

Misuse, Abuse, and Reuse: Economic utility functions for characterising security requirements

Chad Heitzenrater^{†‡}

chad.heitzenrater@cs.ox.ac.uk

[†]U.S. Air Force Research Laboratory
Information Directorate
525 Brooks Road
Rome NY 13441, USA

Andrew Simpson[‡]

andrew.simpson@cs.ox.ac.uk

[‡]Department of Computer Science
University of Oxford
Wolfson Building, Parks Road
Oxford OX1 3QD, UK

Abstract—Negative use cases — in the form of ‘misuse’ or ‘abuse’ cases — have found a broad following within the security community due to their ability to make explicit the knowledge, assumptions and desires of stakeholders regarding real and perceived threats to systems. As an accepted threat modelling tool, they have become a standard part of many Secure Software Engineering (SSE) processes. Despite this widespread adoption, aspects of the original misuse case concept have yet to receive a formal treatment in the literature. This paper considers the application of economic utility functions within the negative use case development process, as a means of addressing existing challenges. We provide a simple demonstration of how existing practice might integrate economic factors to describe the business, management and functional concerns that surround system security and software development.

1. Introduction

Threat modelling has become a critical element within secure software engineering practice. Among the various approaches, the use of *negative use cases* (in the form of misuse and abuse cases) as a means to elicit security requirements has become a common practice among security professionals [1]. Employing the same tools and use-case constructs that are well known to software engineers, these approaches meld security and software engineering in a way that is easily understandable to practitioners from both communities. The application of negative use case analysis to software and system development is widely seen as a software security ‘best practice’ [2].

However, the practice is not without issues. In this paper, we identify specific challenges that have been identified in the negative use case literature: negative use case reuse and premature design specification. We then propose the application of economic-based utility theory as a means of redress, and explore how such a construction can provide a quantifiable basis to link business and engineering practices throughout the Systems Development Life Cycle (SDLC) [3].

This paper presents this argument in the following manner. Section 2 presents the literature on negative use cases and outlines the three issues we seek to address, along with a rationale for the employment of utility constructions to address these concerns. Section 3 outlines specific recommendations and provides the basis for the example that will be employed in demonstrating these concepts (in Sections 4 and 5). We then conclude with a summary and directions for future work in Section 6.

2. Background

In this section we briefly recap the literature, in order to identify and motivate the challenges of negative use case adoption.

2.1. Negative Use Cases

The concept of threat modelling for security requirements is traceable to *intrusion scenarios* as introduced by Ellison et al. in 1999 [4]. Today, this concept is more commonly known as *abuse cases* (as introduced by McDermott in 1999 [5]) or *misuse cases* (as presented in a series of contributions by Sindre et al., starting in 2000 [6]). Such constructs have become a staple of many modern Secure Software Engineering (SSE) practices, including Touchpoints [7], the Microsoft Security Development Lifecycle (SDL) [8], and the Open Web Application Security Project (OWASP) Comprehensive, Lightweight Application Security Process (CLASP) [9]. These processes advocate the extensive application of misuse analysis at both the requirements and architectural levels [10].

Despite arising independently, the conceptualisation of abuse and misuse cases is quite similar. As a direct play on the use case concept, misuse cases invert use case notions — with ‘misusers’ as actors and employing a Unified Modeling Language (UML) notation that is the colour inverse of the standard use case notation [11]. Abuse cases use a similar notion to ‘malefactors’ [12], and while devoid of a visual notion the prose description shares many of the same attributes as misuse cases [13]. Abuse cases specify a four-step

process of iterative development — defining the problem, constructing candidate abuse cases, refining candidates, and ‘refuting’ candidates — with the goal of ending the process with no remaining candidates [12]. Misuse cases add an opening step of identifying assets, with focus on identification of threat and risk, rather than refinement, in their five-step process [11]. Both processes focus on the lightweight, flexible nature of the exercise for non-specialists and, despite minor differences, these concepts have largely merged into a single set of practices generally referred to by either term¹.

Various benefits have been ascribed to the employment of negative use case analysis as a part of security requirement generation.

- *An early focus on security*, permitting threats to be described independently of design. Consideration of threats in the absence of technical constraints can promote creativity when identifying and capturing security needs, enhanced by the informal nature of the process [11]. This helps organisations to see their systems in the same light as attackers, prompting inquiry — and subsequent consideration — of implicit assumptions [2].
- *User/customer assurance, awareness, and education*, as misuse case descriptions can be presented (and understood) in the absence of deep technical detail. This enables more effective communication and, when combined with the previous point, is widely seen as permitting the capture of requirements that may have otherwise been ignored [11].
- *Explicitly identify and consider trade-offs between functional and security requirements*. This is especially critical when considering security implications of functional requirements, which may impact design considerations and aid in the prioritisation of requirements [14], [15]. Furthermore, the ability to visualise such requirements is credited with aiding organisational understanding [11].
- *A mechanism to trace security requirements to their motivation and resulting manifestation*, aiding in security management [13] — although the means to accomplish this is still an area of investigation [16].
- *Enable the re-use of requirements*, providing new development projects a “flying start” with respect to defining security [11]. Focusing on the misuser action sequence, [17] presents an inheritance-inspired hierarchical approach to defining general misuse cases (e.g. ‘Obtain Password’), which can subsequently be ‘specialised’ into instantiations (e.g. ‘Obtain password by race condition’, ‘Obtain password by sniffing’, and ‘Obtain password by phone’). In [18] this concept is taken further to differentiate between the development of misuse cases *for* reuse, and the development of systems *with* misuse case reuse, with the former employing variables (primarily to abstract functional aspects, without much con-

sideration paid to the abstraction of misactors). For their part, McDermott and Fox [5] address the issue of reuse from the start, with abuse cases defined as “a family of undesirable interactions”. Recognition of the relative merits of the two approaches led Sindre et al. to argue for the need to integrate the analysis of security and functional requirements that may result from a combination of misuse and abuse cases [17].

2.2. Problem Statement

Despite these benefits, a number of general issues have been raised, linked to the suitability of the approach to specific contexts and threats [11] and the inability to contribute to ‘high assurance’ development [5]. In order to illustrate these challenges, we employ a subset of the running example employed by Sindre et al. in a series of papers: [11], [13], [15], [17]. Figure 1 illustrates the use case (‘Order Goods’) which is threatened by two identified misuse cases (‘Obtain Password’ and ‘Steal Credit Card Info’), while Figure 2 demonstrates a proposed design decision based on Figure 1. This example serves as the basis for our discussion; the overall example can be found in [17].

Two specific, interconnected weaknesses directly challenge the promise of misuse cases as a threat modelling tool. Each is now considered in turn.

2.2.1. Suitability of misuse/abuse cases in other contexts for which they were not explicitly developed. In practice, this may manifest in one of two ways. The first is in the force-fitting of negative use cases to describe scenarios where some aspect (e.g. misuser) is not readily identifiable or is vague [11]. Alternatively, this may be the case when the description and attributes of a misuser are inappropriately shifted to an effort where the assumptions inherent in the misuser model may not be valid [17]. A failure to appropriately generalise during misuse case generation can lead to repetition in definition, an explosion in scope, and a multitude of cross-references [17], with each scenario potentially leading to incorrect or inappropriate resource expenditure (or lack thereof). For abuse cases this is compounded by the lack of a set template or guidance, leaving the process inaccessible to non-specialists in security and hampering reuse. The approaches specified in [17] and [18] are intended to address such repetition and the copious cross-referencing of misuse cases.

This potential can be seen in Figure 1, where the generic negative use cases could encompass a wide variety of adversary capability and modes of attack. As the desire to reduce development costs calls for increasing reuse, danger arises in the use cases being too general to include useful information (e.g. can the adversary sniff a password off the wireless or wired network, and what methods would be used?), vice project-specific definitions that may not be applicable (e.g. does the misuse case apply if this system is not employing wireless?).

1. In general we will employ the term ‘negative use cases’ when referring to the concept, but will specify one or another where the literature dictates.

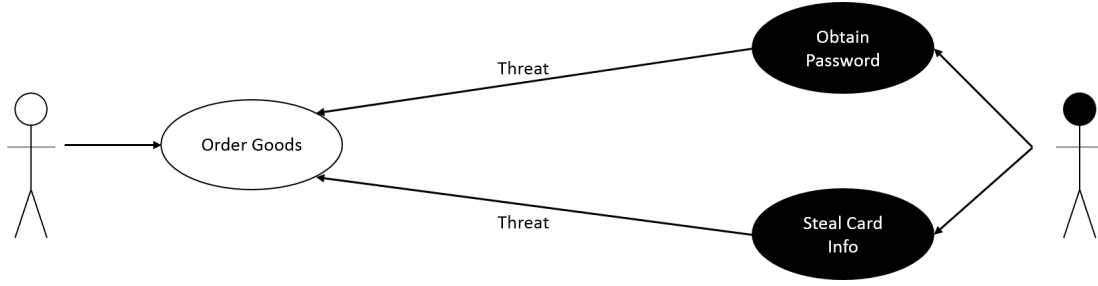


Figure 1. Base use and misuse cases for our example. The construction is a subset of the example used by Sindre, Opdahl, and Brevik [17].

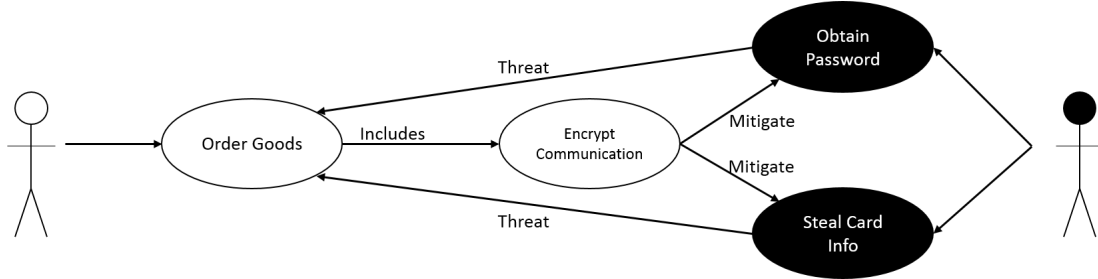


Figure 2. Updated use and misuse cases for this example. The construction is a subset of the example used by Sindre, Opdahl, and Brevik [17].

2.2.2. Premature design decisions stemming from negative use case over-specification. The misuse and abuse case generation processes take pains to make recommendations regarding the translation from concept into design and implementation. It has been suggested that this is best accomplished through iterative application of the process (e.g. in interface- to design- to code-level refinement). However, the early introduction of specific solutions runs afoul of the practice of requirements generation, and may lead to myopic decisions [19]. Alternatively, it has been suggested that capturing potential measures (e.g. in the ‘capture points’ field of the misuse case template) provides ideas that can be later analysed for application within the context of the system. This arrangement has the opposite effect, potentially leading to ‘analysis paralysis’ as stakeholders are overcome with the process, its artefacts, and the interaction with other aspects of secure development [11]. Here it is proposed that focus should be put on the “major use cases, i.e. threats that are likely and mis-actors who might be behind those threats” [13]. However, no guidance is provided on the best approach to identifying or constricting these aspects and, overall, these processes are not well-specified and guidance is unquantified or focuses on system-level trade-off instead of design-centric approaches [18].

This is exemplified in Figure 2. Here, Sindre et al. have specified ‘encrypt communication’ as a specific solution in the ‘order goods’ use case. While it stops short of defining the specific means of encryption, this design decision limits later options for how the system will operate and the policies it must support (e.g. PCI-DSS), unduly pointing the development in the direction of what may turn out to be a less secure configuration.

3. Utility-based Negative Use Cases

To address these deficiencies and realise the envisioned potential of misuse/abuse cases, we propose the incorporation of economic utility to the security software engineering process². The goal is to represent the resources, skills and objectives of malactors, such that the result is explicit but parameterised — making the scope and applicability of the misuse case evident (that is, explicit and calculable) and malleable (so that it is easily altered to fit changing security contexts). To this end, we first present the concept of economic utility, followed by our recommendation for its employment in negative use case analysis.

3.1. Economic Considerations

Utility is a concept that builds on the notion of *preferences* — the ability for a person (in economics, the ‘consumer’) to identify one outcome as more desirable than another. *Utility functions* follow as representations of preference relationships; that is, the ability to identify the preference of one item over another in a *consumption bundle* (the choices available). Under the assumption that the choice is quantifiable (although perhaps not yet quantified) and well behaved, algebraic properties hold; specifically, we can assume (under rationality) that the preferences represented by the utility functions are then complete (any two can be compared), reflexive (any bundle is at least as good as itself), and transitive (if bundle A is better than bundle B, and

2. As exemplified by Touchpoints [7] in which the typical cycle of requirements, design, code, test, and deploy is augmented with security-specific steps such as misuse case analysis.

bundle B is better than bundle C, then bundle A is better than bundle C) [20]. From this simple construct we build sets of expressive relationships that compare two or more choices based on which provides the most benefit, or utility.

Although negative use cases have yet to integrate economic considerations, the basis for incorporating such considerations can be traced to the original contributions on the topic. Sindre and Opdahl identified the need to consider economic aspects in both users and misusers. In [13] the duo propose renaming the standard use case field *Stakeholders and Interests* as *Stakeholders and Risks*, in order to quantify costs and likelihoods “with more ambition” than the use of textual descriptions. In related work, they cite the integration of misuse case analysis with risk analysis, costing and formal methods as a potentially interesting direction to pursue [15]. However, subsequent work has yet to provide any formalism, constructs, or guidance as to the incorporation of these concepts into misuse case analysis processes.

Although McDermott and Fox do not appear to have considered economics as explicitly, a key element of an abuse case are the “resources, skills and objectives” [5] — with the first two arguably unfulfilled economic quantifications. Later work expounds on this idea by considering the assurance resources when constructing abuse cases, in order to utilise that information during refutation — thus connecting assurance to malefactor effort [12]. Unfortunately, the authors do not discuss how this would be done, leaving the method of ranking, matching, and estimation to the reader.

3.2. Integrating Utility and Negative Use Cases

Given the description of utility and the aims set forth in the negative use case literature, there appears to be a relationship between requirements generation and utility as a means of expressing preference; after all, requirements could be considered the capture of user preference (perhaps constrained by invariants imposed by technology, law or policy, which could be thought of as ‘hard’ preferences). We propose the following extensions to textual misuse/abuse case description in [13], [17] and [12] as a means of addressing the challenges outlined in Section 2.2.

3.2.1. Template Separation. As in [17], we advocate specification of misusers (malactors) in a separate template. Further extending the concept of generalisation/specification, we additionally propose generalisation of malactors. Abstracted negative use cases would be linked to specific actors through the *Potential Misuser Profile*.

3.2.2. Utility Incorporation. We advocate changes to the template descriptions to facilitate the expression of attributes as utility functions.

- 1) *Potential Misuser Profile.* Utilising generalised misuser templates permits the misuser profile to be specified in more interesting and potentially quantifiable ways. Examples of ways a misuser might be specified include: likelihood of attack (p_a) and

probability of success given an attack (p_s)³ and attack costs $a \in \{a_1, a_2 \dots a_x\}$. Using just two such simple constructs provides the ability to specify constructions that map to existing breach data sources, and explicitly bound qualitative statements often used in such models. One example might be to specify a probability of loss (what is commonly called ‘likelihood’ or ‘attack’ in other models) as $p_{s|a} = p_s \cdot p_a$. Employing the example from [17], the cited ‘skilled’ attacker could be further specified as someone with p_s above a given threshold. This implicitly creates secondary classes of misusers and permits the requirements to be relatively explicit about the expectations being placed on the system — even if measurement of such values proves difficult. To this end, even a range of values provides more information than the standard ‘high’, ‘medium’ and ‘low’ delineation often used.

- 2) *Stakeholders and Threats* follows suit, rounding out the terms for traditional risk-based calculations by supplying utilities for the consequence portion. While likely to be conceptualised in monetary terms, this may also take the form of any other factor the company measures: website accessibility/connectivity (in the case of an attack on availability) or data record loss (in the case of an attack on integrity). Returning to the example of [17], loss of revenue, bad-will, customer losses, and time wasting are cited — each of which has a direct monetary estimate. We can envision constructions that relate these terms, such as the cost of data loss (d) relative to the liability of the company in the event of data loss (l), such that a relationship can be derived to drive investment (e.g. $0 < d < l$).
- 3) *Mitigation Points* then includes utility forms for estimating resource investment (i.e. costs). The projected effects of the investment on other parameters can then be specified in light of the cost of achieving that level. This could be modelled as a set of costs $c \in \{c_1, c_2 \dots c_x\}$ corresponding to a given $p_{s|a}$, further constrained by the overall budget b . We employ the “Tamper with DB” example from [17] as a simple illustration of these concepts. One can imagine mitigation mp1 (not revealing the database error message) may incur a cost of a few man-hours of testing to implement, but result in a low p_s . However, mp2 (explicitly checked inputs) may mitigate the threat with a high p_s but at a cost that is an order of magnitude higher than that of mp1. The end result is a parameterised relationship between $p_{s|a}$, a , c and d — elements of risk modelling and threat analysis currently practised by security professionals.

3. Note that here we are considering these two events as independent: an attack on the system, p_a , and the state of the system against attack, p_s . This permits the external attacker motive to be considered separate from the internal system state. However, it is recognised that this independence in every situation is open to further investigation and discussion.

3.2.3. Cohesion. Within the negative use case template, ‘Scope’, ‘Abstraction’ and ‘Precision Level’ entries should identify the fidelity to which these utilities have been estimated and captured. Ideally, this would take the form of references to reports, meeting notes and data sources that justify the utility form, as well as any values assigned.

We now examine these concepts against the issues identified in Section 2.2, in the context of our running example.

4. Misuse Case Re-use

Looking first at the question of negative use case re-use, we conceive of an abstraction that combines misuse and abuse cases as described in Section 3.2. To illustrate these concepts we provide a template for a threat actor seeking a password (Table 1) relative to the related abstract negative use case definition (Table 2, modelled on Table 3 of [17]). A generalised specification allows us to define a parameterised misuser, employing utilities to describe relative attributes. This separates and specifies the factors relevant to the malactor/misuser; for instance, this could capture the “skill versus focus” delineation made by Schneier⁴. Other adversary characterisations might lead to other parameterisations, and while they may already be captured in the misuse case template ‘Potential misuser profile’ entry, they are done so qualitatively and statically. In the context of negative use case reuse this may limit their applicability and/or risk incorrect application.

Ideally, such a definition should not only capture some essence of a threat, but should also segment the knowledge required at various stages of design and development. Such generic definitions, along with various abstract utility relationships, can be defined independently and applied to a given project as needed. This segments aspects requiring security-specific knowledge from the engineering steps, removes the desire to over-specify the exact misuser or create a multitude of similar misuse cases, and reduces the impulse to ‘lock-in’ to a specific design.

Using this generalised (or abstract) actor-based negative use case, data can be applied to characterise the nature and effect of the harm anticipated within a specific project. This data may be well known, or rely on estimates; as with traditional risk analysis, this is likely to be a combination of project data, historic data (when available), publicly available estimates (such as the UK’s Information Security Breaches Survey [21], dedicated breach services⁵, or security companies⁶), or expert knowledge. The end result is an attempt to move from vague loss statements towards bounded potential realisations of the threat. To contrast, the misuse case description of Table 2 of [17] identifies ‘loss of revenue’, ‘losing money (at least temporarily)’, and ‘wasting time’ as effects of a threat. Without some basis of estimation (e.g. the cost of a man-hour, or the

daily revenue) the relevance and priority of such a threat is unclear. This integration with the business operations and risk management practices is critical to security considerations. Additionally, this supplies an explicit link to risk-based software engineering approaches (e.g. OCTAVE [22] and AEGIS [23]).

Further integration with business practice is captured in the ‘Stakeholders and Threats’ field, where valuations — monetary or otherwise — can be listed in order to make business-level constraints evident in the engineering process. True valuation of the threat, which may encompass direct harm as well as fines, loss of goodwill, response costs and other considerations, is notoriously difficult to estimate; rather, this serves as an anchor point for later constraints. We have populated this field with a notional value; however, as later illustrated, this field is more useful in providing bounds for the threat analysis to later drive security decision-making.

From this set of abstract considerations, the next step is to move towards relevant, specific refinements to support decision-making within a specific development effort. Continuing to track the exposition of [17], we consider the three misuse case specialisations identified in that work: sniffing, race conditions, and phone acquisition. This is accomplished through the choice of parameters corresponding to the specific system threats expected, with values that reflect the industry, risk attitude, and known data. It is recognised that these values represent the most esoteric aspect of this process, requiring the most security expertise; however, the growing sources of data regarding breach costs, probabilities, and corporate enterprise metrics and measurement present an opportunity for increased fidelity and accuracy of such information within the development process. Tables 3–5 present notional quantitative values, paired with equally notional qualitative values for the negative use case actor profile. These are intended only to illustrate the concept.

At this point we have tracked Sindre et al.’s consideration of a single misuse case focused on a single misuse actor, through its extension to a class of three misuse cases that are still focused on a single actor. Through parameterisation, we have extended the misuser/malactor concept and introduced a virtually unlimited set of potential adversaries conditioned on the variables introduced. This presents a more nuanced view of the system’s security considerations, placing them on a (sliding) scale, rather than on a finish line. Attackers have proven to be nothing if not resourceful, and as recent events have shown⁷ the common approach of classifying likely threats as ‘vandalism’, ‘criminal’ and ‘military’ is no longer sufficient. The disparities between attackers and defenders, along with the increasing value of information held in the public sector, have mandated a wider scope

4. See https://www.schneier.com/essays/archives/2014/12/sony_made_it_easy_bu.html.

5. e.g. <http://breachlevelindex.com/#!breach-database>

6. e.g. <http://www.symantec.com/connect/blogs/data-breach-trends>

7. For instance, the alleged nation-state attack on Sony by North Korea (<http://www.bbc.co.uk/news/entertainment-arts-30512032>), or the acquisition of massive amounts of healthcare data (<http://krebsonsecurity.com/2015/02/china-to-blame-in-anthem-hack/>) and personnel data (<http://www.theguardian.com/technology/2015/jun/04/us-government-massive-data-breach-employee-records-security-clearances>) attributed to China.

Misuser Target: User Password Summary: A misuser who seeks to obtain one or more user passwords. ... Related Use Case(s): Order Goods
Utility Considerations Prob of attack (p_a): Password information is highly desired; if exposed this is likely to be a target. Prob of success (p_s): Dependant on location, skills. Low likelihood of getting caught. Attacker cost (a): Dependant on location, skills, and availability of tools. Constraints: The attacker will attack only if the expected pay-off is higher than cost; $a < p_{s a} \cdot M$, where M is the monetisation value to the attacker. Abstraction Level: Façade ...

TABLE 1. PARAMETERISED MISUSER TEMPLATE. FIELDS NOT RELEVANT TO THIS DISCUSSION HAVE BEEN OMITTED FOR THIS EXAMPLE.

Name: Obtain Password Summary: A crook obtains passwords(s) for user accounts... Known Specializations: Obtain Password by Phone, Obtain Password by Sniffing, Obtain Password by Phone. ...
Assumptions: ... Mitigation Guarantee: The attacker is disincentivised to attack ($a > p_{s a} \cdot M$), or is unsuccessful $p_{s a} \rightarrow 0$ Related Business Rule(s): Only authorized users shall be able to access restricted services.
Stakeholders and threats: The acquisition of a user password carries an average liability of £1,000. ...

TABLE 2. GENERALISED (ABSTRACT) MISUSE CASE FOR ‘OBTAIN PASSWORD’ BASED ON TABLE 3 OF [17]. FIELDS NOT RELEVANT TO THIS DISCUSSION HAVE BEEN OMITTED FOR THIS EXAMPLE.

Name: Obtain Password by Sniffing Summary: A misuser who seeks to obtain the password from the ‘Order Goods’ use case by eavesdropping on the communication between system components. ... Misuser Profile: Obtain Password Prob of attack (p_a): High ($0.8 < p_a < 0.95$); internet-connected Prob of success (p_s): High ($0.95 < p_s < 0.99$); passwords easy to identify & desired Attacker cost (a_i): Low ($1 < a_i < 10mh$). Requires simple, readily available tools.

TABLE 3. MISUSE CASE SPECIALISATION FOR OBTAIN PASSWORD BY SNIFFING.

Name: Obtain Password by Race Condition Summary: A misuser who seeks to obtain the password from the ‘Order Goods’ use case by taking advantage of a race condition in the code base. ... Misuser Profile: Obtain Password Prob of attack (p_a): High ($0.8 < p_a < 0.95$); internet-connected Prob of success (p_s): Medium ($0.5 < p_s < 0.79$); requires some knowledge Attacker cost (a): Low ($1 < a_i < 10mh$). Requires simple, readily available tools.
--

TABLE 4. MISUSE CASE SPECIALISATION FOR OBTAIN PASSWORD BY RACE CONDITION.

Name: Obtain Password by Phone Summary: A misuser who seeks to obtain the password from the ‘Order Goods’ use case by using the phone recovery system. ... Misuser Profile: Obtain Password Prob of attack (p_a): Low ($0.25 < p_a < 0.5$); requires human interaction Prob of success (p_s): Medium ($0.5 < p_s < 0.79$); if user has knowledge Attacker cost (a): Medium ($10 < a_i < 40mh$). Requires some research.
--

TABLE 5. MISUSE CASE SPECIALISATION FOR OBTAIN PASSWORD BY PHONE.

Target	High			Low		
	p_a	p_s	a_i	p_a	p_s	a_i
Sniffing	0.95	0.99	1	0.8	0.95	10
Race	0.95	0.79	1	0.8	0.5	10
Phone	0.5	0.79	10	0.25	0.5	40
Attacker Utilities: $a < p_{s a} \cdot M$						
Utility - Sniffing	$1 < 940.5$			$10 < 760$		
Utility - Race	$1 < 750.5$			$10 < 400$		
Utility - Phone	$10 < 395$			$40 < 125$		

TABLE 6. NOTIONAL MISUSER SPECIFICATION (NO PROTECTION). THE VALUES FOR p_a AND p_s ARE PROBABILITIES, WHEREAS a_i IS DEFINED IN TERMS OF MAN-HOURS FOR THIS EXAMPLE. WE USE THE NOTIONAL VALUE OF £1,000 FOR THE ATTACKER MONETISATION.

of considerations for security professionals. However, this wider view has the danger to exacerbate the existing issues surrounding misuse case development and selection — to which we now turn our attention.

5. Countermeasure Design

Early understanding of architecture can be key to managing large-scale systems and projects, but it is recognised that the choice of requirements can constrain the architecture approach — which in turn will constrain designers [24]. It is argued that at least 50% of all security defects are architectural in nature [25], making the translation from requirements to architecture a critical, and often overlooked, point in the development of a security posture. Through the proposed negative use case approach, we now highlight aspects of countermeasure design that are enabled, or at least enhanced, under this approach.

5.1. Ranking and Selection

One of the critical aspects to controlling costs when implementing security is the proper scoping and specification, driven by the user acceptance of risk and assumptions made regarding the environment. As noted by Gordon and Loeb in their seminal work [26], additional investment in security beyond a specific point is counter-productive; no investment will make a system entirely secure, and investment driven merely by probabilities may not be optimal. This necessitates careful consideration of the generated negative use cases when translating the threat model into a security design. Through the explicit incorporation of cost, loss and probability information we can now make these considerations explicit, and be more precise in respect to what is included and excluded.

Table 6 provides a utility-based attacker profile populated with notional values that correspond to risk-based analysis. Presentation of attacker models in this manner provides the ability to justify emphasis on particular negative use cases over others, with the overall goal of making the prioritisation of negative use cases more straightforward. Following from this specific example, we would rank each of these threats in the order of their presentation based upon their relative values.

Once established, this construct is easily updated as information solidifies during the development process, making it conducive to use in agile or iterative processes. This malleability is also critical for security over the long term, as systems must be built to address today’s threats as well as co-evolve with the ecosystem predators [27]. At the other end of this spectrum, we could envision a scenario where historical data reflects instances of certain attacks, permitting up-front specialisation to meet specific goals.

While this process is similar in concept to McDermott and Fox’s ‘refutation’ process [5], it provides an added element of formality and justification to the analysis. This kind of analysis is straightforward and well known; those familiar with the economics of information security literature will recognise the standard Annual Loss Expectancy (ALE) calculation [28] in the attacker utility, just as those versed in risk analysis will identify with the National Institute of Standards and Technology (NIST) standard definition for risk calculation [29]. While this is partially a result of the simplified exposition, it is also not entirely unexpected: in considering requirements (e.g. use cases) as expressions of user preferences, it follows that misuse or abuse cases would be representative of the attacker utility. While such refutation mechanisms cannot provide absolute guarantees that no exploitable vulnerability remains [27], they enable a more calculated consideration of various design alternatives.

5.2. Open Scope

In Section 4 the use of utility-based constraints to provide a mitigation guarantee was discussed: a binary statement regarding password security was expanded to utility-based statements relating cost, a , attack likelihood or success probability, $p_{s|a}$, and the monetisation value of the attack, M . This wider emphasis highlights the range of design choices that will actually affect system security. While the value of a password may not immediately appear controllable by the system architect, this is essentially the effect of password database encryption — the rendering of the password worthless as a target. In a wider sense, various external system aspects considered holistically offer possibilities for system design that include prevention, detection, and response. A comprehensive design process may recognise that an existing efficient password monitoring/revocation system (were one to exist) might also serve to meet the utility constraints of the attackers, thus fulfilling the goal. Of course, in this instance the cost of operating such a system would become a constraint on the system as well, and would need to be considered, prompting further utility constraints.

This additionally serves as an integration point that incorporates enterprise level considerations within the system development at a functional level. Assuming data regarding effectiveness (increasingly available from a variety of sources, or from the existing security enterprise), it is possible to consider the effects of external controls, third-party additions, and intangible related processes such as policy, training and governance. The resulting decision process is

thus able to be better integrated with the business planning efforts of the organisation, captures a fuller view of the security issues, and places focus where it is most needed to address overall security goals.

5.3. Countermeasure Selection

Returning to the updated use case model in Figure 2, we consider the introduction of the security use case ‘Encrypt Communication’⁸ in light of this construction.

- 1) The probability of attack and probability of success parameters are updated to reflect the mitigation action represented in Figure 2. As an example, it may be decided that the sniffing misuse case probability of success moves to a range of 0.001 – 0.005, depending on the implementation chosen (among other factors). We now see that this misuse case has fallen to the lowest priority of our three considerations, as well as below the utility we have defined for its consideration ($1 \not\leq 0.95$ and $10 \not\leq 4$).
- 2) The ‘Mitigation Point(s)’ field is now populated with information relative to the security use case under consideration. Specifically, this would include a range of costs (ideally based on market research and planning estimates) for the technology (or technologies) under consideration. An example for the ‘Sniffing’ misuse case is provided in Table 7.

These changes result in a re-evaluation of the negative use case processes, noting that the Sniffing misuse case has not disappeared, but has been reduced below the expected utility to be gained (and thus below our risk threshold). Threats are never removed but instead managed, reflecting current risk-driven practices. New information or changes in the environment may later require re-visitation of the assumptions upon which we have performed this calculation, and permit the execution of ‘what-if’ scenarios to satisfy concerns or emerging trends. As such, we have effectively instituted the explicit consideration of security within the development process for as long as development occurs.

Additionally, methods to deal with the inherent trade-offs of the development process now become available. As an example, we note that the options for mitigation via encryption — data-level encryption versus link-level encryption — represent a design choice to be realised by the developer. The current misuse case construct offers no ability to convey information that might affect this preference, while a utility-oriented approach provides such a mechanism; for instance, the ‘better’ security option (in terms of lower P_s) may be more expensive, but may also have a higher instance of error (due to the difficulty to properly implement the protocol, or utilise the API). This is notionally represented by the confidence values in Table 7, using values that reflect reported successful implementation rates for mobile application development (as reported

8. Note that we prefer the ‘Protect Info’ name for this use case as employed in [11]; however, we use the nomenclature of [17] to maintain consistency.

in [30] and [31]). This supplies the system architect with an explicit decision: when does it make sense to spend fewer resources in order to meet security needs — especially when the outcome is in doubt? For this specific case the conceptualisation of misuser actions as utilities permits us to bring to bear constructs of game theory. In doing so, we find that a tenuous equilibrium exists in which link-level security provides the best option; however, perturbation of the probabilities and values involved can move this point, requiring careful consideration by the architect.

6. Conclusions

We have presented an extension to negative use case practice that explicitly incorporates economic considerations via the employment of utility-based expressions. A method for incorporating this information into the misuse case template was proposed, with the goal of providing extensibility to misuse case definition. The employment of utilities as a means to constrain and inform the later analysis and design process was demonstrated. An example was presented using constructs from the literature to highlight the benefits of the proposed approach, which include the promise of reducing premature security design lock-in, the ability to consider multiple paths to achieve security goals, and the consideration of software and system contributions to security — independently and in tandem.

In addition to addressing the three issues identified, this construct supports other positive aspects of negative use case design:

- It supports the clarification of what are often vague, generic descriptions of security (e.g. “The system shall be secure against all attacks”). This is more critical when security requirements inevitably come into conflict with one another, or with functional requirements. These constraints identified by utilities are likely to be a consistent aspect of the project — even as requirements change.
- It provides a natural trade-space for one of the greatest challenges to building and maintaining security: resource constraints. Where security professionals increasingly feel they are inadequately resourced⁹, the quantification and prioritisation provided in utility-based representations permit a more rational approach to decision-making in the face of scarcity and uncertainty in the security landscape.
- In addition to the purely algebraic, risk-oriented operations described here, the economics discipline provides multiple constructs for the calculation, analysis and comparison of utility-based expressions. Constructions such as game theory are “becoming just as important to the security engineer as

9. For example, a recent survey at Blackhat indicated that 73% of security professionals polled don’t believe they will have the resources to adequately deal with attacks they will face; see <http://www.darkreading.com/vulnerabilities-threats/poor-priorities-lack-of-resources-put-enterprises-at-risk-security-pros-say/d/d-id/1321308>.

Name: Obtain Password by Sniffing
Summary: A misuser who seeks to obtain the password from the ‘Order Goods’ use case by eavesdropping on the communication between system components.
...
Mitigation Point(s): Encrypt Communication: Protect password data with encryption
Protect with data-level encryption: $c_D = 10$, $p_s = 0.001$ with a confidence of 0.22
Protect with link-level encryption: $c_L = 8$, $p_s = 0.005$ with a confidence of 0.83
Misuser Profile: Obtain Password
Prob of attack (p_a): High ($0.8 < p_a < 0.95$); internet-connected
Prob of success (p_s): High ($0.95 < p_s < 0.99$); passwords easy to identify & desired
Mitigated to Very Low ($0.001 > p_s > 0.005$) with encryption
Attacker cost (a_i): Low ($1 < a_i < 10\text{mh}$). Requires simple, readily available tools.

TABLE 7. MISUSE CASE SPECIALISATION FOR OBTAIN PASSWORD BY SNIFFING.

the mathematics of cryptography” [32], and provide a rich set of methods for more involved analysis. A burgeoning information security economics research community has produced numerous publications that form security investment considerations as game theoretic trade-offs, such as the consideration of insurance versus protection measures [33], the balance between proactive and reactive security [34], and alternative conceptualisations of security [35] — lending to more insightful security design.

Our work differs from others who have explored augmentations to the negative use case concept. In [16], Heikka and Siponen seek to address many deficiencies they see in misuse case development, to include the integration of risk analysis and security requirement prioritisation. However, to accomplish this they propose an ‘extended’ use case notion that includes fields for priority, frequency, cost of recovery, and description of countermeasure — thus conflating requirements and design, with the potential of contributing to the problem of premature design lock-in. Røstad [36] proposes the introduction of visual and textual conventions for representing vulnerabilities as well as differentiating between insider and outsider threats. While attempting to tackle the question of quantification and risk as part of misuse case development, Røstad largely defers to alternative approaches such as attack trees [37] and falls back on difficult-to-quantify adjectives such as ‘reasonable’ and ‘robust’. The use of utility constructs to make mobile application resource trade-offs is explored in [38], with the goal of enhancing user experience rather than optimal design. While in a similar spirit to our approach, this work does not specify how the proposed approach is to be considered in the context of software development, or how the use of economics relates to security considerations. Finally, Hartong et al. [39] seek to harmonise the representation of misuse and abuse cases into a single, UML-driven meta-model.

It is our belief that the additional dimensions provided by utility-based negative use case specification provide a robust means for expressing the nuances that current system security dictates. In addition to being a vehicle for quantitative justification, these constructs provide a link to the qualitative and risk-based methods currently in use. Recent breaches have reminded us that attackers are resourceful,

and attributing motive to attacks is almost as difficult as attributing actions — rendering a value-free characterisation of threats too simplistic and disjoint from the rest of the security process. It is unlikely that Sony considered a nation-state attack (as has been attributed in the media) when designing their systems. While we do not propose that our approach solves this issue, it renders such considerations explicit and bounded, and provides a construct for examining the possibility space early in the software process.

Future work in this area will more fully explore the incorporation of external, system-level security considerations such as Cyber Essentials [40]. Security relies on compliance, and those developing software must take into account the security environment and information governance policies within which the system is to exist. This is critical both for the control of undesirable emergent properties and proper execution, as well as the maximisation of the security investment. To accomplish this, it is necessary to explicitly convey the security state of the system — the controls it implements, as well as those expected to exist in order to meet the security target. As an augmentation of a widely-adopted methodology that has found use in both traditional and agile development processes, it is worth investigating the contribution of the proposed method to rapid-development and experimental projects in order to balance security–usability challenges. Finally, only direct application of these methods to real-world development will provide the evidence in support of the widespread adoption of these concepts. To this end, we propose further application of these principles in a controlled environment in order to determine their feasibility and effectiveness.

References

- [1] I. A. Tøondel, M. G. Jaatun, and P. H. Meland, “Security requirements for the rest of us: A survey,” *IEEE Software*, vol. 25, no. 1, pp. 20–27, Jan 2008.
- [2] P. Hope, G. McGraw, and A. I. Antòn, “Misuse and abuse cases: Getting past the positive,” *IEEE Security & Privacy*, vol. 2, no. 3, pp. 90–92, 2004.
- [3] NIST Computer Security Division, “Information security: System development life cycle,” PDF, August 2004, pamphlet. [Online]. Available: http://csrc.nist.gov/groups/SMA/sdlc/documents/SDLC_brochure_Aug04.pdf

- [4] R. Ellison, R. Linger, T. Longstaff, and N. Mead, "Survivable network system analysis: A case study," *IEEE Software*, vol. 16, no. 4, pp. 70–77, July 1999.
- [5] J. McDermott and C. Fox, "Using abuse case models for security requirements analysis," in *Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC '99)*. Washington, DC, USA: IEEE Computer Society, 1999, pp. 55–64. [Online]. Available: <http://dl.acm.org/citation.cfm?id=784590.784691>
- [6] G. Sindre and A. L. Opdahl, "Eliciting security requirements by misuse cases," in *Proceedings of the 37th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-Pacific 2000)*, November 2000, pp. 120–131.
- [7] G. McGraw, *Software Security: Building Security In*, 1st ed. Addison-Wesley Professional, 2006.
- [8] M. Howard and S. Lipner, *The Security Development Lifecycle*. Redmond, WA, USA: Microsoft Press, 2006.
- [9] The Open Web Application Security Project (OWASP), "Category:OWASP CLASP project - OWASP," https://www.owasp.org/index.php/Category:OWASP_CLASP_Project, 2015, webpage. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_CLASP_Project
- [10] B. De Win, R. Scandariato, K. Buyens, J. Gregoire, and W. Joosen, "On the secure software development process: CLASP, SDL and Touchpoints compared," *Elsevier Information and Software Technology*, no. 51, pp. 1152–1171, 2009.
- [11] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Engineering*, vol. 10, no. 1, pp. 34–44, January 2005. [Online]. Available: <http://dx.doi.org/10.1007/s00766-004-0194-4>
- [12] J. McDermott, "Abuse-case-based assurance arguments," in *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC 2001)*, December 2001, pp. 366–374.
- [13] G. Sindre and A. L. Opdahl, "Templates for misuse case description," in *Proceedings of the 7th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2001)*, June 2001, pp. 4–5.
- [14] I. Alexander, "Misuse cases: use cases with hostile intent," *IEEE Software*, vol. 20, no. 1, pp. 58–66, January 2003.
- [15] G. Sindre and A. L. Opdahl, "Capturing security requirements through misuse cases," in *Proceedings of the 14th 14th Norwegian Informatics Conference (NIK 2001)*, 2001.
- [16] J. Heikka and M. Siponen, "Abuse cases revised: An action research experience," in *Proceedings of the 10th Pacific Asia Conference on Information Systems (PACIS 2006)*, July 2006.
- [17] G. Sindre, A. L. Opdahl, and G. F. Brevik, "Generalization/specialization as a structuring mechanism for misuse cases," in *Proceedings of the 2nd Symposium on Requirements Engineering for Information Security (SREIS'02)*, 2002.
- [18] G. Sindre, D. G. Firesmith, and A. L. Opdahl, "A reuse-based approach to determining security requirements," in *Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2003)*, 2003, pp. 16–17.
- [19] S. Lilly, "Use case pitfalls: Top 10 problems from real projects using use cases," in *Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS 30)*, August 1999, pp. 174–183.
- [20] H. R. Varian, *Intermediate Microeconomics: A Modern Approach*, 8th ed. W. W. Norton & Company, 2010.
- [21] HM Government, "Information security breaches survey 2015," <https://www.gov.uk/government/publications/information-security-breaches-survey-2015>, June 2015.
- [22] R. Caralli, J. Stevens, L. Young, and W. Wilson, "Introducing OCTAVE Allegro: Improving the information security risk assessment process," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-2007-TR-012, 2007. [Online]. Available: <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8419>
- [23] I. Fléchaïs, "Designing secure and usable systems," Ph.D. dissertation, University of London, February 2005.
- [24] B. Nuseibeh, "Weaving together requirements and architectures," *IEEE Computer*, vol. 34, no. 3, pp. 115–117, March 2001. [Online]. Available: <http://dx.doi.org/10.1109/2.910904>
- [25] K. R. van Wyk and G. McGraw, "Bridging the gap between software development and information security," *IEEE Security & Privacy*, vol. 3, no. 5, pp. 75–79, 2005.
- [26] L. A. Gordon and M. P. Loeb, "The economics of information security investment," *ACM Transactions on Information and Systems Security*, vol. 5, no. 4, pp. 438–457, November 2002.
- [27] J. Peeters and P. Dyson, "Cost-effective security," *IEEE Security & Privacy Magazine*, vol. 5, no. 3, pp. 85–87, 2007.
- [28] K. Hoo, "How much is enough? A risk management approach to computer security," Ph.D. dissertation, Stanford University, Stanford, CA, 2000.
- [29] G. Stoneburner, A. Y. Goguen, and A. Feringa, "NIST SP 800-30. risk management guide for information technology systems," Gaithersburg, MD, United States, Tech. Rep., 2002.
- [30] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith, "Why eve and mallory love Android: An analysis of Android SSL (in)security," in *Proceedings of the 2012 ACM SIGSAC Conference on Computer and Communications Security (CCS 2012)*. ACM, 2012, pp. 50–61.
- [31] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel, "An empirical study of cryptographic misuse in Android applications," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS 2013)*. ACM, 2013, pp. 73–84.
- [32] R. Anderson and T. Moore, "The economics of information security," *Science*, vol. 314, no. 5799, pp. 610–613, October 2006.
- [33] J. Grossklags, N. Christin, and J. Chuang, "Secure or insure?: A game-theoretic analysis of information security games," in *Proceedings of the 17th International Conference on World Wide Web (WWW 2008)*. ACM, 2008, pp. 209–218.
- [34] R. Böhme and T. Moore, "The iterated weakest link – a model of adaptive security investment," in *Proceedings of the 8th Annual Workshop on the Economics of Information Security (WEIS 2009)*, 2009.
- [35] C. Heitzenrater, G. Taylor, and A. C. Simpson, "When the winning move is not to play: Games of deterrence in cyber security," in *Lecture Notes in Computer Science*, vol. 9406. Springer, 2015, pp. 250–269.
- [36] L. Røstad, "An extended misuse case notation: Including vulnerabilities and the insider threat," in *Twelfth Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2006)*. Essener Informatik Beitrage, June 2006, pp. 33–43.
- [37] B. Schneier, *Secrets & Lies: Digital Security in a Networked World*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 2000.
- [38] V. Poladian, D. Garlan, and M. Shaw, "Software selection and configuration in mobile environments: A utility-based approach (position paper)," in *Fourth Workshop on Economics-Driven Software Engineering Research (EDSER-4)*, affiliated with the 24th International Conference on Software Engineering (ICSE 2002), May 2002.
- [39] M. Hartong, R. Goel, and D. Wijesekera, "Meta-models for misuse cases," in *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies (CSHIRW 2009)*. ACM, 2009, pp. 33:1–33:4.
- [40] Department for Business, Innovation and Skills (BIS), "Cyber essentials scheme: Requirements for basic technical protection from cyber attacks," https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/317481/Cyber_Essentials_Requirements.pdf, June 2014.