A Database Framework for Expressing and Enforcing Personal Privacy Preferences

Tania Basso[1], Leandro Piardi[1], Regina Moraes[1], Mario Jino[1], Nuno Antunes[2], Marco Vieira[2]

1 – State University of Campinas (UNICAMP) – Brazil
2 – University of Coimbra (UC) – Portugal

{taniabasso@ft, lpiardi@ft, regina@ft, jino@dca.fee}.unicamp.br,
{nmsa,mvieira}@dei.uc.pt

Abstract. *Nowadays, privacy protection in web applications and services is done, most times, through privacy policies that are presented to users and give them only the options of agreeing or disagreeing. So, it is not possible for users to express, in a detailed manner, their privacy preferences. Having this flexibility would allow users to make more thoughtful choices about the use of their personal information online. This paper proposes a database framework that allows users express their privacy preferences in detail, so that web applications can protect data privacy and manage personal information more securely. We tested the framework and results showed that it can be a simple and effective alternative, avoiding using complex and expensive solutions.*

1. Introduction

Web applications are a quite relevant technology nowadays because they provide a wide range of online services. Usually, to use these services, users need to provide personal private information. Once this information is available, they are no longer under control of their owner in respect to how they are used and the consequences of their indiscriminate availability, raising privacy concerns.

Currently, there is a growth of technologies to guarantee security and privacy of information manipulated by web applications and services. This is mainly due to regulation laws (the companies that hold private data have the obligation and responsibility to protect them) and competitive differentials (the more a company protects the privacy of its users, the better is its reputation). In this context, a highly used resource is privacy policy.

Internet privacy policies describe an organization's practices on information collection, use, and disclosure. Consumers use the stated website policies to guide browsing and transaction decisions [Earp *et al*. 2005]. Nowadays, to make a purchase or use certain online services, the privacy policy is displayed and gives the users only the option to agree or disagree with this policy. If they do not agree, they cannot perform the desired task. Most of the times it is not possible to users express their privacy preferences. This leads to the possibility of the user private data being accessed with purposes different from the ones intended by the users (the real data owners).

To assist data owners and collectors to communicate their privacy concerns, researchers have proposed various privacy policy frameworks and languages such as P3P [P3P 2013] and EPAL [EPAL 2014]. Although these technologies allow users express their preferences, it is done in a very general way, defining, for example, for

which goal their information can be used or who can view their information. Privacy protection based only in these definitions is frequently limited or insufficient. More rules and elements can be necessary to describe protection information decisions. For example, to allow users to express their preferences for each piece of personal information individually, it is necessary to define rules that address this purpose.

This paper presents a database framework for expressing and enforcing personal privacy preferences. The goal is to allow users to express their preferences in a more complete way, where the privacy preference of each piece of their personal information can be defined, based on predefined criticality levels. With this, users can make more thoughtful choices about the use of their personal information online. The framework provides a mechanism to enforce these policies, guaranteeing that the user's privacy preferences will be fulfilled and, thus, contributing to privacy protection.

The policy enforcement is done through an access control mechanism that, based on the predefined policies and the user preferences, allows or denies the information to the person who requests them. This mechanism is integrated in the relational database system and provides an algorithm constructed through database packages. Basically, the algorithm verifies if the data being requested can be retrieved, comparing its associated preference level (set by the user) with the access control policy (specifies the preference level each requester – role – can access). The underlying policies are defined using the policy model proposed in Basso et al. (2013). This model allows users to define the policies without requiring specific or in-depth knowledge about the web application because, instead of the rules defined by other standards, it follows a more user-friendly approach, enabling the users to express preferences about their personal information and even to distinguish among the information they provide.

To evaluate the proposed database framework effectiveness, a prototype tool was implemented and an experimental evaluation was conducted, both in terms of performance and scalability. We used the test process proposed by Mello et al. (2014) and results show that, in practice, the proposed approach allows collecting the user preferences (criticality levels) and enforcing access control considering these preferences. This is done in a non-intrusive way and with an acceptable performance overhead. This suggests that the framework can be used as a simple and effective solution.

The paper is organized as follows. Section 2 introduces related work. Section 3 presents the proposed database framework and Section 4 presents the experimental evaluation. Finally, Section 5 shows the conclusions and future work.

2. Background and Related Work

P3P (*Platform for Privacy Preferences Project*) [P3P 2013] is a protocol that allows websites to declare, in a standard format, privacy policies with the intended use of the information they collect about users, such as what data is collected, who can access those data and for what purposes, and for how long the data will be stored. This information can be retrieved automatically and is easily interpreted. EPAL (*Enterprise Privacy Authorization Language*) [EPAL 2014] allows enterprises to formalize their privacy promises into policies. These policies can define the categories of users and

data, the actions being performed on the data, the business purposes associated with the access requests, and obligations incurred on access. However, as aforementioned, these both technologies do not describe each personal information protection level neither provide mechanisms to enforce the privacy policies.

Agrawal et al. (2003) proposed a server-centric architecture, based on P3P, for matching preferences against policies at the database level. The goal is to establish the infrastructure necessary for ensuring that web sites act according to their stated policies. Byun and Li (2008) proposed a privacy preserving access control model based on P3P *purposes* (elements that defined the intended use of data) for relational databases. Accesses are granted if the access purpose defined by the requestor is among the allowed ones (previously defined by the user). Nevertheless, for this both works [Agrawal et al. 2003, Byun and Li 2008], access decisions based only in the P3P are frequently insufficient and more rules and elements are needed to describe disclosure decisions.

The P-RBAC (*Privacy-Aware Role-Based Access Control*) [Ni et al. 2007] is a model for access control that supports privacy policies. In P-RBAC, permissions are assigned to roles and users obtain such permissions by being assigned to these roles. It implements a structure of privacy permissions, which explicitly states the intended purpose, along with the conditions under which the permission can be given, and the obligations that are to be finally performed. Although P-RBAC is powerful, user preferences are limited and do not specify different levels of protection that each piece of information should have.

Probably, one of the reasons why these patterns (P3P, EPAL, P-RBAC) do not express individual privacy information preference are due to their user friendliness. Some researches try to improve the P3P user agent interface, in order to make the user agents more user-friendly and more effective when communicating the summary and policy warnings [Cranor et al. 2002, Kolter and Pernul 2009]. Obviously, users' interaction with applications that offers finer-grained options is more complicated due to the difficulty in explaining the options and the consequences of choosing them. However, our solution tries to deal with this through a more user-friendliness interface.

When comparing with the alternatives, the presented framework presents the following advantages: (i) instead of guarantee the same amount of privacy to every user, the finer-grained access control gives more flexibility to each user to express their preferences, once each user has different perception of their privacy; (ii) the user interface helps in user-friendliness of the solution. Although we have not yet performed studies about its usability, we believe that it is a first step to facilitate the understanding of the consequences of each privacy choice; (iii) the policies and the mechanism implementation are within the database so, the access control is done without application changes or application awareness of the implementation.

With respect to the validation of the proposed solution, we decided to adopt the process proposed by Mello et al. (2014) because it has already been applied to the evaluation of a privacy solution integrated in a web application. Furthermore, it supports the definition of tests scenarios, their execution and the comparison of the tests results, especially when using and not using the privacy solution.

3. The Database Framework

The database framework is based on the referred privacy policy model described in Basso et al. (2013). This model was selected due to its simplicity and the fact that it was constructed through an extensive study based on the literature and IT professional interviews. Thus, it is based on real needs that a policy model should tackle considering the requirements relevant for implementing a good access control mechanism. However, the model itself does not propose a mechanism to enforce the policies, especially in the persistence layer, which helps improving private information security. Our goal is to define a solution to perform this policy enforcement.

In a broad view, the framework consists of a set of independent tables that can be added to the application's database. These tables will contain the necessary information to perform the access control according to predefined privacy policies, which consider users preferences related to each piece of personal information (i.e., address, credit card number, etc.) to be collected, stored or managed. During the design of the solution, one of the goals was to avoid the introduction of new security or privacy concerns related to the new enforcement system. The access control is performed by implemented database packages and has the advantage of filtering the data directly in the database, contributing to security against possible attacks to the web application or the network. Figure 1 shows a general view of the solution, where the Database Framework is composed by the Framework tables and a Package engine.

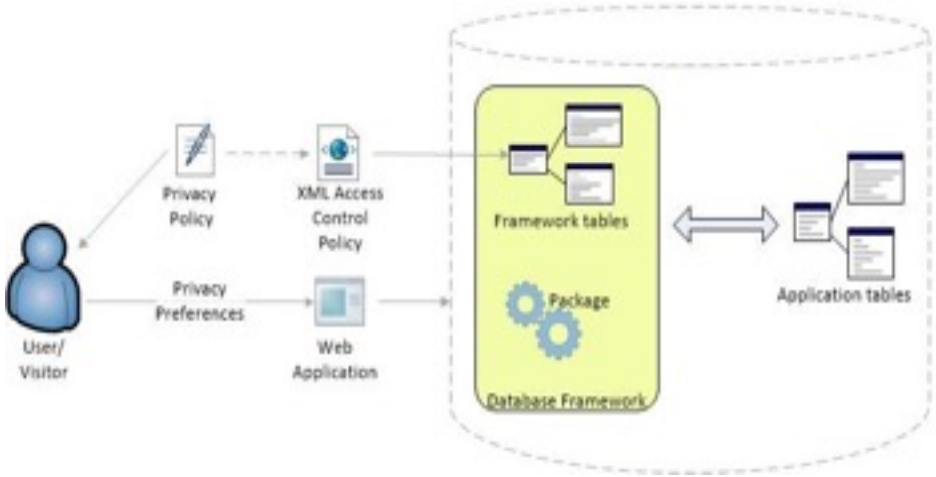

Figure 1. General view of the database framework solution.

Privacy policies are defined in textual manner and presented to the user. Part of the privacy policy usually addresses who can access private dada. Access control policies can restrict unauthorized access to data and thus, protect data privacy. This access control policy is defined through XML files. It expresses the criticality level that each role can access. The XML policies are mapped into the *Framework tables* to address which profile can access certain piece of user information. Based on the privacy policy, the users/visitors can express their privacy preferences through the web application interface, classifying each piece of the personal information with criticality levels (the higher the level, the higher the protection – the criticality levels are explained in section 3.2). The preferences are stored in the *Framework tables*, which are associated to the *Application Tables* (i.e., the tables that stores the users' information).

When some profile tries to access some users' information, the *Package* engine masks this information according to the privacy policies and the users' preferences. More details of this process are given in the next subsections.

3.1. Privacy Policies

In Figure 1, privacy policies are defined through XML files, based on the policy model described in Basso et al. (2013). In summary, the model defines *who* can access certain piece of information, *when, from where, and how* the required information can be accessed, as well as the criticality level of each piece of information. It has an offshoot called *Profile*, with a set of tags that were designed to represent groups and conditions requirements. Among these tags, the *Role* tag represents the user groups. Also, the model has the *Data Access* offshoot, whose set of tags allows determining the restrictions of information in table's columns and rows. The *Level* tag is part of this offshoot and represents the criticality level of the information (see Basso et al. (2013) for more details). For sake of simplicity, we focused on using information about the profiles allowed to access data and the criticality levels of data, disregarding other types of information defined by the model (e.g. *from where* the required information can be accessed).

Besides the simplicity, the model and its respective type of derived file (XML) was adopted because it: (i) allows that policy specifications can be kept under control; (ii) requires less specific knowledge from the person responsible for specifying the policies; (iii) allows easy integration with existing technology.

The XML privacy policies must be created based on textual privacy policies. It is known that natural language is the more adequate manner of communicating users about the privacy policy, but it consists in high-level statements that are difficult to be machine readable. Some works propose methodologies to map policy in natural language to a more low-level and formal representation [Breaux and Rao 2013, Breaux and Anton 2005]. However, this is not the focus of this work and details will not be addressed.

The XML files contain information about the system profiles and the personal data (associated with criticality levels) each profile can access or modify. These criticality levels are a scale of values to define how the information can be protected and will be further explained. Also, updates or new XML policies files can be added to the application: a job (i.e., a combination of a schedule and a program, along with any additional arguments required by the program) is executed periodically to verify, automatically, in a specific application directory, the input of new policies. Then, the job maps these policies to the set of framework's tables, as explained in detail in the section 3.4.

3.2. Criticality Levels

A typical database application manages data with different requirements in terms of security, ranging from non-critical data to data that has to be extremely protected against unauthorized access. These requirements can be represented through data criticality levels. These levels can be configured, added or even removed. In order to identify the different levels of criticality we established, for our study, the 4 levels described in the work of Vieira and Madeira (2005). They are:

- Level 1: non-critical data, i.e., data that does not represent any confidential information.
- Level 2: data in this level must be protected against unauthorized modification (for this class of data unauthorized read is less critical than unauthorized modification). One typical example is the list of products in an online retail store. This information has to be protected against modification (because it is used by the customers to perform orders) and should be open to all users.
- Level 3: data in this level must be protected against unauthorized read and modification. Most of the data in typical database applications is in this criticality level. Some examples are: clients' orders, costumers' information, and employees' information.
- Level 4: critical data that has to be extremely protected against unauthorized read and modification. This data must not be understandable even if someone is able to access the database using a valid username/password (i.e., this data has to be stored encrypted in the database). Typical examples are: usernames/passwords, credit card numbers, patients' files in hospitals, and bank accounts.

The levels we adopted are a good option to represent commercial online applications, but companies and organizations can establish their own levels according to their necessities.

3.3. User Preferences

After defining privacy policies, the normal process used by most companies is to present them to users and visitors, to inform them about company's privacy rules. To allow users and visitors to express their preferences about privacy of each of their personal information, the application must implement some controls while the user interacts with the application during the collection of this personal information. Obviously, this part is a little intrusive in the application, but it is a more intuitive way for users to express their preferences, giving them more flexibility to protect the privacy of their information.

To help users in this process we implemented an interface with simple text box or combo box informing, for each piece of data, the criticality levels that can be chosen. Also, to facilitate the users understanding about the criticality levels and the consequences of their choices, we placed some hints when hovering over them with the mouse, providing an easy visualization of the description.

As suggestions, default values of criticality level are set at first and then can be changed by data owners, via application. For greater security, data that must be extremely protected (as passwords and credit card numbers, for example) have the criticality level set to level 4 and this level must not be changed.

So, according to Figure 1, the user or visitor express his privacy preferences through the web application and information are mapped in the *Application tables* and *Framework tables*, which are explained in the next subsection.

It is important to mention that this is a first implementation of the user interface. Complementary studies about usability and user friendliness, regarding user's point of view, must be developed in order to improve this part of the solution.

3.4. Framework Tables

The *Framework tables* in Figure 1 represent a set of tables to store information about the privacy policies, including the users and visitors preferences. Figure 2 shows the Entity-Relationship Diagram to express the relationship between this set of tables.

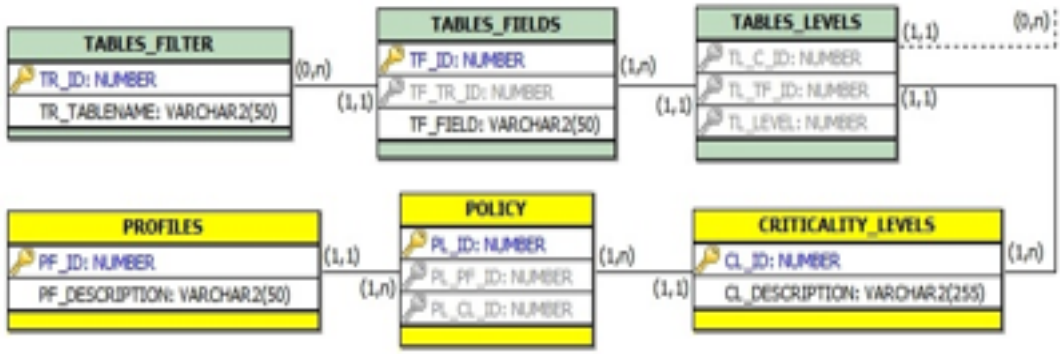

Figure 2. Entity-Relationship Diagram of the database framework.

In Figure 2, the information in the XML policies is mapped to the *Policy* table, addressing the levels of criticality of the information that the profiles can access (e.g. system administrator can access information with levels 1 to 4; trainees can access information with level 1 and 2). The users preferences are stored in the *Tables_Levels*, addressing the criticality level for each information to be protected (e.g. if a user defines the phone number as level 3, this value can be accessed only by the system administrator). More details of the mapping of the policy to the framework and the collection of users' preferences are given along this section. Also, the functions of each of the tables are explained below.

- *Profiles*: stores all the different system profiles existing in the organization as, for example, administrator, customer, vendor, etc.
- *Criticality_Levels*: stores criticality levels, i.e., the default values adopted by the company or organization according to their needs. The definition of such criticality levels must be done in a thoughtful way because they will be associated to each user personal data.
- *Policy*: associates the profiles and criticality levels, specifying, through criticality levels, the information each profile can access. This table stores, in the form of data, the privacy policies defined through XML files and presented to users and visitors.
- *Tables_Filter*: stores the name of the tables whose fields will have the access controlled. Typically, these tables are the ones that stores data that pertains to profiles which express their privacy preferences (e.g. customers) and associated tables (e.g. address, country, phone numbers, etc.)

- *Tables_Fields*: stores the fields of the tables (specified at *Tables_Filter*) that will have restricted access.
- *Tables_Levels*: stores the users (data owners) preferences. Typically, *Tables_Levels* is associated to the table that stores the data of the person or profile subject to data privacy (e.g. customers and its associations).

The mapping from XML policy files to tables is done as follows: the *Profiles* and *Criticality_Levels* tables must be pre-fulfilled according to the company's criteria. As the policies establish the profiles and the levels of criticality of the information that these profiles can access, the content of the *Role* tag in XML is checked to exist in the *Profiles* table. The same verification is done to the criticality level, i.e., the job checks if the content of the *Criticality_Level* tag in XML exists in the *Criticality_Levels* table. If both information are in their respective tables, the *Policy* table is fulfilled, characterizing the privacy policy. If they are not, a message is sent notifying the policy incompatibility.

For collection of users (data owner) preferences, the tables *Tables_Filter* and *Tables_Levels* must also be pre-fulfilled according to the company's criteria. The user specifies her privacy preferences for each piece of data to be collected and managed through the criticality levels and these preferences are stored in the *Table_Levels* table. The records of this table specify the criticality level for each field of each table to be protected. Default values can be set at first and then changed by users, via application, to express their preferences. In Figure 2, *Table_Levels* has a relationship represented by a dashed line, not associated with another table. This line represents the relationship that *Table_Levels* has with the applications tables. These applications tables store the fields that should be protected.

As we mentioned before, during the design of the solution it was a priority to avoid the introduction of new security or privacy concerns related to the new enforcement system. This way, besides placing the enforcement system right inside the database management system, we also plan that the presented tables must be *"implemented"* following the best security practices. For instance, a dedicated user should be created with the sole function of write on these tables and he should be the only one with permissions for that. This can mitigate the probability of the privilege escalation and tampering with the system.

3.5. Policy Enforcement

To guarantee the enforcement of privacy policies and, consequently, to respect users privacy preferences, a mechanism was developed and integrated to the framework. This mechanism is an algorithm integrated to the infrastructure of a relational database system to enforce disclosure control. It provides constructs that allow masking personal information according to the privacy policies and users preferences.

The mechanism was implemented using database packages. Database package is a resource to encapsulate related procedures, functions, associated cursors and variables together as a unit in the database. The packages were used because they provide advantages in terms of performance, since the entire package is loaded into memory when an object from the package is called for the first time. It is also aligned with our

privacy and security concerns, as it does not require additional communication between the database management system and the application level.

The *Package* engine in Figure 1 represents the implemented mechanism. Basically, its operation is: when a query is executed from the application, the *Package* obtains the criticality levels of particular fields returned by this query. Then, the data are masked and presented to the requestor.

To mask the data and, consequently, enforce the policies, the *Package* implements an algorithm, which is based on the following steps:

1. Obtain the identifier of the user that is requesting the private data, the role of this user, and the data that is being requested.
2. From this information (received on step 1), the table that stores the privacy policies (*Policy* table) is consulted to identify the criticality levels this user, with respective role, can access.
3. Obtain the data owner preferences, i.e., the criticality level the owner classified the private data that is being requested (*Table_Levels* is consulted).
4. Verifies if the criticality level that the user (role) can access is higher than the criticality level of the data,
   a. If true, the data is provided to the user who is requesting them.
   b. If false, the data is masked in order to enforce the policies and respect the data owner preferences.

This algorithm is applied to each data request. Obviously, if the user that is requesting the data is the owner of the data, all the information is provided. The package can be easily adapted to different web applications through changes in parameters as the main query that loads the cursor and the variables associated to the new query. For the sake of reducing unnecessary cost performance, the level 1 were not stored in the *Tables_Levels* table, once the corresponding data is non-critical and do not need to be protected.

4. Validating the framework: a case study

To better understand the potential of the proposed solution, a case study to validate it was developed. The tests campaign applied validated the access rules and evaluate the scalability and performance impact of the proposed framework. **Scalability** is expressed in terms of the number of records being processed by the mechanism, i.e., the goal is to understand how much the number of records in the database application affects the performance. The **performance impact** can determine the disadvantage of using the mechanism. For these experiments, the performance was characterized by the average response time and throughput.

4.1 . Experimental Setup

The web application used in the experiments is a Java implementation of a TPC-W [TPC 2015], which is a benchmark for web-based transactional systems where several clients access the website to browse, search, and process orders. To this study, the TPC-W implementation simulates a retail online book store. The components of the TPC-W database are defined to consist of a minimum of eight separate and individual base tables (*Customer, Address, Country, Orders, Order_Line, Author, Item, CC_Xacts*) [TPC

2015]. The proposed framework was integrated with the TPC-W database through the association of the *Tables_Levels* table (from the framework) with the *Customer* table (from the TPC-W). The database used in the experiments is Oracle Database 10g Express Edition Release 10.2.0.1.0 [Oracle 2015] and the mechanism was implemented using PL-SQL (a procedural language extension for SQL). The metrics were collected using the JMeter tool [Jmeter 2015].

The application use scenario to perform the tests simulates a third-party user (that represents a user trying to access unduly data or even a potential attacker) trying to obtain data of a registered customer through a search process. Privacy policies were implemented and the criticality level of each piece of data of each customer was randomly generated through a database script. We performed, previously, independent tests that evaluated the correctness of policy enforcement, respecting the customer's preferences.

For our experimental evaluation it was used, respectively, 500, 5000 and 50000 records in the database. The simulations of threads, that simulate concurrent connections to the server application, ranged from 1 to 128 users for each set of records. Also, in order to understand the performance impact, the tests were performed without the database framework in place (to obtain baseline indicators). For each run of the experiment, the whole system is returned to its initial state in order to avoid cached data.

4.2. Overall Results Analysis

Figure 3 presents the overall results of the study. It shows the average processing time of all requests for the customer search scenario. This average time is given in milliseconds and is presented in Figures 3a, 3b and 3c. Also, Figure 3 shows the throughput results. Throughput is calculated as requests divided by unit of time. The time is calculated from the start of the first sample to the end of the last sample, including any intervals between them, as it is supposed to represent the load on the server. The throughput results are presented in Figures 3d, 3e and 3f. Figures 3a, 3b and 3c shows the tests performed with, respectively, 500, 5000 and 50000 customers recorded in the database. The same for throughput: Figures 3d, 3e and 3f shows the tests performed with, respectively, 500, 5000 and 50000 customers. It is important to evaluate the framework with these different amounts of records because the table *Table_Levels* (Figure 2) has a proportional growth in relation to the amount of customer's records. Also, this growth is related to the amount of fields that the *Customer* table has, including its associations (for example, the *Address* table, whose fields are part of the customer register). To the experiments, the TPC-W customers have 17 fields and address has 7 fields, totaling 24 fields.

Analyzing the processing time requests in terms of performance impact, Figures 3a, 3b and 3c shows that the proposed solution has very low impact when few users are using the web application. Although in some cases the increased time represents a high percentage (for example from 10 to 20 represents a 100% increase), the time is milliseconds and this difference is practically derisive. So, the average time without the database framework is very similar to the other results until around 16 users. As the

number of users increases, little differences arise. The inclusion of the framework affects the performance for higher number of users but the system performance increases in a linear tendency and it is acceptable for high demand.

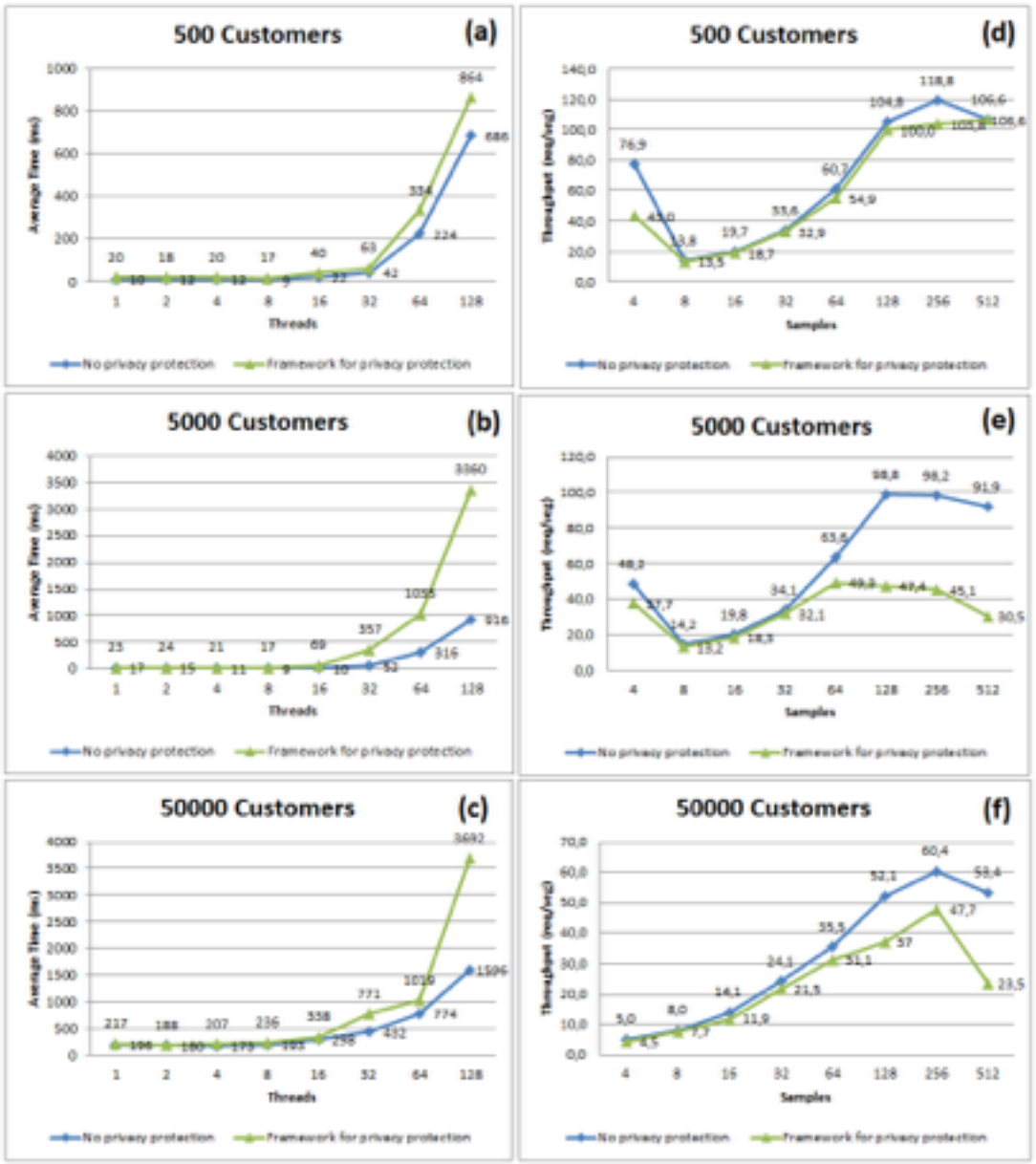

Figure 3. Experiments average time and throughput.

Figures 3d, 3e and 3f show the throughput variation. The samples are given by the number of users multiplied by the number of the requests of the application scenario use. The performance impact in the throughput analysis is also similar to average processing time results: about the 16 first samples the results with and without the proposed database framework are similar and the differences arise as it increases. We believe the differences between the throughputs (for different number of records) arise

due to the randomness of criticality levels, which may not have a realistic distribution of the values.

Through the analysis of the average processing time requests it is possible to observe that, also, the proposed solution is scalable in terms of numbers of records. The use of large number of records can be necessary due to different quantities of information in the databases for different companies or organizations sizes. Figure 4 consolidates the scalability evaluation experiments, aggregating the average time response results presented in Figures 3a, 3b and 3c, all implementing the framework.

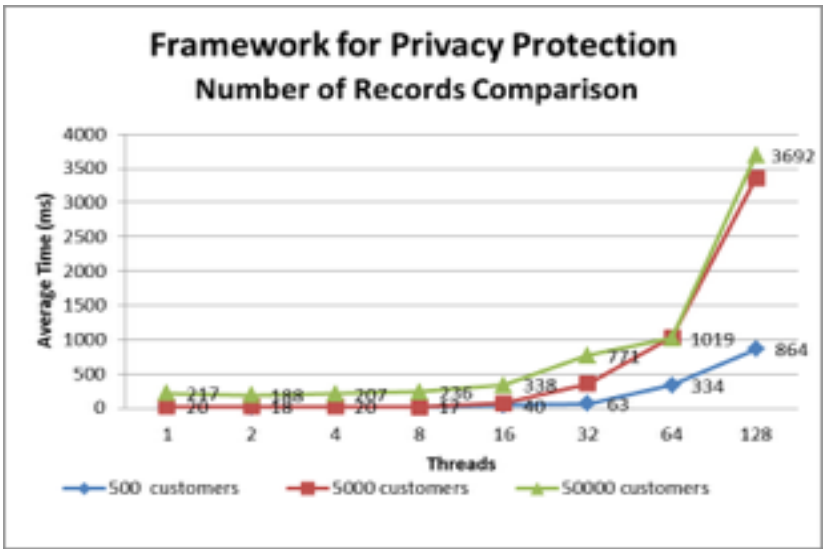

Figure 4. Average time response for different amounts of records implementing the database framework.

In Figure 4 it is possible to observe that the number of records barely affects the system performance for smaller threads. For higher threads, the performance is affected in an expected way (the more records, the more processing cost). As the average time of experiments is in milliseconds, the small variation of values between the data representing 500, 5000 and 50000 customer records, respectively, is acceptable. The proximity of values of 5000 and 50000 records for 64 and 128 threads can be, again, due to the different amount of criticality levels, once they are generated randomly and exclude the level 1. So, the less level 1 generated the more level are recorded and the more processing time is necessary to protect the data .

5. Conclusions and Future Works

This paper proposed a practical solution for privacy policies enforcement, respecting user privacy preferences. It consists in a database framework that stores the privacy policies and the user's preferences (criticality level of protection) and implements a procedure to allow or deny the access to third-party accesses considering these preferences.

This solution has some important features such as allowing users express their privacy preferences in a very flexible way, defining preferences for each piece of

personal information (avoiding limited options like “agree or disagree”, which are still used by many applications) and a proposed interface to this fine-grained preference. Also, executing the access control directly within the database management system makes the solution more secure from attacks and malicious users. The solution is free and easy to implement and use.

We implemented the framework in a TPC-W web application and evaluated its scalability and performance using a predefined process to ensure privacy. The tests results showed that scalability is high because the high number of records did not affect significantly the performance. In terms of the performance impact, it becomes greater as the number of users (threads) increases. We believe that our PL/SQL implementation can present some performance bottlenecks and we are working on identifying and improving it. Also, the randomness of criticality levels can have some influence on the results. However, although we have this performance impact, it can be considered acceptable front of the importance of protecting privacy information, especially considering users privacy preferences in detail and giving them more flexibility while dealing with their personal information.

As future work, we intend to complement the tests, evaluating the solution in large scale applications. Also, we intend to integrate this solution in a more complex environment, where attacks are identified and the information, even under attack, is permitted or denied according to privacy policies. It helps to avoid false positive results from the attack detection tools. This is one of the areas that we are currently working on. Another important future work is to perform studies that consider, in a practical context, the user’s point of view about the usability of the solution. These studies can help improving its user-friendliness and user’s privacy choices.

Acknowledgment. This work has been partially supported by the project DEVASSES - *DEsign, Verification and VAlidation of large-scale, dynamic Service SystEmS*, funded by the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no PIRSES-GA-2013-612569. We also thank the CAPES – Brazilian Federal Agency for Support and Evaluation of Graduate Education within the Ministry of Education of Brazil – and FAPESP – São Paulo Research Foundation process n. 2013/17823–0 – for the support.

References

Agrawal, R., Kiernan, J., Srikant, R., Xu, Y. (2003). "Implementing P3P using database technology," Proceedings of the 19th International Conference on Data Engineering”, pp.595,606.

Basso, T., Antunes, N., Moraes, R., Vieira, M. (2013). " An XML-Based Policy Model for Access Control in Web Applications". In proceedings of 24th International Conference on Database and Expert Systems Applications - DEXA, pp. 274-288.

Breaux, T. D., Rao, A. (2013). "Formal analysis of privacy requirements specifications for multi-tier applications," 21st IEEE International Requirements Engineering Conference (RE), pp.14,23.

Breaux, T.D., Anton, A. I. (2005). "Deriving semantic models from privacy policies,". Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, pp.67,76 (2005).

Byun, J.-W. and Li, N. (2008). "Purpose Based Access Control for Privacy Protection in Relational Database Systems", VLDB J., vol. 17, no 4, p. 603–619.

Cranor, L., Arjula, M., Guduru, P. (2002). "Use of a P3P User Agent by early adopters," in Proceedings of 9th ACM Workshop on Privacy in the Electronic Society, Washington, DC.

Earp, J. B., Antón, A. I., Member, S., Aiman-smith, L., Stufflebeam, W. H. (2005). "Examining Internet Privacy Policies Within the Context of User Privacy Values", IEEE Trans. Eng. Manag., vol. 52, pp. 227–237.

EPAL (2014). "Enterprise Privacy Authorization Language (EPAL 1.2)". [Online]. Available: http://www.zurich.ibm.com/security/enterprise-privacy/epal/Specification/. [Accessed: 23-jan-2014].

Jmeter (2015). "Apache JMeter - Apache JMeterTM". [Online]. Available: http://jmeter.apache.org/. [Accessed: 09-jan-2015].

Kolter, J. and Pernul, G. (2009). "Generating user understandable Privacy Preferences". Proceedings of IEEE International Conference on Availability, Reliability and Security. Fukuoka, pp.299-306.

Mello, V., Basso, T., Moraes, R. (2014). "A Test Process Model to Evaluate Performance Impact of Privacy Protection Solutions". In: XV Workshop de Testes e Tolerância a Falhas (WTF 2014), 2014, Florianópolis. XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos.

Ni, Q., Trombetta, A., Bertino, E., Lobo, J. (2007). "Privacy-aware Role Based Access Control", in Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, New York, NY, USA, 2007, p. 41–50.

Oracle (2015). "Oracle | Hardware and Software, Engineered to Work Together". Available: http://www.oracle.com/index.html. Accessed: 24-jan-2015.

P3P (2013). "P3P: The Platform for Privacy Preferences". [Online]. Available: http://www.w3.org/P3P/. [Accessed: 04-set-2013].

TPC (2015). "TPC-W - Homepage". Available: http://www.tpc.org/tpcw/. Accessed: 08-jan-2015.

Vieira, M., Madeira, H. (2005). "Towards a security benchmark for database management systems", in Proceedings, of International Conference on Dependable Systems and Networks, DSN 2005. p p. 592–601.