

Designing usable and secure software with IRIS and CAIRIS.

FAILY, S.

2018

This is the accepted manuscript for a single representative chapter from the full work. It is distributed under the Springer terms of use for accepted manuscripts: <https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms>

The whole book can be purchased from the publisher's website: <https://doi.org/10.1007/978-3-319-75493-2>

Shamal Faily

Designing Usable and Secure Software with IRIS and CAIRIS

Springer

Contents

Part I Foundations

1	Why designing for usability and security is hard	3
1.1	Empowering the system builders	3
1.2	Ubiquitous technology	4
1.3	Integrating processes	4
1.4	Growing interests in usable security	5
1.5	IRIS and CAIRIS as exemplars for Usability, Security, and Requirements Engineering process and tool integration	5
1.6	Book structure	6
1.6.1	Part 1: Foundations	6
1.6.2	Part 2: IRIS and CAIRIS	6
1.6.3	Part 3: Beyond Requirements	7
2	Usable and Secure Software Design: the State-of-the-art	9
2.1	Towards usable security design	9
2.1.1	Themes in Information Security design	9
2.1.2	Usable security	11
2.1.3	A case for better HCI integration?	14
2.2	Usability design	15
2.2.1	HCI and usability	15
2.2.2	User-centered approaches	17
2.2.3	Human-centered software engineering	21
2.3	Specifying Security	22
2.3.1	Problem Frames	23
2.3.2	Goal-oriented approaches	26
2.3.3	Use cases	33
2.3.4	Other related security and usability approaches	35
2.4	Specification frameworks	37
2.4.1	RESCUE	38
2.4.2	SQUARE	39

2.5	Tool-support	40
2.5.1	Usability design tools	40
2.5.2	Security Requirements Engineering tools	40
2.5.3	Visualising design	42
2.6	Chapter summary	43
3	A Conceptual Model for Usable Secure Requirements Engineering	45
3.1	Introducing NeuroGrid	45
3.2	Overview	46
3.3	Environment meta-model	46
3.4	Asset meta-model	48
3.5	Task meta-model	49
3.6	Goal meta-model	52
3.7	Risk meta-model	54
3.8	Responsibility meta-model	57
3.9	Chapter summary	58
Part II IRIS and CAIRIS		
4	The IRIS Framework	61
4.1	Perspectives	61
4.2	Converging concepts	63
4.3	Framework techniques	63
4.3.1	Grounded Theory	64
4.3.2	Personas	65
4.3.3	Activity scenarios	66
4.3.4	Rich pictures	67
4.3.5	KAOS	68
4.3.6	Volere requirements specification	68
4.3.7	Use Cases	70
4.3.8	AEGIS	70
4.3.9	Misuse cases	71
4.4	Chapter summary	72
5	Introducing CAIRIS: Tool-Support for Designing Usable & Secure Systems	73
5.1	CAIRIS design principles	73
5.1.1	Familiarity	73
5.1.2	Extensibility	74
5.1.3	Process centricity	74
5.1.4	Security and usability centricity	74
5.2	Tool development process	75
5.2.1	Initial CAIRIS prototypes	75
5.2.2	From desktop tool to software platform	75
5.3	Architectural design	76
5.3.1	Visual interface design	76

Contents

5.3.2	High level architecture	77
5.3.3	Physical deployment	79
5.3.4	CAIRIS APIs	80
5.4	Tool-support characteristics	80
5.4.1	Automated analysis	80
5.4.2	Model visualisation	87
5.4.3	Model reusability	95
5.4.4	Model externalisation	95
5.5	Chapter summary	96
6	Adapting Personas and Scenarios for Security and Usability Design	99
6.1	Building personas	99
6.1.1	Step 1: Identify factoids from source data	100
6.1.2	Step 2: Affinity diagram factoids to identify behavioural clusters	100
6.1.3	Step 3 (optional): Affinity diagram behavioural cluster categories	101
6.1.4	Step 4: Categorise behavioural clusters by behavioural variables	101
6.1.5	Step 5: Write persona narrative	102
6.1.6	Persona validation	102
6.2	Assumption Persona Argumentation	102
6.2.1	Toulmin's Model of Argumentation	103
6.2.2	Developing assumption personas	103
6.2.3	Applying and refining the assumption personas	104
6.3	Persona Cases	105
6.3.1	Building persona cases	106
6.3.2	Illustrated example	107
6.3.3	Personifying qualitative models	110
6.4	Supporting the persona creation workflow with CAIRIS	110
6.5	Misusability Cases	111
6.5.1	Eliciting Misusability Cases	113
6.5.2	Applying Misusability Cases	114
6.6	Chapter Summary	115
7	Case Study: Securing a Medical Data Portal	117
7.1	About the Data Support Service (DSS)	117
7.1.1	Beginning the project	118
7.1.2	e-Science and security	118
7.1.3	The DSS user community	119
7.1.4	Stakeholder access	119
7.2	The IRIS Process	120
7.2.1	Persona development and scoping	121
7.2.2	Design sessions	121
7.2.3	Misusability case analysis	122

7.3	Applying IRIS and CAIRIS	122
7.3.1	Persona development	122
7.3.2	Design sessions	123
7.3.3	Misusability case analysis	123
7.4	Lessons Learned	128
7.4.1	Persona development	129
7.4.2	Design sessions	130
7.4.3	Misusability case analysis	132
7.5	Chapter summary	133
8	Case Study: Defending Critical Infrastructure against Stuxnet	135
8.1	About ACME Water	135
8.1.1	Instrumentation and instrument technicians	136
8.1.2	Legacy equipment	136
8.1.3	Software ubiquity	137
8.1.4	Information security	137
8.1.5	Stuxnet	138
8.1.6	Plant operations	139
8.1.7	The enterprise SCADA project	139
8.2	The IRIS Process	140
8.2.1	Scoping	140
8.2.2	Fieldwork	140
8.2.3	Usability and security analysis	141
8.2.4	Risk and requirements review	142
8.3	Applying IRIS and CAIRIS	142
8.3.1	Scoping	142
8.3.2	Fieldwork	144
8.3.3	Usability and security analysis	147
8.3.4	Risk and requirements review	150
8.4	Lessons Learned	151
8.4.1	Scoping	152
8.4.2	Fieldwork	152
8.4.3	Usability and security analysis	153
8.4.4	Risk and requirements review	154
8.4.5	Qualitative model re-usability	154
8.4.6	Incorporating stakeholder activities into the study diagnosis	155
8.5	Chapter summary	155
 Part III Beyond Requirements		
9	Analysing and Managing Architectural Risk	159
9.1	Extending IRIS and CAIRIS for Architectural Design	159
9.2	Approach	160
9.3	Architectural Patterns	161
9.4	Attack Surface Metrics	161

Contents

9.5	Contextualised Attack Patterns	162
9.6	Architectural Risk Analysis	164
9.6.1	Architectural Pattern Specification	164
9.6.2	Attack Resistance Analysis	165
9.6.3	Ambiguity Analysis	165
9.6.4	Weakness Analysis	166
9.7	Example: an Architectural Risk Analysis of WebDAV for NeuroGrid	166
9.7.1	Architectural Pattern Specification	167
9.7.2	WebDAV Attack Surface Metric	169
9.7.3	Attack Resistance Analysis	170
9.7.4	Ambiguity Analysis	171
9.7.5	Weakness Analysis	174
9.8	Chapter summary	174
10	Case Study: Securing an Internet of Things middleware	177
10.1	Learning from Research & Development projects	177
10.2	Lessons Learned from e-Science and NeuroGrid	178
10.3	About webinos	179
10.3.1	The Design and Development process	180
10.3.2	The Usable Security design team	181
10.4	Building Security into webinos	181
10.4.1	Context of Use Analysis	181
10.4.2	Supporting initial Requirements Elicitation and Specification	184
10.4.3	Participatory Risk Analysis	185
10.4.4	Analysing the webinos software architecture	186
10.4.5	Supporting Platform and Application Development	187
10.4.6	Releasing webinos design data	188
10.5	Challenges	188
10.5.1	User Research is not easy	189
10.5.2	Usability is not a priority	190
10.5.3	Technique misappropriation is easy	191
10.5.4	Sustaining adoption through implementation requires creativity	191
10.6	Chapter summary	194
10.6.1	Learn to work with sub-optimal data and expertise	194
10.6.2	Security increases sensitivity to usability problems	194
10.6.3	Designing for usability and security takes time	195
10.6.4	Design research is a key element of designing of usability and security	195
11	Evaluate Security as an Innovation	197
11.1	Security as an Innovation	197
11.2	Innovation and System Building	198
11.2.1	Creativity and design	198
11.2.2	Entrepreneurs as system builders	199

11.2.3	Models of Innovation	200
11.2.4	Social Entrepreneurship analogies	201
11.3	Security Entrepreneurship and the Security Entrepreneur	201
11.4	Security Entrepreneurship Techniques	203
11.4.1	Security Chindōgu	204
11.4.2	Innovation Value-added Chain	207
11.4.3	Social Network Analysis	208
11.4.4	Security Premortems	209
11.5	Towards Security Entrepreneurship	211
11.5.1	Security Entrepreneurship considered harmful?	212
11.5.2	How should Security Entrepreneurship be situated with other design activities?	212
11.5.3	How is Security Entrepreneurship validated?	213
11.5.4	Can Security Economics help?	214
11.6	Chapter summary	215
12	Further applications of CAIRIS for Usable and Secure Software	
	Design	217
12.1	Environments as stakeholder perspectives in Systems of Systems ..	217
12.2	Threat modelling	218
12.3	Trust modelling	220
12.4	Realising “Design as Code”	223
12.4.1	Creating Assured Personas	224
12.4.2	Discovering Risks from Attackers and Threat Models	224
12.4.3	Contextualising the Implication of Security and Usability Design Decisions	227
12.4.4	Implications for CAIRIS	229
12.5	Human-Centred Specification Exemplars	230
12.5.1	Design Principles for Human-Centred Specification Exemplars	230
12.5.2	Evaluating Research with Human-Centred Specification Exemplars	231
12.5.3	Human-Centred Specification Exemplars and Archetypes ..	232
12.6	Chapter summary	232

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 life-cycle [?]. 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.

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

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

1.6 Book structure

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.