Privacy Compliance Enforcement in Email

Quintin Armour, William Elazmeh, Nour El-Kadri, Nathalie Japkowicz, and Stan Matwin*

School of Information Technology and Engineering, University of Ottawa, Canada
{qarmour, welazmeh, nelkadri, nat, stan}@site.uottawa.ca

Abstract. Privacy is one of the main societal concerns raised by critics of the uncontrolled growth and spread of information technology in developed societies. The purpose of this paper is to propose a privacy compliance engine that takes email messages as input and filters those that violate the privacy rules of the organization in which it is deployed. Our system includes two main parts: an information extraction module that extracts the names of the sender and recipients as well as sensitive information contained in the message; and an inference engine that matches the email information against a knowledge base owned by the organization. This engine then applies compliance rules to the information obtained from the extraction and database matching steps of the process. This prototype is currently being developed for a university setting. In this setting, it was shown to obtain a precision score of 77%. The next step of our research will be to adapt our system to the context of a health organization, where privacy rules are more complex and more sensitive.

1 Introduction

Privacy is one of the main societal concerns raised by critics of the uncontrolled growth and spread of information technology in developed societies. On the one hand, the complexity of the modern information systems maintaining our personal data is constantly growing. On the other hand, these systems are often accessed by personnel who are not sufficiently sensitive to the issue of personal data privacy. The fact that private information is in the hands of other people introduces the possibility of human error. To bring order to this complex privacy landscape, most countries have introduced, in the last several years, data privacy laws. In Canada, the main law is the Privacy Information Protection in Electronic Documents Act of 2000. Recently, Ontario has introduced Bill 31 to regulate the issues of privacy and information access in the healthcare sector. This legal framework is normative, and as such, it addresses privacy violations after they have been committed. We believe that IT, in general, and AI in particular, can assist in the development of tools that will detect privacy violations as they happen. Here, we focus on email exchanges initiated from an organization and worry about information breaches with respect to the privacy rules of that organization. Several factors contribute to the fact that information breaches are very likely. The quick pace of information exchanges is the first such factor: lots of information is being exchanged, and emails are

* Stan Matwin is also affiliated with the Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

often sent in a hurry. People put less thought into the content of their messages or in the nature of attachments than they do in the slower, manual correspondence process. Another factor relates to the complexity of the matter: privacy rules can be numerous, unobvious (legal vs. lay language) and very specific, and thus hard to interpret for the variety of staff handling personal information, as is found in a hospital. To illustrate this point, consider that Ontario's Bill 31 is 116 pages long.

There is a lot of interest in offering such capabilities, but most solutions do not seem to go beyond the lexical level for detecting and matching data against encoded privacy rules. Despite the clear potential, little has been done to employ knowledge-based techniques in developing privacy-aware solutions. For instance, Vericept (vericept.com) detects the presence of social security numbers, credit card numbers, and other specific identifiers in messages, yet it is clear that detection of privacy violations often requires inference. Privacy rules must be connected with the knowledge about the people and the types of information involved. It is this added degree of complexity which has motivated our work.

In this paper we describe the research and development of a compliance engine that would, once installed in an organization, warn employees of the potential privacy breaches their email messages may cause. The idea is to flag the various violations and hold the message until the violations are inspected (and potentially corrected) by a human operator. We give an overview of the various components required for such a system and discuss some of the technical details related to their interaction. The current prototype targets the academic environment, where emails are exchanged between students, professors, and administrators, each having different access rights to private information. The final goal of our work is to port this research to the healthcare environment. We are collaborating with The Ottawa Hospital (TOH) on this application of the research.

2 The Envisioned Engine

As mentioned, the compliance engine described in this paper is conceived for a university setting. In such an organization, different people are allowed access to different pieces of information about other people according to the role they play in the organization. These access rules are not always clearly set which can result in frequent privacy violations. For this reason, languages for the internal privacy practices of enterprises and for technical privacy enforcement must offer possibilities for the fine-grained distinction of users, purposes, data categories, purposes and conditions as well as clear semantics. Our engine offers a solution which is consistent with that of the standard EPAL language [1] that addresses all of the aforementioned elements. For illustration purposes, we show that all the elements of the rules addressed in this engine can be represented with EPAL syntax and can take advantage of the XML representation.

Our system considers some of the rules taken from the University of Guelph Privacy Policy on the release of student information [2]. The subset of rules implemented in our system is presented in Section 3.2 along with an EPAL representation.

The following is an example of an email that violates a student's (Student B) privacy in the context of the set rules. The email sender (Professor A), the program's advisor

who is entitled to know the student's personal information such as his home address and phone number, has cc'ed this information to a professor (Professor C) who will only be teaching the student a course and is not entitled to view such information.

```
From: A@uoguelph.ca
To: B@uoguelph.ca
CC: C@uoguelph.ca
Subject: Please Confirm Correct Information

Dear Student name(nameB),
  This message is to confirm your enrollment in CIS 2520,
  your instructor will be professor name(nameC). Also,
  for our records, could you please confirm your current
  contact information below:

  home_address(address)
  home_phone(phoneNum).

  Thank you kindly,
    program_adviser_name(nameA)
```

Our system blocks A's message to C, but it can be sent to B since B owns the personal information in the message. Furthermore, the system informs A of the fact that his message was not sent to C and indicates which rules of the policy were violated.

Building such a compliance engine presents a number of challenges. First, the language used in the drafting of legal documents is often obtuse and difficult to understand. It needs to be interpreted with a certain amount of skill, and then translated into a logical language appropriate for computer processing. Second, the database has to be organized in a way that allows for efficient access to and processing of data and rules. As well, the database should be modular enough so that new rules as well as new facts can be added and removed easily. Finally, the information extraction engine is difficult to implement given the fact that emails are expressed in free-form text and do not follow the types of schemas usually relied upon in typical information extraction tasks [3]. Though some information such as student and phone numbers will be easy to extract, other information pertaining to the context in which these simpler types of information occur will be extremely difficult to assess.

In the final version of the prototype we will incorporate the lessons learned from earlier work on AI and Legal reasoning [4], as well as Information Extraction [5]. Also, as we include other information types for extraction, the methods described in [6] and [7] will become useful.

3 Our Prototype

3.1 Overall Design

The main objective of the prototype is to develop the building blocks necessary to perform privacy compliance enforcement in email. The first element to consider is how

this system, or compliance engine, will interface with its environment. As shown in Fig. 1, for a given enterprise, the system has three elements as inputs: 1) the email, 2) the privacy policy and 3) the database. The only output of the system is whether or not the email conforms to the policy restrictions.

The email is a well-known input type. It is obtained by diverting the flow of emails handled by a mail server. These emails are segmented into the two major parts: header and body. Within each email, the headers of quoted reply chains can be used to establish context and help resolve pronouns.

The privacy policy is the starting point for the compliance engine. The policy can either be expressed in plain English or using IBM's XML-based language EPAL. The benefit to using EPAL is that the privacy policy would be more directly expressed in terms of rules and therefore more easily interpreted. Once the policy has been expressed as a set of rules, the component terms can be extracted from a document and violations detected.

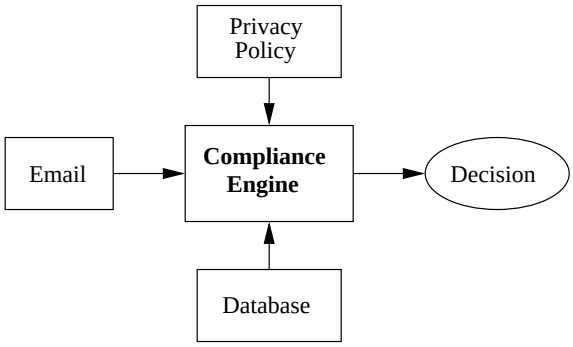

Fig. 1. An overview of the proposed system. The compliance engine is the element of interest.

The database is the place where all the domain information is stored. For all types of domains, a database of items such as employees and customers is usually present. For the academic domain, this includes information about professors, students, administrators, courses, etc. This database is used to help in the identification of the entities extracted by the entity extraction module discussed below.

The core of the compliance engine is composed of two major components. The first is the entity extraction module and the second is the privacy verification module. Each will now be discussed in turn.

The entity extraction module is needed to pull the elements required by the privacy verification module from the email. These elements are delivered to the verification module in the form of two lists. The first list is composed of the recipients and sender and the second contains all of the potentially private information in the email.

The privacy verification module uses the entities extracted by the entity extraction module and verifies their consistency with the database. First, the recipients and sender are identified. Once they have been identified, the types of information that each has

access to is considered. If there are any elements for which one of the parties involved does not have access, it constitutes a violation of the privacy rules. If a violation is detected then the email is flagged and the compliance engine makes the decision of whether to bounce, copy, divert or drop the email.

3.2 Detailed Description of the Prototype

In this section we describe the design and scope of the prototype outlined in the previous section. The focus is primarily on the privacy verification module because it is in this module where the reasoning (to decide whether an email is in violation of the privacy policy) is performed. The entity extraction module is essentially a tool which enables its operation. At this stage in the development, the extracted elements are relatively simple and will become more interesting as the rules become more abstract. The details of the extraction module are first presented, followed by those of the verification module.

At present, the system is only concerned with protecting the privacy of the elements in the database. As such, the task of the entity extraction module is to identify those items of interest in the source email documents. We are currently working to extend the system to consider the privacy of more abstract concepts. Although it complicates the information extraction module, the process is modular allowing more sophisticated methods to be incorporated as needed. In its current form, the system knows what it is looking for, and the challenge comes from the fact that the elements can present themselves in many different forms. The first task it performs is to take from the email header, the email addresses (and names if available) of the sender and the recipients. Since this module resides on the same level as the mail server it has access to all email header information, including bcc. The next step is to extract the information available from the body of the email. In the academic domain, the types of information found in the body of the email are for example, names, phone numbers, social insurance numbers, student numbers, dates of birth, addresses, etc. In order to extract these elements from the body of the email, we work backwards from the database, identifying different possible formats for each type of entity. This extraction can be performed using finite state automata with high accuracy. In later versions, the data will not necessarily be available in the database and more evolved information extraction techniques will be employed. The final step performed generates a series of queries which are then delivered to the privacy verification module. This module then replies as to whether or not privacy was compromised for the particular email.

The privacy verification module[1] we developed is designed to accommodate the storage and processing of three layers of information. The first layer is the data whose privacy the system is trying to protect against privacy breaches. The data here, are the elements in the university domain described above. The second layer describes the additional knowledge needed to assist in the process of determining whether or not a recipient is granted access to the data in the email. This additional knowledge describes the ownership and the type of data being released in allowing the email to be delivered.

[1] The prototype is implemented in Prolog, specifically in SWI-Prolog version 5.2.6 running on a WindowsXP Professional machine. The database, functions, and rules are implemented as Prolog facts, predicates and rules.

The third and final layer describes the rules which restrict access to the data being controlled by the system. These rules define the access levels and recipient privileges for the data in the email based on its ownership and information types. Such access rules are extracted directly from the privacy policy adopted by the university.

In the following sections, we describe the structure of each of the three layers of the database system with an emphasis on the process of developing information access rules based on the privacy policy.

The Data Layer. The database contains several tables and entities that describe each of the following:

1. Personal Details (e.g., ids, names, addresses, etc. of all individuals involved in university activities, such as, students, faculty, and staff)
2. Employee Details (e.g., ids, rank, status etc., for employees of the university)
3. Course Details (e.g., codes and titles of courses offered at the university)
4. Program Details (e.g., program code and department offering the program)
5. Academic Details describes how the different database entities relate to each other. (e.g., what courses a particular student taking or a particular professor teaching)

This database can be implemented to reflect the complete structure of a university; however, our prototype system is designed as a proof of concept consisting of an implementation of the essential parts of the database system. The purpose is to demonstrate the effectiveness of our methods in preserving privacy. Given a comprehensive implementation of the database, the privacy protection methodology described here can be extended to a large scale database information system.

Information Types and Ownership Rules Layer. In order to define information access rules to a data entity D, we must introduce additional knowledge to assist in the decision of whether or not to make D accessible to the user. This additional knowledge must specify the ownership and information type of D.

A data entity D is defined as a primitive Prolog fact or item[2] (a Prolog term listed in the database). For instance, the arguments of the predicate personal_details are each considered as separate data entities. For each data entity we define Prolog rules to determine a type description and an owner identification number. For example, an identification number is a data entity of the type employee_id which identifies the person who owns this personal record. Similarly, the name argument is a data entity of type personal_name owned by the person identified by the identification number and so on for the remaining arguments.

Therefore, in addition to storing the data, the database also stores rules about the type descriptions for each of the data entities and their owners. The privacy policy description directly affects the definitions of the information types and ownership. Laws and regulations govern who owns what type of information. In our case, as mentioned previously, we use the information privacy policy in [2] to extract the following rules to determine data information type and ownership:

[2] We are working to transfer the data to an SQL database

1. Identification numbers, personal names, home addresses, etc. are information types owned by the individuals who own the particular personal record.
2. Employee identification numbers, rank, and status are information types owned by individuals who are listed as being employed by the university.
3. Employee email addresses, office phone numbers, and names are information types owned by the university, which provides public access to the list.
4. Course codes and titles, program codes, degree titles, and departments are information types owned by the university.
5. Student registration information, listed by student names only, is also information owned by the university. This ownership setting allows for anyone to verify whether or not a student is registered at a university.
6. Student email addresses are information types owned by registered students.

Given the above information and ownership types, we can define information access rules to grant users access to the data based on privileges defined by the privacy policy.

Information Access Rules Layer. While they are extracted directly from the privacy policy, access rules must be expressed in a form suitable for representation in the database system. Our database system consists of tables and functions that provide users access to information stored in it. However, we assume that users may not access the general functions of the database directly but rather can only access functions designed to comply with our access rules. Therefore, given the privacy policy in [2], we implemented the following rules to control user access to the database:

1. Students registered in a program at the university are allowed access to public e-mail addresses, phone extensions, and names of employees in the university. In fact, such information is considered public information released by the university.
2. Active employees of the university who are teaching courses in a particular semester can be granted access to the identification numbers, names, and email addresses of only those students enrolled in a course they teach.
3. Student advisers and university staff members are permitted access to any information regarding any student (personal or academic).
4. Any information owned by the university is considered public information and can be released to the public. This rule may not be totally realistic, but given the scope of our prototype system, we felt it was reasonable. This includes any course or program related information, student confirmation of registration, and employee contact information.
5. Any individual has the right to access their own personal or non-personal data.

Please note that the above rules can be easily scripted in EPAL. Following is a translation of the first rule into EPAL. Similarly, all the other rules are represented and are not included in this paper due to space limitations.

```
<rule id = "r1" ruling= "allow">
- <user-category refid = "registered-students"/>
- <data-category refid = "non-personal-email-address"/>
- <data-category refid = "phone-extension"/>
```

```
- <data-category refid = "name"/>
- <purpose refid = "any-purpose"/>
- <action refid = "access"/>
```

Privacy Breach Detection in a Document. To complete the picture, the database now contains data entities, their ownership and information types, and rules to restrict their access. These access rules are consistent with the ownership properties as described in the privacy policy. Our system can now apply the following process to an email document in order to identify the existence of a privacy breach. The first two are provided by the entity extraction module and the third is performed by the privacy verification module.

1. Obtain the list IDs = [identification numbers of individuals receiving or sending the email document]
2. Obtain the list D = [data entities of interest appearing in the document]
3. For each tuple (id, d), where id is in IDs and d is in D, check for the existence of a privacy violation by applying all appropriate access rules to the tuple (id, d). If any such rule denies id access to d, then there exists a privacy breach.

Our system reports all privacy violations by indicating the identification of the individual attempting the access and by stating the data, the information type, and the identification of the owner of the data entity D.

A Complete Example. Consider the email example presented in Section 2. The e-mail was sent from A to B and copied to C. The database stores information about A, B, and C. For instance, A is a program adviser at the university and has access to all information regarding the student B. C is a faculty member who teaches a course in which student B is enrolled. C is not permitted to view any personal information for student B. In this case, our system will build the two lists:

1. IDs = [idA, idB, idC] using a mapping between identifiers and email addresses of A, B, and C found in the database.
2. D = [email(A), email(B), email(C), name($nameA$), name($nameB$), name($nameC$), course($cis2520$), address($address$), phone($phoneNum$)] as extracted from the email text.

Then, the system will determine the privileges for each identifier in IDs and whether these privileges enable access to each of the data entities in the list D. Access privileges are determined once the system resolves the information and ownership types for each of the data entities in the list D.

Although, idB is enrolled in a course taught by idC, and idC is granted access to idB and name($nameB$), our information access rules identify two breaches of privacy by idC. He or she is accessing the address($address$) and the phone($phoneNum$) of idB. Therefore, the output for the first violation will be:

```
check_violation(idC,employee_id,phone(phoneNum),
personal_phone_number,idB)
```

where idC is the user's id of type employee_id. He or she is attempting to access phone($phoneNum$) which is of type personal_phone_number and is owned by idB.

Similarly, the output for the second violation is:

```
check_violation(idC,employee_id,address(address),
personal_home_address,idB)
```

where idC is the user's id of type employee_id. He or she is attempting to access address($address$) which is of type personal_home_address and is owned by idB.

4 Experimentation – Semi-Automatic Processing

In this section, an evaluation of the system prototype is presented. First we describe the prescribed methodology and then present the results and analysis.

4.1 Experimental Setting

The following steps describe the experimental setup and methodology.

1. Obtain a set of actual email exchanges from one of the authors. This set is composed of 407 emails from the incoming mailbox, with 266 containing at least one of the information types of interest.
2. Extract the potentially private information.
3. Map this information to the non-sensitive elements in our hypothetical database.
4. Automatically generate a set of queries to pose to the system.
5. Report how many of the introduced violations were detected and how many non-violations were detected.
6. Introduce 20 privacy violations to non-violating emails. Craft these insertions such that a range of possible privacy violations are covered. Repeat steps 2-5.

4.2 Experimental Results

Here we present the results and analysis for the methodology described above. In the original 266 emails, 44 violations were detected. Of these, 34 were actual violations and 10 were wrongly identified as violations resulting in an overall precision of 34/44 or 77%. The reason for these 10 errors was due to the extraction process. It had identified a teaching assistant and a mass-mail list as external entities and was declaring, since student information was present, that a violation had occurred. Although these errors could be repaired by adding information to the database, the issue of data consistency was raised. In other words, by relying on the database to identify the entities present, we need to ensure that the database has accurate information. Also of note here is the number and type of violations detected. Although the number of violations was fairly significant ($\sim$13%), they were primarily of one type. The most common explanation for a violation was that a professor was allowed to see the student number of a student he or she was not teaching. This unintentional release of information is considered a breach of student privacy. Given that the task was performed using emails from a single

individual, it was logical that we only saw one type of violation repeated several times. This fact led to a recall score of 100%. As more abstract types of privacy violations are introduced to the system this number is expected to fall. For future experiments, we will need to use emails from several people in order to increase the variety and number of violations.

As for the modified set of emails from step 6, all of the introduced violations were detected. For each, the correct reason for the privacy violation was identified. This result was expected as the inserted violations were all in the format expected by both the extraction module and the privacy verification module.

5 Conclusions and Future Work

The purpose of this paper was to present the prototype we constructed for privacy compliance enforcement in email. We described the intended functionality of our software, its overall organization, the details of its implementation and we evaluated its performance on a set of real and modified emails. The system was shown to perform admirably well in the real-world setting for which it was created, obtaining a precision of 77%.

Our experience, thus far, suggests several improvements to our design:

1. The information extraction step can be simplified by considering the rules more closely. Considering the scenarios where proper names should not be disclosed can reduce the need for this more difficult step. In other words, only do name recognition if other information types are present to warrant it.
2. The extraction step can be extended to allow for partial matches to be extrapolated in order for them to match elements in the database. A probabilistic approach is needed to resolve whether or not partial entity A is the same as entity B described in the database.
3. The privacy verification module needs to be less tied to the database. This greater separation will make it easier to augment the system to include items such as student grades. As only final grades would be stored in the database, the detection of these must be done independently of the database. The owner of the grade would also need to be specified "on the fly" so to speak. The system needs to be able to handle this situation and would require more sophisticated text processing techniques.

Although our current system was implemented in a university setting, our ultimate goal is to port the prototype system to a hospital environment. The system would help to protect the privacy of patients (and personnel) from potential disclosure. As email becomes a more and more ubiquitous means of communication, some form of protection against privacy leaks becomes necessary. Imagine for instance an email message specifying the treatment of a patient, sent from one physician to another, and copied to the hospital pharmacy so that specific drugs could be administered as part of the treatment plan. If, inadvertently, an external pharmacy is copied on this message, a potentially serious privacy breach will occur, by releasing patient's name, condition, and treatment to a commercial organization.

6 Acknowledgements

The authors acknowledge the support of the Natural Sciences and Engineering Council of Canada, the Research Partnership Program of the Communications and Information Technology Ontario, and the cooperation of The Ottawa Hospital.

References

1. Paul, A., Hada, S., Karjoth, G., Powers, C.: Enterprise Privacy Authorization Language v1.2. IBM (2003) http://www.w3.org/Submission/EPAL/.
2. University of Guelph: Departmental Policy on the Release of Student Information. (1996)
3. Ciravegna, F., Dingli, A., Petrelli, D., Wilks, Y.: User-system cooperation in document annotation based on information extraction. In Gomez-Perez, A., Benjamins, V.R., eds.: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Lecture Notes in Artificial Intelligence 2473, Springer Verlag (2002)
4. ICAIL-2001: Workshop on AI and Legal Reasoning. (2001) http://www.cs.uu.nl/people/henry/workshop2.html.
5. Hersh, W.R.: Information Retrieval: A Health and Biomedical Perspective. Springer Publishers (2003)
6. Cohen, W., Sarawagi, S.: Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In: KDD 2004. (2004)
7. Borkar, V.R., Deshmukh, K., Sarawagi, S.: Automatic segmentation of text into structured records. In: Proceedings of the ACM SIGMOD Conference. (2001)