Designing usable and secure software with IRIS and CAIRIS.

FAILY, S.

2018

Shamal Faily

Designing Usable and Secure Software with IRIS and CAIRIS

Springer

Contents

Chapter 1
Why designing for usability and security is hard

Abstract In this chapter, I summarise the challenges that make designing for usability and security hard, and outline the structure of this book.

1.1 Empowering the system builders

The effect of Information Technology on our lives can be seen all around us. The increasing ubiquity of technology has also led academic researchers to re-think our interaction with it. In a 2009 Communications of the ACM article [**?**], several leading Human Computer Interaction (HCI) researchers noted that our relationship with computers has changed so radically since the field's inception that even the term *HCI* needs a rethink. In the past, we could reasonably assume that IT involved desktop computers and *users* in commercial organisations. Nowadays, systems are as ubiquitous as the people who use them, who are increasingly connected with different people and other systems. In such a connected world, the users of technology have incalculable opportunities to intentionally or unintentionally interact with a myriad of systems.

One question which has yet to be answered is how much Information Technology has empowered the work of those who build it. Media reports about the growth of high technology industry go almost hand-in-hand with reports about the impact of threats to it. For example, a report commissioned by the UK government estimated that the cost of cyber crime to the UK economy is £27 billion per annum [**?**]. While the methodologies used to devise this figure are debatable, the increased burden of expectation on system designers is not. As consumers, we expect systems to be attuned to the physical and social contexts within which they are used. As a corollary, we would also like our systems to be as secure as they are usable but, as we have discovered, threats to, and vulnerabilities within, this complex network of people and technology make this a challenging task for system builders.

1.2 Ubiquitous technology

Systems can be made vulnerable through a variety of factors, ranging from the accidental introduction of incorrect code, through to an overly complex user interface which may be misused or circumvented. Those who might take advantage of these vulnerabilities have capabilities and motivations which may be unknown to the designers who inadvertently introduced them, together with different abstractions for what the vulnerabilities are and how they can be exploited. So, while our expectations for technology innovation continue to be exceeded, the quality of these systems' security and usability often falls short.

There is no obvious reason why designing secure and usable systems should be so difficult, especially when guidance on applying Security and Usability Engineering best practice is no longer restricted to the scholarly literature. Nielsen claimed that cost was the principal reason why Usability Engineering techniques are not used in practice [**?**], but the financial costs of applying such techniques may not have been reduced by technology advances. Similarly, practical techniques for identifying and mitigating security problems during system design are now available to developers in an easy to digest format, e.g. [**?**, **?**].

1.3 Integrating processes

Problems arise when considering how to use these approaches as part of an integrated process. Accepted wisdom in Software Engineering states that requirements analysis and specification activities should precede other stages in a project's lifecycle [**?**]. However, Information Security and HCI proponents argue that their techniques should instead come first. For example, ISO 13407 [**?**] states that activities focusing on the collection of empirical data about users and their activities should guide early design, but security design methods such as [**?**, **?**] suggest that such stages should be devoted to high-level analysis of the system to be secured. Invariably, the decision of what concern to put first is delegated to the methodology followed by a designer. The designer has many approaches to choose from, some of which include treatment for security or usability concerns. To date, however, no approach treats both security and usability collectively, beyond treating them both as generic qualities contending with functionality.

When weighing up the approaches available, and the effort needed to apply them in their developmental contexts, designers may even choose to simply ignore them. Designers may believe that their knowledge about user goals and expectations negate the need for applying usability design techniques, or their understanding of the system's risks and mitigating controls negates the need for security analysis. In such cases, developers may believe Security and Usability Engineering approaches are useful, but they may not believe the pay-off justifies their cost.

1.4 Growing interests in usable security

There is mounting evidence that the design of usable and secure systems is worthy of specific attention. The US Department of Homeland Security ranked usable security as one of the top cyber-security research topics for governments and the private sector [?], and HCI-Sec (HCI for Security) continues to be an active research topic. Researchers are also beginning to consider how developers should build secure and usable systems [?], and even governments are recognising the role developers play in securing software [?]. Despite this interest in *developers*, yet there remains a lack of guidance available to *designers* about how to design usable systems at a sufficiently early stage in the design process. Fortunately, despite the lack of guidance, the Security Requirements Engineering and HCI communities have proposed a number of individual techniques forming the basis of integrated design approaches. In theory, specifying and designing secure and usable systems involves carefully selecting the right techniques from each community. In practice, each technique is founded in different, potentially conflicting, conceptual models. The level of tool-support for these techniques also varies considerably, and there has been little work on integrating these tools and the conceptual models which underpin them.

The knowledge gleaned integrating design techniques and tools also leads to research contributions beyond the design of usable and secure systems. While there are academic fora devoted to integrating security and software engineering activities, e.g. [?], and HCI and security activities [?], there has been little work describing how usability design techniques can be usefully applied to designing secure systems. It is, therefore, possible that the results of integrating design techniques and tools may lead to design innovation in this area.

1.5 IRIS and CAIRIS as exemplars for Usability, Security, and Requirements Engineering process and tool integration

The book explores how existing techniques and tools might be integrated and improved to support the design of usable and secure systems. It shows how concepts from Usability, Security, and Software Engineering can be harmonised to support the design of secure and usable systems, discusses the characteristics of tool-support needed to support the design of secure and usable systems, and considers how User-Centered Design techniques be improved to support the design of usable and secure systems.

In achieving these goals, the book presents IRIS (Integrating Requirements and Information Security): a process framework guiding the selection of usability, security, and software design techniques for design processes. To complement IRIS, the book also presents CAIRIS (Computer Aided Integration of Requirements and Information Security): a software platform for supporting these processes. I formally introduce both IRIS and CAIRIS in Part 2 of this book.

In Software Engineering, the term *system design* encompasses a broad range of activities; these range from scoping an early vision of what a system needs to do, through to developing models of software components and interfaces. We, therefore, primarily limit our focus on design to the early stages of a system's development for two reasons. First, the term *design* often refers to a plan or an outline of work, upon which a structure is built [**?**]; agreeing and specifying the nature of this plan is both required, and something best carried out as early as possible. Second, each discipline contributing techniques to the design of usable and secure systems argues for their own approaches preceding all others. Consequently, there is value in exploring how early design techniques interoperate together.

1.6 Book structure

1.6.1 Part 1: Foundations

Chapter 2 reviews the current state-of-the-art in the design of usable and secure systems. I consider existing work from the HCI Security community, and its limitations, before reviewing prevalent HCI concepts relevant to the design of secure systems. Given this book's focus, I review several Requirements Engineering approaches from a security and usability perspective, before reviewing existing design *frameworks* for eliciting security and usability requirements. I conclude the chapter with a brief review of the available tool-support for facilitating Usability and Requirements Engineering activities.

Chapter 3 presents a conceptual model for usable secure Requirements Engineering. This work builds upon practical work in usability design, and research on meta-models for Security Requirements Engineering. Collectively, the meta-model concepts help structure and manage Usability, Security, and Requirements Engineering activities in different contexts. After presenting a brief overview of the conceptual model itself, I sub-divide the model explanation into six different views: Environment, Asset, Task, Goal, Risk, and Responsibility. For each view, I present and justify the related concepts and their relationships.

1.6.2 Part 2: IRIS and CAIRIS

Chapter 4 introduces a process framework for selecting techniques when specifying usable and secure systems. Building on the meta-model described in Chapter 3, I present and describe the different *perspectives* of IRIS. Finally, I propose a number of exemplar techniques for each perspective.

Chapter 5 presents CAIRIS (Computer Aided Integration of Requirements and Information Security), a software tool designed to embody the characteristics needed to support the IRIS framework. I briefly discuss the principles motivating the design of CAIRIS before presenting its high-level architecture. I then present several characteristics of tool-support for the specification of usable and secure systems, and illustrate how CAIRIS supports these.

Chapter 6 describes how personas and scenarios can be adapted for secure system design. I present a technique for aligning personas with the design of secure and usable systems. This technique, *Assumption Persona Argumentation*, describes how structured argumentation can be used to support the development and evolution of assumption personas. Building on this work, I present *Persona Cases*: an approach for developing personas both grounded in, and traceable to, their originating empirical data. Also building upon both the IRIS meta-model and this argumentation structure, I present a technique called *Misusability Cases*, scenarios which describe how design decisions lead to usability problems subsequently leading to system misuse.

Chapter 7 describes a study where IRIS and CAIRIS were used to specify security requirements for a portal facilitating the sharing of medical study data.

Chapter 8 reports on a study where IRIS and CAIRIS were used to analyse the security and usability impact of a security policy for control system software at a UK water company. This analysis was used to identify missing security policy requirements.

1.6.3 Part 3: Beyond Requirements

In Chapter 9, I illustrate how IRIS can be used to not only elicit and specify security requirements but also, with the aid of architectural and attack patterns, and complementary work on attack surface metrics and architectural risk analysis, we can secure software architectures as well.

Chapter 10 describes a long term case study where IRIS and CAIRIS played a role in the design and development of webinos: a web-based middleware for the Internet of Things.

In Chapter 11, I look at the role *innovation* can play when designing for security and usability, and introduce the paradigm of *Security Entrepreneurship*. Much has been written about the role of economics for information security, but entrepreneurship and economics go hand-in-hand, and there is much that can be learned from technology and social entrepreneurship that can be applied to security problem solving. This chapter shows how theory and models from technology innovation and entrepreneurship can be used and, in some cases, how IRIS and CAIRIS can be of assistance.

For much of this book, CAIRIS plays a supporting role to IRIS. However, the platform has the potential to support usability and security more broadly. Therefore, I conclude this book in Chapter 12 by presenting several additional applications of CAIRIS, to show the direction that future tools for usable and secure software design can take.