Secure System? Challenge Accepted: Finding and Resolving Security Failures Using Security Premortems

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

Simon Parkin
School of Computer Science
Newcastle University
s.e.parkin@ncl.ac.uk

John Lyle
Department of Computer Science
University of Oxford
john.lyle@cs.ox.ac.uk

Risk-driven approaches are dominant in secure systems design; these aim to elicit and treat vulnerabilities and the threats exploiting them. Such approaches, however, are so focused on driving risks out of system design, they fail to recognise the usefulness of failure as a vehicle for security innovation. To explore the role of failure as a design tool, we present the security premortem: a participative design technique where participants assume that a system has been exploited, and plausible reasons are given for explaining why. We describe this approach and illustrate how software tools can be used to support it.

Risk,Premortem,CAIRIS

1. INTRODUCTION

Many of the approaches associated with secure system design are driven by the elicitation and mitigation of risks. These are concerned with identifying vulnerabilities which expose *assets* of value, together with threats which exploit them. While useful concepts for both design and information security management, focusing too much on risks may draw undue attention to the symptoms of security failures, rather than their root causes. To better understand these causes we must elevate security failures to concepts worthy of analysis in their own right.

Because there are many reasons for why a system might be exploited, it has been argued that security is what social planners call a *wicked problem*. This is because we lack clarity about what it means to secure systems, tests for proving a system is secure, and a grasp of all possible solutions for satisfying a specified security problem (Faily and Fléchais 2010b). Therefore, while assurances may be given that a system's specification is secure, we can never be certain that circumstances won't arise where these assurances fail to hold. What specifying a design does do is force designers to make value judgements about what might be a *good enough* solution. Even if these judgements lead to ineffective design decisions, knowledge about failures still provide insights to designers about the nature of the problem space.

It seems nonsensical that we might want to make design decisions knowing that they are doomed to fail, but doing so is also emancipatory. While it is generally accepted that security is a *weak link* problem in that attackers will find and exploit this weak link, reflecting on the different ways a chain *might* break can lead to insights that would otherwise be missed if the weak link is allowed to fail and then quickly replaced with another functional — but still imperfect — link. The idea of thinking about the potentially broken chain rather than its weak link is analogous to the business scenario planning metaphor of the *premortem*. These operate on the assumption that a solution has failed and, rather than reflecting on what might have gone wrong, designers instead generate plausible reasons for explaining the solution's failure (Klein 2007). Even when ambiguity shrouds the reason for this failure, the lack of clarity provides clues about what additional evidence is needed before the "cause of death" can be established.

In this paper we present an approach for planning, running, and evaluating the results of a *security premortem*: a participative design technique where participants assume that a system has been exploited, and plausible reasons are given for

Proceedings of BCS HCI 2012 Workshops
Designing Interactive Secure Systems

explaining why. The objective of security premortems is to identify the reasons for a failure, rather than attempting to mitigate them. In Section 2, we present our approach, before illustrating how the software tools can support this technique, and integrate its results into the broader secure system design process in Section 3. Finally, in Section 4, we reflect on some of the consequences that might arise from running and evaluating premortems.

2. APPROACH

Our approach for running security premortems is loosely based on the three-step process proposed by Klein (Klein 2007); this is described in more detail in the following sections.

2.1. Presenting the scenario

In the first step the project team is brought together and informed that the project has failed because of security problems. Careful thought needs to be given to the "breaking-news" scenario being presented; it must be significant enough to cause the project's failure, believable enough for participants to take the failure seriously, yet also imprecise enough to yield several causes of failure. Based on an imaginary software platform we shall call ACME, an example of a possible scenario is described below:

A major news provider picked up a story based on blog reports by angry mobile phone users; these complain about increased email spam and phishing mails since they started using ACME services. This spam is sufficiently targeted that it evades spam filters. These incidents led to irate twitter posts appearing on the twitter feed on the ACME home page, especially from developers who users blame for this problem. As the bad press grew, major partners began to leave the project, and funding was cut. The cuts meant that the project was forced to stop work.

2.2. Stating potential causes of death

In the second step, team members are given time to independently write down every reason they can think of for the failure; this includes reasons they would normally consider inappropriate. Following this, the facilitator asks each person in turn for a reason, starting with the team leader or most senior team member present. These reasons may correspond to problems at different levels of abstraction. For example, one possible reason might be: *Hardcoded administrator accounts and secrets were, as a result of testing, committed in a major release of ACME that is used in most installations of ACME. This allowed attackers to target cloud services hosting ACME services, and leaking personal information to pastebin.*

2.3. Incorporating reasons into the design

After each reason has been recorded, participants review a hard-copy collection of project specification, reports, and models. Where these artefacts correspond with a possible reason, these are tagged by affixing a "reason" post-it note to the appropriate location in the physical document. Where an artefact does not but should, ideally, exist, then these are noted on a white-board or flip-chart and post-it notes are attached. At the end of the workshop, the team leader reviews the reasons; the tags are used to determine how these reasons cross-cut the system design. Based on this, an action plan is proposed for addressing these reasons.

3. SUPPORTING TOOLS

We now illustrate how software tools can be re-used or extended to support security premortems. In particular, we consider the open-source CAIRIS (Computer Aided Integration of Requirements and Information Security): a Requirements Management tool for supporting the elicitation and specification of usable and secure systems. CAIRIS was developed to implement the IRIS (Integrating Requirements and Information Security) meta-model, which integrates concepts from HCI, Requirements Engineering, and Information Security (Faily and Fléchais 2010a). CAIRIS was designed to be a research tool and can be extended to support new design concepts and techniques. A more detailed overview of CAIRIS is beyond the scope of this paper, but more information about its design and evolution can be found in (Faily and Fléchais 2012).

3.1. Presenting the scenario

Software tools like CAIRIS can be used to support the elicitation and specification of scenarios used in premortem workshops. These scenarios might be consequences of *misusability cases*: scenarios which describe how design decisions cause usability problems that might lead to system misuse (Faily and Fléchais 2011). These scenarios are motivated by argumentation models, the grounds of which might be requirements or architectural components which specify how the system should behave, or behavioural characteristics of personas — descriptions of archetypical user behaviour (Cooper 1999) — that use it. As such, these scenarios describe contexts where a system which satisfies the designers' intentions might be unintentionally exploited.

Scenarios might also arise from specified *misuse cases*; these describe the consequences of specified risks being realised (Sindre and Opdahl 2005). For example, (Atzeni et al. 2011) describes how, with the aid of CAIRIS, open-source threat data from the OWASP (The OWASP Foundation 2011) project was used to create personas for attackers and elicit the attacks they might employ.

3.2. Stating potential causes of death

The causes of failure naturally fit with the requirements engineering concept of *domain properties*; descriptive statements about the problem world. In the case of premortem scenarios derived from misusability cases, CAIRIS can associate these reasons with the both the scenarios and its argumentation model. This is illustrated by the example in Figure 1, which forms the basis of the scenario presented in Section 2.

3.3. Incorporating reasons into the design

To support the integration of premortem reasons into the requirements and architectural design, we have extended CAIRIS in two ways.

First, we have extended CAIRIS to support the association of one or more *tags* with model concepts. In addition to providing a means for interrogating CAIRIS models based on these tags as a search criteria, several of CAIRIS' visual models have also been updated to support the annotation of tags to different model elements.

Second, if risks are not evident then these tagged artefacts can be analysed in more detail. To allow this, we have built qualitative data analysis capabilities into CAIRIS. This allows us to assign codes — words or phrases that assign a summative, essence-capturing attribute for a portion of language based data (Saldaña 2009) — directly to design artefacts stored in CAIRIS. For example, based on the reason we gave in Section 2.2, we might wish to better understand the factors that might lead to this reason; as Figure 2 shows, this might include coding persona descriptions. Relationships drawn between these thematic concepts might be used to motivate vulnerabilities, in the same way that argumentation models can motivate premortem scenarios.

4. CONCLUSION

This paper presented an approach for applying premortems for finding security failures in a secure system design. We have also shown how CAIRIS can support this technique by facilitating scenario generation, categorising models according to failure reason and, based on these reasons, analysing model data to find ways of addressing their root causes.

We are currently evaluating both this technique and CAIRIS' ability to support it as part of the *webinos* project. As part of this evaluation, we are exploring possible stimuli that might be used by participants for eliciting reasons. These stimuli includes adopting the perspective of an attacker with the aid of pre-developed attacker personas. We are also evaluating the physical settings where premortem processes can be run. By running premortem workshops, participants gain respect from their colleagues by suggesting insightful reasons, and healthy team dissent is encouraged rather than discouraged (Klein 2009). However, workshops can be difficult to set up when team members are distributed, and a successful outcome is often dependent on the effectiveness of the group's facilitator. For this reason, we are currently investigating how effective the premortem process might be if reasons are elicited on a one-to-one basis outside of a workshop, and what sort of factors might lead to the elicitation of insightful reasons given the change of setting.

By tool-supporting premortems, we also raise the question of how far tools can go before they obstruct, rather than stimulate, creativity and innovation? The *innovation design dilemma* suggests that structure might stymie creativity but, without it, creative output might become too disruptive (Hobek 1988). In this respect, we believe CAIRIS strikes a balance. By providing only modest tool-support during workshop settings, the tool provides little to obstruct group dynamics. Also, by aligning reasons and their rationale with CAIRIS models, the impact of security innovation arising from premortem scenarios can be explored.

If qualitative data analysis is to be used to find the root causes of failure then CAIRIS will need to be used more visibly in group settings; this will help mediate discussions around qualitative models stored within the tool. While techniques for using software for supporting qualitative *research* are well known, their use for supporting *design*, especially for security, is ill-explored. Consequently, future work will also explore the effectiveness of qualitative data analysis techniques in conjunction with premortems to more directly support secure system design activities.

5. ACKNOWLEDGEMENTS

The research described in this paper was funded by the EU FP7 *webinos* project (FP7-ICT-2009-05 Objective 1.2).

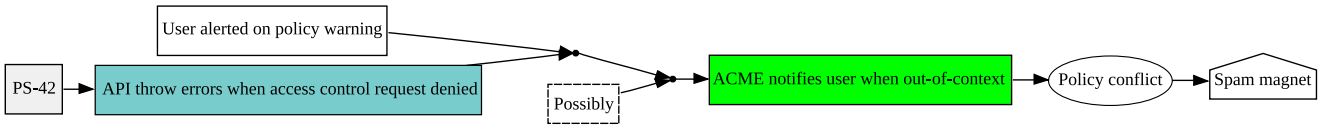

Figure 1: *Misusability case argumentation model motivating a security premortem*

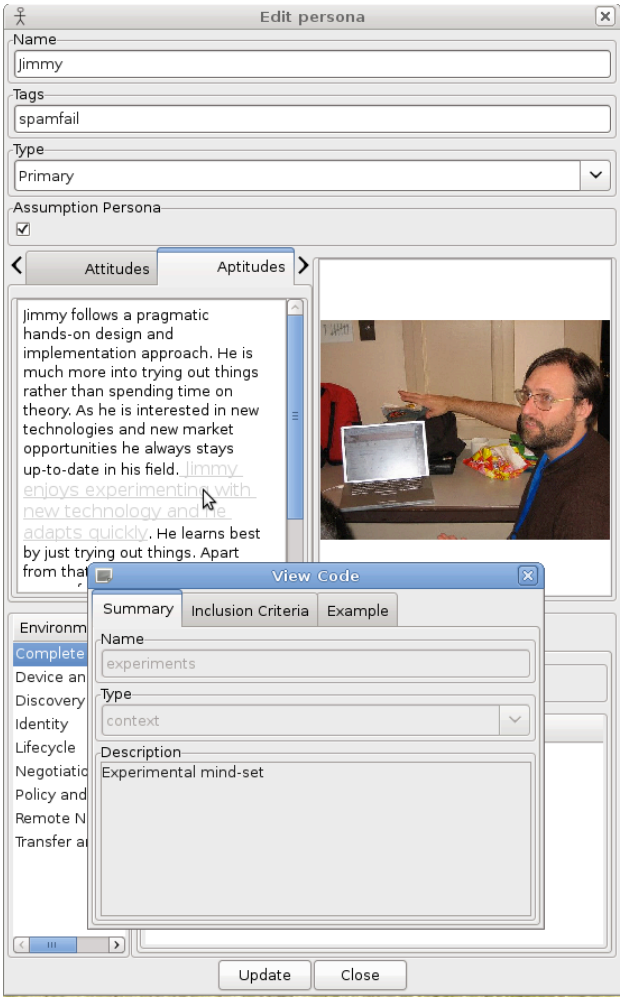

Figure 2: *Coding a persona based on a premortem reason*

REFERENCES

Atzeni, A., Cameroni, C., Faily, S., Lyle, J., and Fléchais, I. (2011). Here's Johnny: a Methodology for Developing Attacker Personas. In *Proceedings of the 6th International Conference on Availability, Reliability and Security*, pages 722–727.

Cooper, A. (1999). *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity (2nd Edition)*. Pearson Higher Education.

Faily, S. and Fléchais, I. (2010a). 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. (2010b). To boldly go where invention isn't secure: applying Security Entrepreneurship to secure systems design. In *Proceedings of the 2010 New Security Paradigms Workshop*, pages 73–84. ACM.

Faily, S. and Fléchais, I. (2011). Eliciting Usable Security Requirements with Misusability Cases. In *Proceedings of the 19th IEEE International Requirements Engineering Conference*, pages 339–340. IEEE Computer Society.

Faily, S. and Fléchais, I. (2012). Software for Interactive Secure System Design: Lessons Learned Developing and Applying CAIRIS. In *Designing Interactive Secure Systems: Workshop at British HCI 2012*. To Appear.

Hobek, J. (1988). The innovation design dilemma: some notes on its relevance and solution. In Grønhaug, K. and Kaufmann, G., editors, *Innovation: a cross-disciplinary perspective*. Norwegian University Press.

Klein, G. (2007). Performing a project premortem. *Harvard Business Review*, 85(9):18–19.

Klein, G. A. (2009). *Streetlights and shadows: searching for the keys to adaptive decision making*. MIT Press.

Saldaña, J. (2009). *The coding manual for qualitative researchers*. Sage.

Sindre, G. and Opdahl, A. L. (2005). Eliciting security requirements with misuse cases. *Requirements Engineering*, 10(1):34–44.

The OWASP Foundation (2011). Open Web Application Project (OWASP) web site. `http://www.owasp.org`.