Enhancing Utility and Privacy of Data for Software Testing

Boyang Li
College of William and Mary
Williamsburg, VA 23185
boyang@cs.wm.edu

***Abstract*—A fundamental problem in test outsourcing is how to allow a database-centric application (DCA) owner to release a smaller subset of its private data along with the application. In addition, the DCA owner needs tangible guarantees that the entities in this data are protected at a certain level of privacy, while retaining testing efficacy. We are trying to solve this problem by balancing four important dimensions: testing coverage, privacy, semantic correctness, and data minimization. We built a novel approach that enhances both utility and privacy of data for software testing using a novel combination of program analysis, clustering, and association rule mining approaches. To the best of our knowledge, there exists no prior approaches that synergistically address all of the aforementioned dimensions. We also proposed several avenues for future work based on our existing work.**

I. Problem Introduction

Currently, many organizations use DCAs to support and manage their businesses. Consider a scenario where a large organization, such as a hospital, insurance company, or bank needs to perform maintenance testing of their product. In this case they may hire a large software consulting company to do so since outsourcing these tasks maybe potentially cost-effective and lead to higher quality. One obstacle is that application and data owners can no longer easily share confidential data with outsourcing service providers because of recent data protection laws passed in several countries [7]. In addition, it is also very time-consuming to test applications that include large databases. Since the application owners always want testing data to be efficient, a trivial method can be to delete some records and modify any sensitive data in the database. However, simply removing and sanitizing data often leads to significantly worsened test coverage metrics and fewer uncovered faults, thereby reducing the quality of software applications [6]. For instance, if values of the attribute `Nationality` are replaced with the generic value "`Human`," DCAs may execute some paths that result in exceptions or miss certain paths [6].

Therefore, a fundamental problem in test outsourcing is how to allow a DCA owner to release a smaller subset of the private data assuring that the entities in this data (e.g., people, organizations) are protected at a certain level, while retaining testing efficacy. More specifically, we focus on balancing the following four dimensions:

- **Testing coverage**. Statement and branch coverage are significant metrics to measure the testing quality. Higher testing coverage usually reveals more bugs in the code.
- **Privacy**. For testing databases, our goal is to limit the information that can be inferred by attackers.
- **Semantic correctness**. The generated (or synthetic) data should semantically match the original data. For example, there should be no entries in the database in which a male patient suffers from gestational diabetes.
- **Data minimization**. Fewer representative records in the database vs. complete set of records would increase testing efficiency.

Although the problem is important for software testing, there is a limited number of papers currently published on this topic. Two main reasons may account for this. First, researchers started to realize the importance of electronic information privacy in the recent years. Meanwhile, many laws were introduced that force companies to protect clients' sensitive information [7]. Second, it is only in the past decade that outsourcing is becoming mainstream. Clearly, research at the intersection of software engineering and data privacy is a new but growing research area.

II. Previous Work

To address this issue, we presented a novel approach for *Protecting and mInimizing databases for Software TestIng taSks (PISTIS)* , which can both sanitize and minimize a database that comes along with an application [9]. PISTIS uses a weight-based data clustering algorithm that partitions data in the database using information obtained via program analysis. The weight of an attribute can indicate how this data is used by the application. The approach also computes a centroid object for each cluster, which represents different persons or entities in the cluster. Intuitively, the centroid object is the most representative node in the cluster. In addition, we use associative rule mining to compute constraints, and then use these constraints to ensure that the centroid objects are representative of the general population of the data.

A. Architecture and Process

Due to space limits, we do not present all the technical details here. The architecture of PISTIS is shown in Figure 1. The inputs to PISTIS are application's source code and DB_0 which is the original database.

PISTIS first performs control- and data- flow analyses `(2)` using the Soot toolkit[1] in order to establish how the DCA uses values of different database attributes. The ranks of attributes `(3)` reflect how much these attributes impact data and control flow in a given program. Ideally, higher rank means that the attribute affects more statements in DCAs.

For the clustering algorithm component, the inputs are DB_0, ranks of attributes, and the number of clusters k. First,

[1]http://www.sable.mcgill.ca/soot

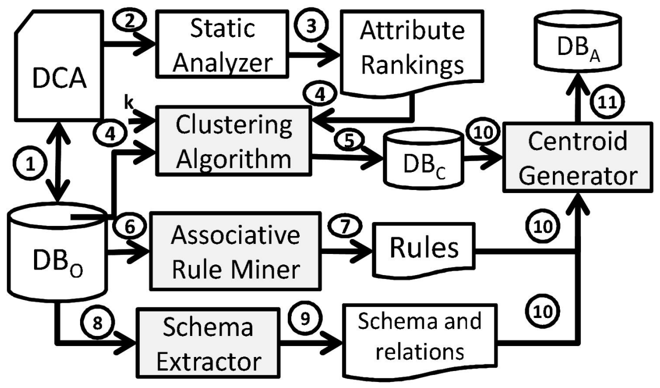

Fig. 1: The architecture of PISTIS

PISTIS transforms the data, so that distance can be computed between different types of data. For numerical attributes, we normalize the value dividing by the maximum value of the attribute in the database. For categorical attributes, the situation is different, since it is not straightforward to define the order on the values of these attributes. Therefore, we follow a general approach to assume that every distinct value for a given attribute is equally different from one another [2]. We represent each distinct value either as one if it is present for a given row, or zero otherwise. In the next step, PISTIS applies weighted k-means clustering to group a normalized DB$_0$ based on the ranks. Once clustering is done, the clustered original database DB$_C$ is outputted `(5)`. Also, PISTIS computes the centroid records by computing the average of the values for each column of each cluster. Once it is done, it maps the resulting values back to specific distinct values and generates a new dataset.

PISTIS also uses `(6)` an associative rule mining algorithm, Apriori [1], to generate association rules `(7)`, which describe semantic constraints that are obtained from data in DB$_0$. On the other hand, PISTIS extracts the database schema and its constraints from DB$_0$ using metadata services `(8)` `(9)`. The association rules and constraints can be used to help in correcting the new dataset that is generated `(10)`. For example, assume that we obtain an association rule "hysterectomy→female" and a record that indicates a "male" with "hysterectomy", we can correct the record to "female" by applying the rule. The final output of PISTIS is a sanitized and minimized database with semantically correct data records `(11)`.

B. Experimental Evaluation

We evaluated the proposed approach on two open-source Java applications, `RiskIt` and `DurboDax` [6]. We evaluated statement coverage and branch coverage over different numbers of clusters k, which we varied between 10 to 2000. We show that a reduction in statement coverage of no more than 25%, while minimizing the size of the database by more than an order of magnitude. In addition, we also evaluated the disclosure rate of our approach, which indicates less information leakage. More detailed results are presented in our ICST'14 paper [9].

III. Research Plans

For future work, we are planning on attacking the following problems: i) We only clustered 4000 random records in each database. The time is limited by the number of records and the number of attributes. It would be interesting to create a faster algorithm. Alternatively, a weight threshold could be utilized to determine the attributes for computation, since currently all of the attributes are weighted. ii) We evaluated the disclosure rate by computing average similarity between the generated data and the original data. It would be interesting if we could refine the disclosure rate definition. More specifically, defining new privacy metrics requires further research. iii) We weighted attributes based on their impact on the number of statements. The attributes also can be weighted by other factors, such as impact on the number of branches or hybrid factors. iv) We used association rules to correct data. In the experiment, one example of the rule we obtained was GQTYPE=(-inf-0.5]→ FARM='All'. It would be interesting if we can generate more useful rules, however, we understand that this depends on the underlying databases. v) In the previous experiments, we used k-means algorithm for clustering. An interesting investigation would be to compare the results of k-means algorithm with k-medians algorithm.

Acknowledgment

I would like to thank my advisor Denys Poshyvanyk for his tremendous help and support. I also thank my collaborator Mark Grechanik from University of Illinois at Chicago for his inspiration and help.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.
[2] J.-W. Byun, A. Kamra, E. Bertino, and N. Li. Efficient *k* -anonymization using clustering techniques. In *DASFAA*, pages 188–200, 2007.
[3] M. Castro, M. Costa, and J.-P. Martin. Better bug reporting with better privacy. In *ASPLOS*, pages 319–328, 2008.
[4] J. Clause and A. Orso. Penumbra: Automatically identifying failure-relevant inputs using dynamic tainting. In *ISSTA*, pages 249–260, 2009.
[5] J. A. Clause and A. Orso. Camouflage: automated anonymization of field data. In *ICSE*, pages 21–30, 2011.
[6] M. Grechanik, C. Csallner, C. Fu, and Q. Xie. Is data privacy always good for software testing? In *ISSRE*, pages 368–377, 2010.
[7] InformationShield. International data privacy laws. 2012.
[8] G. M. Kapfhammer and M. L. Soffa. A family of test adequacy criteria for database-driven applications. In *ESEC/FSE*, pages 98–107, 2003.
[9] B. Li, M. Grechanik, and D. Poshyvanyk. Sanitizing and minimizing databases for software application test outsourcing. In *ICST*, 2014.
[10] S. Mizzaro. Relevance: The whole history. *JASIS*, 48(9):810–832, 1997.
[11] S. Mizzaro. How many relevances in information retrieval? *Interacting with Computers*, 10(3):303–320, 1998.
[12] D. Peters and D. L. Parnas. Generating a test oracle from program documentation: work in progress. In *ISSTA*, pages 58–65, 1994.
[13] F. Peters and T. Menzies. Privacy and utility for defect prediction: Experiments with morph. In *ICSE*, pages 189–199, 2012.
[14] Y. Rachlin, K. Probst, and R. Ghani. Maximizing privacy under data distortion constraints in noise perturbation methods. In *PinKDD*, pages 92–110, 2008.
[15] D. J. Richardson. Taos: Testing with analysis and oracle support. In *ISSTA '94*, pages 138–153, New York, NY, USA, 1994. ACM Press.
[16] D. J. Richardson, S. Leif-Aha, and T. O. O'Malley. Specification-based Test Oracles for Reactive Systems. In *14th ICSE*, pages 105–118, 1992.
[17] K. Taneja, M. Grechanik, R. Ghani, and T. Xie. Testing software in age of data privacy: a balancing act. In *SIGSOFT FSE*, 2011.
[18] K. Taneja, Y. Zhang, and T. Xie. MODA: Automated test generation for database applications via mock objects. In *ASE*, 2010.
[19] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.
[20] S. Yoo and M. Harman. Regression testing minimisation, selection and prioritisation: A survey. *STVR*, 2011.