Towards tool-support for usable secure requirements engineering with CAIRIS.

FAILY, S. and FLÉCHAIS, I.

2010

Towards Tool-Support for Usable Secure Requirements Engineering with CAIRIS

Shamal Faily, University of Oxford, UK

Ivan Fléchais, University of Oxford, UK

ABSTRACT

Understanding how to better elicit, specify, and manage requirements for secure and usable software systems is a key challenge in security software engineering, however, there lacks tool-support for specifying and managing the voluminous amounts of data the associated analysis yields. Without these tools, the subjectivity of analysis may increase as design activities progress. This paper describes CAIRIS (Computer Aided Integration of Requirements and Information Security), a step toward tool-support for usable secure requirements engineering. CAIRIS not only manages the elements associated with task, requirements, and risk analysis, it also supports subsequent analysis using novel approaches for analysing and visualising security and usability. The authors illustrate an application of CAIRIS by describing how it was used to support requirements analysis in a critical infrastructure case study.

Keywords: *HCI Security, Misuse Cases, Personas, Requirements Management, Risk Analysis, Security Requirements Engineering, User-Centered Design*

INTRODUCTION

Frequent reports of human and technical security failures in systems highlight the need for designing usable security, but specifying usable and secure systems is easier said than done. Understanding why security controls are unusable means factoring in the characteristics of people using controls, the work they carry out while using controls, and the physical, social, and even cultural contexts within which the controls are used. While it is accepted wisdom that these concerns should be treated as early as possible, eliciting and specifying requirements for secure and usable controls remains a hit-and-miss affair.

Requirements Engineering involves understanding the problem domain within which a system is situated, obtaining data from stakeholders in this domain, analysing this data to elicit a set of requirements, validating these requirements, and managing their evolution. When properly applied, techniques from HCI and Information Security complement these early stages of Requirements Engineering. Techniques used by usability professionals are grounded in observational, performance,

DOI: 10.4018/jsse.2010070104

and other qualitative and quantitative data. If used properly, this usability data can immerse analysts and stakeholders in the problem domain and help explore assumptions held about threats and vulnerabilities. Similarly, Goal-Oriented Requirements Engineering techniques are not only useful for eliciting requirements from goals, but also threats from anti-goals (Lamsweerde, 2004). Even the traditional workshop setting, where requirements are often elicited and validated, can support the design of usable security; participative approaches to risk analysis, e.g., (Fléchais et al., 2007; Braber et al., 2007) help stakeholders take a situated approach to security by alerting them to threats and vulnerabilities, identifying risks in their environment, and directing mitigating specification and design decisions.

The challenge of specifying usable and secure software systems comes not only from choosing the right combination of techniques, but also from analysing and managing the data arising from them. For non-trivial systems, risk and requirements analysis precipitate voluminous amounts of data. A requirement may be the leaf node of a large goal-tree, the root goal of which may be derived from mitigating a particular risk; this mitigation response may arise as a result of a chain of risk and requirements analysis. Furthermore, empirical usability data needs to contribute to any design decisions; if we mitigate one risk, the resulting usability impact of this design decision may introduce others. Risk and usability ratings for a system design are also coloured by analyst perceptions; this allows human error to creep into any valuation.

Without tool-support, the security-usability balance can become uneven and overly subjective as risk analysis becomes more advanced. We need tool-support to manage security, usability, and requirements data, automate its analysis, and convey the results to stakeholders. This paper discusses CAIRIS (Computer Aided Integration of Requirements and Risk Analysis): a tool for managing the elements arising from usability, requirements, and risk analysis. This tool supports the elicitation of requirements from goals and tasks, and risks from threats and vulnerabilities. By structuring elicited data according to a meta-model for usable secure requirements engineering (Faily & Fléchais, 2010), meaningful traceability links between different model types can be automatically maintained, allowing data to be quickly analysed and visualised in a participative workshop setting. In the next section, we describe the related work motivating CAIRIS. In the subsequent sections, we introducing the tool, and we describe how CAIRIS was used to elicit requirements in a Critical Infrastructure case study.

RELATED WORK

We are unaware of any single tool purporting to support the analysis of usability, requirements, and risk analysis. Some coverage is, however, provided by existing tools in each of these areas, and presented in the following sections.

Conceptual Tools for Usability

Designing usable system requires an early focus on users and their goals (Preece et al., 2007). Although many engineers consider usability as synonymous only with user interface design (Seffah & Metzker, 2004), it is also a quality concerning the people interacting with these interfaces, and how they use them to perform work tasks. Unfortunately, we currently lack tool-support allowing analysts and developers to inform secure system design with usability insights.

Qualitative usability data can be represented as personas: fictitious, specific, concrete representations of target users (Pruitt & Adlin, 2006). By describing how personas carry out these scenarios, we can represent usability data in a meaningful way to stakeholders and inform the subsequent analysis accordingly. A step towards managing personas and the scenarios they participate in involves devising a suitable means of structuring and categorising this interaction. Such categorisations might also help measure the impact to usability of security design decisions, and vice-versa.

Security Requirements Engineering Tools

Many tools for Security Requirements Engineering are general Requirements Management tools, which have been augmented for security. Such tools are often based on the spreadsheet metaphor, where a table is used to enter the attributes of a natural language requirement. By applying this metaphor, requirement attributes, such as its description, type, and rationale, can be quickly specified. Unfortunately, the generic strength of a Requirements Management tool is also its weakness; the lack of distinct semantics means an analyst must manually maintain traces between requirements and non-requirements artifacts.

Model-based approaches support traceability between different artifacts. If a tool conforms to the requisite meta-model then, as data is entered into the tool, it can be structured in a manner that facilitates automated analysis and visualisation. Tool-support for model-based approaches exist for risk analysis (Braber et al., 2007; Meland et al., 2008) and goal modelling (Respect-IT, 2007), but the task of integrating different model-based approaches for security requirements engineering is non-trivial. One problem is the diversity of the models to be integrated; tools are often based on different meta-models, making understanding and agreeing interfaces difficult. If we assert that misuse cases <<threaten>> use cases, do we agree what it means for a use case to be threatened? Does a misuse case threaten the work carried out by a use case, or the assets associated with it? Houmb et al. (2009) describes some of the challenges faced when integrating these different techniques.

One strategy for integrating these approaches is to consider how Secure Requirements Engineering and HCISec might complement each other. Thimbleby (2007) argues that usability approaches are necessary, but far from sufficient for critical systems. The sheer size of the state space associated with interactive devices is so big that empirical evaluation on its own is unsustainable. It is, however, possible, to supplement usability analysis with basic technical methods.

Visualising Secure Systems Design

Before stakeholders can measure the impact of usability of secure system design decisions, they need to understand the rationale underpinning a design. Previous work has illustrated how different visualisation techniques can both explain the results of analysis, and explore the resulting impact. While we are unaware of work purporting to visualise the analysis and resulting impact of usability and security, there has been work on independently visualising analysis in each area.

The canonical visual notation for modelling tasks as scenarios is the UML Use Case Diagram. These diagrams show the relationship between human or non-human actors, represented as stick figures, and coherent units of functionality, represented as ellipses. The diagrams have also been extended to display Misuse Cases, typically represented as black ellipse, and attackers, typically represented as an attacker with a filled black head (Alexander, 2002). Røstad has proposed an extended notation for dealing Misuse Cases, such that threats and vulnerabilities are modelled as separate entities in a Use Case Diagram (Røstad, 2006). With the aid of stereotyped associations and different attacker types, this notation allows more information cogent to a risk analysis to be modelled, and inside and outside attacks to be distinguished.

Techniques from information visualisation have also been used to support risk analysis. Hogganvik (2007) has concluded that colours are a useful means of distinguishing the value of different risks, and Feather et al. has used bar charts to portray information from Defect Detection and Prevention (DDP) models to make problematic areas more evident to stakeholders (Feather et al., 2006).

CAIRIS

CAIRIS is a step towards tool-support for usable secure requirements engineering. CAIRIS supports the elicitation of usability, require-

ments, and risk analysis data before and during participative design activities. Data is entered into the CAIRIS front-end, and stored in a back-end database; the constraints in the database are based on the IRIS meta-model (Faily & Fléchais, 2010).

Requirements Management

A recent survey on techniques for describing security requirements (Tøndel et al., 2008) concluded that there was no consensus on what a security requirement is. We have, therefore, decided to represent all requirements, including security requirements, as natural language text: the lingua franca for requirements specifications in industry. CAIRIS includes an editor for specifying natural language requirements, which is based on the spreadsheet metaphor; the table columns conform to the Volere Requirements Shell (Robertson & Robertson, 2009). Each table of requirements is associated with an asset or an environment. This enables large specifications to be structured according to the concern most closely related to it.

CAIRIS also supports many of the features found in commercial requirements management tools, such as versioned changes to requirements, forward and backward traceability, and automatic requirements document generation. CAIRIS does not, however, support ad-hoc traceability between all artifacts; almost all traceability links are automatically generated and maintained as part of the modelling process. Manual links can only be created where they are meaningful. For example, it is meaningful to associate a task with a vulnerability; some aspect of a task might be open to exploitation, and it is difficult to cull such a relationship from the textual narrative of the task. However, it is invalid to manually associate a task with a role. Although this relationship may exist implicitly, it is as a corollary of a relationship between tasks and personas. This latter relationship can be generated automatically by CAIRIS when stating a persona participates in a task.

Task Analysis

Empirical data about how target users plan to use the system-to-be is modelled in CAIRIS using personas (Pruitt & Adlin, 2006) and task based scenarios (Rosson & Carroll, 2002). These personas fulfil one or more roles. Although there is no agreed way of measuring the usability of a task with respect to its participating personas, a number of persona and task attributes match attributes found in the ISO 9241-11 (ISO, 1998) framework. ISO 92411-11 describes how usability goals can be evaluated using the goals of effectiveness, efficiency, and satisfaction. Based on this framework, we have devised a set of task usability properties (Figure 1); these can be used to evaluate how usable a task is to a persona. When defining tasks, these four properties are set for each persona participating in a scenario. Each of these properties map to one of the usability components of ISO 9241-11.

Each property has an associated value x which maps to a natural number in the range $0 \leq x \leq 3$; this corresponds to the qualitative values of None, Low, Medium, and High respectively. To ensure equal weighting for all 3 usability components, the usability of a task U_t is computed using the equation

$$U_t = \frac{\overline{a+b}}{2} + \bar{c} + \bar{d}$$

where $\frac{\overline{a+b}}{2}$ is the mean task efficiency, $\bar{c}$ is the mean task satisfaction, and $\bar{d}$ is the mean task effectiveness. Variables a, b, c, and d refer to the task duration, frequency, demands, and goal conflict respectively. The mean value is taken across all personas carrying out the task in question. The higher the value of U_t, the less usable a task is for the personas associated with it. More meaningful values must be used for duration and frequency because values like low, medium, and high are ambiguous. For duration,

Figure 1. Task (left) and countermeasure task (right) usability properties

Property	ISO 9241-11 Usability Component	Description	Values
Duration	Efficiency	The time taken by a persona to complete the task.	• None • Seconds • Minutes • Hourly or longer
Frequency	Efficiency	The frequency a persona carried out the task.	• None • Hourly or more • Daily - Weekly • Monthly or less
Demands	Satisfaction	The mental or physical demands on a persona.	• None • Low • Medium • High
Goal Conflict	Effectiveness	The degree to which the task interferes with the persona's work or personal goals.	• None • Low • Medium • High

Property	ISO 9241-11 Usability Component	Description	Values
Duration	Efficiency	The degree to which the countermeasure helps or hinders the time taken by a persona to complete the task.	• High Help • Medium Help • Low Help • None • Low Hindrance • Medium Hindrance • High Hindrance
Frequency	Efficiency	The degree to which the countermeasure increases or decreases the frequency a persona needs to carry out the task.	• High Help • Medium Help • Low Help • None • Low Hindrance • Medium Hindrance • High Hindrance
Demands	Satisfaction	The degree to which the countermeasure increases or decreases the mental or physical demands on a persona while carrying out the task.	• High Help • Medium Help • Low Help • None • Low Hindrance • Medium Hindrance • High Hindrance
Goal Conflict	Effectiveness	The degree to which the task helps or hinders the persona's work or personal goals.	• High Help • Medium Help • Low Help • None • Low Hindrance • Medium Hindrance • High Hindrance

the qualitative ratings used are *Seconds*, *Minutes*, and *Hours or Longer* are associated with the values 1, 2, and 3 respectively. For frequency, the ratings used are *Monthly or less*, *Daily - Weekly*, and *Hourly or more*.

When mitigating risks, one or more roles are associated with each mitigating countermeasure; these roles will, in some way, be directly affected by the countermeasure being designed. By associating roles with countermeasure within IRIS, candidate personas and their tasks can be identified. For each task-persona pairing, countermeasure usability properties can be specified.

Based on this countermeasure usability data, it is possible to calculate the countermeasure usability factor TU_t. The right hand side of the equation computing TU_t is identical to U_t, i.e.

$$TU_t = \frac{\overline{a+b}}{2} + \bar{c} + \bar{d}$$

The values are, however, different. $\frac{\overline{a+b}}{2}$ is the mean contribution to task efficiency, $\bar{c}$ is the mean contribution to task satisfaction, and $\bar{d}$ is the mean contribution to task effectiveness. Like U_t, the variables a, b, c, and d refer to the task duration, frequency, demands, and goal conflict respectively. The mean contributing value is taken across all countermeasures affecting the task in question. However, unlike U_t, each qualitative value x associated with a property maps to an integer in the range $-3 \leq x \leq 3$.

Based on these equations, we compute the task summative usability SU_t to be

$$SU_t = U_t + TU_t$$

Like U_t, the higher the score, the less usable the task is for the associated personas. After calculating U_t and SU_t, the score is normalised to a natural number in the range $0 \leq n \leq 9$. Given the potential of a task to increase or decrease usability, this value remains unchanged irrespective of it being a high positive or negative number.

Risk Analysis

In CAIRIS, we define a risk as the likelihood of a threat exploiting a vulnerability to cause an impact. Threats are synonymous to attacks, and vulnerabilities are properties of a system making it liable to exploitation.

A risk rating can be assigned based on likelihood and severity tables in IEC 61508 (IEC 1998-2005) (Figure 2). However, this rating does not reflect values held about individual assets or threats. To score risks with respect to the perceived value of the assets threatened, we define a security property as a row vector $\begin{bmatrix} c & i & a & o \end{bmatrix}$, where *c*, *i*, *a*, and *o* represent the values held for confidentiality, integrity, availability and accountability respectively. Each element *n* is valued $0 \leq n \leq 3$ based on whether the value held for that element is none, low, medium or high. The likelihood of the threat being realised, L_r, is computed using the equation

$$L_r = L_t - \overline{m_t}$$

where L_t is the likelihood of the threat *t* associated with risk *r*, and $\overline{m_t}$ is the mean likelihood value for the set of countermeasures mitigating the likelihood of L_t occurring. The values of L_t and $\overline{m_t}$ exist within the range $0 \leq n \leq 5$, and map to the likelihood categories in Figure 2. The severity of the vulnerability exposed by risk *r* is computed using the equation

$$S_r = S_v - \overline{m_s}$$

where S_v is the severity of the vulnerability *v* associated with risk *r*, and $\overline{m_s}$ is the mean severity for the set of countermeasures mitigating the severity of S_v. Like threat severity, vulnerability values exist within the range $0 \leq n \leq 3$ and map to the vulnerability categories in Figure 2.

Risk impact is described by a security property, representing the values held in the assets at risk from risk *r*. Risk impact is computed using the equation

$$P_r = (P_t \times P_a) - \overline{m_p}$$

where P_t is the security property of the threat associated with risk *r*, P_a is the security property of the vulnerable or threatened assets at risk, and $\overline{m_p}$ is the mean security property for the countermeasures targeting the risk's threat or vulnerability.

Finally, the calculation for the Risk Score of risk *r*, R_r, is computed, as the product of the threat likelihood, the severity of the vulnerability, and the risk impact to the threatened assets.

$$R_r = L_r \times S_r \times P_r$$

Each element of row vector is added together, and the sum is normalised to an integer between 1 and 9. If, during the above computations, negative numbers are calculated, these values are resolved to 0.

CAIRIS MODELS

One of the main differences between CAIRIS and related tools for security modelling is the model-driven nature of visualisation; models are automatically generated from specified, declarative data rather than via direct manipulation. This frees analysts from the tedious task of manually maintaining a variety of different models and the traceability relations between them.

Data elicited by CAIRIS is stored in a MySQL database conforming to the IRIS Metamodel (Faily & Fléchais, 2010), a conceptual model for usable secure requirements engineering. Rendering a tabular representation of the model data using the open-source Graphviz framework generates each model view. By using Graphviz's xdot output format, the position

Figure 2. IEC 61508 tables for threat likelihood, vulnerability severity, and risk categorisation

Score	Threat Likelihood
0	Incredible
1	Improbable
2	Remote
3	Occasional
4	Probable
5	Frequent

Score	Vulnerability Severity
0	Negligible
1	Marginal
2	Critical
3	Catastrophic

Score	Risk Rating
1	Intolerable
2	Undesirable
3	Tolerable
4	Negligible

	Consequence			
Frequency	**Catastrophic**	**Critical**	**Marginal**	**Negligible**
Frequent	1	1	1	2
Probable	1	1	2	3
Occasional	1	2	3	3
Remote	2	3	3	4
Improbable	3	3	4	4
Incredible	4	4	4	4

of different model elements is retained and, consequently, hit-testing can be supported in CAIRIS model viewer components; this allows analysts to click on nodes in a model viewer to obtain more information about the related model elements.

Several models can be automatically generated by CAIRIS based on elicited data. These are described in the following sections.

Asset Model

The CAIRIS Asset Model is represented as a UML class model, where assets are represented as classes. If an asset is used within a task then the persona associated with the task is also displayed in the asset model as an actor. Similarly, depending on the level of zoom used in the model, a comment node is also displayed to indicate the traceability origin of the asset and its relationship.

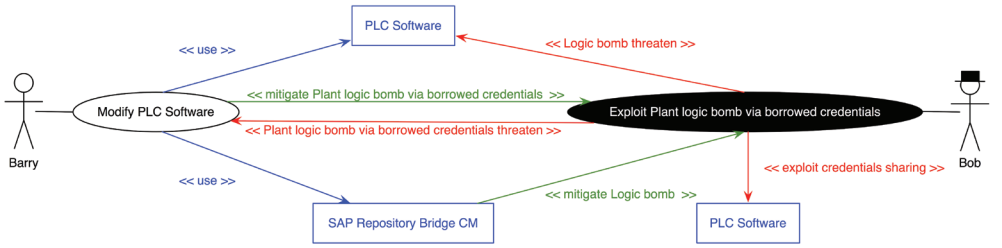
Assets may be generated from countermeasures as part of risk analysis. If this occurs, then an association between the asset at risk and the countermeasure protecting it is generated; this association is labelled with a <<safeguard>> stereotype.

Task Model

Personas and their task associations are represented using a modified form of UML Use Case diagram, where tasks are modelled as use cases, and personas are modelled as actors. This model also displays Misuse Cases and the attackers who realise them; attackers are displayed as actors wearing a black hat and Misuse Cases are represented as black ellipses with white text. Figure 3 is an example of a CAIRIS task model.

In the same manner that assets are associated with tasks, Misuse Cases are also, albeit indirectly, associated with assets. Each Misuse

Figure 3. Task Model example



Case in CAIRIS is associated with a risk and, by extension, with a single threat and vulnerability. Consequently, if an asset is used by a task and also exposed by a vulnerability or threatened by threat, then we can model this Asset-Misuse Case relationship in CAIRIS. Because these associations potentially add to model clutter in a large task model, these are associations are displayed based on the model's current zoom factor.

Goal, Obstacle, and Responsibility Models

We adopt a multi-model view of Requirements Engineering based on the goal-oriented KAOS methodology (Lamsweerde, 2009). KAOS defines goals as prescriptive descriptions of system intent, which are used as vehicles for refining requirements. The KAOS modelling notation is compliant with UML and, by extension, compatible with modelling notations commonly used by industry. The polygon KAOS model elements are also comparatively trivial to render visually. This is an important property for tool-support, which needs to rapidly compute and visualise the products of analysis without hindering participative design activities.

High-level goals stipulated by stakeholders at the beginning of a project can be refined by CAIRIS using goal trees; leaf goals can be refined as requirements, which can then be operationalised as tasks. Alternatively, a bottom up approach may also be taken where goals or requirements are elicited from tasks and retrofitted into the goal tree.

Obstacles can be identified from requirements and goals; these are conditions representing undesired behaviour and prevent an associated goal from being achieved (Lamsweerde & Letier, 2000). By refining obstacles, candidate threats and vulnerabilities may be defined.

Horizontal traceability is implemented using concern links. If assets or asset relationships of concern are identified in goals or tasks, these are automatically generated in the asset model.

Risk Analysis Model

The Risk Analysis model provides a quick-look view of the current risk analysis. This model only displays the elements of risk analysis, and other non-risk analysis elements associated with them. This view also compresses goal trees arising from risk responses, thereby making it easier to trace risks to mitigating responses, the requirements they treat, and the countermeasures they refine.

Risk Analysis model nodes are both colour coded and encoding with multidimensional data. As Figure 4 illustrates, information about the security properties is coded within asset and threat elements. Histograms indicate whether or not values are held for each property and, if so, whether that property is low, medium, or high. The colours selected for the confidentiality, integrity, availability, and accountability histograms are the 3 primary colours -- red, blue, and green -- together with black; the use of black and primary colours provide the maximum differentiation between property types (Tufte, 1990).

Risk analysis elements are colour coded with information, such as threat likelihood, vulnerability severity, and risk impact. Threats, vulnerabilities, and risks are also coloured based on their criticality: the more critical the element, the deeper the hue of red.

The Risk Analysis model also visualises metrics on requirements quality. These metrics are based on requirements completeness, the presence of an imperative phrase, and ambiguity. These are displayed using cartoon *Chernoff Faces* (Chernoff, 1973), and described in more detail by (Wilson et al., 1996). Eye-brow shape indicates the completeness of a given requirement. If no text is found in certain fields, or phrases like *TBC*, *None*, or *not defined* are present, the completeness score is marked down accordingly, and the eye-brows convey a negative mood. The eye shape indicates whether or not an imperative phrase exists in the requirement description. If such a phrase exists then the eyes becomes vertically elongated. The mouth indicates the presence of weak or fuzzy phrases, such as *mostly*, *appropriate*, *normal*, or *adequate*; the presence of these phrases turn the smile into a frown.

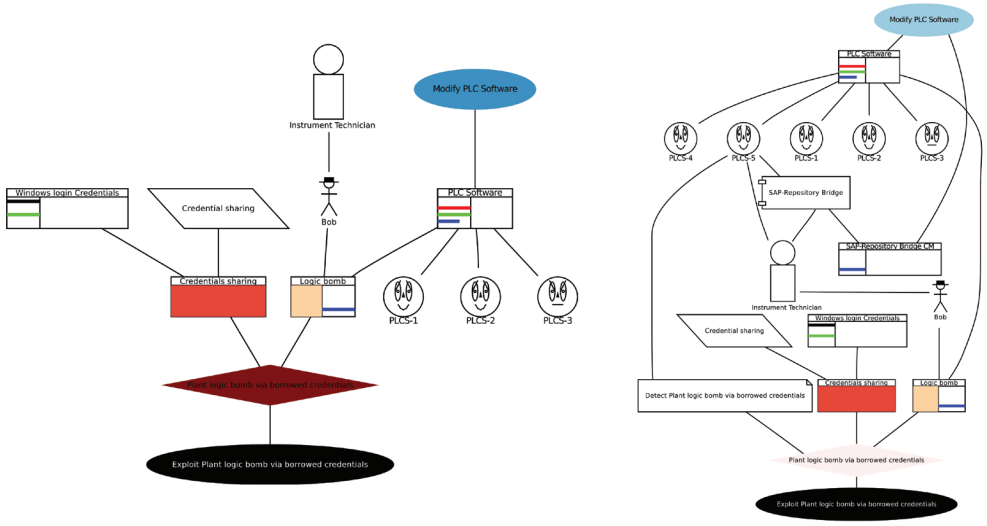
Clutter Management

With so much information associated with the different model views, visual clutter can become a problem as models grow. Minimal distinctions in colour can be used to reduce visual clutter, and small contrasts enrich the visual signal increasing the number of possible distinctions (Tufte, 1997). To take advantage of this, we map the normalised values for R_r and SU_t to the respective risk (red) and task usability (blue) colour charts. The higher the risk or task usability score, the deeper the hue of red or blue.

Threat likelihood and vulnerability severity scores map to a colour chart similar to that of risk. An example of how these colours are applied to elements on the IRIS risk analysis model is provided in Figure 4.

CAIRIS also uses geometric and semantic zooming to make efficient use of the available viewing area as model data increases. Geometric zooming magnifies detail at the cost of loss of context; semantic zooming conveys additional model information as it is zoomed (Spence, 2007). The risk analysis model supports 3

Figure 4. Risk Analysis model before (left) and after (right) risk mitigation



levels of zooming. At the lowest zoom factor, only the most salient elements are displayed. As the zoom factor increases, the further elements are displayed, together with the associations between them. At the highest zoom factor, all remaining elements are displayed, together with additional information about assets and threats. At this level of granularity, textual labels are also displayed; at lower levels, such detail would be unreadable and distracting. Zooming is also supported in the task model. Associations between personas and attackers and Tasks and Misuse Cases respectively are displayed at low zoom factors. Associations indicating that a Misuse Case threatens a Task, or a Task mitigates a Misuse Case are displayed at medium zoom factors. Associations indicating assets used by a Task, exploited or threatened by a Misuse Case, or assets mitigating a Misuse Case are displayed at high zoom factors. Figure 3 illustrates a task model at a high zoom factor.

Even with support for zooming, clutter remains a problem in large models when viewed at a high zoom factor. Consequently, filtering is also supported in certain models. Models can be filtered by task in the task model, and by node name and type, i.e., risk, threat, vulnerability, etc, in the risk analysis model; in these models, only the filtered model node and associated nodes are displayed. In the goal model, filtering is supported by goal name; when the filter is applied, the goal tree is re-displayed such that the filtered node becomes the root node.

CASE STUDY

In this section, we report on a case study where CAIRIS was used to support the specification of requirements for a central repository for control software; this repository was designed to support the work of instrument technicians at a UK water company.

Water and sewage treatment is controlled by a substantial amount of control software. This software runs on many different devices and locations across wide geographic areas. As part of their responsibility for maintaining the water network, instrument technicians make software modifications to telemetry outstations, PLCs (Programmable Logic Controllers), and SCADA (Supervisory Control and Data Acquisition) workstations. Without a central strategy for controlling such software, water treatment integrity may be compromised if software is lost, or incorrect software is accidentally, or deliberately, installed on critical instrumentation. However, because maintaining the water network can be physically and mentally taxing, any new technology needs to be situated for the contexts within which these technicians work.

Following an initial scoping workshop, empirical data was elicited from 3 contextual interviews (Holtzblatt & Jones, 1993) with instrument technicians, 2 on-site qualitative interviews, and 2 telephone interviews with related stakeholders. Transcripts of these interviews were analysed using qualitative data analysis, the results of which were used to identify the behavioural characteristics of potential users. From these behaviour characteristics, a number of personas were developed. This qualitative data was also used to inform a number of candidate requirements, vulnerabilities, and threats.

Three one-day workshops were held to carry out requirements analysis; participants included instrument technicians, software engineers, IT support staff, and information security officers. Each workshop began by validating the results of previous sessions before undertaking requirements analysis, and supplemental task and risk analysis. CAIRIS was used by a joint facilitator/scribe in each of these workshops to specify the artifacts of task, requirements, and risk analysis, and display different models to facilitate discussion and subsequent analysis activities.

For the purposes of brevity, this section focuses on how CAIRIS was used to elicit and analyse requirements relating to the modification of PLC software.

Category Definition and Asset Modelling

In the initial workshop, definitions were agreed and candidate assets were elicited. The meanings of Low, Medium, and High values were agreed for security properties, and categorical values for vulnerability severity were agreed and entered into CAIRIS. The IEC and ISO categories for threat likelihood and usability were also discussed with participants.

At this early stage, information about assets of value was also elicited and entered into CAIRIS. For this example, we focus only on two particular assets: PLC control software and repository access credentials. The security properties associated with PLC Software were Integrity (High), Availability (High), and Accountability (Medium). The properties associated with the repository access credentials were Confidentiality (High), and Availability (High).

Usability Analysis

For reasons of brevity, we focus on the work of Barry, the primary persona in the case study. Barry represented an instrument technician who modifies software as part of his day-to-day work. In several tasks, Barry made infrastructure changes to plant equipment, which led to control software modifications and, consequently, interaction with the software repository. One of these tasks, *Modify PLC Software*, began with Barry examining the details of the task on the SAP-based planning system, determining the required plant changes, and speaking to plant operators about the work. Barry then carried out the necessary modification work and commissioned (tested) the changes. When this task was completed to the satisfaction of the plant operators, Barry closed the job on the planning system, and uploaded the modified programs to the software repository.

Given Barry's profile, this task takes several hours, but only occurs on an infrequent basis. Due to the amount of work involved in this task, coupled with the importance of the task itself, this is a high demand task, which does not interfere with his goals. Based on this information, CAIRIS can compute the usability of this task:

$$\begin{aligned} U_t &= \frac{a \mp b}{2} + \bar{c} + \bar{d} \\ &= \frac{3+1}{2} + 3 + 1 \\ &= 6 \end{aligned}$$

Goal and Requirement Elicitation

From the scoping workshop, a high-level goal for maintaining control software was elicited; this was broken down to sub-goals for maintaining different classes of software. In this case study, we focus only on the elicitation of security requirements following the analysis of a goal for downloading PLC software.

In this simple example, a number of functional requirements (PLCS-2, PLCS-3, and PLCS-4) were directly elicited from the Download PLC software goal. However, in this context, an instrument technician must be authorised to make any software downloads and potential software modifications. We chose to generate obstacles on each of these goals to explore their consequences. Workshop participants were interested in exploring the unauthorised access of repository login credentials as a cause of unauthorised downloading. While analysing this goal, the assumption that organisational login credentials would be used for accessing the software repository was explicated. This domain assumption was modelled elsewhere in the goal model, and further obstacle refinement proceeded on the basis of this assumption. A number of leaf obstacles, one of which pertained to login credentials sharing, were identified. Based on this obstacle, a Credentials Sharing vulnerability was specified. Given the resources these credentials facilitate access to this vulnerability was scored as Critical.

The Chernoff Faces for these requirements in Figure 4 suggest quality problems with some of these requirements. In some cases, the eyebrow shape indicates that attributes, such as

fit criteria and rationale, are missing in some cases. In the case of PLCS-3, the requirements description ("A user shall be able to download the latest version of the PLC software for a site") is also ambiguous. This is due to the presence of the weak-phrase *be able to*, which is open to multiple interpretations.

Risk Analysis

Not all threats and vulnerabilities arose from requirements analysis. A *Logic Bomb* threat was defined based on the concern that an inside-attacker instrument technician might intentionally plant malicious code in PLC software to compromise water treatment. An example of such malicious logic involves turning off particular pumps in a water treatment plant at a designated time late one evening; this ensures the malevolent instrument technician receives a financially lucrative call-out to fix the problem. Alternatively, if another instrument technician's credentials are used, the Logic Bomb could undermine the company's confidence in his abilities. In the worse case scenario, turning off critical safety controls can lead to pollution of the water supply or substantial environment damage if raw sewage is released into the surrounding ecosystem. When carrying out this threat, the attacker wishes to hide his Logic Bomb within an innocuous code change carried out by another instrument technician. As such, the attacker looks to target the accountability property of this asset. All participants agreed that, dangerous as this threat is, its likelihood was low.

We defined a *Plant Logic Bomb via borrowed credential risk*. This risk occurs when an attacker carries out a *Logic Bomb* threat by exploiting the *Credentials Sharing* vulnerability. CAIRIS assessed this risk quantitatively by calculating its risk score. The likelihood and severity scores mapped to 1 and 2 respectively. The threat targeted only the accountability property of PLC software. The Windows login credentials, used to access the repository, are exploited by the *Credentials Sharing* vulnerability. Using this information, CAIRIS calculated the risk score R_r:

$$
\begin{aligned}
L_r &= L_t - \overline{m_t} \\
&= 1 - 0 \\
&= 1 \\
S_r &= S_v - \overline{m_s} \\
&= 2 - 0 \\
&= 2 \\
P_r &= (P_t \times P_a) - \overline{m_p} \\
&= ([0\ 0\ 0\ 3] \times [0\ 3\ 3\ 2]) \\
&= [0\ 0\ 0\ 6] \\
R_r &= L_r \times S_r \times V_r \\
&= 1 \times 2 \times [0\ 0\ \mathbf{0}\ 6] \\
&= [0\ 0\ 0\ 12]
\end{aligned}
$$

After rounding R_r down, the normalised score resolved to 9.

This risk was mitigated with a detective mitigation response, such that occurrences of this risk would be detected after the event. A goal was generated to reflect the objective of detecting this risk and, after goal refinement, requirements for peer-reviewing PLC software changes were elicited. One of these requirements stipulated that an instrument technician making a software modification cannot be the technician selected to carry out a peer review. Based on this, a *SAP-Repository bridge* countermeasure was defined. This countermeasure was a software component for cross-checking a peer reviewer with an instrument technician responsible for a modification. This countermeasure is considered reasonably effective at targeting the *Logic Bomb* threat, motivating the value of 2 (Medium) for $\overline{m_t}$. This countermeasure also fosters a high value of accountability, giving rise to a score of $[0\ 0\ 0\ 3]$ for $\overline{m_p}$. Based on this information, the risk score can now be re-evaluated.

$$
\begin{aligned}
L_r &= L_t - \overline{m}_t \\
&= 1 - 2 \\
&= -1 \\
S_r &= S_v - \overline{m}_s \\
&= 2 - 0 \\
&= 2 \\
P_r &= (P_t \times P_a) - \overline{m}_p \\
&= ([0\ 0\ 0\ 3] \times [0\ \ 3\ 3\ 2]) - [0\ 0\ 0\ 3] \\
&= [0\ 0\ 0\ 3] \\
R_r &= L_r \times S_r \times V_r \\
&= 0 \times 2 \times [0\ 0\ 0\ 3] \\
&= [0\ 0\ 0\ 0]
\end{aligned}
$$

These results show that while the accountability security value remains threatened, the likelihood of the threat was rendered inert, thereby reducing the risk score to the lowest possible value. As this countermeasure appeared to be effective, a new asset was defined for this component. The security property of this asset was based on the values placed on the countermeasure.

This countermeasure also positively influenced the task of modifying PLC software. A software component linking the repository with the SAP based work system means that job data can be used to support modification comments and, potentially, the job can be closed off by uploading a modification to the repository; this changes leads to the task being slightly less mentally taxing. Therefore, the summative task usability can now be re-evaluated.

$$
\begin{aligned}
SU_t &= U_t + TU_t \\
&= 6 + \frac{0+0}{2} - 1 + 0 \\
&= 5
\end{aligned}
$$

The risk analysis model before and after risk mitigation in Figure 4 illustrates how the differences in R_r and SU_t are represented using different colours; the risk node colour has changed from a dark to a light shade of red, while the task node in the mitigated model is now a lighter shade of blue.

CONCLUSION

Reasoning about security and usability is a challenge during requirements analysis, not least because analyst bias and data explosion can occur as specification and design activities progress. This challenge motivates the need for tool support to manage the results of this analysis, and use these results to further inform security and usability requirements analysis.

This paper has introduced CAIRIS: tool-support for usable secure requirements engineering. Although CAIRIS incorporates much of the functionality found in classic Requirements Management tools, elicited data is structured using a conceptual model for usable security. This allows CAIRIS to analyse risk and task analysis data as it is specified, and automatically generate different views of collected data. Our approach has shown that empirical usability and risk analysis data can be put to good use by applying simple qualitative and quantitative techniques to evaluate risks and tasks. By using this data with simple visualisation techniques, we can validate assumptions underpinning analysis, and explore the impact of certain specification and design designs.

Future work will examine the challenges associated with integrating CAIRIS with other tools, which support downstream secure software engineering activities.

ACKNOWLEDGMENT

The research described in this paper was funded by EPSRC CASE Studentship R07437/CN001. We are very grateful to QinetiQ Ltd for their sponsorship of this work.

REFERENCES

Alexander, I. (2002). Initial industrial experience of misuse cases in trade-off analysis. In *Proceedings of Requirements Engineering, IEEE Joint International Conference* (pp. 61-68). Washington, DC: IEEE.

Chernoff, H. (1973). The Use of Faces to Represent Points in K-Dimensional Space Graphically. *Journal of the American Statistical Association*, 68.

den Braber, F., Hogganvik, I., Lund, M. S., Stølen, K., & Vraalsen, F. (2007). Model-based security analysis in seven steps - A guided tour to the CORAS method. *BT Technology Journal*, *25*(1), 101–117. doi:10.1007/s10550-007-0013-9

Faily, S., & Fléchais, I. (2010). A Meta-Model for Usable Secure Requirements Engineering. In *Proceedings of the Software Engineering for Secure Systems* (*SESS '10)*.

Feather, M. S., Cornford, S. L., Kiper, J. D., & Menzies, T. (2006). Experiences using Visualization Techniques to Present Requirements, Risks to Them, and Options for Risk Mitigation. In *Proceedings of Requirements Engineering Visualization (REV '06), the First International Workshop* (p. 10).

Fléchais, I., Mascolo, C., & Sasse, M. A. (2007). Integrating security and usability into the requirements and design process. *International Journal of Electronic Security and Digital Forensics*, *1*(1), 12–26. doi:10.1504/IJESDF.2007.013589

Hogganvik, I. (2007). *A graphical approach to security risk analysis*.

Holtzblatt, K., & Jones, S. (1993). *Contextual Inquiry: a participatory technique for systems design* (pp. 177-210).

Houmb, S. H., Islam, S., Knauss, E., Jurjens, J., & Schneider, K. (2009). Eliciting security requirements and tracing them to design: an integration of Common Criteria, heuristics, and UMLsec. *Requirements Engineering*, 1–31.

IEC. (1998-2005). *IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems. Parts 1-7*. Geneva, Switzerland: International Electrotechnical Commission.

ISO. (1998). *ISO 9241-11. Ergonomic requirements for office work with visual display terminals (VDT) s - Part 11 Guidance on usability (Tech. Rep.)*. Geneva, Switzerland: ISO.

Lamsweerde, A. v. (2009). *Requirements engineering: from system goals to UML models to software specifications*. Hoboken, NJ: John Wiley.

Meland, P. H., Spampinato, D. G., Hagen, E., Baadshaug, E. T., Krister, K.-M., & Velle, K. S. (2008). SeaMonster: Providing tool support for security modeling. In *Proceedings of NISK 2008*.

Preece, J., Rogers, Y., & Sharp, H. (2007). *Beyond Interaction Design: Beyond Human-Computer Interaction*. New York: John Wiley & Sons, Inc.

Pruitt, J., & Adlin, T. (2006). *The persona lifecycle: keeping people in mind throughout product design*. Amsterdam: Elsevier.

Respect-IT. (2007). *Objectiver*. Retrieved from http://www.objectiver.com

Robertson, J., & Robertson, S. (2009). *Volere Requirements Specification Template: Edition 14 - January 2009*. Retrieved from http://www.volere.co.uk/template.htm

Rosson, M. B., & Carroll, J. M. (2002). *Usability engineering: scenario-based development of human-computer interaction*. San Francisco, CA: Academic Press.

Røstad, L. (2006). An extended misuse case notation: Including vulnerabilities and the insider threat. In *Proceedings of REFSQ, the 12th International Working Conference on Requirements Engineering*.

Seffah, A., & Metzker, E. (2004). The obstacles and myths of usability and software engineering. *Communications of the ACM*, *47*(12), 71–76. doi:10.1145/1035134.1035136

Spence, R. (2007). *Information Visualization: Design for Interaction*. Upper Saddle River, NJ: Pearson Prentice Hall.

Thimbleby, H. (2007). *User-centered methods are insufficient for safety critical systems*.

Tøndel, I. A., Jaatun, M. G., & Meland, P. H. (2008). Security Requirements for the Rest of Us: A Survey. *Software, IEEE*, *25*(1), 20–27. doi:10.1109/MS.2008.19

Tufte, E. R. (1990). *Envisioning information*. Cheshire, CT: Graphics Press.

Tufte, E. R. (1997). *Visual Explanations: Images and Quantities, Evidence and Narrative*. Cheshire, CT: Graphics Press.

van Lamsweerde, A. (2004). Elaborating Security Requirements by Construction of Intentional Anti-Models. In *Proceedings of the 26th International Conference on Software Engineering (ICSE '04)* (pp. 148-157).

van Lamsweerde, A., & Letier, E. (2000). Handling obstacles in goal-oriented requirements engineering. *Software Engineering*, *26*(10), 978–1005. doi:10.1109/32.879820

Wilson, W., Rosenberg, L., & Hyatt, L. (1996). Automated quality analysis of natural language requirement specifications. In *Proceedings of Fourteenth Annual Pacific Northwest Software Quality Conference*.

Shamal Faily is a doctoral student at the Computing Laboratory at the University of Oxford. His doctoral research involves understanding how factors relating to 'context of use' impact security, and how these factors can be applied to secure systems design. Shamal graduated with a BSc in Business Computing Systems from City University, and spent nearly 10 years as a software engineer at Logica UK.

Ivan Fléchais is a Departmental Lecturer in the Software Engineering Programme at Oxford University and his main lecturing and research interests are in the area of computer security. In particular, given that people are the weakest link in the security chain, this involves researching how secure systems can be designed, implemented and tested to take human needs into account. Prior to this, he graduated with a BSc in Computer Science from University College London and then stayed on at UCL to achieve a PhD researching how to design secure and usable systems which resulted in the creation of the AEGIS secure system design methodology.