A Workbench for Privacy Policies

Lucas C.J. Dreyer & Martin S. Olivier

Department of Computer Science, Rand Afrikaans University,
PO Box 524, Auckland Park, Johannesburg, South Africa 2006
lucas.dreyer@eskom.co.za
molivier@rkw.rau.ac.za

Abstract

This paper describes a tool that may be used to create and analyse privacy policies based on the InfoPriv model (proposed by us elsewhere). It is called the Privacy Workbench and consists of four modules: the Privacy Policy, Graph Module, Inference Engine and Rule Base. The Workbench maps privacy policies to graphs called information can-flow graphs. The vertices of an information can-flow graph represent entities while the arcs depict the potential information flow. It is the purpose of the Inference Engine to analyse the graph for all possible information flow between entities including conflicting information flow. It does this by using graph-traversal algorithms. The Inference Engine further resolves conflicting information flows. It uses a rule-based approach to choose the 'best' arcs to remove in order to resolve conflicts. These rules are contained in the Rule-base and make use of the information can-flow graph's structure and specifics of the privacy policy.

1. Introduction

Computer privacy is concerned with the protection of personal information against misuse [1, 2]. The recent publication of reports that outline the misuse of personal information in government institutions of the USA only serves to emphasise the importance of privacy-related issues. See [7, 8] for further information regarding privacy violations in the Internal Revenue Service (IRS) and National Crime Information Centre (NCIC).

We propose a set of techniques (contained in the Privacy Workbench) for assisting in the creation, maintenance and conflict resolution of privacy policies. These techniques will help to solve many of the above-mentioned problems. A rule-based approach will be given that can be used to improve the intelligence of the conflict resolution mechanism.

The Workbench is based on the InfoPriv model for privacy [5, 6] and draws heavily on graph theory and logic programming (Prolog [9] here). Note that the Workbench has been implemented in Prolog and examples of screen output will be shown to illustrate the operation of the Workbench.

The following section will give an overview of InfoPriv. An overview of the Workbench will be given in Section 3. This overview contains a description of the main components of the Workbench: the Graph module, Privacy Policy, Inference engine and Rule base. The Graph module will be briefly discussed in Section 4. Section 5 is concerned with the Inference Engine and conflict resolution.

2. Background

The purpose of this section is to give a brief overview of the InfoPriv model. Note that a complete discussion of InfoPriv can be found in [5, 6]. We will only outline its main features here.

2.1 Entities and information flow

The basic elements of InfoPriv are entities and the information flow between the entities. An entity is defined as a uniquely identifiable container of information about itself and other entities. The information flow between entities is caused by interaction between the entities. We will now illustrate the concepts of entities and information flow by means of an example.

Consider the three individuals: John Smith, Sarah Parker and Jane Ullman. John Smith is an ordinary taxpayer while Sarah and Jane are employees of the Internal Revenue Service (IRS) of the USA. Sarah Parker is both John's sister-in-law and a friend of Jane. What happens when Jane Ullman is assigned to process the tax information of John Smith?

Jane is permitted to see John's tax information since she is not related to him. However, we mentioned that Jane and Sarah are friends. It is, therefore, very likely for Jane to disclose John's tax information to Sarah (which is strictly prohibited because John and Sarah are relatives). Note that such an informal description of permissible information flow is called a *Privacy Policy*. A way to model such a policy is by making use of an information can-flow graph.

The information can-flow graph W = (E, $\rightarrow$) is a directed graph where E is the set of entities and $\rightarrow$ is the set of directed arcs between the entities. The arcs represent the possible information flow (see [3] for a description of graphs and graph theory). Note that quotation marks will be used to refer to an entity (for instance "John Smith's Tax").

Figure 1 depicts an information can-flow graph that represents the IRS scenario. The four vertices ("John Smith", "John Smith's Tax", "Sarah Parker" and "Jane Ullman") in Figure 1 are connected by positive and negative arcs to indicate valid and invalid (potential) information flow. Note that "John Smith's Tax" is separated from "John Smith" so that the information flow from it can be controlled separately from "John Smith".

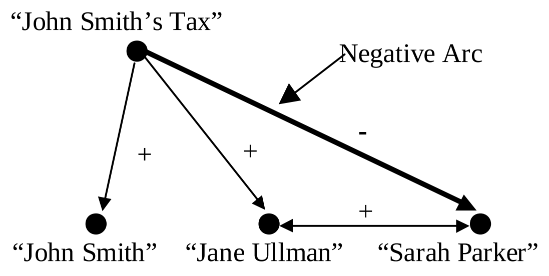

Figure 1 An information can-flow graph

The positive arcs in Information in Figure 1 indicate that information may flow from "John Smith's Tax" to both "John Smith" and "Jane Ullman". "Jane Ullman" and "Sarah Parker" are further free to exchange information since they are friends. The negative arc from "John Smith's Tax" to "Sarah Parker" indicates that she may not see John's tax information. It is possible for information to flow *indirectly* from "John Smith's Tax" to "Sarah Parker" via "Jane Ullman". Negative arcs are used to control indirect information flow.

Note that the information can-flow graph represents the potential information flow (hence a can-flow graph). It is, therefore, a representation of the static aspects of InfoPriv. A discussion of the dynamic aspects (information flow as it occurs) is beyond the scope of this paper.

The use of an information can-flow graph to represent a privacy policy has a number of advantages. First, it is a visual representation of the policy and is as a result easier to understand than a textual description. A large base of graph algorithms may further be used to analyse and manipulate the information can-flow graph.

Attributes of entities (for example, John's home address) are supported by the InfoPriv model but are outside the scope of this paper. We will assume that any attributes are stored separately by means of entities.

The rest of this paper is devoted to a description of a set of techniques (referred to as the Privacy Workbench) that can be used to analyse an information can-flow graph.

3. System overview

Figure 2 shows the structure of the Workbench. It consists of the following modules: Privacy Policy, Graph Module, Inference Engine and the Rule Base.

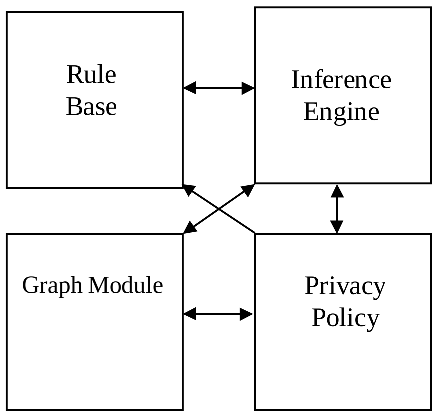

Figure 2 Structure of the InfoPriv Workbench

Central to the Privacy Workbench is the *Privacy Policy.* It contains a description of the entities and the information flow (positive and negative) between the various entities. The purpose of the Workbench is to permit the System Security Officer (SSO) to analyse the Privacy Policy for indirect and unauthorised information flow between entities.

The Inference Engine is responsible for managing indirect information flow. It can be used to answer questions such as 'can information flow from one entity (say "John Smith's Tax") to another entity (for instance "Sarah Parker")?' Another purpose of the Inference Engine is the resolution of conflicts in the Privacy Policy.

An example of conflict resolution is when the Privacy Workbench recommends that Jane Ullman should not see John Smith’s tax information. This will prevent Sarah Parker from obtaining the information. Note that this type of conflict resolution mechanism is limited to a static analysis of a privacy scenario.

It is not always clear what information flow to remove in order to resolve conflicts. We present a rule-based approach that can be used to choose what information flow to remove in order to resolve conflicts in the most optimal way. These rules are contained in the *Rule Base.*

The last module is the Graph Module. We noted in Section 2.1 that graphs and graph theory play a central role in InfoPriv. It is, therefore, sensible to isolate as much as possible of the graph-related operations in a single module. This module is the *Graph Module* and contains several operations that are dedicated to the maintenance of graphs. We will discuss the Graph Module in the following section.

4. Graph module

The Graph Module contains a number of operations that are used for graph maintenance. These are divided into three groups: *miscellaneous, path-related and reachability* operations. The miscellaneous operations include the deletion, copying and comparison of graphs as well as the adding and removing of vertices and arcs. Path-related operations are used to find paths between vertices in a graph. We will only concentrate on the reachability functions in this paper. Refer to [4] for a description of graph and path-finding algorithms.

The reachability set of an entity (say “John Smith’s Tax”) refers to the set of entities that may receive information from that entity. Information from “John Smith’s Tax” can reach “John Smith”, “Jane Ullman” and “Sarah Parker” if negative arcs are ignored. This type of reachability is referred to as “full-reachability”.

However, it is also necessary to know which entities can receive information from a specific entity while taking negative arcs into account (hence “limited reachability”). For instance, information from “John Smith’s Tax” can only reach “John Smith” and “Jane Ullman” if the negative arc of Figure 1 is taken into account. The full-reachability and limited-reachability sets of “John Smith’s Tax” are:

full-reach (“John Smith’s Tax”) = {“John Smith”, “Jane Ullman”, “Sarah Parker”}

limited-reach (“John Smith’s Tax”) = {“John Smith”, “Jane Ullman”}

Note that we will use 'reachability' to mean 'limited-reachability' in the rest of this paper unless otherwise stated.

Reachability is used by the Workbench to detect potential conflicting information flow. Note that information on how to determine the full- and limited-reachability sets of an entity may be obtained in [5, 6]. The following section contains a description of the Inference Engine of the Workbench. We will illustrate how a privacy policy can be statically analysed by means of the Inference Engine.

5. Inference engine

It happens frequently that the System Security Officer (SSO) has to implement and maintain a privacy policy for a large computer system. Such a privacy policy has to comply with the privacy policies of the company while catering for individual needs at the same time. As the complexity of the privacy policy increases it becomes increasingly difficult for the SSO to maintain it and make sure that it complies with the company wide privacy policies. For example, can information flow from “Entity1243” to “John Smith” be permitted without introducing conflicts in the privacy policy or violating the company privacy policies? It may be very difficult to quickly answer such a question.

The following three sections present tools for performing a static analysis of a privacy policy as well as conflict resolution.

5.1 Can-reach and conflicts

The Workbench includes the *conflict* predicate that can be used to detect conflicts in a privacy policy. “Conflict” examines the limited- and full-reachability sets of all the entities to find possible unauthorised information flow. It was shown in Section 4 that the full-reachability set of “John Smith’s Tax” contains “Sarah Parker”. However, the reachability set of "John Smith's Tax" does not contain "Sarah Parker" (Information flow is prohibited from “John Smith’s Tax” to “Sarah Parker” according to the negative arc in Figure 1). The possible information flow from “John Smith’s Tax” to “Sarah Parker”, therefore, forms a conflict.

Figure 3 shows the use of the *conflict* predicate in a Prolog session. The session shows that there may be an unauthorised information flow from “John Smith’s Tax”, to “Sarah Parker”. The following section describes how the Workbench can be used to automatically resolve conflicts.

The Workbench further includes the *canreach* predicate. Its purpose is to determine whether information can flow between two entities, taking negative information flow into account. Canreach examines the reachability set of an entity for this purpose.

```
Goal: conflict (1, A, B)
A=John Smith’s Tax, B=Sarah Parker
1 Solution
Goal:
```

Figure 3 Use of the *conflict* predicate

The SSO can, therefore, use canreach to determine whether unauthorised information flow can occur between two entities even if the privacy policy does not explicitly prohibit it (by means of negative arcs). Figure 4 shows the use of the canreach predicate.

```
Goal: canreach (1, A, B)
A=Jane Ullman, B=Sarah Parker
A=Jane Ullman, B=Jane Ullman
A=John Smith's Tax, B=John Smith
A=John Smith's Tax, B=Jane Ullman
A=Sarah Parker, B=Jane Ullman
A=Sarah Parker, B=Sarah Parker
6 Solutions
Goal:
```

Figure 4 Use of the *canreach* predicate

Consider two entities: “John Smith’s Tax” and “Jane Ullman”. “Canreach” determines whether information can flow from “John Smith’s Tax” to “Jane Ullman” by testing whether “Jane Ullman” is in the reachability (not full-reachability) set of “John Smith’s Tax”. According to Figure 1 this is the case so canreach concludes that information can flow from “John Smith’s Tax” to “Jane Ullman”.

5.2 Elimination of conflicts

It is necessary to resolve conflicts as soon as they are detected. The SSO can either remove them manually or the Workbench can attempt to resolve conflicts automatically. The algorithm of Figure 5 is used by the Workbench to resolve conflicts.

This algorithm uses the *conflict* predicate to search for entities (in the privacy policy) between which unauthorised information flow can occur. In the policy of Figure 1 the two entities are “John Smith’s Tax” and “Sarah Parker”. A path-finding algorithm can be used to find a path between the two entities. Such a path for “John Smith’s Tax” and “Sarah Parker” is:

{(“John Smith’s Tax”, “Jane Ullman”), (“Jane Ullman”, “Sarah Parker”)}

One of the arcs in the path between “John Smith’s Tax” and “Sarah Parker” should be removed and this greatly depends on the nature of the privacy policy. We adopted a rule-based approach in the Workbench that permits the SSO to specify rules for deciding which arcs to remove from the privacy policy. The next section will discuss the rule base component of the Workbench.

```
ResolveConflicts (Policy)

Inputs: Policy is the privacy policy
Outputs: Policy without conflicts

1       while conflict (Policy, Entity1, Entity2) do
2           FindPath (Policy, Entity1, Entity2, Path)
3           delete an arc from the path stored in Path
```

Figure 5 Resolving conflicts in a privacy policy

5.3 Rule base

We will now briefly discuss various rules that can be used to choose the best arcs for removal during conflict resolution. The information can-flow graph of Figure 6 will be used in this section to illustrate these rules.

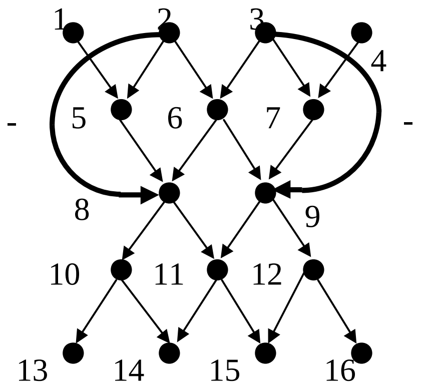

Figure 6 An example information can-flow graph

Conflict resolution rules can be roughly divided into two groups: *graph-related rules* and *policy-related rules.* The first group of rules analyses the structure of the information can-flow graph. For instance, arcs should be removed in such a way as to have the smallest impact on the information flow between entities. The second group consists of rules that depend on the nature of the specific privacy policy. For example, a manager should be affected the least during conflict resolution as compared to a regular employee.

Removing one or more arcs from an information can-flow graph can disrupt the information flow between entities. We refer to the extent of such disrupted information flow as the 'impact on information flow'. For instance, information can no longer flow from "1" to "5" if the arc (1, 5) is removed from Figure 6. However, information from several entities can no longer flow to "10" if the arc (8, 10) is removed. This definition is rather vague and serves only to illustrate the type of heuristics that can be used.

Consider the information can-flow graph depicted in Figure 6. Information is prohibited to flow from “2” to “8” and from “3” to “9” (see the negative arcs). We will now illustrate how the properties of the graph can be used to select the best arcs for removal in order to prevent the above-mentioned information flows.

The Workbench can be used to resolve this conflict. It will do so by determining the paths from “2” to “8” which are {(“2”, “5”), (“5”, “8”)} and {(“2”, “6”), (“6”, “8”)}. At least one arc must be removed from both paths in order to prevent the information flow from “2” to “8”. Likewise the Workbench will determine the paths from “3” to “9” which are {(“3”, “7”), (“7”, “9”)} and {(“3”, “6”), (“6”, “9”)}. Figure 7 shows the use of the "analysis" predicate of the Workbench to resolve conflicts.

```
Goal: analyse (1)
remove: Entity5, Entity8
remove: Entity6, Entity8
remove: Entity6, Entity9
remove: Entity7, Entity9
Yes
Goal:
```

Figure 7 Use of the analyse predicate to resolve conflicts

Note that the arcs shown in Figure 7 (for instance the arc from “Entity5" to "Entity8”) are removed from the privacy policy. However, the arcs chosen by the Workbench are not optimal since the information can-flow graph is partitioned into two graphs after conflict resolution. We can improve the selection of the Workbench by ordering the arcs according to some criteria. The criteria that will be used in this example is related the degree of the vertices.

The degree of a vertex in a graph is defined as the number of arcs that are incident with the vertex [3]. For instance, the degree of “2” is 2 while the degree of “5” is 3. Consider now the arc (“2”, “5”). The “arc-degree” of the arc can be calculated by adding the degrees of the two vertices that are connected by the arc. For (“2”, “5”) the “arc-degree” is 5. Figure 8 shows a simple rule that can be used to order two arcs according to their arc-degrees.

```
IF arc-degree (Arc1) <= arc-degree (Arc2)
THEN remove Arc1 first
ELSE remove Arc2 first
```

Figure 8 A rule to order two Arcs

Figure 9 shows the arcs that are removed by the Workbench according to the rule in Figure 8. These arcs are better choices since their removal will have a smaller impact on the (potential) information flow of the can-flow graph.

```
Goal: analyse (1)
remove: Entity2, Entity5
remove: Entity2, Entity6
remove: Entity3, Entity6
remove: Entity3, Entity7
Yes
Goal:
```

Figure 9 Improved result by ordering the arcs

Note that the rule of Figure 8 can be extended to take into account the specifics of the privacy policy. An example of such specifics is that the information flow to and from system administrators, managers and other key figures should be the least affected. The extended rule is depicted in Figure 10. Note that the value of “manager-count (Arc1)” indicates whether zero, one or both of Arc1's vertices are managers.

```
IF  manager-count (Arc1) =
    manager-count (Arc2) THEN
BEGIN
    IF  arc-degree (Arc1) <=
        arc-degree (Arc2)
    THEN remove Arc1 first
    ELSE remove Arc2 first
END

IF  manager-count (Arc1) <
    manager-count (Arc2) THEN
    remove Arc1 first

IF  manager-count (Arc2) <
    manager-count (Arc1) THEN
    remove Arc2 first
```

Figure 10 The extended rule

Assume that “Entity2” in Figure 6 is a manager. Figure 11 shows the arcs removed by the Workbench according to the extended rule. We can see that the arcs (“Entity2”, “Entity5”) and (“Entity2”, “Entity6”) are not deleted since they are incident to “Entity2”.

```
Goal: analyse (1)
remove: Entity5, Entity8
remove: Entity6, Entity8
remove: Entity3, Entity6
remove: Entity3, Entity7
Yes
Goal:
```

Figure 11 Result by using the extended rule

The architecture of the Workbench is flexible enough to support any user-defined rules. This gives the SSO enough freedom to refine the conflict resolution of a privacy policy according to any set of circumstances.

6. Conclusions

A model called InfoPriv was proposed in [5, 6] that can be used to model the privacy of a computer system. The basic constructs of InfoPriv are entities and the information flow between them. Negative information flow can be used to explicitly prevent information flow between specific entities. Policies (that are defined by using InfoPriv) translate directly to graphs, with the entities forming the vertices and the information flow forming the arcs. Graph traversal algorithms can then be used to analyse the graph for possible invalid or unwanted information flows.

We proposed the Privacy Workbench that consists of a set of tools that can be used for the creation and maintenance of privacy policies as well as conflict resolution. This Workbench is based on the InfoPriv model and is implemented in Prolog. The Workbench consists of four modules: the Graph Module, Privacy Policy, Inference Engine and Rule base.

The Workbench goes further than other automated tools by offering a facility for the resolution of conflicts. It is the purpose of the Inference Engine to use the facilities of the Graph Module to analyse the privacy policy for conflicts. A rule-based approach is used to choose arcs in the information can-flow graph for removal.

We presented rules that analyse the structure of the information can-flow graph in order to choose arcs that will have the smallest impact on the graph when they are removed. Properties of the privacy policy can also be used to choose more optimal arcs for removal.

The Workbench was implemented with Prolog in a total of 568 lines of source code. The use of Prolog made the development of the prototype relatively easy. However, performance issues still need to be addressed since the prototype performs rather slowly with moderate to large privacy policies. It does not really scale to a large number of entities. The whole purpose of the Prolog prototype was to illustrate that the Privacy Workbench can indeed be constructed. It was not intended for practical use on large systems. Further work is still needed to implement some of the modules with a language such as C++.

7. References

[1] “Options for Promoting Privacy on the National Information Infrastructure”, Information Policy Committee, Information Infrastructure Task Force, Washington, 1997

[2] “Directive 95/46/EC on the Protection of Individuals With Regard to the Processing of Personal Data and on the Free Movement of such Data”, 24 October 1995, European Union

[3] **C. Chartrand, L. Lesniak,** Graphs & Digraphs Third Edition, Chapman & Hall, London, 1996

[4] **T. H. Cormen, C. E. Leiserson, R. L. Rivest**, Introduction to Algorithms, McGraw-Hill Book Company, MIT, 1994

[5] **L. Dreyer, M. Olivier**, “An information-flow model for Privacy (InfoPriv)”, IFIP WG 11.3 Working Conference on Database Security, Chalkidiki, Greece, July, 1998

[6] **L. Dreyer, M. Olivier**, “Dynamic Aspects of the InfoPriv Model”, Dexa 98 International workshop on SECURITY AND INTEGRITY OF DATA INTENSIVE APPLICATIONS, Vienna, Austria, August, 1998

[7] **IRS Systems Security: Tax Processing Operations and Data Still at Risk Due to Serious Weaknesses,** United States General Accounting Office, Washington, Document GAO/AIMD-97-49, 1997

[8] **National Crime Information Center: Legislation Needed to Deter Misuse of Criminal Justice Information,** United States General Accounting Office, Washington, Document GAO/T-GGD-93-41, 1993

[9] **C. Townsend**, Introduction to Turbo Prolog, Sybex, Alamenda, 1986