Requirements Eng (2016) 21:209–223
DOI 10.1007/s00766-014-0217-8

ORIGINAL ARTICLE

Finding and resolving security misusability with misusability cases

Shamal Faily · Ivan Fléchais

Received: 5 March 2014 / Accepted: 25 November 2014 / Published online: 2 December 2014

Abstract Although widely used for both security and usability concerns, scenarios used in security design may not necessarily inform the design of usability, and vice-versa. One way of using scenarios to bridge security and usability involves explicitly describing how design decisions can lead to users inadvertently exploiting vulnerabilities to carry out their production tasks. This paper describes how misusability cases, scenarios that describe how design decisions may lead to usability problems subsequently leading to system misuse, address this problem. We describe the related work upon which misusability cases are based before presenting the approach, and illustrating its application using a case study example. Finally, we describe some findings from this approach that further inform the design of usable and secure systems.

Keywords Goals · Personas · Scenarios · Use cases · Security

1 Introduction

In 1994, Nielsen claimed that cost was the principle reason why usability engineering techniques are not used in practice [47]. While the financial costs of applying many usability techniques have been reduced by technology advances, other tangible and intangible costs have arisen to take their place. Many of these costs arise because of the expectations we increasingly place on these systems; we expect systems to be secure, but we also expect them to be usable. Techniques from secure software engineering and usability engineering help reason about how effective design decisions might be at mitigating risks or improving the effectiveness of a user's tasks, respectively. Yet, developers may believe that their knowledge about user goals and expectations negate the need for interaction design, or their understanding of the system's risks and mitigating controls negates the need for any security analysis. In such cases, developers may feel security and usability engineering approaches are useful, but they do not believe that the pay-off justifies their cost.

Scenarios are widely used by both security and usability professionals, but for different reasons. To usability professionals, they describe how people use a system to carry out activities that achieve their personal or occupational goals. To security professionals, scenarios describe how a system might be misused towards an attacker's own ends. Although scenarios are flexible enough to be used in both contexts, an artefact from one context is not necessarily useful in another. A scenario describing how a student returns a borrowed book to a library may provide no more insight into the security of a library's loan management system than a scenario describing how a professional hacker might carry out a denial of service attack on the library's web-server provides insight into the usability of the same system.

One way of engaging developers might be to encourage them to consider the negative impact that their design decisions could have on the systems they are building. A system could go wrong in many different ways; it could be exploited by an insider or external attacker, or it could be sufficiently unusable that key stakeholders lose confidence in its capabilities. By focusing on the different unintended

S. Faily (✉)
Bournemouth University, Poole, UK
e-mail: sfaily@bournemouth.ac.uk

I. Fléchais
University of Oxford, Oxford, UK

consequences of a system, developers may be encouraged to track problems back to their root causes. There is, however, a need to see how security and usability contribute to each other during system design activities.

This paper describes misusability cases: scenarios which describe how design decisions may lead to usability problems subsequently leading to system misuse. Section 2 describes the related work upon which our technique is built. We present misusability cases in Sect. 3 together with a validating example in Sect. 4. We conclude in Sect. 5 by discussing how the findings from this work further inform design approaches for usable and secure systems.

2 Related work

2.1 Designing usable security

In their seminal paper on information security, Saltzer and Schroeder [50] espouse several design principles for protection mechanisms. One of these is the principle of psychological acceptability, which states: *The interface to the protection mechanism should be designed for ease of use, so users routinely and automatically apply the mechanism correctly*. In recent years, there has been an enormous amount of research in the area of HCI security; seminal work in this area has looked at the usability of password policies [1] and security controls [59]. The HCI security community has considered the challenge of designing usable security from two different standpoints: design principles and user-centred security.

2.1.1 Design principles

The first standpoint is characterised by the application of usable security design principles. Examples of these include Yee's design guidelines and strategies for secure interaction [60]. These guidelines are written as principles a designer should consider when making decisions about secure interface design; examples include: *Match the most comfortable way to do tasks with the least granting of authority*, and *Indicate clearly the consequences of decisions that the user is expected to make*. More recently, attempts have been made to synthesise the many principles that exist towards general guidelines for usable cybersecurity [48].

While the contributions of Yee and others are sensible and well meaning, they are not the final word in designing usability security as many of these principles and idioms are also truisms. For example, [48] claims that *Cybersecurity usability should be considered early on*—it is hard to imagine anyone advocating a contrary view. The need for considering usability in security early on is not contested, what is not so obvious is *how* security usability should be considered and incorporated into a design or broader intervention early and in such a way that it does not interfere with productivity. Siponen [52] observes that the main value of such "common sense principles" is to change the conventional attitude that security is a human rather than technical problem, rather than providing rigorous guidelines with empirical evidence to support their suitability.

2.1.2 User-centred security and AEGIS

The second standpoint is characterised by Zurko and Simon's work on user-centred security [61]. User-centred security refers to "security models, mechanisms, systems, and software that have usability as a primary motivation or goal". Zurko and Simon claim that secure systems have, traditionally, been indifferent to the needs of users—irrespective of whether these are end-users, developers, or administrators. Like Yee [60], Zurko and Simon argue for security models conducive to the mental models of different types of users, but they also propose synthesising security design techniques with established design techniques from HCI. This work inspired, and continues to inspire, much of the current research in usable security. Surprisingly, the bulk of work in this community focuses on studying the usability of security controls rather than activities associated with designing systems that are usable and secure. For example, a survey by Birge [6] found that many HCI security papers are divided into studies about usability testing on security controls, mitigating security controls, conceptual investigations about terms such as "trust" and "privacy", experience studies about user attitudes, and models and guidelines demonstrating trusted interactions and trusted user interfaces. Birge also notes that much of this research focuses on the needs of the end-user rather than the needs of the designer, and few studies have attempted to tackle the question about how designers should approach security concerns.

Attempts to synthesise HCI and security techniques continue to raise issues for researchers. To understand why, we should consider one example of follow-on work: the Appropriate and Effective Guidance for Information Security (AEGIS) design method [30]. AEGIS assumes that secure systems are not merely software systems, but socio-technical systems: systems of technology used within a system of activity. To this end, AEGIS was designed as a lightweight process which augments existing software development methods that provide guidance to developers for designing secure systems. This process is applied within a focus group setting, where stakeholders gather,

identify, and model the system's assets in different contexts. These assets are evaluated according to values held by the participants about them. Vulnerabilities, threats, and risks affecting these assets are elicited, before possible security controls mitigating these risks are selected. The costs and benefits of situating these countermeasures for each of the affected contexts is considered and, if unmitigated risks remain, this process is repeated.

At a superficial level, AEGIS appeals to Zurko and Simon's canons for user-centred security. A more critical analysis of AEGIS does, however, raise issues; these affect not only AEGIS but the paradigm of user-centred security in general.

First, AEGIS assumes that usability will follow by taking a participative approach to design. However, work by Irestig et al. [36] found that while participative practices are effective for eliciting usage culture values that might have otherwise remained tacit, resulting systems tend to be comparatively chaotic, small-scale, and imbued with the power relationships of the organisation.

Second, although conceptually simple, UML class diagrams (which are used in AEGIS to model assets) cannot model alone the many elements which also impact secure systems, such as goals, tasks carried out by users, and other potentially relevant relationships, such as dependencies between different users. Moreover, as analysis progresses, models are likely to grow and become unwieldy without dedicated tool-support.

Finally, although treating environments as first-class modelling objects is an important step towards contextualising risk analysis, workshops alone may not be enough to elicit information about threats and vulnerabilities within different environments. Although Zurko and Simon have proposed applying techniques such as contextual inquiry [35] to elicit such data, we are unaware of any peer-reviewed literature which has attempted to do this in a security context.

2.1.3 Evaluating the design of usable security

Some researchers in the security and HCI communities have explored how evaluating the design of security might lead to more usable security controls. For example, Jøsang et al. [37] devised a selection of security usability vulnerabilities associated with security tasks, and the assessment of a systems security state. Such vulnerabilities can then be compared with security solutions, and possibly combined with relevant threat sources, to see where security might fail. [37] also consider strategies for improving the security and usability of security technologies. They accept that interface changes alone may not be enough and entirely new designs may be necessary to implement a usable security design.

Jøsang and his colleagues suggest that re-designing a security control with poor usability can be helped with suitable usability metrics. Subsequent work by Braz et al. [9] presented an approach where different steps of a task scenario are associated with supplemental security problems, usability criteria, and a collection of usability factors and metrics. This approach can be used by usability specialists and security designers to rate the impact of security and usability issues associated with different aspects of a particular task. This approach relies primarily on the expertise of security and usability experts, so does not require access to design models or systems stakeholders. However, while this approach is useful for evaluating different aspects of a design, it remains primarily an evaluation approach, with no guidance provided for a revising a systems requirements based on the evaluation.

2.2 Addressing usable security with user-centred design and requirements engineering

One proposal suggested by Zurko and Simon involves synthesising security engineering practices with *user-centred design*. User-centred design is concerned with an early focus on user goals and tasks, empirical measurement of users, and iterative design [33]. Several user-centred design techniques have also been usefully appropriated by requirements engineering practitioners. Two of the most prominent of these are *personas* and *scenarios*.

Personas are narrative descriptions of archetypical users that embody their goals and needs [15]. Because they are comparatively easy to develop and use, they are becoming popular for summarising user research about prospective system stakeholders [12]. Personas have also been proven useful when engaging stakeholders in security [24]. The activities necessary to create personas are also conducive to a security analysis because in addition identifying affordances for use, affordances for misuse and possible vulnerabilities can be identified at the same time [26].

Scenarios are stories about people carrying out an activity [49]. Their allusory power has not been lost on the requirements engineering community. They can be used to impart knowledge about human activities at any level of abstraction, provided the activities can be rendered in a narrative structure. Consequently, scenarios are a universal language in requirements engineering and can be used to support the elicitation, specification, and validation of requirements [3]. Sindre and Opdahl [51] have also proposed using scenarios to describe unwanted behaviour in a system. Such behaviour can be encapsulated in a *misuse case*: a sequence of actions, including variants, that a system or other entity can perform, interacting with misusers of the entity, and causing harm to some stakeholder/s if the sequence is allowed to complete.

While popular, user-centred approaches are not without their flaws. From a security design perspective, it is important to understand these weaknesses; these may lead to the unintentional introduction of vulnerabilities into a requirements specification.

First, user-centred approaches assume that usability design should precede general system design. Cockton [14] argues that invention will precede innovation, and while concurrent design may be possible, an iterative design process beginning with human factors work is an unrealistic aspiration.

Second, user-centred approaches may be implicitly biased against software engineers. For example, Cooper [15] argues that developers will, given the opportunity, design any given software product for themselves, and that many examples of bad usability can be attributed to software developer indifference to usability. In particular, Cooper's argument is founded on the failure of a particular Microsoft project described in [45], where a culture of usability allegedly clashed with a culture of engineering. However, an alternative reading of [45] also suggests that poor requirements engineering practices on the part of the usability designers and a failure to consistently respond to developer requests for a working requirements specification may have been as much to blame. Consequently, it is equally possible that [45] strengthens, rather than rebuts, Cockton's more recent observations. A similar anti-engineer bias has been observed by Thimbleby [55], who claims that many usability professionals believe that technologists are the origin of usability problems, and that these problems can be solved by extolling the virtues of user-centricity and acting as user proxies.

Third, user-centred approaches may, in some cases, be methodologically weak. For example, Chapman and Milham [11] report little peer-reviewed discussion of the Personas technique validity and, as a result, it may be impossible to verify their accuracy. Moreover, because personas are fictional representations, there is no easy way to falsify them. Consequently, if a persona is developed using questionable methods, or less than accurate empirical data, it is difficult to disprove a persona's validity.

Finally, knowing about usability problems is different from being willing and able to fix them. Knowing that there are usability problems is, however, important, but while user-centred approaches are necessary and useful, they are not in themselves sufficient. This is because user-centred design techniques appear to focus solely on the world external to the user interface without considering the usefulness of formal specifications for modelling complexity and ambiguity; such specifications can, potentially, highlight the causes of poor user interface design [56].

Sutcliffe [54] has compared and contrasted the traditions in both HCI and software engineering concerning design and its theoretical underpinnings. Sutcliffe concluded that scenarios are indeed boundary objects between these disciplines, but that both occasionally carry out research within each other's bailiwick. In one sense, this might be seen as competition, but it could also be viewed as convergence between disciplines. Both disciplines do appear to converge when dealing with interaction concerns, although the viewpoints of each differ in both cases.

These different viewpoints suggest that care needs to be taken when using scenarios to address concerns that cross-cut these disciplinary perspectives, particularly when security is involved. To see why, we should consider Alexander's proposed use of misuse cases for examining usability issues [2]. Alexander presents an example where confusion about the use of an interface causes a novice user to become a negative agent. A corollary of this approach is that the user is typecast as an attacker; this is analogous to treating the user as the cause of poor interface design. However, Brostoff and Sasse argue that this position is tantamount to blaming users for a security design they might compromise, and analogous to blaming the cause of safety-critical system failures on human error rather than bad design [10].

2.3 Aligning user-centred design with requirements and security engineering using IRIS

Sutcliffe [54] claims that synthesis between different disciplinary design perspectives may be possible if instead of focusing on heavyweight methods, a spectrum of complementary approaches is adopted. To explore this claim, our research has examined how complementary user-centred design practices can be integrated with both requirements and security engineering. This work has led to the development of the Integrating Requirements and Information Security (IRIS) meta-model: a conceptual model for usable secure requirements engineering [22]. The IRIS meta-model extends existing work in user-centred design, security and requirements engineering by including concepts which allow the usability of tasks, and the usability of impact of security design decisions to be modelled. The IRIS meta-model extended related meta-models in security requirements engineering, such as [29, 43, 44] to ensure reuse of existing concepts, but was as parsimonious as possible when declaring model concepts, and model relationships were simplified to make conceptual associations as clear as possible. The meta-model itself sub-divided into five views: task, goal, risk, responsibility, and environment. Together, these views make it possible for security designs to be assessed in different contexts of use.

The IRIS meta-model formed the basis of the IRIS process framework, which guides technique selection when specifying usable and secure systems [26]. The process

framework is also complemented by the open-source Computer Aided Integration of Requirements and Information Security (CAIRIS) requirements management tool, which demonstrates how model elements from security, requirements, and usability engineering can be managed and analysed [21].

Goals are a central feature of IRIS, and form the basis of a specification that a particular system needs to satisfy. As such, IRIS' definition of goal, which is based on the definition used by the KAOS approach [39], is analogous to a system goal. These goals are progressively refined, and leaf goals become the responsibility of stakeholder roles; these are responsible for ensuring the goal is satisfied. Goals can be operationalised as *tasks*; these are scenarios exploring the relationship between the specified system and the intended users. These intended users are modelled as *personas*, which use the functionality specified by the system goals to carry out activities of importance to them.

The same system goals facilitating these tasks can be threatened using KAOS *obstacles*: conditions representing undesired behaviour that prevent an associated goal from being achieved [40]. Subsequent refinement of obstacles may lead to the elicitation of vulnerabilities or possible threats. When attackers carry out threats exploiting these vulnerabilities, risks can be defined. Misuse cases act as a validation of this risk analysis exercise. If a risk is valid, then a believable misuse case should be written; this describes how the attack associated with the risk exploits the risk's vulnerability to harm the endangered or exploited assets.

Although user goals are not an explicit concept in IRIS, subsequent work [20] has shown how the characteristics of personas and the activities they engage in align with the i* based goal-oriented requirements language (GRL) [4]. This alignment makes it possible to generate GRL models based on personas and use cases. With appropriate tool-support, this may facilitate complementary analysis of trade-offs arising from a reconfiguration of persona activities, goals, and the dependencies between them [19]. Further exploration of this alignment between IRIS and GRL is outside the scope of this paper.

Although quantitative data analysis based on this meta-model allows the impact of security decisions on the usability of tasks to personas to be modelled, the only way misusability can be explored is by treating personas as attackers, or eliciting obstacles giving rise to secure misusability. If we consider obstacles as exceptional behaviour, then use cases might form the basis of eliciting such obstacles. Use cases are sequences of actions a system performs yielding an observable result of value to a particular *actor*, i.e. someone or something outside the system that interacts with the system [38]. These are synonymous with scenarios and are a commonly used requirements elicitation technique. Although they often describe the normal course of an actor's usage of a system, extensions for exceptional behaviour can be associated with individual steps [13]. Using this approach, accidental misuse of a system can be elicited without typecasting a persona as an attacker, but only if the requirements giving rise to the misuse are known. It may be case that all we have are clues to what this misuse might be. These might include possible ambiguity in a specification, or some assumptions about a persona's behaviour that, in some cases, might cause certain types of behaviour.

2.4 Designing for the negative

To discover the contributing factors to inadvertent misuse or abuse of a system, we must look beyond the classic view of systems used precisely as their designers intended. An example of such thinking is Dunne and Raby's work on *Design Noir* [18], which argues that our emotions and needs are played out in technology across a much broader spectrum of use originally envisaged by its designers.

Although previous work has considered the abuse and misuse of social agents [8], Nathan et al.'s work on Value Scenarios [46] is one of the few examples of work where scenarios are used to describe both the positive and negative systematic effects of technology without considering users as malevolent. Value scenarios are vignettes which describe the systematic effects of a system to both direct and indirect users over an extended period of time. These scenarios describe the negative impact of forgetting about certain values, such as prejudice and inequality.

Not all software systems are as divisive, pervasive, or long-running as those typically described by value scenarios. Nevertheless, it may be possible to stimulate similar narratives with more supplemental information about the system, its users, and its contexts of use. This information may not be available during the early stages of design where it is envisaged that value scenarios should be employed, but it might be available from the data collected during later stages.

2.5 Bridging techniques with assumptions

The IRIS meta-model supports the KAOS concept of *domain properties* to capture assumptions expected to hold by the system, but it does not consider assumptions that underpin the meta-model concepts themselves. Without a specific means of associating these assumptions with design, identifying the associations between misuse and the system design decisions that contribute towards them remains an ad hoc affair. Subsequent work [25] has, however, examined how structured assumptions can be used to better ground the construction of *assumption*

personas. Building on Toulmin's work on developing practical arguments [57], we have illustrated how assumptions can be structured and aligned with persona characteristics. This work suggests that a characteristic is analogous to a *claim* being made as part of an argument, and a proposition reflecting an assumption about this characteristic may act as evidence *grounds*, or a *warrant*: a rule of inference describing how the grounds contribute to the claim. The origin of a warrant's assumption is the *backing* knowledge for believing the claim, and assumptions may also be used as *rebuttals*, which challenge the validity of the claim. Finally, a *modal qualifier* indicates the degree of certainty about the claim. Our use of Toulmin's model of argumentation is inline with growing security requirements engineering best practice for visualising design rationale, e.g. [32, 34].

In a case study where personas were used to help elicit secure system requirements [24], it was observed that the contextual data used to elicit personas also inform the elicitation of vulnerabilities. Based on the effectiveness of the Toulmin model for grounding the characteristics of personas, it may be possible to structure contextual data suggesting possible system or user ambiguity in a similar manner. Such characteristics may form the basis of establishing contexts where security misusability can occur.

3 Approach

Misusability cases, which were first proposed by [27], are scenarios where a persona achieves a personal or work objective, but inadvertently exploits one or more vulnerabilities in order to do so. Misusability cases have two objectives. First, to identify cases of insecure misuse within the context of use where activities are carried out using a designed system. Second, to elicit the root causes of this misuse, together with the system goals which mitigate them.

As Fig. 1 illustrates, misusability cases do not exist in isolation and are tightly integrated into the IRIS meta-model. Before applying the technique, we assume system goals have been elicited; these correspond to the requirements the systems needs to satisfy. We also assume that one or more personas have also been elicited to reflect different roles the system needs to be designed for. As the figure also illustrates, some of these goals in this goal model are operationalised as use cases. A misusability case realises a use case describing a particular episode of system behaviour carried out by an actor. Each actor associated with the use case is realised by a persona.

Misusability cases are modelled using the IRIS concept of task. The factors leading to this exploitation are myriad; a user might accidentally circumvent application error checking, misread software instructions, or carry out behaviour that conflicts with organisational security policies. However, in all cases, these factors arise because of vulnerabilities in the context of use within which the task takes place.

To facilitate misusability cases within IRIS, two modest changes were made to the task view of the IRIS meta-model published in [22]; this revised model is presented in Fig. 2.

The first change involved repurposing the changes made to the IRIS meta-model in [25] for assumption personas to also support misusability cases. These concepts are shown on the left-hand side of Fig. 2. *Characteristic* was introduced to describe a motivated characteristic of a persona. These are, as indicated in Fig. 1, grounded in assumptions.

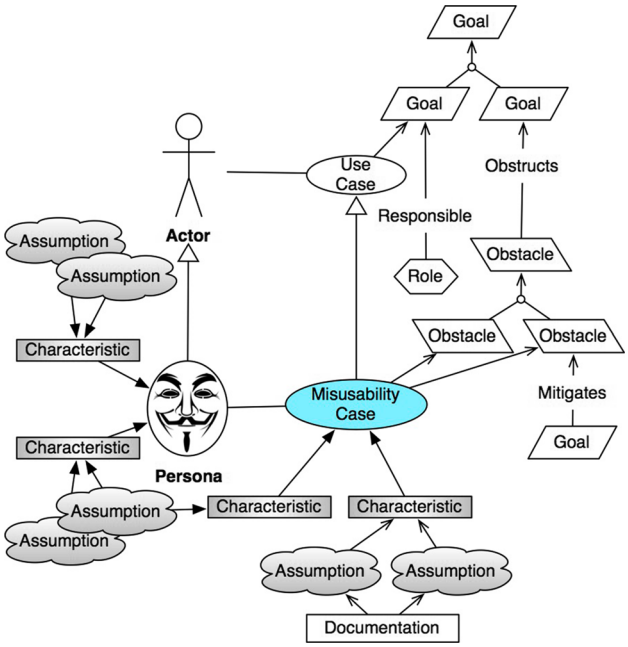

Fig. 1 Misusability case with other design concepts

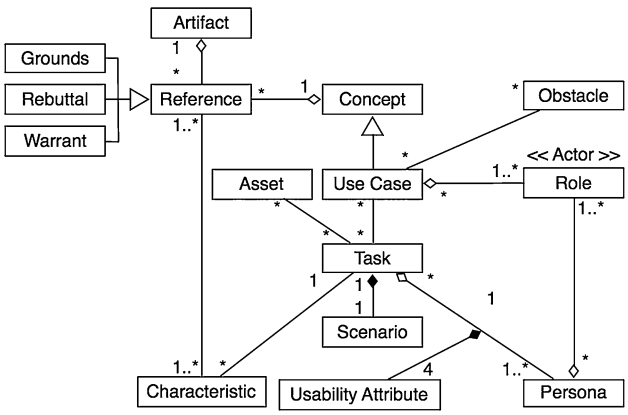

Fig. 2 IRIS task meta-model updates for misusability cases

These assumptions are modelled as *references* (be these grounds, warrants, or rebuttals), which were backed up either by some externally documented *artefact* or by some pre-existing security, usability, or requirements *concept* within an IRIS model.

The second change involves introducing the concept of *use case* to the meta-model, and the re-purposing of the existing concept of *role* to encompass use case actors. Use cases were not incorporated into the initial version of the IRIS meta-model because, given the presence of the *task* concept, use cases appeared to be superfluous when the IRIS meta-model was validated in two initial industry case studies (described in [22, 26]) where requirements were elicited and specified. In these studies, however, the IRIS process framework was used exclusively, i.e. there were no constraints for the IRIS model to be interoperable with any existing or planned design models. Given the ubiquity of use cases for specifying, managing, and validating requirements, not including this concept affects the scaleability of the IRIS meta-model. Moreover, pre-existing use cases which are incomplete or ambiguously defined may provide a useful source of misusability data. These may be specified at an early stage of design, but not updated as a design evolves, even though they may be used as an authority when considering user needs.

In the following sections, we describe how misusability cases are elicited and applied.

3.1 Eliciting misusability cases

The first step involves identifying implicit assumptions being made about the design related to a use case. A variety of techniques can be used to discover these assumptions. When analysing documentation, such as architectural design documents or user manuals, techniques proposed by Dewar's assumption-based planning methodology [17] are particularly useful. These techniques include using journalist questions (Who? What? When? Where? Why?, and How?) about items of data, and looking for instances of text where the words *will* and *must* are used.

Using the conceptual model described by Fig. 2, references are created for each assumption. Each reference contains a statement summarising the assumption, a link to the source material, together with an excerpt from the source material justifying the assumption. References may also be elicited from design artefacts, such as personas.

Once a collection of references have been elicited, the characteristics of a convincing misusability case are developed. The process for developing misusability case characteristics are analogous to those used for developing persona characteristics. A claim is made about some characteristic of the system which might be liable for misuse. The references are used to act as grounds or a warrant to this claim or, if necessary, a rebuttal. Finally, a modal qualifier is associated with the characteristic based on the analyst's confidence in the claim.

The final stage involves writing a supporting task satisfying these characteristics while, simultaneously, carrying out the steps within the use case. Enacting the task is the persona fulfilling the use case actor's role. The behaviour exhibited by the persona should be commensurate with the characteristics built into the task; if the characteristics of the misusability case as such that they conflict with the persona's objectives then this should be reflected in the task narrative.

3.2 Applying misusability cases

The next stage involves identifying the obstacles directly contributing to the different aspects of misusability in the misusability case. Based on these obstacles, the higher-level obstacles these lower-level obstacles help satisfy are elicited. This step continues until system goals are identified, or new goals are elicited, which are obstructed by these obstacles. Although this step could be construed as an exercise in bottom-up analysis, fitting the misusability case and its contributing obstacles into the larger goal model necessitates both top-down and bottom-up analysis.

Once the misusability case has been reconciled with the system goal model then one of two actions may be possible. Eliciting both the misusability case and contributing obstacles may have provided insights suggesting new goals to resolve the obstacles identified. If this is the case then these are added to the goal model. Alternatively, it may not be possible to mitigate the elicited obstacles because further investigation into the problem domain is needed, or the controls needed to mitigate the obstacles are out of scope. In such cases, the obstacles are assigned to a particular role. This role is responsible for further analysis leading to eventual mitigation of the obstacle. Explicitly assigning the obstacle to a role mitigates the possibility of *diffusion of responsibility*, where unresolved problems are ignored because no single agent is responsible for them [16].

4 Misusability cases in practice

We now provide an example of how misusability cases were used in a project to develop a portal for sharing medical study data [42]. The portal was primarily designed to serve two particular user communities: academic researchers and data managers. Academic researchers use the portal and its resources to find re-usable data sources, and contribute to the portal's documentation server and portal. Data managers work within particular study units

and are responsible for curating study data, its meta-data, and making authorised study data available on the portal.

To support portal development, we elicited additional security requirements for the portal's meta-data repository (MDR): a database allowing researchers to discover meta-data about different studies. Ideally, both security and usability should be designed into a system at a very early stage. Our involvement with the project commenced not at its initial inception, but once the main architecture and component sub-systems had been outlined. Moreover, despite the fact that contact and engagement from representative stakeholders could have made an invaluable contribution to our work, the project scope was such that data could not be collected from prospective researchers or data managers. It was, therefore, necessary to use the portal development team, who had spent considerable time working with the different user communities, as proxy users. We also attempted to make best use of the available project documentation. This documentation included requirements specification that was developed for the portal at the very early stages of the project. These requirements were elicited by a team of analysts with access to representative data managers. The analysts elicited a collection of ten scenarios illustrating how researchers and data managers might use the portal and, on the basis of these, 70 functional and non-functional requirements were derived. These non-functional requirements included 11 security requirements which, with two exceptions, were exclusively concerned with access control of the MDR data. This bias for access control and confidentiality is understandable; the nature of the project was such that certain aspects of meta-data were sensitive and likely to be harmful to study participants if disclosed.

4.1 Methods

The process for eliciting the additional security requirements for the MDR, which the creation and application of misusability cases was part of, is illustrated in Fig. 3 and detailed in the sub-sections below.

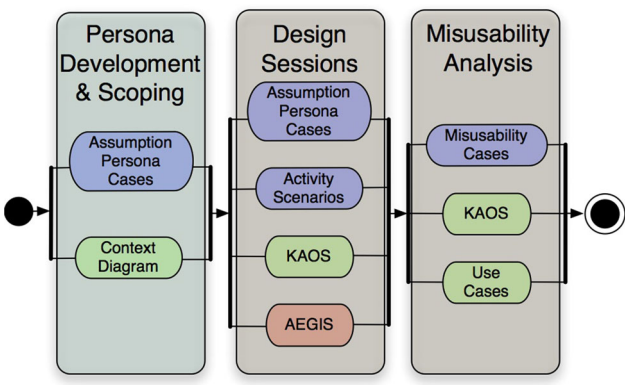

Fig. 3 Instantiated process for the MDR specification

4.1.1 *Persona development and scoping*

While the project team's responsibility formed a natural scope of analysis around the MDR, there was concern that security issues might cross-cut organisational boundaries; such issues may be the responsibility of one team, but the adverse impact could affect others. It was also necessary to understand the different expectations held about the prospective MDR user-community. To deal with both of these issues, it was decided to identify implicit assumptions in the available documentation, and use these to form the basis of assumption personas. In doing so, the expectations about end-users are made explicit, and subsequent discussion around these confirm a useful boundary for the analysis to be carried out in later stages.

For each role relevant to the scope of analysis, the available documentation was reviewed to elicit references for each role. These are used to establish persona characteristics and, based on these, assumption personas. The process for eliciting these assumption personas is described in more detail in [25].

Once the assumption personas were developed, these were presented to the project team for review. Issues raised by the team were used to revise the assumption personas or correct any misinterpretations held about the MDR.

4.1.2 *Design sessions*

By carrying out this intervention in parallel with on-going project activity, using participative design sessions alone was infeasible. Conversely, however, limited documentation artefacts meant that group-based design sessions would be required to elicit the data contributing to the requisite IRIS concepts.

The design sessions stage entailed holding small focus groups with project team members. Each session focused on the use of activity scenarios, KAOS, or AEGIS. An activity scenario session involved modelling scenarios carried out by the elicited assumption personas in their respective contexts. A KAOS session involved eliciting goals needing to be satisfied to enable the elicited scenarios to be realised. In both session types, assumption personas were used as an authority for user expectations; these were modified if aspects of the analysis challenge their characteristics. AEGIS sessions involved carrying out asset modelling for the different environments, and discussing possible attackers, threats and vulnerabilities that might arise due to environmental factors; based on these, several risks were identified.

Each design session was recorded and, following each session, the transcripts were analysed to elicit additional information about the scenarios, goals, and asset models elicited.

After the final session, each goal was examined and assigned a responsible role. Following this, a specification document was generated and sent to the project team members for review.

4.1.3 Misusability analysis

In order to help make assumptions more transparent and, simultaneously, help inform system usability and security using the work of the project team, a misusability analysis phase was appended to this process.

Using both previously analysed and new documentation produced by the project team, assumptions were identified leading to the elicitation of misusability cases. These artefacts were used to elicit contributing obstacles, together with the goals these obstacles obstruct. This activity stimulated innovative thinking about new goals for resolving these obstacles.

Elicited goals were refined to requirements and assigned a responsible role. Following this, a revised specification document was generated and sent to the project team members for review. Once the team was given sufficient time to review the analysis, a final wrap-up session was held where the final results were presented.

4.2 Results

The available project documentation was analysed to identify assumptions and, based on these, assumption personas for a researcher (Alex) and a data manager (Brian) were elicited.

Four in-situ design sessions were held with project team members; each session lasted between 40 min and 1 h. During each session, 1–2 developers worked with the primary author to elicit the use cases and tasks carried out by the personas, together with the goals that the portal would need to satisfy to support them. These sessions were supported by CAIRIS, which was used to manage the design data and evolve the system goal model as the sessions progressed. After the final design session, a requirement specification was generated by CAIRIS, and sent to the project team members for review.

Once it had been established that there were no issues with the analysis carried out to date, the goal model and related design artefacts, together with the project documentation were used as data sources for misusability case elicitation and mitigation. CAIRIS was also used to store elicited references, characteristics, misusability cases, and the obstacles and goals elicited on the basis of this additional analysis.

After the misusability cases had been elicited, and the final wrap-up session was held, a total of 42 requirements and 21 obstacles had been elicited. Of these 21 obstacles, 15 were elicited using misusability cases. Of the 42 requirements elicited, twice as many security requirements (22) were elicited using this process than were specified in the original portal requirements specification.

In the following sections, we describe an example of how one of these misusability cases was elicited and used during this study.

4.3 Developing the misusability case

To identify the functionality associated with importing study meta-data into the MDR, the following use case, *batch import meta-data*, was specified.

The data manager completes the fields of a mapping file and, from his web-browser, enters a URI to the meta-data upload page. When the upload page is displayed, the data manager enters the location of the mapping file and clicks on the Upload button. The system uploads the meta-data to the MDR based on the data in the mapping file and, after several minutes, acknowledges the successful import of the meta-data to the data manager.

The mapping file describes properties of the meta-data, such as the file names, locations, and security policies associated with the meta-data. Associated with this use case was a pre-condition that the meta-data itself has been prepared and ready for import.

While the use case suggests little to form the basis of security misuse, a number of clues were found in related artefacts. The MDR implementation guide suggested that utilities to support data managers in preparing their meta-data *may* be made available via a central repository on the portal. This central repository also hosted forums and best practice documentation about how to make best use of the portal. Other clues were found in the related persona description. Brian was found to irregularly use the portal and, because of his unfamiliarity with the mechanics of sharing data outside of his study unit, is unfamiliar with the process of importing data to the MDR. Consequently, a lack of documentation and best practice was likely to be a cause of frustration. We also know that Brian was adept at scripting, and was likely to script the process of satisfying the use case preconditions.

These assumptions were recorded as references and, using the argumentation structure described in Sect. 3.1, characteristics were elicited that unpinned a misusability case undermining the use case; this argumentation structure generated by CAIRIS is illustrated in Fig. 4. Using these

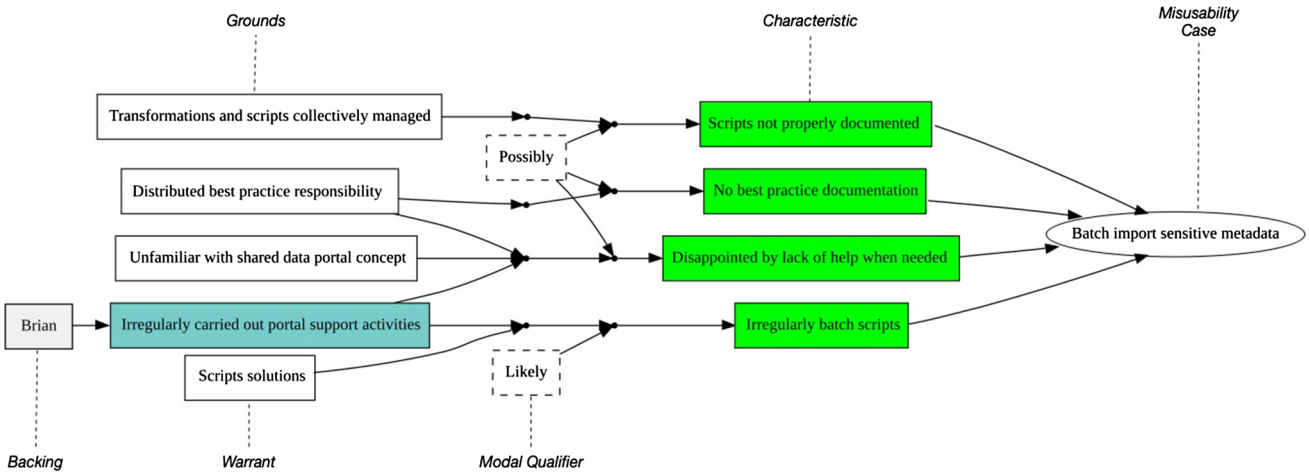

Fig. 4 Characteristics and argumentation structure underpinning a misusability case

characteristics as guidance, the *batch import sensitive meta-data* misusability case was written:

Brian had spent most of the morning preparing data-sets ready for export to various sources. Some of the meta-data was for deep [sensitive] meta-data for local databases, while others were shallow [summarised] meta-data targeted for the MDR. He hoped to use standards and guidelines on the gateway, but he was disappointed by the lack of anything useful that would help him. Nevertheless, Brian managed to organise his meta-data into the layout he inferred from some XSLT scripts he was able download. After finally finishing the preparation of his data-sets and meta-data, Brian created the mapping files needed for the data import process. Fortunately, most of them were very similar so most of the files he used were based on an initial template he created for one of his data-sets. Brian entered a URI he had been provided for uploading meta-data to the MDR, and logged in using his Data Manager credentials. Brian then specified the mapping file corresponding to the meta-data he wanted to upload and hit the Upload button. Several minutes after clicking the Upload button, Brian received a message from the portal indicating that the meta-data had been uploaded.

Although not written into the misusability case itself, Brian has inadvertently uploaded a mapping file for public meta-data, which points to sensitive and private meta-data. As a result, unauthorised meta-data had been made publicly available on the portal.

4.4 Mitigating the misusability case

Figure 5 presents an excerpt from the goal model generated by CAIRIS that is associated with the aforementioned misusability case; this misusability case is represented by the dark blue ellipse at the bottom of the figure. The colour of this figure is based on how usable Brian finds the activities described in this task; the darker the shade of blue, the less usable the task is. Further explanation of how the usability of tasks are calculated is outside the scope of this paper, but is detailed in [23].

As the figure illustrates, we identified three contributing obstacles which cause the misusability case to be realised. In this figure, obstacles are modelled as yellow rhomboids.

The first of these obstacles is the lack of documentation about the import layout. If we consider possible root obstacles, then we discover that satisfying this obstacle contributes to an obstacle of the MDR documentation being unavailable. One reason that this documentation is unavailable is because no one is explicitly responsible for publishing anything. We can, therefore, mitigate this first obstacle by specifying a goal (layout documentation) stating that the expected data layout shall be published when an import tool is made available to data managers.

The second obstacle points to a lack of contributed best practice documentation. The frustration caused by this, like the first obstacle, lengthened the time taken to complete the activity; this possibly increased the likelihood of the slip occurring during the misusability case; this slip was reflected by the inadvertent specification of the mapping file data policy as public. Although the management of portal documentation was largely de-scoped from this analysis, the impact of the obstacle affects the MDR. For this reason, the obstacle was assigned to the portal administrators to ensure it is addressed by the portal design team.

The third obstacle relates to the upload of the inappropriate meta-data to the MDR due to mis-specification of the mapping file template. The obstacle states the sensitive meta-data was specified as publicly accessible. This obstacle is too granular to immediately suggest a mitigating

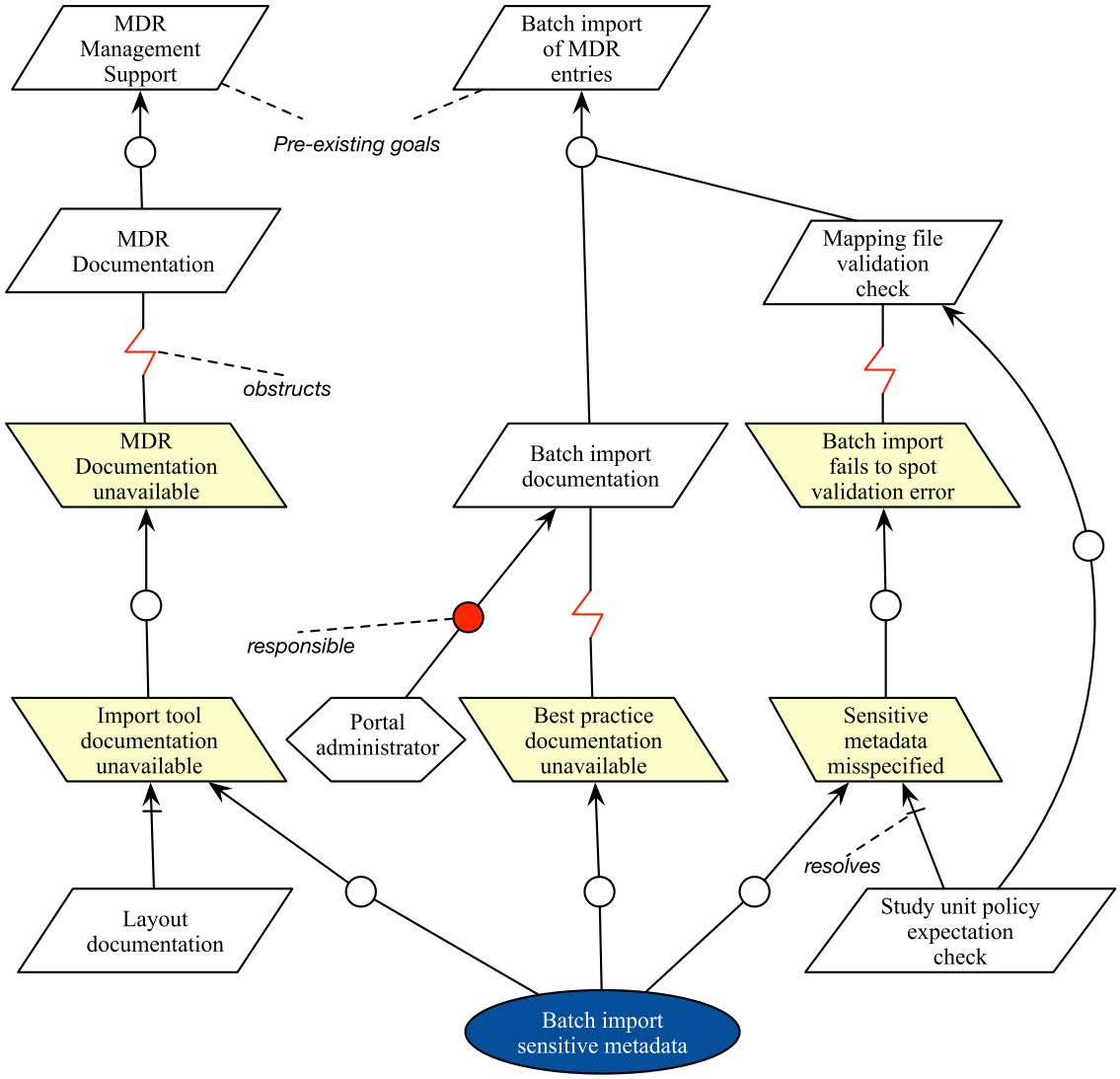

Fig. 5 Misusability case contribution to goal model

goal, thereby requiring further thought about root obstacles this might be satisfying. Because the obstacles are based on system rather than user errors, it would be inappropriate to define the immediate parent obstacle as a slip on the part of Brian. However, the consequences of the slip suggest that some form of validation safeguarding against such slips might have failed. A root obstacle that might be satisfied by the contributing obstacles states that the batch import process failed to spot the validation error. Yet, this obstacle raises the question: What such a validation error might entail? One means of providing this validation involves stating *a priori* expectations about the data manager's study unit's policy for exporting different classes of data and meta-data. By consulting these expectations, discrepancies between unit data and the unit data policies can be highlighted during a pre-import validation check. To realise these requirements, two goals were added to the goal model. The first of these stipulates that a mapping file validation check shall be carried out in order to satisfy the goal of batch importing meta-data into the MDR. The second goal resolves the meta-data policy mis-specified obstacle; this involves stating that study policy expectation check shall be carried out as part of the validation process.

5 Discussion

5.1 Pairing for engagement

One of the benefits of the misusability case technique was its ability to increase developer engagement towards understanding how poor usability can both hinder take-up

of the system they were trying to build, and compromise security. This engagement led developers to consider security issues they might have otherwise considered out-of-scope, and the responsibility of another team.

The misusability case below, which is similar to that described in Sect. 4.3, was developed to explore the impact of corrupt meta-data causing the import process to misinterpret the quality of study data.

Brian has been preparing data-sets ready for ingestion into the MDR. He hoped to use standards and guidelines on the portal, but he was disappointed by the lack of anything useful that would help him. Sill, Brian managed to organise his meta-data into the layout he inferred from some the XSLT scripts he downloaded.

Unfortunately, in some cases, some of the meta-data came from MS Office files and a few invisible character codes managed to find their way into the prepared meta-data files. Fortunately, Brian thinks he managed to catch most of them before he finally collated a meta-data file ready for upload.

Brian entered a URI he had been provided for uploading his meta-data and, after logging in using the Data Manager credentials, Brian enters an online form describing some of the characteristics of the meta-data, together with the location on his PC where meta-data can be found. Several minutes after Brian clicked on the Upload button, Brian received a message from the gateway saying the meta-data had been uploaded.

Consequently, although draft data was imported into the MDR, this was interpreted as real study data. When this misusability case was presented to developers, together with goals which would mitigate this problem, the developers were adamant that the obstacle should not be mitigated. Doing so, they argued, might disengage data managers; the developers did not want to pre-empt what data managers should or should not wish to import into the MDR.

Following this discussion, a second, consequential, misusability case (below) was presented to the developers.

Alex is currently sat in his office, in front of Safari. Alex is currently trying to add substance to an unfinished paper about the time parents spend with their sons in South Africa.

Alex has come across a paper in PubMed, which points to a particular data-set. This dataset is referenced in the paper as a URI. Alex clicks on the URI, which takes him into the portal The publicly available data about the dataset is loaded into the browser. This data includes a thumbnails of the questions being asked, and some statistics on who the question was asked to, i.e. the number of responses, together with the mean or variance. Unfortunately, the metadata about the data quality meta-data has been corrupt, which means that he fails to spot the lack of quality information for the data set. There are, however, links about the study which produced the data-set, and links to a PDF version of the question it came from.

This data-set isn't precisely what Alex is looking for, but it is similar. As a result, Alex decides to look at the study in more detail. He clicks on the details of the study to find out more about the characteristics of the population and who the study is funded by. After looking at these details, Alex bookmarks the study URI for future reference.

Later, Alex will obtain this dataset and notice enough similarities in his research that elements of the data will be applicable to his study.

This misusability case was a corollary of the first and described the impact of the corrupt data from Alex's perspective. In this misusability case, the invisible control characters in the imported meta-data caused the portal to leave the quality indicator field blank when information about a study is viewed by an end-user. As a result, Alex obtained the data and used it in his own research without realising that it wasn't real.

After this second misusability cases were presented, the developers acknowledged the seriousness of the MRD contributing to the publication of research grounded in invalid scientific data. The developers did not have an immediate solution to this problem, but they did identify additional usability concerns that Brian might have, and they acknowledged that Brian would only use the portal if it was seamlessly integrated into their work processes. Although the system documentation does not specifically allude to draft data being uploaded to the MDR, encouraging data managers to use the portal would require further thought about how synthetic data could be imported, and distinguished from actual study data.

5.2 Scope and responsibility

The misusability case example illustrated how security issues can lead to a scope of analysis review. During earlier design sessions, documentation related goals and assets were deemed to be out of scope for the MDR. As such, they were removed from the analysis data altogether. However, in hindsight, it was necessary for such goals to be present; this included making explicit who was responsible for ensuring their satisfaction, and how these goals could impact goals and obstacles which were within scope.

The example also highlighted the importance of ensuring that unresolved obstacles were also assigned to responsible agents. Previous work in Responsibility Modelling has considered roles held by responsible agents towards securing systems, and modelling responsibility relationships between these agents, e.g. [5, 7]. Despite a comparatively recent resurgence of interest in the role of responsibility modelling to elicit requirements during the

early stages of design [19, 53], vulnerabilities are still considered a consequence of responsibility failure, rather than something which can be brought to account. Although risk management approaches deal with the idea of *transferring* unmitigated responses to one or more agents, we believe that assigning ownership of obstacles during the early stages of design will ensure that vulnerabilities giving rise to security and usability problems are promptly addressed. This entails making the assigned stakeholders liable for addressing the obstacle. Flechais and Sasse [31] argue that this motivates the assigned stakeholders. This is because failure to act responsibly damages both the project's assets and its reputation; this reputation loss may lead to loss of trust in the whole system.

5.3 Misusability as an innovation tool

Another benefit of explicitly introducing the concept of misusability into design are the opportunities afforded for innovation. Ensuring sensitive data was not offered to an external interface was raised as a concern in the MDR requirements specification, but much of the focus on security within the design itself was focused on providing identity assurance. The misusability case example in Sect. 4 led to the un-envisaged leverage of different types of access control policies to help safeguard against the disclosure of sensitive data.

Scenarios have already been proposed as a vehicle for stimulating innovation [41]; by viewing technology from the perspective of marginal communities, a fresh perspective can be obtained which lead to innovative ideas. In misusability cases, rather than looking at *marginal* communities, we instead consider how *marginalised* personas are affected by design ambiguity. Like the transfer scenarios described by [41], misusability case narratives are grounded in data, but are also supplemented with goal-oriented techniques from requirements engineering to show how a system design contributes to misusability. Rather than treating misusability ephemerally, perhaps we should use also misusability cases as a grounded innovation technique?

5.4 Limitations of approach

Although we have validated Misusability Cases using a real case study example, two limitations of this approach are worth highlighting.

First, the analyst applying this approach in the case study example was one of the paper authors. Although this author has several years of industry experience in software engineering, which includes the elicitation, specification, and validation of software requirements, he also possesses broader usability and security engineering expertise. While such expertise would not typically be available to a single practitioner, there is, as Sect. 2.2 highlights, a growing body of work that aligns both user-centred design and security with general requirements engineering practice. As a result, the requisite tools and techniques underpinning misusability cases are accessible to the practitioner community. For example, misusability cases were used by the EU FP 7 *webinos* project to re-evaluate software requirements from a usability security perspective [58]. A team of ten practitioners and researchers (including the paper authors) elicited 5 misusability cases. The methodology described in this paper was broadly followed, but while design consequences of the misusability cases are presented, no requirements were explicitly updated as a result.

Second, several goals were elicited for mitigating the misusability cases in the study described in Sect. 4, the security impact of these mitigating goals were not considered. This is because the scope of investigation was to elicit and specify additional security requirements. It was then left to the developers to decide whether or not to realise these requirements. However, if the development team had decided to realise these requirements, KAOS could have been used to elicit and specify further goals and obstacles based on these, which could be complemented by an architectural risk analysis using CAIRIS. Such an approach is demonstrated by [28].

6 Conclusion

Scenarios have been used to support both security and usability engineering. To date, there is little evidence that scenarios used to support design in one context informs design activities in another. In this paper, we have described misusability cases which, rather than treating misusability as a corollary of bad design, explicitly identifies the causes of misusability and inform the design of systems to resolve them. In doing so, we demonstrate how a particular scenario can support both security and usability design activities.

Our work makes four specific contributions towards addressing misusability in secure systems. First, we describe how existing work in usability and security engineering can be aligned to form the basis of eliciting usable security requirements engineering concerns, and why simply typecasting users as attackers may be less effective than modelling both the impact of misusability in users' activities, and their causes. Second, we illustrate how assumptions identified using existing user-centred design and requirements engineering techniques can be modelled using argumentation models, the elements of which help identify examples of unintentional misuse. Third, we present a case for explicitly assigning responsibility for the causes of misusability leading to system

misuse irrespective of whether or not discharging this responsibility is within the scope of analysis or not. Finally, we describe how misusability cases can benefit design activities by re-sensitising developers with usability concerns, and stimulating innovative thinking towards hitherto unidentified design requirements fostering both security and usability.

References

1. Adams A, Sasse MA (1999) Users are not the enemy. Commun ACM 42:41–46
2. Alexander I (2003) Misuse cases: use cases with hostile intent. IEEE Softw 20(1):58–66
3. Alexander IF, Maiden N (eds) (2004) Scenarios, stories, use cases: through the systems development life-cycle. Wiley, New York
4. Amyot D, Ghanavati S, Horkoff J, Mussbacher G, Peyton L, Yu E (2010) Evaluating goal models within the goal-oriented requirement language. Int J Intell Syst 25(8):841–877
5. Backhouse J, Dhillon G (1996) Structures of responsibility and security of information systems. Eur J Inf Syst 5(1):2–9
6. Birge C (2009) Enhancing research into usable privacy and security. In: Proceedings of the 27th ACM international conference on design of communication. ACM, pp 221–226
7. Blyth A (1999) Using stakeholders, domain knowledge, and responsibilities to specify information systems' requirements. J Organ Comput Electron Commer 9(4):287–296
8. Brahnam S, de Angeli A (2008) Special issue on the abuse and misuse of social agents. Interact Comput 20(3):287–291
9. Braz C, Seffah A, M'Raihi D (2007) Designing a trade-off between usability and security: a metrics based-model. In: Proceedings of the 11th IFIP TC 13 international conference on human-computer interaction—volume part II. Springer, New York, pp 114–126
10. Brostoff S, Sasse MA (2001) Safe and sound: a safety-critical approach to security. In: Proceedings of the 2001 new security paradigms workshop. ACM, pp 41–50
11. Chapman CN, Milham RP (2006) The persona's new clothes: methodological and practical arguments against a popular method. In: Proceedings of the human factors and ergonomics society 50th annual meeting, pp 634–636. http://cnchapman.files.wordpress.com/2007/03/chapman-milham-personas-hfes2006-0139-0330
12. Cleland-Huang J (2013) Meet Elaine: a persona-driven approach to exploring architecturally significant requirements. IEEE Softw 30(4):18–21
13. Cockburn A (2001) Writing effective use cases. Addison-Wesley, Boston
14. Cockton G (2008) Revisiting usability's three key principles. In: CHI '08 extended abstracts on Human factors in computing systems. ACM, pp 2473–2484
15. Cooper A (1999) The inmates are running the asylum: why high tech products drive us crazy and how to restore the sanity, 2nd edn. Pearson Higher Education, Upper Saddle River
16. Darley JM, Latané B (1970) Norms and normative behaviour: field studies of social interdependence. In: Berkowitz L, Macaulay J (eds) Altruism and helping behaviour. Academic Press, San Diego
17. Dewar JA (2002) Assumption-based planning: a tool for reducing avoidable surprises. Cambridge University Press, Cambridge
18. Dunne A, Raby F (2001) Design Noir: the secret life of electronic objects. August/Birkhaeuser, Basel
19. Elahi G, Yu E (2009) Trust trade-off analysis for security requirements engineering. In: Proceedings of the 17th IEEE international requirements engineering conference. IEEE Computer Society, pp 243–248
20. Faily S (2011) Bridging user-centered design and requirements engineering with GRL and persona cases. In: Proceedings of the 5th international i* workshop. CEUR Workshop Proceedings, pp 114–119
21. Faily S (2013) CAIRIS web site. http://github.com/failys/CAIRIS
22. Faily S, Fléchais I (2010) A meta-model for usable secure requirements engineering. In: Proceedings of the 6th international workshop on software engineering for secure systems. IEEE Computer Society, pp 126–135
23. Faily S, Fléchais I (2010) Analysing and visualising security and usability in IRIS. In: Proceedings of the 5th international conference on availability, reliability and security. IEEE Computer Society, pp 543–548
24. Faily S, Fléchais I (2010) Barry is not the weakest link: eliciting secure system requirements with personas. In: Proceedings of the 24th BCS interaction specialist group conference, BCS '10. British Computer Society, pp 124–132
25. Faily S, Fléchais I (2010) The secret lives of assumptions: developing and refining assumption personas for secure system design. In: Proceedings of the 3rd conference on human-centered software engineering, vol LNCS 6409. Springer, New York, pp 111–118
26. Faily S, Fléchais I (2011) Eliciting policy requirements for critical national infrastructure using the IRIS framework. Int J Secur Softw Eng 2(4):114–119
27. Faily S, Fléchais I (2011) Eliciting usable security requirements with misusability cases. In: Proceedings of the 19th IEEE international requirements engineering conference. IEEE Computer Society, pp 339–340
28. Faily S, Lyle J, Namiluko C, Atzeni A, Cameroni C (2012) Model-driven architectural risk analysis using architectural and contextualised attack patterns. In: Proceedings of the Workshop on Model-Driven Security. ACM 3:1–3:6
29. Firesmith D (2004) Specifying reusable security requirements. J Object Technol 3(1):61–75
30. Fléchais I, Mascolo C, Sasse MA (2007) Integrating security and usability into the requirements and design process. Int J Electron Secur Digit Forensics 1(1):12–26
31. Fléchais I, Sasse MA (2009) Stakeholder involvement, motivation, responsibility, communication: how to design usable security in e-science. Int J Hum Comput Stud 67(4):281–296
32. Franqueira V, Tun TT, Yu Y, Wieringa R, Nuseibeh B (2011) Risk and argument: a risk-based argumentation method for practical security. In: Requirements engineering conference (RE). 2011 19th IEEE international, pp 239–248
33. Gould JD, Lewis C (1985) Designing for usability: key principles and what designers think. Commun ACM 28(3):300–311
34. Haley CB, Laney R, Moffett JD, Nuseibeh B (2008) Security requirements engineering: a framework for representation and analysis. IEEE Trans Softw Eng 34(1):133–153
35. Holtzblatt K, Jones S (1993) Contextual inquiry: a participatory technique for systems design. In: Schuler D, Namioka A (eds) Participatory design: principles and practice. Lawrence Erlbaum Associates, New Jersey, pp 177–210
36. Irestig M, Eriksson H, Timpka T (2004) The impact of participation in information system design: a comparison of contextual placements. In: Proceedings of the 8th conference on Participatory design. ACM, pp 102–111

37. Jøsang A, Alfayyadh B, Grandison T, Alzomai M, McNamara J (2007) Security usability principles for vulnerability analysis and risk assessment. In: Computer security applications conference, 2007. ACSAC 2007. Twenty-third annual, pp 269–278
38. Kruchten P (2003) The rational unified process: an introduction, 3rd edn. Addison-wesley, Boston
39. van Lamsweerde A (2009) Requirements engineering: from system goals to UML models to software specifications. Wiley, New York
40. van Lamsweerde A, Letier E (2000) Handling obstacles in goal-oriented requirements engineering. IEEE Trans Softw Eng 26(10):978–1005
41. Ljungblad S, Holmquist LE (2007) Transfer scenarios: grounding innovation with marginal practices. In: Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '07. ACM, pp 737–746
42. Matthews BM, Duncan A, Jones CM, Bicarregui JC (2009) MRC data support service—data gateway requirements: version 1.0
43. Mayer N (2009) Model-based management of information system security risk. Ph.D. thesis, University of Namur
44. Mellado D, Fernández-Medina E, Piattini M (2007) A common criteria based security requirements engineering process for the development of secure information systems. Comput Stand Interface 29(2):244–253
45. Moody F (1996) I sing the body electronic: a year with microsoft on the multimedia frontier. Penguin, New York
46. Nathan LP, Klasnja PV, Friedman B (2007) Value scenarios: a technique for envisioning systemic effects of new technologies. In: CHI '07: extended abstracts on Human factors in computing systems. ACM, pp 2585–2590
47. Nielsen J (1994) Guerrilla HCI: using discount usability engineering to penetrate the intimidation barrier. In: Bias RG, Mayhew DJ (eds) Cost-justifying usability. Morgan Kaufmann, San Francisco, pp 242–272
48. Nurse J, Creese S, Goldsmith M, Lamberts K (2011) Guidelines for usable cybersecurity: past and present. In: 2011 third international workshop on cyberspace safety and security (CSS), pp 21–26
49. Rosson MB, Carroll JM (2002) Usability engineering: scenario-based development of human-computer interaction. Academic Press, London
50. Saltzer JH, Schroeder MD (1975) The protection of information in computer systems. Proc IEEE 63(9):1278–1308
51. Sindre G, Opdahl AL (2005) Eliciting security requirements with misuse cases. Requir Eng 10(1):34–44
52. Siponen M (2002) Designing secure information systems and software. Ph.D. thesis, University of Oulu
53. Sommerville I, Lock R, Storer T, Dobson J (2009) Deriving information requirements from responsibility models. In: van Eck P, Gordijn J, Wieringa R (eds) CAiSE '09: proceedings of the 21th international conference on advanced information systems engineering, vol LNCS 5565. Springer, Berlin, pp 515–529
54. Sutcliffe A (2005) Convergence or competition between software engineering and human computer interaction. In: Seffah A, Gulliksen J, Desmarais MC (eds) Human-centered software engineering: integrating usability in the software development lifecycle. Springer, New York
55. Thimbleby H (2007) User-centered methods are insufficient for safety critical systems. In: HCI and usability for medicine and health care, third symposium of the workgroup human-computer interaction and usability engineering of the Austrian Computer Society, vol 4799 LNCS. Springer, New York, pp 1–20
56. Thimbleby H, Thimbleby W (2007) Internalist and externalist HCI. In: Proceedings of the 21st British HCI group annual conference. British Computer Society, pp 111–114
57. Toulmin S (2003) The uses of argument. Cambridge University Press, Cambridge
58. webinos Consortium (2011) webinos report: user expectations of security and privacy phase 2. http://webinos.org/2011/11/01/webinos-repot-user-expectations-of-security-and-privacy-phase-2/
59. Whitten A, Tygar D (1999) Why Johnny can't encrypt: a usability evaluation of PGP 5.0. In: Proceedings of the 8th USENIX security symposium. USENIX Association, pp 169–184
60. Yee KP (2005) Guidelines and strategies for secure interaction design. In: Cranor LF, Garfinkel S (eds) Security and usability: designing secure systems that people can use. O'Reilly Media, Sebastopol, pp 247–273
61. Zurko ME, Simon RT (1996) User-centered security. In: Proceedings of the 1996 new security paradigms workshop. ACM, pp 27–33