for whom the content was not initially meant for. Or, other applications that benefit from the online social network can use private information for marketing or presentability purposes.

A well-known example is about an individual’s location being identified from geographic information attached to the pictures she uploads. While the online social network is not explicitly revealing the location information itself, through inferences the location is being deciphered without the users’ knowledge and consent [1]. Overall, the leaked information is not a simple privacy agreement violation. That is, the OSN operator does not immediately violate a privacy agreement. However, the relations among users and the inferences that are possible among the content make it possible for third parties to become aware of information that is actually private. How can a user be sure that the content it publishes does not reach individuals that she doesn’t want? The above discussion clearly shows that it is not enough to check the privacy agreement of the system. In addition to this, the user’s relations with other users and what these relations enable the other users to do should be systematically analyzed to reach a conclusion.

Accordingly, we propose PROTOSS, a run time tool to detect privacy breaches in online social networks. The main technique underlying our approach is model checking, which given a model of the system checks whether certain properties hold. Our system model represents the privacy agreements of a system with users as well as the relations of users formally. Using this model, we can check interesting properties such as whether a certain user’s content will ever reach a certain individual even when that individual is not an acquaintance, or whether the relations among individuals can lead to information to be revealed to individuals for which the content was not intended.
2. Technical Background

There are two important techniques underlying our approach. The first one is the abstraction of commitments among individuals when carrying out agreements. The second one is checking models of systems to verify if certain properties hold in the system.

**Commitments:** A commitment is an agreement from a debtor to a creditor to bring about a certain property, if a specified condition is achieved [2], [3] and formally represented as $C(\text{debtor}, \text{creditor}, \text{condition, proposition})$. For instance, a social network operator may commit to one of its users Charlie to share his location with Linus, if Charlie and Linus are friends ($C(\text{operator, charlie, friend(charlie, X)}, \text{shareLocation(charlie, X)})$). This commitment becomes active when Charlie and Linus actually become friends. If the proposition holds, that is if Charlie’s location is shared, then the commitment is fulfilled, otherwise it is violated.

**Model Checking:** Model checking is a computational method to automatically verify whether a given property holds for a system [4]. The system under consideration is modeled as a state transition graph in some formal language and the property that is aimed to be verified is represented as a logic formula in a suitable language. Given the system model and the logic formula of the investigated property, a model checking algorithm checks whether the system model satisfies the desired property. For instance, a model can represent that two users are friends in a social network and evolution of this model can be modeled according to the operations provided on this relation by the modeled social network. Similarly, a property of the social network about privacy could be that the location of a user is not revealed to users that are not friends. In our work we use NuSMV, a state of the art model checker based on binary decision diagrams [5], which uses computation tree logic (CTL) [6] to represent properties that are aimed to be verified.

3. Privacy-Aware OSN Architecture

We are interested in online social networks (OSNs) that are administered by an OSN operator. Each user can post content as it sees fit. One can post personal information such as her location, the people she is with, and so on as well as links to news, jokes and so forth. Our primary target in this work is the first set of information since we are interested to see how private information can float in the system.

Since it is a social network, users are related to each other. As in newer social networks, users can be related to each other through different relations. For example, Charlie could be a friend of Linus but a colleague of Sally. These relations identify how much and of what type of content would be shared with other users. For example, Charlie would share his whereabouts with his friends, but may not want to share this with colleagues. Essentially, the OSN operator is responsible for ensuring that these expectations are met, by making sure that only the users with the right privileges are shown private content.

Figure 1 depicts our privacy-aware OSN architecture, where among each user and the OSN operator, there exists a privacy agreement. This is an agreement that contains clauses about which relations are entitled to which privileges. For example, an agreement between Charlie and the OSN operator can state that all friends of Charlie are entitled to see his location. This is not a static agreement. That is, as Charlie creates more relations with other users, this privacy agreement is updated accordingly. Since the OSN operator is responsible for realizing the clauses in the privacy agreement, it needs a mechanism to check whether it can honor the agreement. We call this the privacy checker (**PROTOSS**).

**PROTOSS** uses the network information as well as the agreement information to decide whether the agreements can be honored in the system. For example, if Charlie does not want any of his colleagues to see his location and if he has identified Linus both as a colleague and a friend, then OSN will end up disclosing the location to a colleague. In this case, OSN should let Charlie know that a colleague will hear of his location and maybe let him decide what to do.
4. Running Example

Consider the following social network with specific components as outlined above:

i. Users: We have three users of the OSN; charlie, sally, and linus.

ii. Relations: The users of the system can initiate relations among themselves. This is typical of online social networks [7]. We assume that the following relations exist:

- colleague(X, Y): Users X and Y are colleagues.
- friend(X, Y): Users X and Y are friends.

We instantiate them as follows: friend(sally, linus) and colleague(charlie, linus).

We assume that if a relation is not instantiated, then it doesn’t hold. Hence, for example, one can conclude that sally and charlie are not friends.

iii. Content: To complement the relations of the system, we have two types of content. These are:

- location(X, W): User X is at location W.
- with(X, Y): User X is with user Y.

iv. OSN operator: There is a single OSN operator in the system. It is responsible for displaying appropriate information to the users based on its own policies; i.e., decides what will be visible to each user. For example, visible predicate below describes whom to grant access to:

- \( R_1: \text{visible(with(X, Y), Z) \leftarrow friend(X, Z) \lor friend(Y, Z)} \): This rules states that any friend of the user X or Y knows who X or Y is with.
- \( R_2: \text{visible(location(X, W), Y) \leftarrow friend(X, Y)} \): This rule states that any friend of the user X knows the location of X.

v. Privacy agreements: Privacy agreements contain the clauses for disclosing information. We essentially represent these through commitments among the OSN operator and a particular user. The commitments capture how visibility will be released to other parties. For each of the scenarios below, we use a subset of the following commitments:

- \( C_1: C(\text{osn, sally, friend(sally, X)}, \text{visible(with(sally, Y), X)}) \): The OSN operator commits to the user sally that her friends will be able to see who she is with.
- \( C_2: C(\text{osn, sally, friend(sally, X)}, \text{visible(location(sally, W), X)}) \): The OSN operator commits to sally that her friends will be able to see where she is.
- \( C_3: C(\text{osn, charlie, colleague(charlie, X)}, \neg \text{visible(location(charlie, W), X)}) \): The third commitment is slightly different from the first two. Here, the OSN operator commits to the user charlie that his colleagues will not see where he is.

Next, we describe two scenarios that might occur in such a system. The OSN operator, three users, two relations, and two contents all exist in the below scenarios.

First, we begin with a setting in which no location information is passed among the users. Example 1 demonstrates this case.

Example 1: Consider a setting where rule \( R_1 \), and commitments \( C_1 \) and \( C_3 \) exist. In this setting, friends know who each other is with, but they do not know where they are. Hence, if Charlie is in Montreal for a conference, then Linus should not be shown this information.

Next, we extend the first setting where we allow location information to be passed among users. Example 2 demonstrates this case.

Example 2: Consider a setting where rules \( R_1 \) and \( R_2 \), and commitments \( C_1 \), \( C_2 \), and \( C_3 \) exist. In this setting, friends both know who each other is with, and where they are. Hence, if Charlie is in Montreal, then Linus will have access to this information, if they are friends.

5. PROTOSS

The reasoning necessary for the examples above are done within the privacy checker of the OSN operator (Figure 2). PROTOSS is a tool to realize this reasoning. As input, it uses the privacy agreements of the users, user relations, the content they upload as well as some inference rules. Using this input, it decides
whether a certain privacy property that is of concern will be violated.

Figure 3 shows a screen-shot of the PROTOSS interface, which is built in Java\(^1\). From the interface, we can create the social network (together with its relations) according to the number of agents set or we can simply upload an existing social network specification, similar to the one in Section 4. Once the model of the social network is created (on the left pane), we can check whether the properties of interest are satisfied by the model. After the execution is completed, the output of the check is shown with relevant performance statistics (on the right pane). This output specifies whether the property of interest (e.g., whether OSN’s commitment to hide a user’s location) can be violated or not in a given social network. A user can then use this output to decide its actions.

The PROTOSS engine uses NuSMV model checker as a core component. However, NuSMV is not by itself capable of checking models with commitments in them. Hence, we have first introduced a commitment module into the NuSMV model checker, based on Telang and Singh’s work [8].

Three instances of the commitments module are shown above, that represent the commitments described in Section 4.

Inference rules are an important aspect of PROTOSS. An example inference rule is the following:

\[
\text{visible (location (Y, W_1), Z)} \leftarrow \\
\text{visible (with (X, Y), Z)} \land \\
\text{visible (location (X, W_2), Z)}
\]

This rule states that if X and Y are together and this fact is visible to Z, then when Z knows the location of X, he will also know the location of Y. These rules define semantic relations among concepts in the real world so that PROTOSS can make further inferences beyond its privacy agreement.

Given the above as input, PROTOSS can then check whether a privacy condition of concern takes place or not. Below we give an example property. Note that this property is phrased from charlie’s point of view. It checks whether there is a chance that the OSN operator’s commitment to charlie is violated at some point after it has become active. Remember that this commitment states that charlie’s location is not going to be visible to his colleagues. A violation of this commitment means that charlie’s privacy may be jeopardized. The property is described by the following CTL formula.

\[
\text{SPEC} \\
\text{AG (colleague_charlie_linus} \rightarrow \\
\text{EF c15.status = VIOLATED)};
\]

When we run PROTOSS on the setting described in Example 1, we see the formula returns false. That is, the commitment \(C_3\) (c15 in the NuSMV code listings) will not be violated in this setting, meaning that the location of charlie will not be visible to a colleague of his. This is an expected outcome, since there is no way of knowing where another user is (rule \(R_2\) does not exist). This is an example where the relations of users and hence their actions do not affect each other’s privacy. Hence, the OSN operator only checks that its commitments are safe with this set of relations.

When we run PROTOSS on the setting described in Example 2, however, we see that the formula returns true. That is, there is indeed a progression of events that will lead to the violation of \(C_3\). This means that \(C_3\) is no longer completely safe for the OSN operator. A simple case that depicts the violation is the following: Assume charlie is a colleague of linus. If they are also friends, then charlie’s location will be visible to linus. Thus, the \(C_3\) will be violated.

---

1. The full implementation can be downloaded from http://mas.cmpe.boun.edu.tr/ozgur/code.html, under Section “4. Experiments for Model Checking Privacy Agreements”.

---
6. Discussion

Fang and LeFevre point out that following a privacy agreement for a novice user is difficult and there is a need for easy-to-use privacy specification tools for such users [9]. In our work, we assume that privacy agreements can be represented and processed formally so that we can focus on the interplay between privacy agreements and user relations.

Krishnamurthy and Wills study the leakage of personal identifiable information in social networks [10]. The types of leakages they are concerned with are generally based on HTTP side effects that allow information to appear in URLs or cookies that can be used by other applications. While those identified leakages are important, the types of leakages we are concerned here are more high-level in the sense that even when such system level details are fixed can still exist.

Akcora, Carminati, and Ferrari measure how risky an individual in a social network is in terms of privacy [11]. They develop a method in which a user’s friends’ friends are analyzed in terms of their potential for learning and misusing personal information. While their aim is to identify risky individuals, our aim in this work is to help a system decide on potential privacy violations and inform the user appropriately.

Model checking has been used to identify a variety of failures in commitment-based systems. Menshawy et al. use model checking to verify the compliance of agents to their commitments [12]. Telang and Singh [8] model several business patterns as commitment interactions and map them onto CTL specifications. Then, using model checking they verify whether the underlying operational model (built with commitment semantics and its operations) supports the business specifications. Compared to these works, in addition to commitment compliance, we are also interested in effects of inferences and user relations on commitment violations. This means that even if an agent doesn’t violate a commitment through explicit actions, its relations with other agents can make the commitment violated. In our future work, we plan to evaluate the scalability of our approach on OSNs with various sizes.

Acknowledgment

This research is supported by Bogazici University Research Fund under grant BAP5694, and the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610. Pınar Yolum and Akın Günyay are partially supported by TÜBİTAK Scholarships 2219 and 2211, respectively.

References


