Identifying Transitivity Threats in Social Networks

Sorren Hanvey  
Lero - The Irish Software Research Centre  
Limerick, Ireland  
sorren.hanvey@lero.ie

Néstor Cataño  
Madeira Interactive Technologies Institute  
Funchal, Portugal  
nestor.catano@m-iti.org

Abstract—Transitivity threats refer to the unintended disclosure of information to unintended recipients as a consequence of an unrelated action. Transitivity threats stem from the automated transmission of data/content due to user actions within the social network. For example, commenting on some content within the social network makes the principal content visible to the recipients of the comment, thereby breaching the privacy policy under which the principal content was shared. This paper presents a novel approach for modelling and comparing social-network privacy policies to deal with transitivity threats. Our approach differs from existing approaches in its use of formal methods to address privacy threats. This work builds on a predicate calculus definition for social networking, modelling social network content, people, friendship relations, and privacy policies as access permissions to content. We have implemented our approach as a tool called Poporo. The tool extends on a previous version of the Poporo tool that checked a third party application's compliance with system invariants, described in [5]. We present privacy policy verification examples using the Poporo tool that validates our approach.

Categories and Subject Descriptors
Privacy Aware Social Networking

General Terms
Privacy, Transitivity, Social Networking Sites

Keywords
Privacy, Formal Methods, Transitivity, Tool, Poporo

I. INTRODUCTION

A. Motivation

With the ever increasing number of users on social networking sites, and the fact that user link their multiple social groups with each other the privacy of the content they share is of utmost importance. A typical social network user profile features personal information (e.g. gender, birthday, family situation), a continuous stream of activity logged from actions taken on the site (such as messages sent, status updated, games played) and media content (e.g. personal photos and videos). The privacy of this information has become a significant concern [15]. Users may upload content they wish to share with specific friends, but do not wish to be widely distributed to their network as a whole. Privacy means something different to everyone. Based on the diverse types of privacy rights and violations, it is evident that technology has a dual role in privacy: new technologies give rise to new threats to privacy rights, at the same time, new technologies can help preserve privacy [20]. Formal Methods provides a mechanism to model the functionality of social networking sites allowing a user to reason about the consequences of their actions in terms of the set of access privileges granted over some content.

In this paper we focus on Transitivity threats. Transitivity threats are caused due to the automated transmission of content within a Social Networking Site and the granting of privileges over the content that do not respect the privacy policy associated with the content. For Example, User A (Primary User) sets his privacy policy to allow his friends to see some specific content PC. User B, a friend of User A, comments on the content PC and sets his privacy policy to allow his friends to view the comment. As per the design of most social networking sites, e.g. Facebook, User B’s action of commenting results in the original content PC being shared with all his friends. That is, the content PC is shared with a set of users that are not friends with User A, thereby breaching User A’s privacy policy.

Such back-doors for accessing content exist due to the conflicting goals of social networking sites. Although respecting the privacy of their client base is important, they must also grow and expand the connections between their users in order to be successful. This is achieved by allowing users to connect over common interests by exposing content through less restrictive policies.

Social-networking sites are constantly evolving and changing in keeping with user demand for additional functionality. This constant change leaves users in the dark as to how the social networking site handles their content and the consequences of their actions. The consequences of user actions refers to the access privileges granted to other users as a result of that action. Social networking sites provide users with the ability to specify their privacy policies but these are not always effective or are not always applied. The inadequacy of privacy policies stems from the fact that users find stipulating detailed privacy settings to be challenging [3]. Additionally, it is not always possible to trust the social-networking site to uphold user’s policies as became evident from Facebook privacy breaches in 2009 [13] when Facebook changed its privacy policies without informing its users, resulting in content from private groups, user defined “friend” groups and users being made public. Therefore, the users require a user-friendly mechanism that informs them of the consequences of their actions, allowing them to make informed decisions.
B. Contribution

In this paper, we present a formal methods approach to identify transitivity threats in social networking sites. The approach offers a mechanism to formally model privacy policies of social-networking sites, for reliably restricting user content, and for verifying whether a user’s actions breach the privacy policy associated with the content. The approach was implemented as the Poporo tool. Poporo extends on our previous work, wherein we presented a mechanism for checking third party applications to ensure they satisfied the system defined privacy policies. System defined privacy policies refer to system rules that ensure privacy. For example, a system rule might state that if a user can view some content, they must have previously been granted view privileges over said content. The work was presented in [5]. The Poporo tool extends the previous work to provide users with a mechanism to check the consequences of their actions against user defined privacy policies. User defined privacy policies deal with who should be granted privileges over content.

C. Paper Structure

This paper is structured as follows. Section II presents the components and methodologies adopted from the previous versions of the Poporo tool. Section III presents the approach behind the new version of the Poporo tool, describing the process behind privacy policy definition and compliance checking. Section IV presents the application of the approach to an example. Section V discusses related work, comparing them to our approach. In Section VI we present our concluding thoughts on the work.

II. PRELIMINARES

A. Matelas

The Poporo tool relies on Matelas [6], a social-networking model specified in Event-B [1]. Matelas is a predicate calculus abstract specification layer definition for social networking, modelling social-network content, privacy policies, social-network friendship relations, and how these effect the policies with regards to content and other users in the network [6]. Matelas has been implemented into a close-to-code library implementing the core functionality for social networking. Matelas models access permissions as invariants [1], and EventB-enabled tools can generate proof obligations to ensure that networking operations adhere to these invariants. The events defined in this model are used to represent the user actions that make up a privacy policy.

An example of a Matelas event/method, modelled in Event-B, is shown in Figure 1. This example presents the transmit_tolist event. This event is triggered to transmit (share) some content with users in a user defined list within the social network. The event specifies a set of guards that must hold before the transmit actions are executed. The guards are placed between the WHERE and THEN. The guards state that the set of users, ps, in the list, l, must exist within the social network as must the content, rc, being transmitted. Additionally, the content being transmitted along with the list being transmitted to, must be owned by the same user calling the event. Owning content means a specific user has being granted all privileges over the content. Ownership is aware to the user that adds the content to the social network. If these guards are satisfied then the actions defined are executed. The actions are placed between the THEN and END and counting single variable clauses. In this case, the content, rc, is placed on the pages of all the members of list l and all the members of l are granted view privileges over said content.

Most online social networks are built on Java or Java based technology platforms. It is necessary for the Matelas model to be accessible and understood by such platforms. Therefore, the Matelas model is translated into the Java Modeling Language (JML) [4] which is a model-based language for specifying the behaviour of Java classes. This translation is performed through a custom built translator presented in [17].

B. Translating JML to Yices

The definitions of the social networking operations generated by translating the Matelas model into JML are then translated into the Yices Input Language. The primary motivation for mapping the JML model into Yices is to be able to use an efficient solver to verify the JML model for social networking, i.e., the Yices SMT Solver [12], [11]. The solver also used when checking whether a users action might breach any of the system privacy rules. The solver accepts as input the user actions in an appropriate format and provides as output whether the system rules are satisfied. The JML definitions are translated into the Yices Input Language via translation rules defined in [7]. The definitions of the social networking operations in the Yices Input Language is referred to as the Yices Prelude. This Yices prelude is used to verify privacy policies for compliance (whether a policy can coexist with another).

Figure 2
Social Networking Site

**C. The Verification Condition Generator**

The verification condition generator (VCGen) takes the user-defined privacy policy and calculates a weakest precondition predicate based on the program instructions \([10]\), which can be assignments, conditionals, variable declarations, or calls to predefined Matelas rules. We show some of the weakest precondition rules below, where \(m.P\) and \(m.Q\) are the precondition and the post-condition of method \(m\) respectively.

- **Assg:** \(WP(x = E, Q) = WP(E, Q[x\rightarrow result])\)
- **Seq:** \(WP(S; T, Q) = WP(S, WP(T, Q))\)
- **MCall:** \(WP(m(y), Q) = m.P \land m.Q[x, y] \Rightarrow Q\)

The VCGen parses the privacy policy statement by state-generation creating a verification condition \(VC_i\) for every statement \(S_i\). \(VC_i\) take the form shown below, where \(r_i\) is the pre-condition of instruction \(S_i\) and \(t_i\) is its post-condition. The consolidated \(VC\) is passed to the Yices SMT solver and is checked for satisfiability. If \(S_i\) is a method call (MCall), \(VC_{i+1}\) represents the property \(Q\) (calculated through a weakest precondition rule) that the method postcondition \(t_i\) must verify.

```scheme
(define postcondition-transmit-tolist::(-> variable datatypes (-> jmlset int int))
(lambda (Pre-State variables)
  (lambda (Post-State variables)
    (lambda (prs::jmlset rc::int l::int)
      (mk-record p:: persons-prestate
        c:: contents-prestate
        o:: owner-prestate
        pp:: (jmlrel-union pages-prestate (jmlrel-product-singleton-set rc prs)))
      wp:: (jmlrel-union view-prestate (jmlrel-product-singleton-set rc prs))
      ep:: editt-prestate
      prn::principal-prestate
      fld::field-prestate
      req::required-prestate
      wc:: wallcontents-prestate
      w:: wall-prestate
      wa:: wallaccess-prestate
      wc:: wallcontents-prestate
      lp:: listpe-prestate
      lo:: listow-prestate
      pol:: policies-prestate
      disj:: disjoints-prestate
      b:: (and (jmlset-is-subset persons-prestate prs)
                (jmlset-is-equal (jmlrel-apply-to-elm owner-prestate rc)
              )
                (jmlset-is-empty prs)
                (jmlset-is-member (jmlrel-get-domain listpe-prestate) rc)
                (jmlset-is-equal (jmlrel-apply-to-elm listow-prestate l)
              )
                (jmlset-is-member contents-prestate rc)
                (jmlset-is-member contents-prestate rc)
                (jmlset-is-equal (jmlrel-get-domain listpe-prestate) rc)
                (jmlset-is-equal (jmlrel-apply-to-elm listow-prestate l)))))))
```

**III. THE POPORO TOOL**

The Poporo tool attempts to address some of the concerns presented regarding privacy in online social networks. The Poporo tool provides a mechanism for the definition of privacy policies within a Matelas based social network. The tool can compare the user’s current privacy/sharing policy against any other privacy policies regarding associated content to identify transitivity threats. Poporo’s architecture is presented in Figure 3.

**A. Policy Definition in Poporo**

In the context of Social Networking Sites (SNS), from a user’s perspective, a privacy policy is a statement that discloses some or all of the ways a system shares and manages the user’s data. Personal information can be anything that can be used to identify an individual, not limited to but including name, address, date of birth, marital status, contact information or any content shared by a user within a SNS. From the perspective of the SNS system, it is a statement that declares a policy on how it collects, stores, and releases personal information. It informs the user what specific information is collected, and whether it is kept confidential or shared with partners and if so, how.

More specifically in the context of Poporo, privacy policies are defined on a per-user basis. A user’s privacy policy defines a set of other users within the SNS, which can be granted view or edit privileges over some content. With the concept of user defined lists, a user can specify which list of users they wish to share said content with and which set of users must never be granted any privileges over the content. A user might have multiple different policies regarding content sharing. This is due to the flexibility required within social networking sites when it comes to content sharing. A look at trends in current SNS shows that a user can either use a pre-defined policy or define a new one, every time they want to share some content. For example, when using Facebook, a user may add some content to the SNS and then wish to share it. When sharing, the user’s default policy is set to the policy last defined by the user, the user then either selects this policy or define a new policy on-the-fly.

A new policy, the policy to be checked for compliance, can therefore be defined as the consequences of any actions performed by a user with a SNS. More specifically a policy is defined by the set of users granted privileges over some specific content.
A user defined policy might stipulate that when a user \(ow\) publishes certain content \(rc\), it must be added to the page of all users in list close_friends. The policy definition would be as below, where create_account creates users accounts, upload_principal adds network content, create_list creates used-defined lists and transmit_tolist shares content with a specified subset of users.

```c

OriginalPolicy()
{
    create_account(ow, rc);
    upload_principal(rc, ow);
    create_list(close_friends, ow);
    transmit_tolist(close_friends, ow);
}
```

An example of a transmission policy adopted by a user commenting on the content \(rc\) above would be as shown below. In this policy the users chooses to associate the comment \(cmt\) with the content \(rc\) and share it with the user-defined list of users, work.

```c

CommentPolicy()
{
    create_account(work,rc);
    upload_principal(content,ow2);
    create_list(work,ow2);
    transmit_tolist(content, work);
    transmit_tolist(cmt, work);
}
```

A transmission policy for a comment as described above would then be compared to one of the Original policy to check whether it breaches it. It is necessary for a policy to contain all the relevant actions, pertaining to the content, performed by a user as the Poporo tool is not aware of the context in which the content is being shared. It is therefore necessary to inform the tool that the content in question was uploaded to the network by a user who previously created an account and the content is being shared with a list of users which was also previously created.

The Poporo tool is not provided the state of the system in its entirety. We rather provide it with the entities involved in the privacy policy and their relationships. We ensure that the conditions required for the various methods are satisfied by first checking whether the policy satisfies the system invariants. Only once the individual policies are deemed correct are they checked for compliance.

### B. Policy Compliance

The aim of the Poporo tool is to provide a mechanism for users to compare policies to check for compliance. Users constantly alter their privacy policies based on the context of the content they are sharing. Due to the nature of social networking sites, multiple users and their content are intricately linked. Therefore it is paramount for a user to be able to compare their privacy policy against the transmission policy of other users to ensure it won’t breach their privacy. SNS are constantly evolving providing their users with more functionality. This change, at times, takes the form of changing the interaction paradigm or the privacy policy paradigm within the SNS. To aid users the SNS will update a user’s privacy policy to reflect the new policy paradigm. A user must be able to confirm that the new policy adheres to their original policy. Based on the above there exist four major kinds of privacy breaches that a user might need to compare privacy policies for compliance to avoid:

- **SNS Privacy Breach:** As SNS evolve, the internal mechanisms, which determine how the SNS work, might change. These changes might effect how and to whom some content is transmitted or might change the way privacy policies are defined, thereby leaving the user open to privacy breaches.

- **User Privacy Breach:** Wherein a user inadvertently breaches their own privacy by not realising that consequences of their own actions due to a lack of understanding of the internal working of the SNS.

- **User to User Privacy Breach:** Wherein a user might unintentionally breach another user’s privacy policy due either sharing that user’s content or due to the automated transmission policy adopted by the SNS. e.g. Commenting on content can make it visible to an unintended audience.

- **Application Privacy Breach:** A user might use an external application or plug-in developed by a third party developer which might not adhere to either the user’s privacy policy or the policy enforced by the SNS.

There is at present no system in place within SNSs, such as Facebook, to allow the user to compare the consequences of their actions to those of a pre-existing policy. In addition to the above shortcoming, policy enforcement is only employed when sharing some content explicitly. The privacy issue raised by such selective policy enforcement is, there exist other actions a user might perform which alters the set of users the said content is visible to. For instance, in Facebook, a user might tag some content thereby making it visible to all the "friends" of the tagged user or a comment on some content by a user might make the content visible to the user’s friends. Therefore there is a need for policy comparison and compliance every time a user performs an action that might alter the set of privileges over the content in question.

Any action or operation performed by a user, either directly within a SNS or via an external plug-in, can be considered as the definition of a new transmission policy. For instance, if a user were to upload some content \(rc\) and share it with a list of users, List\(_A\), the policy defined would state that only the users in List\(_A\) should have view privileges and only the owner of \(rc\) must have edit privileges. Next if a user comments on \(rc\), the new policy would specify a list of users, List\(_B\), who now have view privileges over the comment and original content. It is necessary to check that the list List\(_B\) is a subset of List\(_A\). If this is not true the action of commenting would breach the original privacy policy specified.

Poporo allows a user to check the compliance of two policies. The new policy can stipulate that the user may share content with a list of users that is known to be a subset of the
The relation among social-networking variables (the lists) is represented as a JML property and it is translated to Yices. This translation is carried out by a custom JML-to-Yices translator based on the work presented in [7]. This translator translates a subset of the JML. The acceptable subset is restricted to the JML constructs and data-structures necessary to adequately define and compare privacy policies. The Yices property is combined with the Yices verification conditions (VCs) obtained for the translation of both policies, and is then passed to the Yices solver. If the VCs and the translation of the JML relationship are satisfiable, then the two policies comply with Matelas. During the process of verification of a privacy policy, the Yices solver generates a post-state of the execution of each policy. The Poporo tool then checks whether the post-state privileges afforded by the new policy is a subset of the post-state privileges afforded by the old policy. More concretely, it checks whether access permissions after the execution of the new policy are a subset of the access permissions after the execution of the old policy. If it is the case, the new policy complies with the old policy. This approach is more complete than simply comparing the lists of users that the content is shared with at transmission, as this accounts for any change made to access permissions or privileges through any action carried out by the user, another user, a plug-in or the SNS.

IV. APPLICATION EXAMPLE

Let us assume, a user has defined a privacy policy which states that when their content is shared, it must only be shared with the users in the user defined list close_friends. To this effect the user defines a policy as shown below. This policy is referred to as the OriginalPolicy. In this policy first a user ow1 is created with some required content rc1. ow1 refers to the user in question who is defining their policy. Next the policy uploads some content to the network via the upload_principal() operation allocating ownership to ow1. Then the policy creates a list close_friends before transmitting content to the users in list close_friends.

OriginalPolicy(){
    create_account(ow1, rc);
    upload_principal(content, ow1);
    create_list(close_friends, ow1);
    transmit_tolist(content, close_friends);
}

Now, another user defines a CommentPolicy, wherein when they comment on some content it must be shared with the user defined list work. The comment they are sharing is linked to some existing content, content. To this effect the policy is defined as shown below. This policy is referred to as the CommentPolicy and reflects the mechanisms adopted by social networking sites such as Facebook. In this policy first a user ow2 is created with some required content rc2. ow2 refers to the user in question who is defining their policy. Next the policy adds some content to the network, i.e. the comment cmt via the upload_principal() operation allocating ownership to ow2. Then the policy creates the list work before commenting on content. The consequence of commenting is as follows:

- The users in list work are granted view privileges over content. This is implemented via the transmit_tolist method.
- The comment cmt is similarly transmitted to the users in list work.

CommentPolicy(){
    create_account(ow2, rc);
    upload_principal(content, ow2);
    upload_principal(cmt, ow2);
    create_list(work, ow2);
    transmit_tolist(cmt, work);
    transmit_tolist(content, work);
}

To identify whether a transitivity threat causes a privacy breach, we use the Poporo tool to check whether the CommentPolicy complies with the OriginalPolicy. The tool will need to ascertain the relationship between ow1’s list close_friends and ow2’s list work. This relationship can be provided by the user or can be determined automatically from the social networking site. The relationship between lists refers to whether a given used-defined list is a subset of another user-defined list.

If the list work is a subset of the list close_friends, the JML property passed to the tool would be “ensures work.isSubset(close_friends)”. The set of access permissions granted after the execution of the CommentPolicy and OriginalPolicy are presented below. The access permissions are represented as bit-vectors. The bit-vectors represent a relationship between a content and a user, a 1 representing that the user has been granted that privilege over that content while a 0 represents that no such privilege exists. As the set of access permissions granted with respect to content by the CommentPolicy is a subset of the permissions granted by the OriginalPolicy, the Poporo tool informs the user that there is no privacy breach.

(`=` viewp_Original 0b000001000000000000000001
           000000010000000000000001
           000000001000000000000001
           000000000000000000000001)

(`=` viewp_Comment 0b000010000000000000000001
           000000001000000000000001
           000000000000000000000001
           000000000000000000000001)

If the list work is not a subset of the list close_friends, the JML property passed to the tool would be “ensures !(work.isSubset(close_friends))”. The set of access permissions granted after the execution of the CommentPolicy and OriginalPolicy are presented below. As the set of access permissions granted with respect to content by the CommentPolicy is not a subset of the permissions granted by the OriginalPolicy, the Poporo tool informs the user that there is a privacy breach. The user can then choose to take the appropriate action to mitigate the breach or not.

(`=` viewp_Original 0b000001000000000000000001
           000000010000000000000001
           000000001000000000000001
           000000000000000000000001)

(`=` viewp_Comment 0b000010000000000000000001
           000000010000000000000001
           000000001000000000000001
           000000000000000000000001)
defined based on criteria such as a user’s clearance level, etc. Based on these rules when a user attempts to access some content the rules associated with the content are checked and if satisfied the system will allow the access. This differs from social networks as the user is not attempting to access some content, rather a set of content is shared with them. Therefore based on the rules/privacy policy content is shared with some users. The difference in the approach presented by the authors and our approach is in their application. T. Sans, et.al., apply policy definition within information systems wherein all the context is self contained. Within online social networks a lot of context might not be contained within the system but rather in the offline world.

VI. Conclusion

With the ever-growing complexity of social network relationships and the way social networking sites link content and their sharing, having a mechanism to identify potential privacy breaches is crucial. The complex social relations and connections that exist in SNS make it hard for users to identify what a large global policy entails. Rather, a user with multiple policies regarding specific content can have greater control over their content. This in turn raises the issue of non-compliance between policies over associated content leading to transitivity based breaches. Privacy breaches due to transitivity threats can very often be unintentional. Providing users with feedback as to the consequences of their actions can help mitigate such transitivity based privacy breaches. To overcome this problem, we have presented an application the Poporo tool to check for compliance of privacy policies across the social network.

REFERENCES


