Trust Obstacle Mitigation for Database Systems

Victor Page[1], Robin Laney[2], Maurice Dixon[1], and Charles Haley[2]

[1] Department of Computing,
Communications Technology and Mathematics,
London Metropolitan University, 31 Jewry Street, London, EC3N 2EY
{Vic.Page, M.Dixon}@Londonmet.ac.uk
[2] Department of Computing
The Open University, Walton Hall, Milton Keynes, MK7 6AA
{R.C.Laney, C.B.Haley}@Open.ac.uk

Abstract. This paper introduces the Trust Obstacle Mitigation Model (TOMM), which uses the concept of trust assumptions to derive security obstacles, and the concept of misuse cases to model obstacles. The TOMM allows a development team to anticipate malicious behaviour with respect to the operational database application and to document a priori how this malicious behaviour should be mitigated.

1 Introduction

The Lowell Database Research Self-Assessment [1] discusses "trustworthy systems that safely store data, protect it from unauthorized disclosure, protect it from loss, and make it always available to authorized users". It also suggests that "the information management community should play a central role in addressing these needs and enhancing DBMSs with mechanisms to support these capabilities".

This work reports progress on addressing these needs at the application level without the need to enhance the DBMS. To do this we have derived a model, the Trust Obstacle Mitigation Model (TOMM), for analysing the detection and mitigation of security obstacles within a database system. An obstacle is something that, should it occur, will invalidate a trust assumption and result in a deviation between a use case and the realisation of the use case in the operational system. Security obstacles are caused by malicious agents, external to the system, that might destroy, reveal, modify, or block information assets. This is in line with the security requirements of confidentiality, integrity, availability, and authentication as presented in [2] and given the acronym CIAA.

The TOMM draws on three existing concepts, obstacle analysis [3, 4, 5], trust assumptions [6] and misuse cases [7, 8]. Trust assumptions are used to derive obstacles and misuse cases are used to provide a diagrammatic and textual representation of an obstacle. The contribution of this work is the bringing together of these concepts coupled with a 'traffic light' approach to ranking obstacles and their consequences.

The paper is structured as follows. Section 2 provides an overview of the TOMM. Finally conclusions are presented in Section 3 along with future work.

D. Bell and J. Hong (Eds.): BNCOD 2006, LNCS 4042, pp. 254 – 257, 2006.

2 The Trust Obstacle Mitigation Model

Fig. 1 presents an activity diagram for the TOMM. The swim lanes show the three phases of the TOMM Elicit Use Cases, Derive Obstacles, and Derive Mitigations. The swim lanes are present to show that each phase could be carried out in a different facilitated workshop.

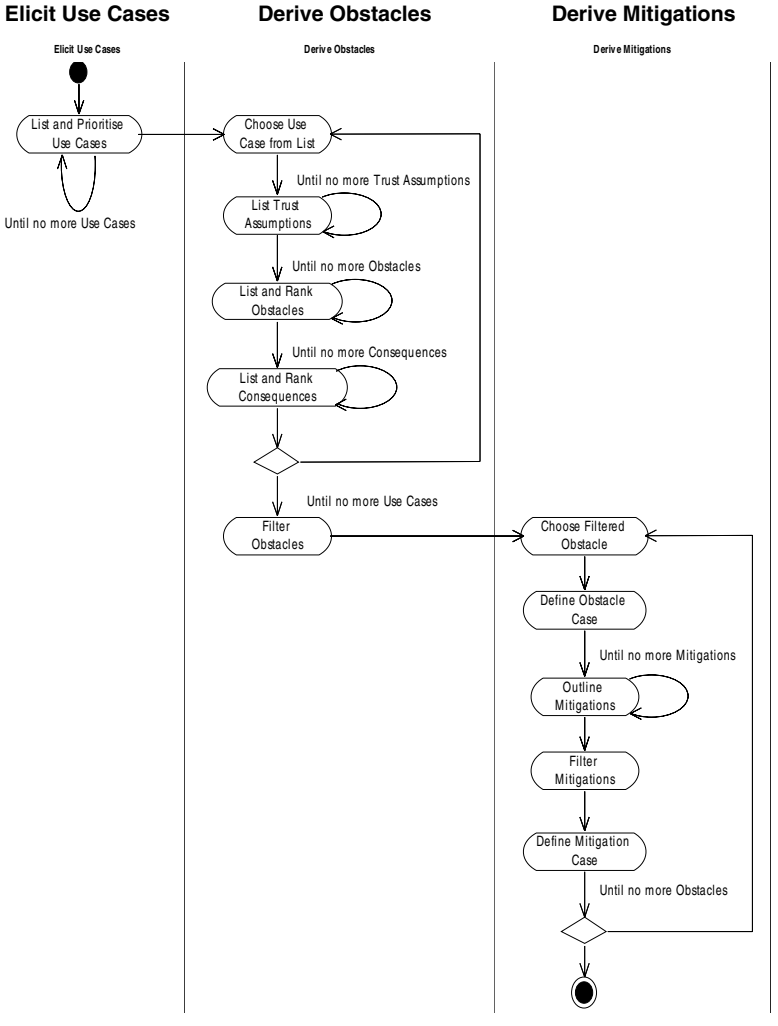

Fig. 1. Activity diagram of the TOMM

The above is not a strictly sequential process and there is an implied iteration in some of the activities. Specifically the development team may wish to re-prioritise the use case list either at the end of the Derive Obstacles phase, or at the end of the Derive Mitigations phase.

The phases and activities of the TOMM allow a development team to anticipate malicious behaviour with respect to the operational system and document a priori how this malicious behaviour should be mitigated. This is achieved by requiring the development team to reason about how the trust assumptions on which the database system will be built could be undermined.

The following lists alphabetically the definitions of the concepts and terminology that are used within the TOMM:

- **Consequence:** A consequence defines what would happen to the operational system if a trust assumption is invalidated due to the manifestation of an obstacle. Consequences are given a RAG code that signifies their severity to the operational system.
- **Filter:** Filter means to choose based on available data. For obstacles we choose those obstacles that need to be mitigated, based on the RAG codes. For mitigations we choose the most effective mitigation based on an overview of the mitigation along with its estimated cost and duration.
- **Mitigation:** A mitigation is something, that should it be implemented, will counter the effect of an obstacle.
- **Mitigation Case:** A Use Case that shows what should be done to counter the effect of malicious behaviour on the operational system.
- **Obstacle:** An obstacle is something that, should it occur, will invalidate a trust assumption and result in a deviation between a use case and the realisation of the use case in the operational system. Obstacles are given a RAG code that signifies the likelihood of them occurring.
- **Obstacle Case:** A Use Case that shows the effect of malicious behaviour on the operational system. The main function of the obstacle case is to decide and document a priori how the operational system would react to malicious use. This is based on the concept of a Misuse Case.
- **RAG Code:** RAG codes form an intuitive 'traffic light' approach to ranking obstacles and their consequences. RAG is an acronym for Red, Amber, and Green. The RAG codes are classified as follows;
  - **R:** signifies either a high likelihood of an obstacle occurring or a fatal consequence should an obstacle occur.
  - **A:** signifies either a medium likelihood of an obstacle occurring or a non-fatal consequence should an obstacle occur.
  - **G:** signifies either little likelihood of an obstacle occurring or low negative consequence should an obstacle occur.
- **Security Obstacle:** An obstacle that is caused by the malicious behaviour of an external agent (human or machine).
- **Trust Assumption:** Documents the way in which a use case, when realised in the operational system, can be trusted to have certain stated properties and/or behaviour.
- **Use Case:** A representation by diagram and text of a sub-set of the database system functionality.

3 Conclusions and Future Work

Use Case diagrams are at the heart of the Trust Obstacle Mitigation Model. They provide a simple yet powerful diagrammatic representation of database system requirements at a level of granularity appropriate for reasoning about security obstacles in facilitated workshops. The TOMM is visually intuitive and it can be adopted by projects where use case modeling would normally be applied. Also the TOMM provides an intuitively direct approach to ranking obstacles and their consequences – via the use of RAG codes.

The TOMM can be improved by deriving taxonomies of trust assumptions, obstacles, consequences and mitigations, along with heuristics to support their use. We have incorporated trust assumptions in the TOMM, which are assumptions by the development team that a requirement, when realised in the operational system, will cause that system to have certain stated properties and/or behaviour [6]. This suggests that the obstacles caused by this trust being misplaced can be classed as trust obstacles. The incorporation of the formal semantics described in UMLSec [9] will provably show that the model is consistent, correct and optimal for its purposes. Future work will focus on these improvements. A tool will also be developed to support the model.

References

1. Abiteboul, S., Agrawal, R., Bernstein, P., et al: The Lowell Database Research Self-Assessment. Communications of the ACM, Vol. 48. No. 5. (2005) 111-118
2. Stallings, W.: Business Data Communications 5th edn. Pearson Prentice Hall, Upper Saddle River USA (2005)
3. Page, V., Dixon, M., and Bielkowicz, P.: Object-Oriented Graceful Evolution Monitors: In: Konstantas, D., Leonard, M. and Patel, S. (eds): Proceedings of the Ninth International Conference on Object-Oriented Information Systems, University of Geneva Geneva Switzerland (2003) 46-59
4. Anton, A.: Goal Identification and Refinement in the Specification of Software-Based Information Systems. Ph.D. Thesis, College of Computing Georgia Institute of Technology (1997)
5. Lamsweerde, A., Letier, E.: Integrating Obstacles in Goal-Driven Requirements Engineering. ICSE'98 – 20th International Conference on Software Engineering, Kyoto Japan (1998) 53-62
6. Haley, C., Laney, R., Moffett, J., Nuseibeh, B.: (2004), The Effect of Trust Assumptions on the Elaboration of Security Requirements. Proceedings of the 12th International Requirements Engineering Conference (RE'04), Kyoto Japan (2004) 102-111
7. Alexander, I.: Misuse Cases: Use Cases with Hostile Intent. IEEE Software, Vol. 20, Issue 1. (2003) 58-66
8. Sindre, G., and Opdahl, A.: Eliciting Security Requirements by Misuse Cases. Proceedings of the 37th International Conference on Technology Object-Oriented Languages and Systems, Sydney Australia (2000) 120-131
9. Jürjens, J.: UMLsec: Extending UML for Secure Systems Development. In Jézéquel, J., Hussmann, H. and Cook, S. (eds): Proceedings of the 5th International Conference on The Unified Modeling Language, Dresden Germany (2002) 412-425