Context-Sensitive Requirements and Risk Analysis

Shamal Faily
Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD, UK
shamal.faily@comlab.ox.ac.uk
http://web.comlab.ox.ac.uk/people/Shamal.Faily

Abstract

When a system's context of use changes, the security impact may be felt in other contexts. Risks mitigated for one operational context may continue to pose a danger in others due to contextual differences in assets, threats and vulnerabilities. The research presented will identify security factors in these contexts of use, and describe their application to secure systems design. The contributions of this research lead to conceptual models, processes and tool-support which better situate requirements and risk analysis for different contexts of use.

1 Introduction and Motivation

Contexts of use [1] are constantly evolving: people, tasks, and artifacts may alter in line with changes to their related environments. A myriad of possible physical, cultural, or temporal environments exist; instances of a system may be situated in different countries, or be used by operatives with different cultural backgrounds. Even if a system appears to be static, the temporal environment influences the overall context. For example, consider ESA's Rosetta spacecraft [12]. At first glance, it appears this system will remain unchanged for its 11 year life-span, as it travels along a complex trajectory in space before its rendezvous with Comet 67P/Churyumov-Gerasimenko. However, the passage of time influences the ground control team, the tasks they perform, together with Rosetta's onboard software components. The corollary of contextual change impacts both security and usability. From a security perspective, the existence, value and severity of system assets, threats, and vulnerabilities in one context may vary in others. From a usability perspective, the properties of goals, the tasks achieving them, and the people carrying them out, may equally vary between contexts.

Contexts of use are socio-technical systems; these are systems of technology used within a human activity system [5]. Techniques for eliciting requirements from socio-technical systems exist, but the design of secure systems involves more than simply eliciting and specifying requirements. Not meeting a functional requirement leads to missing functionality, but not meeting a security requirement may introduce exploitable vulnerabilities. Moreover, trading-off different qualities can quickly become unmanageable as the emergent system grows, and possible contexts of use are enumerated. Secure Software Engineering, the branch of research investigating the integration of security concerns into Software Engineering practices [24], can help defeat the complexity arising from the design of secure systems. To date, however, this research has been biased towards the technical aspects of the socio-technical balance, thereby neglecting the analysis of people, their tasks, and related environments. This balance can be redressed by integrating techniques from User-Centered Design (UCD), which is characterised by an early focus on users and tasks, empirical measurement, and iterative design [17].

UCD is predominantly an empirical discipline, with little support available for conceptual models and tool support. Recent work has argued for a better understanding of how contexts of use impact security, and the process of designing secure systems [13]. This understanding can inform the design of models and tools, leading to more situated approaches to secure systems design.

2 Research question and Contributions

The issues raised in the previous section motivate the following research question:

What factors relating to context of use impact security, and how can these be applied to secure systems design?

Given the multi-faceted aspects of both socio-technical con-

[1] By 'context of use', we mean the users, tasks, equipment, and the environments where a system is used [1]

texts and secure systems design, we choose to break this research question into the following three smaller questions.

- RQ1: What key concepts and values can be drawn from a socio-technical context of security to inform the design of secure systems?
- RQ2: How can introducing context integrate UCD and Secure Software Engineering?
- RQ3: How can secure systems design approaches be augmented with better support for requirements elicitation practices?

These questions aim to investigate the context of security (RQ1), and explore how these can be better integrated into conceptual models and process associated with secure systems design (RQ2 and RQ3).

Answering RQ1 contributes a theoretical model of the cultural context of security, together with key concepts and values informing it. The novelty of this model is its grounding in both the literature and empirical data; this empirical data was derived from a recently completed e-Science project. e-Science not only exemplifies issues found in large, distributed and heterogeneous systems, their de-centralised nature allows a model to be grounded in data not usually found in the literature. Insights from this model can be used to inform research into the design of security, which is sensitive to the influence of security culture.

Answering RQ2 contributes a meta-model for integrating UCD and Secure Software Engineering. The elements of context are woven into this model, together with concepts from Requirements Engineering which help bridge these disciplines. This contribution is practical, as it is validated by a software prototype building on this model, and supporting the analysis of usability, requirements, and risk analysis artifacts; this analysis includes context-specific quantitative usability and risk analysis, and the visualisation of different contexts of use. We are unaware of any conceptual models or tool-support which integrates concepts from UCD and Secure Software Engineering.

Answering RQ3 contributes a UCD process, incorporating requirements elicitation techniques to capture the contributions of RQ2. Empirical data from this methodological contribution will also be analysed to inform the answer of the high-level research question.

3 State of the art

The least explored elements of *context* described in [1] are the environments underpinning the other elements, specifically the social and cultural environments; these environments need to be understood before the socio-technical balance described in section 1 can be redressed. These particular environments are often packaged under the moniker of a *security culture*, however previous research has focused on its consequences, rather than its intrinsic nature, e.g. security culture as a synonym for security awareness [2]. Work examining its intrinsic nature has done so through the lens of organisational culture during day-to-day use of secure systems within individual organisation, e.g. [21]; we are unaware of previous work drawing insights from security culture based on the design of socially and technically heterogeneous systems. Like security engineering, which has inherited ideas from safety engineering, the safety culture literature suggests insights which may be relevant to security. However, while models exist on the intrinsic nature of safety culture [7], no such model exists for security culture.

We are unaware of previous work which tries to use context as a means of bridging HCI and Secure Software Engineering, but attempts to bridge this gap have been made by each of these disciplines.

The HCI community universally agree that design should be user-centered. Zurko's work on User-Centered Security [30] served as a crucible for subsequent research into HCI Security design, eventually informing the design of AEGIS (Appropriate and Effective Guidance in Information Security) by Fléchais et al [14]. AEGIS takes a participative approach to handling security and usability concerns during risk analysis. However, although AEGIS mandates the elicitation of requirements, workshop participation is the primary means of eliciting empirical data. Reasoning about security for contexts of use requires richer data about users and their tasks; this cannot be easily elicited in a workshop setting. For this reason, user-centered security approaches need to be supplemented by other techniques. Some members of the HCI community argue that supplemental techniques can be borrowed from Computer Science; these include information theory, graph theory, and finite-state automata [27]. Formal methods have also been proposed as a means of precisely expressing contributions from Psychology, Sociology, and Software Engineering [19]. To date, however, the application of such techniques have been limited, and the promise of formal methods as a vehicle for communication between disciplines has yet to be fulfilled.

The Software Engineering community has traditionally considered 'Usability' and 'User Interface' as synonymous. Rafla et al [25] suggest that architectural patterns like Model-View-Controller help sustain the assumption that usability design can be decoupled from architectural design. Ferre et al [11] also acknowledge the disconnect between Software Engineering and HCI, and have proposed Requirements Engineering techniques to help bridge this gap. Requirements Engineering is a prominent contributor to contemporary Secure Software Engineering research; this is

illustrated by the work of Haley and his colleagues at the Open University [18], and proponents of Goal-Oriented approaches [29, 16]. However, these approaches treat usability as another class of non-functional requirement, and fail to recognise the unresolved methodological issues integrating usability requirements with other requirement types, such as security [23].

Like Ferre, Diaper [9] recognises the need to bridge the gap between HCI and Software Engineering. Diaper states that two elements are needed to successfully integrate HCI with Software Engineering:

- Tool-support.
- Output representations compatible with Software Engineering.

For tool-support to be forthcoming, a conceptual model is needed to link HCI and Software Engineering artifacts. Although meta-models have been proposed for dealing with security requirements engineering concerns, e.g. [22], these models lack support for certain constituent elements of context of use, such as tasks and environments.

As an output representation, requirements have the potential to be a common language between the disciplines. However, a recent survey of approaches for Security Requirements Engineering [28] concluded that approaches for security requirements elicitation were not prescriptive enough to be usable by software developers. Engineers frequently confuse security requirements with other developmental artefacts due to a lack of clarity about what a security requirement is. In their roadmap paper, Cheng & Atlee [6] note the lack of consensus about whether *Security Requirements* should be realised during analysis, or detailed at design time together with other competing non-functional requirements. The critical nature of secure systems design means that concepts related to requirements and security must be clear and unambiguous.

4 Research methods

An analytical induction approach was taken to draw concepts and values from cultural contexts of security for RQ1. Grounded Theory [8] was selected as the methodology for this approach, as this prescribes procedures for generating theory from observed real-world phenomena. Two sets of data were analysed: the first was a selection of literature on safety and security culture; the second was a set of transcripts from interviews held with participants of the NeuroGrid e-Science project [15]. The methodology consisted of building models for each data sample, before comparing and consolidating both into a single model of security culture. The literature based model of security culture was grounded in a sample of 21 peer-reviewed papers from research literature. The empirical model was grounded in qualitative interview transcripts, amounting to approximately 500 minutes of transcribed data.

In answering RQ2, the IRIS (Integrating Requirements and Information Security) framework was devised. The framework consists of a meta-model integrating the notion of context with concepts from Requirements and Risk Management, together with a software prototype implementing it. The process involved constructing an initial, non contextualised meta-model for Requirements and Risk Management based on existing work, and developing a software prototype implementing these concepts. By iteratively applying the prototype to the different contexts evident in a real-world exemplar [4], additional concepts and associations were induced, re-informing both the prototype and the meta-model.

A UCD process has been devised, which builds on previous work by the HCI and HCI Security communities. This process shall be evaluated by an interventionist methodology, specifically Action Research [3], for two evaluation cycles. The first cycle will involve a case study in industry; the evaluation phase of this cycle will inform any necessary changes to the process, the meta-model, or the software tool. The second cycle will apply the design process in a pedagogical context; this will involve teaching IRIS and the associated design process to post-graduates students attending the 'Design for Security' course at the University of Oxford.

5 Progress

The research necessary to answer RQ1 is complete, and early results of this work have been presented [10]. A Grounded Theory model of security culture was induced, together with a number of influencing concepts. Classes of role and responsibility were found to be reflective of values within different sub-cultures. The cultural context itself was also found to influence security values and perceptions towards artifacts and norms. These ideas were used to inform the meta-model design for RQ2

The main contributions from RQ2 are complete, and will be presented as a poster at RE'09. The IRIS meta-model extends existing security RE meta-models by introducing the notion of security values, roles, personas, goals, and tasks. The IRIS software prototype was developed using open-source components such as Python, MySQL, and GraphViz. The requirements managed by this tool conform to the VOLERE requirements specification template [26]. The tool supports many of the requirements stipulated by Hoffman's list of requirements for requirements management tools [20], such as versioned changes to requirements, traceability between model elements, and automatic document generation. Additionally, traceability between most

model elements is automatically maintained by the IRIS data model. Values held about different model elements, such as threat likelihood, vulnerability severity, and task usability are used to quantitively score and qualitatively rate different risks and tasks. Metrics associated with usability, risk, and requirements artifacts, together with their traceability relations, can also be visualised.

A UCD process, which will be used to answer RQ3, has recently been devised. The process is currently being evaluated using an industry case study.

References

[1] *ISO/IEC 13407: Human-Centered Design Processes for Interactive Systems*. ISO/IEC, 1999.

[2] E. Albrechtsen. A qualitative study of users' view on information security. *Computers and Security*, 26(4):276–289, 2007.

[3] R. L. Baskerville. Investigating information systems with action research. *Commun. AIS*, page 4, 1999.

[4] BBC News. Thefts puncture Paris bike scheme. `http://news.bbc.co.uk/1/hi/world/europe/7881079.stm`, 10 February 2009.

[5] P. Beynon-Davies. *Information Systems: An Introduction to Informatics in Organisations*. Palgrave, 2002.

[6] B. H. C. Cheng and J. M. Atlee. Research directions in requirements engineering. In *FOSE '07: 2007 Future of Software Engineering*, pages 285–303, Washington, DC, USA, 2007. IEEE Computer Society.

[7] M. D. Cooper. Towards a model of safety culture. *Safety Science*, 36(2):111–136, 2000.

[8] J. M. Corbin and A. L. Strauss. *Basics of qualitative research : techniques and procedures for developing grounded theory*. Sage Publications, Inc., 3rd edition, 2008.

[9] D. Diaper. Understanding Task Analysis for Human-Computer Interaction. In D. Diaper and N. A. Stanton, editors, *The Handbook of Task Analysis for Human-Computer Interaction*, pages 5–47. Lawrence Erlbaum Associates, 2004.

[10] S. Faily and I. Fléchais. Making the invisible visible: a theory of security culture for secure and usable grids. UK e-Science All Hands Conference 2008, Edinburgh, UK (Oral Presentation), 2008.

[11] X. Ferre, N. Juriste, and A. M. Moreno. Framework for integrating usability practices into the software process. volume 3547, pages 202–215, 2005.

[12] P. Ferri and G. Schwehm. Rosetta: ESA's Comet Chaser Already Making its Mark. *European Space Agency Bulletin*, (123):62–66, 2005.

[13] I. Fléchais. *Designing Secure and Usable Systems*. PhD thesis, University College London, 2005.

[14] I. Fléchais, M. A. Sasse, and S. M. V. Hailes. Bringing security home: a process for developing secure and usable systems. In *NSPW '03: Proceedings of the 2003 workshop on New security paradigms*, pages 49–57, New York, NY, USA, 2003. ACM.

[15] Geddes et al. Neurogrid: using grid technology to advance neuroscience. *Computer-Based Medical Systems, 2005. Proceedings. 18th IEEE Symposium on*, pages 570–572, June 2005.

[16] P. Giorgini, H. Mouratidis, and N. Zannone. Modelling Security and Trust with Secure Tropos. In H. Mouratidis and P. Giorgini, editors, *Integrating Security and Software Engineering*. Idea Group, 2007.

[17] J. D. Gould and C. Lewis. Designing for usability: key principles and what designers think. *Communications of the ACM*, 28(3):300–311, 1985.

[18] C. B. Haley, R. Laney, J. D. Moffett, and B. Nuseibeh. Arguing satisfaction of security requirements. In H. Mouratidis and P. Giorgini, editors, *Integrating Security and Software Engineering*, chapter 2, pages 16–43. Idea Group, 2007.

[19] M. Harrison and H. Thimbleby. The role of formal methods in human-computer interaction. In M. Harrison and H. Thimbleby, editors, *Formal Methods in Human-Computer Interaction*, pages 1–8. Cambridge University Press, 1990.

[20] M. Hoffmann, N. Kuhn, M. Weber, and M. Bittner. Requirements for requirements management tools. *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*, pages 301–308, Sept. 2004.

[21] S. Kraemer and P. Carayon. Computer and information security culture: findings from two studies. *Proceedings of the 49th Annual Meeting of the Human Factors and Ergonomics Society*, pages 1483–1487, 2005.

[22] N. Mayer. *Model-based Management of Information System Security Risk*. PhD thesis, University of Namur, 2009.

[23] D. J. Mayhew. Two Contrasting Case Studies in Integrating Business Analysis with Usability Requirements Analysis and User Interface Design. In C. Righi and J. James, editors, *User-Centered Design Stories: Real-World UCD Case Studies*. Morgan Kaufmann Publishers, 2007.

[24] H. Mouratidis and P. Giorgini. Integrating security and software engineering: An introduction. In H. Mouratidis and P. Giorgini, editors, *Integrating Security and Software Engineering*, chapter 1, pages 1 – 14. Idea Group, 2007.

[25] T. Rafla, P. N. Robillard, and M. Desmarais. A method to elicit architecturally sensitive usability requirements: Its integration into a software development process. *Software Quality Journal*, 15(2):117–133, 2007.

[26] J. Robertson and S. Robertson. Volere Requirements Specification Template: Edition 13 - August 2007. `http://www.volere.co.uk/template.pdf`.

[27] H. Thimbleby. *Press on: principles of interaction programming*. MIT Press, Cambridge, Mass., 2007.

[28] I. Tøndel, M. Jaatun, and P. Meland. Security requirements for the rest of us: A survey. *Software, IEEE*, 25(1):20–27, Jan.-Feb. 2008.

[29] E. Yu, L. Liu, and J. Mylopoulos. A social ontology for integrating security and software engineering. In *Integrating Security and Software Engineering*, chapter 4, pages 70–105. Idea Group, 2007.

[30] M. E. Zurko and R. T. Simon. User-centered security. In *NSPW '96: Proceedings of the 1996 workshop on New security paradigms*, pages 27–33, New York, NY, USA, 1996. ACM.