Software for interactive secure systems design: lessons learned developing and applying CAIRIS.

FAILY, S. and FLÉCHAIS, I.

2012

Software for Interactive Secure Systems Design: Lessons Learned Developing and Applying CAIRIS

Shamal Faily
Department of Computer Science
University of Oxford
shamal.faily@cs.ox.ac.uk

Ivan Fléchais
Department of Computer Science
University of Oxford
ivan.flechais@cs.ox.ac.uk

As systems become more complex, the potential for security vulnerabilities being introduced increases. If we are to provide assurances about systems we design then we need the means of analysing, managing, and generally making sense of the data that contributes to the design. Unfortunately, despite ongoing research into tools for supporting secure software development, there are few examples of how tools can be used to help build and support design models associated with security and usability. This paper summarises some of our experiences developing and applying CAIRIS: a requirements management tool for usable and secure system design. We describe our motivation for building CAIRIS, summarise how it was built and evaluated, and present our experiences applying it to real world case studies.

CAIRIS,Requirements Management,Security,Usability

1. INTRODUCTION

As systems become more complex, the potential for security vulnerabilities being introduced increases. This means that if we are to provide any assurances about systems that we design then we need some means for analysing, managing, and generally making sense of all the data that contributes to a system's design to ensure such vulnerabilities are not unintentionally introduced. While there has been ongoing research into software tools to support the development of secure software, there has been comparatively little work on tools for reasoning about security and usability models. Without this ability, it is difficult to predict the usability implications of security design decisions and vice-versa, the security implications of usability decisions; this becomes particularly difficult when considering how these implications might change in different contexts of use.

To help understand how software tools can support security and usability design techniques, we developed CAIRIS (Computer Aided Integration of Requirements and Information Security): a requirements management tool for secure and usable system design. Since developing the initial prototype in 2009, we have evolved CAIRIS based on our experiences in several real-world case studies. In this paper, we reflect on some of these experiences building and applying CAIRIS. In Section 2, we briefly describe some of the challenges that motivated our approach to designing CAIRIS, before summarising how the tool has been developed and evaluated in Section 3. In Section 4, we discuss some of our experiences and problems faced in using and maintaining CAIRIS.

2. RELATED WORK

Because *requirements* are a recognised boundary object across security, usability, and software engineering models, requirements management tools have been proposed as a basis for supporting security and usability design activities. Their potential for extensibility is illustrated by the DOORS requirements management tool (IBM 2010) which, with the aid of its DXL scripting language, supports extensions for specifying positive and negative scenarios of user behaviour(Alexander 2002). However, the lack of distinct semantics for the underlying concepts associated with these techniques means that analysts need to manually maintain links between requirements and non-requirements artifacts.

By structuring the data being managed according to a specific meta-model, model-based approaches address this traceability management problem. However, modelling languages and tools tend to consider requirements only as a notational concept. For example, while UMLSec—a UML profile for secure system development(Jürjens 2005)—supports the concept of a *security requirement*, this is depicted only as a UML stereotype. Rather than being concerned with how these might be analysed, the notation is concerned with the requirement's deployment rather than its specification.

Further problems arise when trying to integrate tools grounded in similar, but subtly different, conceptual models. For example, if we assert that a misuse case *threatens* a use case, do we agree what it means for the use case to be threatened? Does the misuse case threaten the work carried out by the use case, or the assets associated with it? Houmb et. al (Houmb et al. 2010) faced some of these problems when integrating tools based on different techniques. Their experiences indicate that while building heuristics into tools to help with integration is useful, these alone can't replace the expertise needed to apply the techniques themselves. Consequently, while integrating tools and concepts can help verify requirements, few tools provide support for eliciting or validating them.

3. BUILDING AND EVALUATING CAIRIS

Based both on the IRIS meta-model (Faily and Fléchais 2010)—which characterised our ideas about how concepts from requirements, security, and usability engineering might interrelate—and lessons learned from existing tools, we designed and developed CAIRIS to appeal to the following design principles:

- Familiarity: The tool itself should not add to pre-existing cognitive burdens; given the difficulty associated with grasping new concepts and learning new notations, the tool and its artifacts should require no more cognitive overhead than learning how to use the techniques associated with IRIS meta-model.
- Extensibility: Because the tool was to be used in several case studies, new insights might arise from its use; this could include identifying unnecessary functions, or the need for new functionality. Moreover, it should be possible to quickly modify the tool to implement the suggested changes and assess their impact during, or shortly after, an intervention.
- Standardisation: As well as structuring the collected data, we wanted the tool to be used to support a variety of existing analysis techniques. Crucially, we wanted to ensure that the tool supported each without changing standard concepts or the manner in which each technique normally operated. This meant that different people might use the tool to support different techniques, according to their expertise and responsibilities. Consequently, given that the meta-model allowed data collected through one technique to inform another, traceability between model concepts needed to be automatic.

We developed CAIRIS using a prototyping approach, over five iterations.

In the first iteration, CAIRIS was developed in parallel with the IRIS meta-model. The tool was used to elicit data using contemporary examples where multiple contexts of use were evident. One of these examples involved analysing contemporary news reports and documentation about the Vélib bicycle sharing system to elicit security requirements which would not compromise the usability of Vélib. The objective of this phase was to determine whether the concepts necessary to model the different problems were reflected in the IRIS meta-model.

Based on early feedback from the Requirements Engineering community (Faily and Fléchais 2009), additional concepts were added to the IRIS meta-model and the tool was evolved to support these. As the meta-model became more elaborate, additional model views were incorporated into the tool, and the architecture was re-factored to allow scaleability should further model elements and associations need to be added.

In the second iteration, we created a specification exemplar based on the NeuroGrid e-science project (Geddes et al 2006) to validate whether the tool was capable of modelling a complete, non-trivial problem. Using CAIRIS, the resulting NeuroGrid model was validated with one of the previous project stakeholders.

The final three iterations involved applying CAIRIS in three separate case studies; these studies are described in more detail in (Faily 2011b)

CAIRIS was written primarily in Python, and used the open-source wxPython and pyGTK frameworks for windowing and visualisation support, and NumPy for matrix manipulation. MySQL was used for management and access to model data. Although CAIRIS is primarily maintained by members of the Security research group at the University of Oxford, it has been released to github as an open-source project under an Apache Software license.

4. EXPERIENCES

Space constraints mean we are unable to reflect on all of our experiences with CAIRIS. We do, however, highlight three particular experiences that, we believe, might provide useful insights to designers of future software tools for secure and usable system design.

4.1. Building on similarities rather than differences

One of the first challenges we needed to address was reconciling the several different security requirements engineering meta-models that had previously been proposed; this would be particularly challenging given the lack of consensus about what a security requirement is, e.g. (Tøndel et al. 2008). Rather than trying to tease out lessons learned from all these models, we instead decided to select a particular model and attempt to scale this given the additional usability elements we wanted to consider. We chose the Information Systems Security Risk Management (ISSRM) modelling language Mayer (2009) because not only was it largely orthogonal to usability, it also attempted to align with goal-oriented requirements languages. While the concepts associated with those languages were incompatible with IRIS, we believe that on-going research in both CAIRIS and their languages might lead to synergies in the future.

Our approach came at the cost of simplifying the risk analysis approach adopted by ISSRM. This ruled out the ability for modelling sophisticated risk analysis strategies, such as blended threats where an attacker might exploit two seemingly innocuous vulnerabilities to achieve catastrophic results. Nonetheless, by keeping the IRIS meta-model as simple as possible without compromising any of the fundamental concepts, we were able to progressively incorporate concepts over time.

For example, based on what was originally an unconnected research finding about the usefulness of argumentation models for supporting persona development Faily and Fléchais (2011), we were eventually able to align concepts from IRIS with what were incompatible goal-oriented requirements techniques; this is described in more detail in Faily (2011a). We believe that the success of this approach relied not on seeing how social-goal models could be built into CAIRIS, but on how CAIRIS could add value to complementary tools which are better suited to analysing these in more detail.

4.2. Specifying contexts of use

From the outset, the concept of an *environment* was supported in CAIRIS; this was introduced to make it possible to specify and reason about the elements of several different contexts of use for a given system. However, based on the results of our case studies, the results of trying to specify formal contexts of use were mixed.

While environments could be of any type, those of most value tended to be social or cultural rather than physical. When considering one example based on the NeuroGrid exemplar, we noticed that security properties associated with certain assets had markedly different security properties in different environments. In the three case studies, the security properties varied less, and the environments were used to compartmentalise the analysis of activities according to the context of most relevance. Occasionally, however, some discussion arose by comparing the same tasks carried out in different social contexts, or discussing how the tasks carried out in one environment had an impact in others.

We were also interested in how variations of context could be composed to introduce new contexts of use. Individually, such environments might be innocuous but, when combined, new phenomena might be observed that might not otherwise be seen in the separate models. To investigate this, CAIRIS allowed composition of an environment based on one or more other environments. In practice, however, this did not prove to be very useful. Although the environments modelled in the case study examples were non-trivial, they were also distinct enough that combining them added little value to the analysis carried out. However, reasoning about dependencies between two environments was occasionally useful when considering how attackers might exploit knowledge about one environmental context to cause a shift from one environment to another, or how a risk in one context lead to a subsequent risk being introduced in another.

4.3. Using CAIRIS

CAIRIS was not designed to be a general purpose tool. One of the initial motivations for building the tool was to support participative workshops where stakeholders could discuss different models, and examine the impact of model changes in real time. However, when CAIRIS was used for supporting design activities for the EU FP 7 webinos project, we faced two new challenges. First, many of the CAIRIS users didn't have the necessary technical background to install and setup CAIRIS. Second, the users were only knowledgeable in some of the capabilities of CAIRIS. As a result, of the users that did install and use CAIRIS, all used CAIRIS for little more than a tool for specifying and managing personas.

Because there was little time available on the project to run training sessions, we used the project wiki to capture structured data that could then be imported into CAIRIS. We created structured page templates for design artifacts like scenarios, use cases, and personas. We then created scripts that could be used to convert this content into compatible XML models that could be imported into CAIRIS. In addition to this, we provided guidance and support for the rest of the project on the use of the templates. Eventually, this approach was extended to security and requirements models as well. As a result, although the wiki was still used to browse data, most of the webinos security, usability, and requirements model was stored as text in a project git repository, with a *build* script used to create a consolidated CAIRIS model on demand. More details about this process can be found in Faily et al. (2012).

5. CONCLUSION

In this paper, we described some of our experiences in designing, building, and using the CAIRIS requirements management tool. In providing our candid experiences with CAIRIS, we aim to begin filling the hitherto unnoticed gap in the literature on tools for secure and usable system design.

While the tool was designed to support only a single researcher (the main author), the CAIRIS user community is slowly beginning to grow. Following the introduction of the tool to the Oxford Software Engineering Programme's *Design for Security* course, practitioners are beginning to use CAIRIS to address their own security design challenges. As industrial take-up grows, we plan to evaluate how well CAIRIS, and software tools in general, tackle the security usability design problems practitioners currently face.

6. ACKNOWLEDGEMENTS

The research described in this paper was funded by EPSRC CASE Studentship R07437/CN001, and the EU FP7 webinos project (FP7-ICT-2009-05 Objective 1.2).

REFERENCES

Alexander, I. (2002). Initial industrial experience of misuse cases in trade-off analysis. In *Proceedings of the IEEE International Requirements Engineering Conference*, pages 61–68. IEEE Computer Society.

Faily, S. (2011a). Bridging User-Centered Design and Requirements Engineering with GRL and Persona Cases. In *Proceedings of the 5th International i* Workshop*, pages 114–119. CEUR Workshop Proceedings.

Faily, S. (2011b). *A framework for usable and secure system design*. PhD thesis, University of Oxford.

Faily, S. and Fléchais, I. (2009). Context-Sensitive Requirements and Risk Management with IRIS. In *Proceedings of the 17th IEEE International Requirements Engineering Conference*, pages 379–380. IEEE Computer Society.

Faily, S. and Fléchais, I. (2010). A Meta-Model for Usable Secure Requirements Engineering. In *Proceedings of the 6th International Workshop on Software Engineering for Secure Systems*, pages 126–135. IEEE Computer Society.

Faily, S. and Fléchais, I. (2011). Persona cases: a technique for grounding personas. In *Proceedings of the 29th international conference on Human factors in computing systems*, pages 2267–2270. ACM.

Faily, S., Lyle, J., Paul, A., Atzeni, A., Blomme, D., Desruelle, H., and Bangalore, K. (2012). Requirements sensemaking using concept maps. In *Proceedings of the 4th International Conference on Human-Centered Software Engineering*. Springer. To Appear.

Geddes et al (2006). The challenges of developing a collaborative data and compute grid for neurosciences. *Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on*, pages 81–86.

Houmb, S. H., Islam, S., Knauss, E., Jürjens, J., and Schneider, K. (2010). Eliciting security requirements and tracing them to design: an integration of Common Criteria, heuristics, and UMLsec. *Requirements Engineering*, 15(1):63–93.

IBM (2010). IBM Rational DOORS.

Jürjens, J. (2005). *Secure systems development with UML*. Springer, Berlin.

Mayer, N. (2009). *Model-based Management of Information System Security Risk*. PhD thesis, University of Namur.

Tøndel, I. A., Jaatun, M. G., and Meland, P. H. (2008). Security requirements for the rest of us: A survey. *IEEE Software*, 25(1):20–27.