A Case for the Economics of Secure Software Development

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

Over the past 15 years the topic of information security economics has grown to become a large and diverse field, influencing security thinking on issues as diverse as bitcoin markets and cybersecurity insurance. An aspect yet to receive much attention in this respect is that of secure software development, or 'SWSec' — another area that has seen a surge of research since 2000. SWSec provides paradigms, practices and procedures that offer some promise to address current security problems, yet those solutions face financial and technical barriers that necessitate a more thorough approach to planning and execution. Meanwhile, information security economics has developed theory and practice to support a particular world-view; however, it has yet to account for the investments, constructs and benefits of SWSec. As the frequency and severity of computer misuse has increased, both areas have struggled to impart a new mindset for addressing the inherent issues that arise in a diverse, connected and functionality-driven landscape.

This paper presents a call for the establishment of an economics of secure software development. We present the primary challenges facing practice, citing relevant literature from both communities to illustrate where commonalities lie — and where further work is needed. Those challenges are decomposed into a research agenda, deriving from the application of principles in both themes a lack of models, representation and analysis in practice. A framework emerges that facilitates discussions of security theory and practice.

CCS Concepts

•**Security and privacy** → **Economics of security and privacy;** *Software security engineering;* •**Software and its engineering** → *Software design tradeoffs;*

Keywords

information security investment, secure software engineering, software security economics

Approved for Public Release; Distribution Unlimited: 88ABW-2016-3869 20160803
NSPW '16 September 26–29, 2016, C Lazy U Ranch, Colorado, USA
© 2016 ACM. ISBN .
DOI:

1. INTRODUCTION

There is a growing realisation that many issues faced in computer security today are rooted in our approach to developing software and systems. The fields of secure software engineering and information security economics are both built on this assertion, albeit to different ends: secure software development (otherwise known as 'SWSec' or secure software engineering) leverages software engineering practice and risk management to raise the quality of security decision making in all phases of software and system development [51], while information security economics identifies the externalities and misaligned incentives that drive how those decisions are made [3]. An approach to secure software that properly integrates various processes while aligning these incentives is thought to be the key to addressing the 'Three Headed Cyber Cerberus' of cyber crime, cyber espionage and cyber war [54].

Building systems 'properly' is difficult and fraught with problems that straddle both of the aforementioned fields. There is growing evidence that cost continues to impede the implementation of security best practice by corporations[1]. This is no wonder, as the additional processes and tools required for SWSec can be costly, requiring expertise, resources and time often unavailable to the development organisation (if available at all). At the heart of secure software engineering are processes that rely on expert judgement, require specialised expertise, and exhibit variability in their effectiveness [45]. Even when budgeted for and successfully executed in a particular project, there will be dependencies; components not built or owned by the developer, and the need to connect to systems developed with less rigour are examples of this phenomenon. Decisions are made at every step that impact the security balance of the programme, and current practices aren't well suited to make these trade-offs — or even articulate when they have been made. For all of the guidance on what to do in the name of software security, there is little on applying these precepts in practice.

This paper attempts to capture the salient concerns when considering the economics of secure software development. Following an outline of the paradigm in Section 2, we systematically develop each aspect of the concept. The nec-

[1]For example, recent reports have cited cost as a barrier to the adoption of the NIST Cybersecurity Framework; see http://www.darkreading.com/attacks-breaches/nist-cybersecurity-framework-adoption-hampered-by-costs-survey-finds/d/d-id/1324901.

essary elements of the secure software engineering field are introduced in Section 3, serving as the basis for the remainder of the discussion. In Sections 4–6, we investigate the economic aspects exhibited by software security practice. These are summarised in Section 7, in which we also present thoughts on possible future directions in this area.

2. PARADIGM

When considering relevant factors that inform the realisation of system security, various aspects spring to mind: desires to balance functionality and the need to ship product, the emergent behaviour of complex systems, and — most visibly — post-deployment processes (e.g. patching and detection) that attempt to maintain secure system operation. Threatening each of these factors are errors in system design, implementation, or configuration. While the software engineering community has long understood that costs to remediate errors escalate as one moves through system development [12, 79], security remains largely focused on post-deployment measures late in the system life-cycle. Achieving higher returns on security investment by building security into the software engineering process would seem intuitive, however there is little in the way of proof to support such an assertion [2].

Even under this assumption, there are significant barriers to realising such a vision. Foremost, most developers are not security professionals (and vice-versa), resulting in a lack of understanding as to how development activities relate to end-product security [33]. Instead, security is often presented as a set of theoretical concepts or is overly-focused on top-10 lists [5]. Prescriptive requirements for implementing security are often lacking — setting up developers for failure [48]. Management is often reluctant to make investments in security processes or procedures without a business case demonstrating a tangible return on investment[2]. Perhaps most tellingly, the tools to enable anyone within the process to make a compelling argument for such investment are lacking — especially in the case of investments within secure development processes.

The nature of such tools must then balance the technical and non-technical (e.g. managerial) concerns that security raises. Information security economics enables a structured approach to understanding the current state of software security through the lens of the various technical and managerial factors that drive practice. Our attention is placed on three questions that capture the essence of a practice of secure software development economics and frame the contribution of this paper:

- *How can SWSec be better represented within investment models?* To date, investment models have largely failed to incorporate early life-cycle development costs, rendering investment into SWSec unaccountable. Although no easy task, the literature pertaining to the equally difficult task of enterprise investment is active and growing; we explore the motivations for, and approaches to, addressing this paucity.

- *What is the proper balance between security and functionality expenditure?* A common critique of both information security investment and SWSec is that security is paramount, yet other factors (such as time-to-market or system features) often take precedence. Secure software paradigms fail to address this trade-space, while investment models incorporate this assumption (often implicitly) into their operation [70]. We examine how this balance may be explored, starting with the question of a sufficient construct for such an analysis.

- *When is security better served by 'building-in' rather than 'building-around'?* As with any other security construct, SWSec is but part of a larger picture that addresses security at multiple points. SWSec distinguishes application and software security practices that enable, but cannot replace, the enterprise, IT-oriented practices that are the focus of standards and practice today. Nuance in this space is often overshadowed by condemnation of the current status quo. An approach that establishes the proper scope and balance while recognising that today's security apparatus — as misguided as it may be — will not, and probably should not, disappear entirely, is required. We consider the barriers to improved practice in this area.

A framework rooted in utility theory, decision support and cost-benefit analysis will enable a more systematic and rational planning of software development, and of security overall. Through the inclusion of security investments made during development, the overall security investment is rendered more cost-effective [34], implementation decisions can be made coherently with security concerns [35], and the effectiveness of the overall security enterprise can be analysed [36]. To anchor further discussion, we present a summary of the current state of secure software engineering practice.

3. THE STATE OF SOFTWARE SECURITY

While software security as a practice is typically attributed to post-2000 efforts to address the growing rise of computer intrusions [54], it has roots in the software management field where cost modelling, defect reduction and reliability have long been of concern to software practitioners [13]. SWSec differentiates itself from application security by the latter placing focus on securing software after it is written (and therefore being network-centric), while the former is concerned with building better software [51]. While software management seeks quality as an inherent attribute, software security recognises the emergent nature of security and elevates security over features and functionality [74]. Yet, little guidance is provided by the software security community to guide the implementation of various practices.

3.1 SWSec Practise

Software management and software security unite on the premise that security within systems fundamentally rests on the elimination of defects that make an application vulnerable to attack [73]. To this end, the software security community has fostered a number of efforts to address defect removal. For example, in 2003 the Open Web Application Project (OWASP) group[3] initiated their 'Top 10 Project' to generate awareness of poor coding practices. Organisa-

[2]http://www.cio.com/article/3063738/security/it-leaders-pick-productivity-over-security.html

[3]https://www.owasp.org/index.php/Main_Page

Domain	Practices
Governance	Strategy & Metrics (SM) Compliance & Policy (CP) Training (T)
Intelligence	Attack Models (AM) Security Features & Design (SFD) Standards & Requirements (SR)
SSDL Touchpoints	Architecture Analysis (AA) Code Review (CR) Security Testing (ST)
Deployment	Penetration Testing (PT) Software Environment (SE) Configuration Management & Vulnerability Management (CMVM)

Table 1: The BSIMM domains and associated practices [56]

tions such as Microsoft[4] and Adobe[5] have generated their own development paradigms, as OWASP itself has sought to codify its own[6]. Recently, initiatives such as the IEEE Center for Secure Design's "Avoiding the Top 10 Software Security Design Flaws" [6] have extended this concept to architectural considerations. Despite criticism for failing to consider security in a global context [75], these initiatives are widely heralded for bringing security considerations into the development process.

Perhaps the most prevalent framework for thinking about software security is the Building Security In Meta-Model (BSIMM), heralded as supplying "science to security" by way of a systematic survey of security best practice [56]. BSIMM decomposes into four domains, each comprised of three practices that are further comprised of activities that associate with a level (1 to 3) to indicate the maturity demonstrated through the execution of the activity. The four domains and associated 12 practices are summarised in Table 1.

Key within BSIMM are the Secure Software Development Lifecycle (SSDL) Touchpoints (Figure 1). Originally published in [51], Touchpoints is a set of seven practices that overlap the BSIMM decomposition, directly addressing the need for 'building security in'. Unlike BSIMM, which merely catalogues practice (while supplying a notion of maturity), Touchpoints takes the additional step of ordering practices according to their effectiveness[7]:

1. Code review
2. Architectural risk analysis
3. Penetration testing
4. Risk-based security tests
5. Abuse cases
6. Security requirements
7. Security operations

While collectively the Touchpoints are widely identified, individually they are also well-known, if not standard, software engineering practice. The Touchpoints construct supplies an additional emphasis on the security aspects of these practices, where code review, architectural analysis, testing and requirements methodologies have long been practised under the guise of software quality and reliability. Each Touchpoint is intended to focus recognised engineering steps to specifically address security concerns. As such, the Touchpoints have their own mapping to standard system and software engineering practice (see Table 2). For this discussion it is not essential to understand the Touchpoints approach other than as a recognised collection of security-focused software engineering practices that is core to SWSec as a discipline.

The tandem of BSIMM and Touchpoints provides a basis to address the global and emergent nature of software security, although some will note that collections of 'best practice' do not constitute formally evaluated methodologies [80]. Touchpoints is often presented as an ordered list in order to impart effectiveness, and while BSIMM appears to bear out this ranking, the original basis for this ordering lies with the experience of the Touchpoints developer [55]. In addition, these constructs leave to the practitioner the exercise of identifying how much, or how little, to apply to each activity. This is certainly not to say that such approaches are without value, but, rather, are incomplete in their depiction of security practice. Missing from a complete, robust SWSec tradition are the measures of effectiveness and tools for analysis to guide both the research and practice of SWSec.

3.2 Modelling SWSec

In order to foster discussion, we present a simple model for discussing SWSec in the context of security economics. As identified in [84], any effort to model or measure security requires us to identify a relationship with a specific definition of security. We focus on the SWSec emphasis on software defects as the root cause of computer security failures writ large [51], rendering the identification and removal (or mitigation) of defects as the measure of successful computer security.

3.2.1 Types of failure

It is often the case that the term 'security' is used to describe failures that stem from the actions of people, policy, and systems. Properly scoping the contribution of SWSec to overall security requires that we are nuanced in our discussion of security failures. For this paper we adopt a version of the classification presented by Aslam *et al.* in [7] and later adopted for economic considerations by Camp and Wolfram in [17]. This model employs two primary categories, each with two sub-categorisations:

- *Coding faults*, introduced during development as a result of errors in programming logic, missing or incorrect requirements, or design errors. This can be further decomposed into *synchronisation errors* and *condition validation errors*. Note that the definition for such a category encompasses what the SWSec community considers *bugs* (implementation-level problems) and *flaws* (design-level problems) [83]. Bugs such as

[4]The Microsoft Secure Development Lifecycle (MS SDL), http://www.microsoft.com/en-us/sdl/.
[5]The Adobe Secure Product Life Cycle (SPLC), http://www.adobe.com/security/proactive-efforts.html.
[6]The Comprehensive, Lightweight Application Security Process (CLASP), https://www.owasp.org/index.php/Category:OWASP_CLASP_Project.
[7]From http://www.swsec.com/resources/touchpoints/.

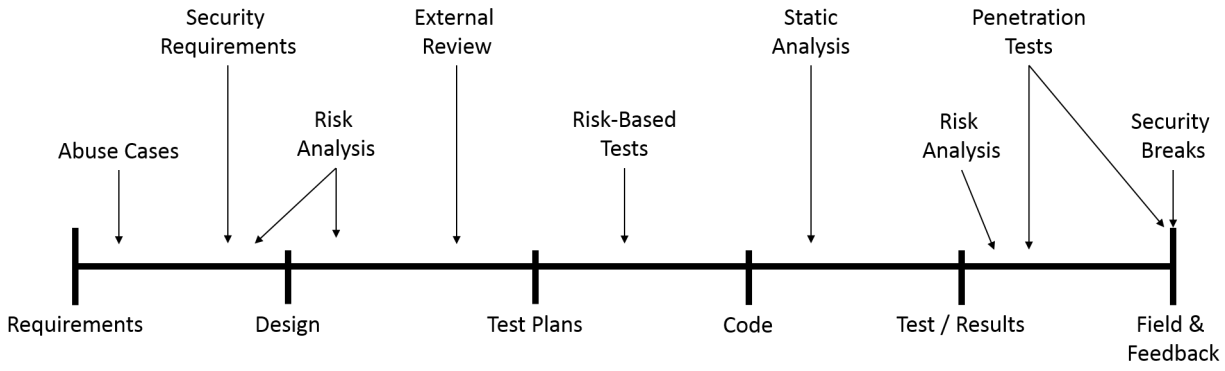

Figure 1: The SDDL Touchpoints (figure adapted from [53], and used with permission).

buffer overflows are a common error (and are the focus of many SWSec efforts such as the OWASP Top 10[8] list), while flaws such as the failure to employ cryptography are often more subtle (the identification of which is focus of recent efforts by the IEEE [6]).

- *Emergent faults*, where the software performs to specification yet still causes failure, due to installation errors, integration incompatibilities, or a misunderstanding of the run-time behaviour. *Configuration errors* and *environment errors* make up this category of fault. Common examples include the failure to close ports, or the misconfiguration of cryptography that causes insecurity (e.g. key re-use). Many standards and accreditation check-lists focus on this type of fault.

Despite the convention — and as apparent in the examples cited above — it would be a fallacy to assume SWSec is concerned only with coding faults. Conspicuous in the BSIMM practices and Touchpoints processes are activities related to the security environment, configuration & vulnerability management, and operations (Table 1).

3.2.2 SWSec Scope

In order to structure a consideration of SWSec processes and their implications to security expenditure, a mapping between the System Development Life-Cycle (SDLC) [24], software engineering practice, Touchpoints and the BSIMM has been provided in Table 2. From the BSIMM mapping, an estimate of the various investment categories can be derived [56], to include:

- (M)anpower, e.g. man-hours of expertise,
- (T)ools, such as specialised software, and
- (E)quipment, or the acquisition of hardware.

The magnitude of these costs vary on many parameters, to include the size of the organisation, the in-house security and development expertise, and the risk appetite endured. We limit focus to the Touchpoints activities, exploiting the cursory prioritisation they provide as a starting point to considering the economic implications of SWSec.

[8]https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

These dual concepts of fault categorisation and phase investment provide the necessary foundation upon which a framework for SWSec economics can be examined, analysed and developed. Although not a definitive categorisation, by placing our framework within the context of existing practice we seek to develop the theoretical underpinnings that unify the empirical research into software engineering with the applied examination of efforts such as BSIMM. This will be accomplished by considering investment models, utility-derived analysis, and pre-/post-deployment security balance in support of the SWSec paradigm.

4. SWSEC INVESTMENT MODELS

We now turn to the consideration of SWSec within security investment models. Unfortunately, defending from all credible threats — being 'truly secure' — is prohibitively expensive, and as a result the optimisation of how to apply limited resources is a central challenge in unifying software and systems engineering [23]. Resource allocation, particularly in the face of various incentives and externalities, has been central to the emerging practice of information security economics since its inception [3]. Security investment models have previously been classified into different types [71], to include accounting models [29, 40, 14], game-theoretic models [18, 62], and macroeconomic-focused input/output models [4]. Unfortunately, these standard cybersecurity economic model archetypes are not easily applied to software security [61].

4.1 The problem of risk

Even good models are useless without good data, and the challenge of choosing appropriate models and applying them in practice is compounded by the availability of robust, relevant and reliable data [71]. This has led some to question the ability to quantify security, at least in an operational sense [84]. As a result, the application of security investment models in practice is difficult, even in the case of post-deployment, enterprise security investment [70].

Yet software engineering practice has engaged in quantified economic research for years, primarily under the guise of quality, reliability and safety [13]. Recent investigations have highlighted the effectiveness of code reviews [25], static analysis [9, 30, 73] and penetration testing [8, 73] in eliminating vulnerabilities. Coupling such findings with long-

SDLC	Software Phase	Touchpoints	BSIMM	Costs
Initiation	Scope Requirements	Abuse cases Security requirements	AM SR, CP, SFD	M M
Acquisition & Development	Architecture Design Code	Architectural risk analysis Code review	AA CR	M M + T
Implementation & Assessment	Test Accreditation	Risk-based security tests	ST	M + T
Operations & Maintenance	Deployment	Penetration testing Security operations	PT SE, CMVM	M + T M + E + T
Sunset (Disposal)				

Table 2: Touchpoints alignment to the NIST SDLC, software engineering practice, the Touchpoints practices and the BSIMM. Cost types are those identified by the BSIMM sub-activities [56].

standing knowledge regarding the fault removal effectiveness of code reviews, architectural analysis and testing (e.g. [45, 12]) makes secure software process investment ripe for quantified analysis.

Why, then, have economic models for secure software development not yet emerged? Examining model types (as defined in [71]), we find a common theme to their quantification approach: risk-based estimation. While risk serves an important role in both SWSec and security economics, as with many engineering constructs it is often applied for purposes beyond its intended use. Risk plays a key role in modern SWSec and computer security in general; however, little research in assessing and ranking risks during the early part of the SDLC, where investment planning is performed, has been accomplished [85].

Information security investment models reliant on risk estimation inherently estimate operational risks, often requiring a definition of the target system (in terms of hardware and software) [15, 86]. Other methods, such as the NIST Cybersecurity Framework, rely on risk estimation in order to gauge resource estimates such as staffing and funding to "achieve cybersecurity goals in a cost-effective, prioritized manner" [1]. While this may align business needs and set overall budget, it inherently assumes post-deployment investment and provides little guidance on individual investment priorities. Such techniques do not provide guidance at the component or environment level, leaving a gap between their outcomes (for instance, an Annual Loss Expectancy (ALE) calculation) and a detailed software security mitigation definition [52]. As a result, SWSec approaches based in risk are often necessarily qualitative, exemplified by techniques such as CAIRIS [26], SAEM [16], and the (Touchpoints-cited) ARA process[9]. Transforming such assessments into meaningful and credible qualitative statements is typically beyond the ability of the development team [16]. While critical to security, such analyses are but a "yardstick by which we can judge our security design effectiveness" [83].

Fundamental to qualitative accounting measures (such as ALE) is the need to understand the system's assets, their costs and the probability of loss — undetermined quantities in systems still under development [59]. Effectiveness, a characteristic difficult to characterise at the operational level [36], becomes even more ambiguous; where most enterprise security investments can be expected to operate deterministically, investments into SWSec are largely manpower-focused and compounded by factors such as expertise [10] and tool(s) employed [30]. The uniqueness of the vulnerabilities uncovered by each method [8] further affects the overall number and balance of coding faults and emergent faults (Section 3.2.1). Complexity and a lack of data emerge as a barrier to accurate analysis [15].

Any risk assessment is itself risky, as findings are potentially multiple steps removed from the environment under analysis [31] and the pre-artefact, variable nature of software development renders the modelling of SWSec investment as a risk-based venture particularly daunting. While appropriate uses of risk analysis within development remains a best-practice, SWSec investment requires understanding of what is being purchased, and what quantity is appropriate — necessitating a re-evaluation of traditional risk-based models and metrics.

4.2 New models or new thinking?

In addition to issues surrounding qualitative risk estimation, existing information security investment models share common assumptions that undermine their validity when applied to SWSec investment:

- *They assume that investment is made in its entirety*, which may not always be the case [21] — especially for software security investments, which are process-oriented and reliant upon both effort and expertise. SWSec investments occur at different phases of the development process (requirements, architecture, design, etc.), and throughout the SDLC, while current investment models optimise investment at a fixed point (generally, post-delivery during Operations and Maintenance). As a result, these models do not readily apply to the incremental investment decisions made during development (e.g. determining the number of code reviews to conduct, or the amount of manpower to devote to security testing during development).

- *They require the existence of process artefacts.* Alternative metrics such as the Cost to Break (CTB) [74] or the Return on Attack (ROA) [19] are enabled by concrete outcomes, such as system designs, attacker models, or functional systems. During the early phases of a development effort such artefacts may not exist (or may be constantly changing), and modern development approaches may eschew customary items for

[9]The Architectural Risk Assessment process, https://www.cigital.com/presentations/ARA10.pdf.

others that are not as easily employable.

- *They fail to reflect temporal change.* By definition, the state of security within a system will vary during its construction, rendering fixed point metrics such as Return on Security Investment (ROSI) [2, 58] problematic even if probabilities can be assigned. Formulations such as Net Present Value (NPV) and Internal Rate of Return (IRR) have been employed [57], however their appropriateness has been called into question [67]. Such calculations must either be continuously re-evaluated with updated metrics (requiring continuously re-calculated data), or generalised (e.g. made qualitative). Even then, such approaches are not guaranteed to be accurate and pose a risk that the small changes naturally occurring during development will result in vastly different outcomes for the analysis.

Frameworks such as that described in [63] attempt to articulate how the merits of different models can be rationalised, but to date these efforts lack the rigour required for effective SWSec decision-making. In order to overcome the barriers of risk-based models in this space, it is necessary to conceive of how software security contributes to the ecosystem outside of risk.

Following from the security model in Section 3.2.1, candidate measures that combine SWSec and engineering practice include 'security defects removed' — if only such a measure could be accurately quantified, given unknowns such as the number of faults without process or the number of faults that are exploitable. It is widely accepted that the removal of defects (security or otherwise) is correlated to improved security [73], with many advocating the treatment of security flaws as a special case of quality flaws [41]. This correlation provides the basis for applying software engineering measures to augment, rather than replace, the role of operational or business risk evaluation.

Rather than affecting risk, investments in SWSec could instead be considered as affecting the endogenous aspects of the system such as its inherent vulnerability to attack, or stakeholder valuation of the system in the form of uncertainty reduction and perception of quality. This view is separate, but coherent, with the risk-management view of security investment as a form of risk reduction [15]; however, such a view provides a quantitative basis upon which such appraisals can be measured and compared. Examples of such estimations include the Iterated Weakest Link model [14], which employs an uncertainty parameter σ and a surrogate for quality (the 'attack gradient', Δx) as input parameters. The Gordon-Loeb model [29] employs security breach probability functions with a vulnerability parameter v conveying the quality of the system relative to an attacker class.

However, this perspective requires the development of new or extended models for the evaluation and optimisation of SWSec investment. In current models, the establishment of the identified parameters is an employment choice left to the security manager to set; this must be replaced with sound approaches based on SWSec evaluations such as those identified in Section 4.1. Models that capture the benefit of SWSec investment in terms relative to the BSIMM practices — and relate this investment to current and emerging enterprise models — are necessary for the development information security investment overall. Ultimately, the evaluation of several models in concert in order to define an overall security strategy is likely to be the best approach [67] — although rational investment demands better definition and orchestration of models to paint a broader picture inclusive of SWSec. Beyond investment, achieving a truly complete picture requires the consideration of security in the context of the system and the derivation of technical decisions from determined investment objectives.

5. BALANCING SECURITY WITH FUNCTIONALITY

The ultimate end goal of investment analysis is to allocate expenditure [33]. However, once targets are set, the execution of the security investment is often non-obvious and easily misguided. There is a recognised difficulty in defining the goals, expectations and objectives of security [82] that is tied to the lack of a unified definition for security requirements, or even how concrete or verifiable they should be [81]. As a result, it is very rare for organisations to provide developers with requirements that guide them down the path towards secure software, resulting in the developers being set up for failure [48]. More often, security requirements are confused with architecture and design, leading to premature (and often sub-optimal) design decisions [72].

5.1 Life-cycle considerations

Contributing to this state of affairs is the moving-target nature of security during system development. While modern software development has increasingly moved toward non-traditional approaches to software construction, security considerations remain relegated to fixed points within the life-cycle. Traditional SWSec processes are difficult to apply in modern development, often leading to processes that are repetitive or lack focus when applied to agile methodologies [42].

In addition to the Touchpoints activities, process-driven methods such as the aforementioned MS SDL, Adobe SPLC, and OWASP CLASP specify practices that largely follow traditional development paradigms. Activities conducted within these Security Development Life-cycles (SDLs) are largely focused on addressing coding faults within particular segments of the overall system life-cycle, and while there is evidence that they do reduce safety bulletins [47] SDLs alone are insufficient for addressing the emergent nature of security. While a necessary component to practice, the inability of SDLs to represent and account for the role of the system within the enterprise is a limitation and their failure to consider the architectural context of the enterprise is a limitation of current processes [44]. However, these activities will impact the nature of resulting faults, and, as discussed in the previous section, early investment is key to effective security investment.

Only when placed within higher level meta-models (such as BSIMM or the OWASP OpenSAMM[10]) do these process provide a wider view of security — losing specificity as processes in return. Such constructs are, by definition, collections of 'best practice' in contrast to formally evaluated methods [80]. By the same token, compliance-based approaches (such as the Common Criteria[11] or NIST Au-

[10]The Open Software Assurance Maturity Model, http://www.opensamm.org/.

[11]https://www.commoncriteriaportal.org/

tomated Security Self-Evaluation Tool (ASSET)[12]) could equally be criticised for focusing only on post-construction concerns. Such approaches lead only to very simple solutions [42], and have been criticised for being neither dynamic nor cost-effective [43].

Balancing informal practices with established software development process requires a broader view of security. Integrating, conveying and reasoning about security concerns requires constructs that transcend any one model, providing mechanisms that are employable throughout the lifecycle and applicable to various actors at every stage, starting with project inception. Representation and evaluation of secure development investments within the software engineering process using an economics-based approach provides a common basis upon which functionality and security can be compared.

5.2 Security in an economic context

As early process analysis imparts fluctuations during early development, the ripple effect can be felt throughout the development process and beyond — to deployment and maintenance [20]. An amount of traceability (not explicitly supported by methods such as Touchpoints) is needed in order to support change management [20]. Increasingly, there is recognition that economics plays a role in considering not only current attacks, but possible future attacks [23]. This is especially true as the return on per unit investment continues to shift for both defenders and attackers, who continually adapt and alter their approaches. We conceptualise a computable *security context* that is employable in different phases within the system life-cycle such that it incorporates the technical-economic concerns that security demands.

As SWSec does not seek to replace, but rather augment, decades of software engineering practice, security contexts are likewise best considered as an extension of current practice. Such a construct is best captured within development artefacts in a manner that is unambiguous, recognised, and understandable to developers and security managers alike. Approaches for capturing security contexts include the following:

- Within the specification of security requirements as constraints on functional concerns [72]. These could be developed as requirements in their own right, or more likely in the vein of fit criteria [69]. The latter would necessitate that security requirements are measurable and testable — a challenging task, as there is no standard measure of security [42].

- Within current attack modelling techniques, such as attack trees [76], which have been parameterised and extended to include return on attack/investment considerations and the notion of defence trees (a summary of work in this area can be found in [46]). However, proper attack tree generation requires multiple rounds of iteration to gain fidelity, and expert knowledge to execute [81].

- Within extended current attack modelling techniques, such as those advocated by the SWSec community. Within abuse cases, McDermott and Fox call for specification of the "resources, skills and objectives" [50], while Sindre and Opdahl propose a *Stakeholders and Risks* field within misuse cases to quantify costs and likelihoods "with more ambition" than the use of textual descriptions [78]. The latter is taken a step further in related work, with the integration of misuse case analysis with risk analysis, costing and formal methods identified as a potentially interesting direction to pursue [77]. However, subsequent work has yet to provide any formalism, constructs, or guidance as to the incorporation of these concepts into misuse case analysis processes. As misuse cases serve as a complement to use cases, these concepts could enable functionality-security analysis [37].

Specifying security requirements in the language of utility theory offers a solution. Security expressed as utility relationships provides calculable constraints, supports parameterised assertion of requirements and relationships that is employable within various modelling constructs, and supplies a common basis by which functional, non-functional and policy-based demands can be rationalised. This can be thought of as 'non-functional fit criteria', supplying constraints on the architectural and design decisions through explicit statements related to management concerns such as resources or cost. Preliminary work has demonstrated how such a construct could be used to choose a compliant cryptographic mode that balances functionality and security in order to foster system resilience in Internet of Things (IoT) devices [35]. Expressing security requirements and attacker/defender concerns in this way supports the critical process of refutation [65], and supplies information critical for extending understanding beyond *what* needs to be protected, to include *why*. This enables continued analysis and re-evaluation as the system specification, threat picture and regulatory landscape are clarified over the course of development and deployment.

Additional benefit can be found in the potential to form security constructs that are explicitly tied to system function. Rectifying the constraints presented by functional, non-functional, policy and regulatory security concerns requires a common language with the flexibility to support alternative definitions of security that meet the varied goals of confidentiality, integrity, availability, etc. Forming the basis for techniques such as game-theoretic analysis, utility theory provides the ability to directly link requirements and models to a defined context. The information security economics literature has examined diverse security goals such as protection and self-insurance [32] and deterrence [38]; however, the ability to tie equilibria to software design and development practice has proved difficult, as current utility-based, multi-attribute and 'first principles' approaches (e.g. [11, 66, 68]) have yet to provide widely employed approaches.

Critically, beyond the security demands of a given system, security contexts support overall organisational security goals. This is critical to a comprehensive notion of security, providing a link between the business enterprise and the system implementation. Holistic security demands the existence of the security enterprise, with optimal solutions requiring methods to examine SWSec within the context of the greater security picture.

6. BUILDING-IN VS. BUILDING-AROUND

Many have come to view the problem of security as one of insurance rather than one of investment, with the primary

[12]http://csrc.nist.gov/archive/asset/

goal being the reduction of potential risk [52]. Given the necessity of early expenditure with undetermined benefit and no active adversary, processes underpinning software security may be best viewed in this light. While the practices described in Sections 4 and 5 support improved planning and understanding of security within the scope of SWSec, practical security requires more: the means to specify, analyse and justify the security of software within and without the context of the security enterprise. Central to this are the standards that define the current security landscape.

6.1 Role of standards and guidance

In spite of their limitations, standards and guidance play a central role in security: driving accreditation and licence to operate (e.g. Common Criteria[13]), setting security requirements for industries (e.g. PCI DSS[14]), and establishing minimal guidance for doing business (e.g. UK Cyber Essentials [22]). Unfortunately, often they are also taken as the last word in security (rather than as the starting point), leading some to elevate conformance to standards as the "highest degree of confidence" that can be achieved with regard to software security [28]. Such ex ante regulation has been found to be sub-optimal when those issuing the requirement are uncertain about what minimum standards should entail [60], as is the case in cybersecurity. While conformance is likely to be preferable to an absence of security, mis-implementation can result in elevated costs [36] and has not been shown to unilaterally stem the tide of breaches[15]. Best practice dictates that standards must be evaluated for adoption, rather than blindly accepted [67].

Taking PCI-DSS as an example, we find that such standards are often narrowly focused. As demonstrated by Table 3, this commonly results in an undue emphasis on post-deployment security and emergent (specifically, configuration) faults. It may be argued that this is due to the scope of this particular regulation, but it serves to highlight a danger in designing security to standards: doing so risks a failure to account for, or at the very least under-represent, entire classes of threats and vulnerabilities. If the entirety of the security investment is standards-focused, it becomes evident how compliance is not sufficient for security. Schemes such as ISO/IEC27001[16] or the NIST Cybersecurity Framework[17] offers less prescriptive guidance at the price of ambiguity. This places an onus on system designers to develop, describe and justify their security architecture, while mechanisms to incorporate SWSec investment remain a challenge.

6.2 Spanning the life-cycle

The need to consider operational the mitigation supplied by the deployment environment during development has long been recognised (even if not quantified) [39]. While a primary tenet of SWSec is the notion that software bugs and flaws are at the heart of many security failures, there is also recognition that overall security is an emergent property [51]. Combating the stream of breach reports requires coordinated effort throughout the system's life-cycle. This necessitates methods for reasoning about the overall investment, placing security within the context of the overall system and business requirements, and reasoning about the applicability and cost of adherence to standards.

Taking these points together, a sketch of an augmented SWSec process that explicitly incorporates economic considerations emerges. Figure 2 exemplifies the inclusion of explicit economic concerns within secure software engineering, using BSIMM and Touchpoints as a basis in order to illustrate ties to existing processes.

- The establishment of a security context would occur within the BSIMM Intelligence domain, in parallel to the other activities undertaken in that domain. These activities drive the problem definition and are critical to the overall security approach. The combined result forms the basis for the development of the security context.
- Likewise, the practice of integrating SWSec economic modelling could also occur as a result of the practices within the Intelligence domain. This would necessarily follow the definition of the context, but is also imperative for setting the parameters by which the Intelligence domain activities are conducted. This 'bootstrapping' problem of requiring security expenditure in order to establish the necessary amount of security expenditure is an open problem, but conceptually similar to other problems within software engineering (such as prototype development) that are generally solved by the establishment of best practice.
- These constructs are then referenced and enhanced over the course of the SDDL Touchpoints processes. Specifically, the security context serves as a reference for requirements and design decision-making, as well as a repository for decisions and constraints as they are identified. Investment modelling guides development by establishing the amount and type of resources dedicated to each of the activities.
- Context and investment come together with the Deployment domain activities to inform the development (or augmentation) of the enterprise, its configuration and maintenance.

Taken together, this approach constitutes the emergence of a new consideration within SWSec methodology: the need for the establishment of a practice of economics, alongside the current practices of Table 1. A natural fit for such a practice would be in the Governance domain, as it serves to inform and define the practices that follow — just as Strategy & Metrics, Compliance & Policy, and Training define the overall approach governed by BSIMM adherence. We recognise that the introduction of such a practice into a model such as BSIMM runs counter to its purpose, which is an empirical study of SWSec as opposed to a theoretical approach to practice. This underscores the earlier point regarding the use of meta-models to drive software and security engineering standards: the practice remains reactionary, the state-of-the-art is slow to adapt, and the possibility for

[13] https://www.commoncriteriaportal.org/

[14] The Payment Card Industry Data Security Standard, https://www.pcisecuritystandards.org/pci_security/.

[15] Notably, the US retailer Target was PCI DSS compliant prior to their November 2013 breach; see http://www.computerworld.com/article/2486879/data-security/after-target--neiman-marcus-breaches--does-pci-compliance-mean-anything-.html.

[16] http://www.iso.org/iso/iso27001

[17] http://www.nist.gov/cyberframework/upload/cybersecurity-framework-021214.pdf

Control	Summary	SDLC Phase	BSIMM Activity	Fault Target
1	Install and maintain a firewall configuration to protect cardholder data	Operations	SE	Emergent
2	Do not use vendor-supplied defaults for system passwords and other security parameters	Operations	CMVM	Emergent
3	Protect stored cardholder data	Operations	SE	Emergent
4	Encrypt transmission of cardholder data across open, public networks	Acq/Devel	SR	Coding
5	Use and regularly update anti-virus software	Operations	SE	Emergent
6	Develop and maintain secure systems and applications	Acq/Devel Operations	CMVM	Emergent Coding
7	Restrict access to cardholder data by business need-to-know	Operations	CP	Emergent
8	Assign a unique ID to each person with computer access	Operations	CP, CMVM	Emergent
9	Restrict physical access to cardholder data	Operations	CP, T	Emergent
10	Track and monitor all access to network resources and cardholder data	Operations	SM, SE	Emergent
11	Regularly test security systems and processes	Operations	ST, PT	Emergent
12	Maintain a policy that addresses information security	Operations	CP	Emergent

Table 3: Summary of the PCI-DSS Controls [64], mapped to SDLC and fault target.

change is hindered. As cross-disciplinary entities such as the newly-formed IEEE CSD[18] gain traction, the promise for establishing standards and meta-models that incorporate the best of practice and theory is raised and the opportunity to fundamentally change security thinking in such ways becomes tangible.

7. DIRECTIONS FOR THE ECONOMICS OF SWSEC

We have presented three high-level topics central to establishing a practice of software security, which come together in SWSec models (using the Touchpoints processes as an example) to augment and enhance the security decision-making process. This vision can be decomposed into key areas of research, each with challenges to be overcome.

While economics has been applied to various aspects of the security problem, its application to SWSec has been limited. Realising economically-informed secure software practices will require further study into three broad areas.

- *Models*: SWSec requires the development of models that capture the essence of software process contribution. Due to the variability and dependency in SWSec processes, the effects of software process on the prevalence of vulnerabilities are more characterisable as effects on the model conditions (uncertainty, quality, etc.), rather than as estimates of exogenous properties or indirect measures. In addition to increasing the accuracy of estimates that rest on the existing software engineering body of knowledge, this will also require the integration of investment considerations with current enterprise models and risk practices within information security economics. Necessary to the validity of such models are further quantified studies into tools and processes such as [30, 9, 73], in order to properly tune and validate model assumptions.

[18]The IEEE Center for Secure Design (CSD), http://cybersecurity.ieee.org/center-for-secure-design/

- *Representation*: Once the targets for SWSec have been characterised, these must still be rectified against the overall goals of the project and the enterprise. While the conveyance of economic considerations within current processes is well supported (if not well utilised), the methods and models for juxtaposing security and functionality in order to inform investment are underdeveloped. As agile methods fail to address these concerns and formal methods remain complex and expensive, utility-based characterisations may provide an alternative that integrates functional and security constructs through economics in order to allow reasoning and enable decision-making within a variety of approaches.

- *Analysis*: Critical to rationalising SWSec investment is the ability to identify the contribution. Starting with the execution of the effort, SWSec investment must be examined in light of the scope of the overall context: business, regulatory, technical and process. This requires not only a common representation for the various aspects of security (i.e. coding vs. emergent fault reduction), but also the means to compare contributions against competing objectives. Multi-attribute and risk-based methods provide such constructs, but require the careful selection of inputs. SWSec analysis requires methods that can incorporate the varied indicators more effectively than current risk-based methods, yet retain compatibility with business practice.

Research in these three areas forms a solid basis for the establishment of an economics of secure software development, rooting the practice within the software engineering and information security fields while incorporating key information security economics concepts in the realisation of security in practice. However, advancement toward this goal must address the data, usability and abstraction challenges that have generally plagued information security investment model adoption. Representing SWSec concerns risks exacerbating issues with model complexity and rendering model

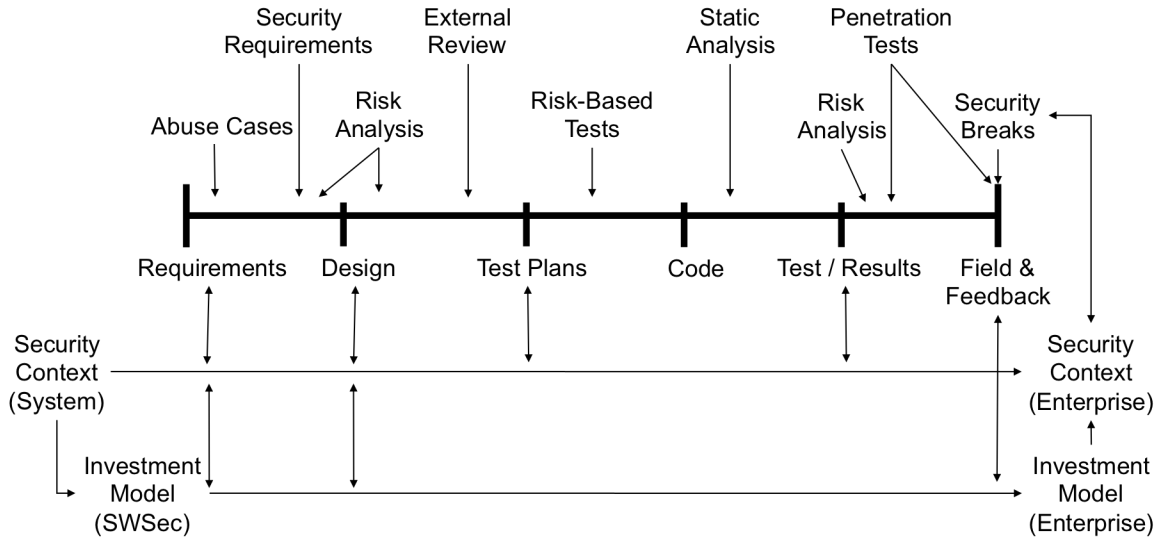

Figure 2: The Touchpoints process overlaid with the SWSec economic constructs described in this paper.

selection and usage decisions more difficult [70]. Just as a lack of security expertise amongst developers hinders software security [81], a lack of software engineering expertise will hinder the use of such models.

Out of these goals and obstacles come concrete research directions to pursue, with their own requisite challenges:

- *Development of new models, or extension of current models, to accurately represent the contribution of secure software engineering* — which is essential to the rational allocation of resources by system developers. Section 4 examined issues with current investment models, which fail to account for incremental investment, can often only be applied to completed systems, and are static views of the system security state. Models of SWSec investment must realistically represent the contribution of process (e.g. the addition of code reviews), resources (e.g. the number of code reviews, number of reviewers per review and time dedicated to reviews), technology (e.g. manual and automated tools used in the code review) and characterise their contribution in terms of effectiveness (e.g. the expertise of the reviewers, number and type of implementation errors found) and cost. As each of these aspects is dependant on the others, this may result in complex model relationships. These may take the form of novel investment models in their own right, or instead be developed as functions and models that complement existing enterprise investment models. While the latter may prove a more effective means to integration with broader system life-cycle security investment, the 'correct' means to model such investment in current models may be unclear. The first such attempt at such a model has focused on SWSec's potential to increase costs for the attacker, while reducing defender uncertainty regarding where to invest [34]; other formulations focused on increased resistance to vulnerability, reduced attack surface, or other inherent system properties are possible.

- *Further work in security-specific software engineering practice.* Section 4 also identified the problem of a dearth of software security studies. While software engineering has long embraced empirical evaluation and qualitative research, this work has been notoriously difficult to reproduce and generalise. Efforts to specialise such work are no different, and while the contributions cited in Section 4.1 represent a good start, much more needs to be done. Efforts that investigate and quantify the effects of SWSec practices throughout a number of enterprises and a variety of projects are key to developing employable models and metrics, mindful of the challenges posed by control [27], repeatability and validity [49]. Such research must additionally address the aforementioned issues of process resource, effectiveness and cost demands, and must result in published, accessible datasets. This inherently requires a change in the security research mindset, away from 'absolute security' and toward a concept of security based in measurable experimentation [82]. Inspiration — and a basis for further work — can be found in years of qualitative research within the software engineering domain (e.g. [13]).

- *A means to convey, in a computable but manipulatable way, the security context.* Section 5 highlighted the need to integrate SWSec throughout the system life-cycle, with Section 5.2 identifying potential constructs that enable SWSec integration with the early stages of system development. The form of such a construct must allow for reasoning over a diverse set of inputs: standards, policy, functional requirements, managerial constraints, and risk-based probabilities. In addition, re-evaluation and validation of the security expectations throughout the system life-cycle must be supported. While utility theory offers a potential language [35], further validation of this approach on larger and diverse developments is necessary. Application of these techniques must then be balanced with

applied engineering experience, and enabled through the development of analytical and specification tools and techniques in order to promote adoption.

- *Progress toward 'composable' decision-making mechanisms*, reflecting the means by which security investments, deployment and management are measured and analysed. As discussed in Section 6.2, given the phased nature of software development and the knock-on effects of decisions made during development methods for decision support that weight and manage varied life-cycle phase inputs are necessary. This runs counter to much of the information security investment literature, which assumes complete, instantaneous investment or single-run games. However, many such models have shown promise in their ability to optimise post-deployment security investment; as discussed in Section 6.1, such investments are crucial to standards compliance and accreditation techniques that are core to current notions of computer security. Models that explicitly incorporate temporal and dependent aspects from multiple phases of the SDLC offer the promise of coupling independent investment decisions to enhance overall return on investment, as demonstrated in [34]. Further work that analyses various considerations while integrating pre- and post-deployment dependencies will be central to developing a holistic approach to a firm's overall security investment strategy.

- *Recognition that theory is necessary* to the development of scientifically-rooted approaches, while security is ultimately a practical endeavour. While software engineering has long been a practice-oriented discipline, information security economics suffers from a dichotomy of theory-based concepts punctuated with devoted empirical studies. Adoption and use of SWSec economics will require a balance of theory and application that will only come with the cross-fertilisation of data and ideas, necessitating more discussion between these disciplines. A first step in this direction will be the coordination of the best SWSec practices — such as the BSIMM – with further theory to move practice beyond what can be conceived of today.

As SWSec moves from purely empirical considerations toward the incorporation of theoretical underpinnings, these research areas support a deeper understanding of how software process impacts overall security goals. Applying these tenets in practical approaches will be essential to their adoption by practitioners, and ultimately drive their adoption and impact. Incorporation of software engineering, security practice and enterprise operations provides a basis for a true science of software security — and increasingly the necessary conditions for practical security — with economics poised to supply the enabling insights.

Acknowledgements

The authors would like to thank Christian Probst and the reviewers for their insightful comments.

8. REFERENCES

[1] Framework for improving critical infrastructure cybersecurity. Tech. rep., National Institute of Standards and Technology (NIST), February 2014.

[2] Al-Humaigani, M., and Dunn, D. A model of return on investment for information systems security. In *IEEE International Symposium on Micro-NanoMechatronics and Human Science* (2003), vol. 1, pp. 483–485.

[3] Anderson, R., and Moore, T. The economics of information security. *Science 314*, 5799 (October 2006), 610–613.

[4] Andrijcic, E., and Horowitz, B. A macro-economic framework for evaluation of cyber security risks related to protection of intellectual property. *Risk Analysis 26*, 4 (2006), 907–923.

[5] Apvrille, A., and Pourzandi, M. Secure software development by example. *IEEE Security and Privacy 3*, 4 (July 2005), 10–17.

[6] Arce, I., Clark-Fisher, K., Daswani, N., DelGrosso, J., Dhillon, D., Kern, C., Kohno, T., Landwehr, C., McGraw, G., Schoenfield, B., Seltzer, M., Spinellis, D., Tarandach, I., and West, J. Avoiding the top 10 software security design flaws. PDF, 2014. http://cybersecurity.ieee.org/images/files/images/pdf/CybersecurityInitiative-online.pdf.

[7] Aslam, T., Krsul, I., and Spafford, E. Use of a taxonomy of security faults. Tech. Rep. 96-051, Purdue University, 1996.

[8] Austin, A., and Williams, L. One technique is not enough: A comparison of vulnerability discovery techniques. In *2011 International Symposium on Empirical Software Engineering and Measurement (ESEM)* (September 2011), pp. 97–106.

[9] Baca, D., Carlsson, B., and Lundberg, L. Evaluating the cost reduction of static code analysis for software security. In *Proceedings of the Third ACM SIGPLAN Workshop on Programming Languages and Analysis for Security (PLAS '08)* (2008), ACM, pp. 79–88.

[10] Baca, D., Petersen, K., Carlsson, B., and Lundberg, L. Static code analysis to detect software security vulnerabilities - does experience matter? In *International Conference on Availability, Reliability, and Security (ARES 2009)* (2009), IEEE Computer Society, pp. 804–810.

[11] Beresnevichiene, Y., Pym, D., and Shiu, S. Decision support for systems security investment. In *IEEE/IFIP Network Operations and Management Symposium (NOMS) Workshops* (April 2010), pp. 118–125.

[12] Boehm, B., and Basili, V. R. Software defect reduction top 10 list. *IEEE Computer 34*, 1 (January 2001), 135–137.

[13] Boehm, B. W. *Software Engineering Economics*. Prentice-Hall Advances in Computing Science and Technology Series. Prentice Hall, Englewood Cliffs, N.J, 1981.

[14] Böhme, R., and Moore, T. 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).

[15] Böhme, R., and Nowey, T. *Dependability Metrics: Advanced Lectures*. Springer Berlin Heidelberg, 2008,

ch. Economic Security Metrics, pp. 176–187.

[16] Butler, S. A. Security attribute evaluation method: a cost-benefit approach. In *2002 International Conference on Software Engineering (ICSE 2002)* (2002), W. Tracz, M. Young, and J. Magee, Eds., ACM, pp. 232–240.

[17] Camp, L. J., and Wolfram, C. Pricing security. In *Proceedings of the CERT Information Survivability Workshop* (2000), pp. 31–39.

[18] Cavusoglu, H., Mishra, B., and Raghunathan, S. A model for evaluating it security investments. *Communications of the ACM 47*, 7 (July 2004), 87–92.

[19] Cremonini, M., and Martini, P. Evaluating information security investments from attackers perspective: the return-on-attack (ROA). In *Proceedings of the 4th Annual Workshop on the Economics of Information Security (WEIS 2005)* (June 2005).

[20] De Win, B., Scandariato, R., Buyens, K., Gregoire, J., and Joosen, W. On the secure software development process: CLASP, SDL and Touchpoints compared. *Elsevier Information and Software Technology*, 51 (2009), 1152–1171.

[21] Demetz, L., and Bachlechner, D. *The Economics of Information Security and Privacy*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, ch. To Invest or Not to Invest? Assessing the Economic Viability of a Policy and Security Configuration Management Tool, pp. 25–47.

[22] Department for Business, Innovation and Skills. Cyber essentials scheme: overview. https://www.gov.uk/government/publications/cyber-essentials-scheme-overview, June 2014.

[23] Devanbu, P. T., and Stubblebine, S. Software engineering for security: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering (ICSE 2000)* (2000), IEEE / ACM, pp. 227–239.

[24] Division, N. C. S. Information security: System development life cycle. PDF, August 2004. http://csrc.nist.gov/groups/SMA/sdlc/documents/SDLC_brochure_Aug04.pdf.

[25] Edmundson, A., Holtkamp, B., Rivera, E., Finifter, M., Mettler, A., and Wagner, D. An empirical study on the effectiveness of security code review. In *5th International Symposium on Engineering Secure Software and Systems (ESSoS 2013)* (2013), Springer Berlin Heidelberg, pp. 197–212.

[26] Faily, S. *A framework for usable and secure system design*. PhD thesis, University of Oxford, Oxford, UK, 2011.

[27] Geer, D. For good measure: The undiscovered. *;login: 40*, 2 (April 2015), 50–52.

[28] Gilmore, S., Sondhi, R., and Simpson, S. Principles for software assurance assessment: A framework for examining the secure development processes of commercial technology providers. Tech. rep., Software Assurance Forum for Excellence in Code (SAFECode), 2015.

[29] Gordon, L. A., and Loeb, M. P. The economics of information security investment. *ACM Transactions on Information and Systems Security 5*, 4 (November 2002), 438–457.

[30] Goseva-Popstojanova, K., and Perhinschi, A. On the capability of static code analysis to detect security vulnerabilities. *Information and Software Technology 68*, C (December 2015), 18–33.

[31] Grimes, R. A. Implementing a data-driven computer security defense. Tech. rep., Microsoft IT Information Security and Risk Management, November 2015.

[32] Grossklags, J., Christin, N., and Chuang, J. Secure or insure?: A game-theoretic analysis of information security games. In *17th International Conference on World Wide Web (WWW '08)* (2008), ACM, pp. 209–218.

[33] Hein, D., and Saiedian, H. Secure Software Engineering: Learning from the Past to Address Future Challenges. *Information Security Journal: A Global Perspective 18*, 1 (2009), 8–25.

[34] Heitzenrater, C., Böhme, R., and Simpson, A. The days before zero day: Investment models for secure software engineering. In *15th Workshop on the Economics of Information Security (WEIS 2016)* (June 2016).

[35] Heitzenrater, C., King-Lacroix, J., and Simpson, A. Motivating security engineering with economics: A utility function approach. In *To be published at the 1st IEEE Workshop on Cyber Resilience Economics (CRE 2016)* (August 2016).

[36] Heitzenrater, C., and Simpson, A. Policy, statistics, and questions: Reflections on uk cyber security disclosures. In *Proceedings of the 14th Annual Workshop on the Economics of Information Security (WEIS 2015)* (2015).

[37] Heitzenrater, C., and Simpson, A. C. Misuse, abuse, and reuse: Economic utility functions for characterising security requirements. In *to appear in the Proceedings of The Second International Workshop on Agile Secure Software Development (ASSD 2016)* (August 2016).

[38] Heitzenrater, C., Taylor, G., and Simpson, A. When the winning move is not to play: Games of deterrence in cyber security. In *2015 Conference on Decision and Game Theory for Security (GameSec 2015)* (2015).

[39] Hoglund, G., and McGraw, G. *Exploiting Software: How to Break Code*. Pearson Higher Education, 2004.

[40] Hoo, K. J. S. *How much is enough? A risk management approach to computer security*. PhD thesis, Stanford University, Stanford, CA, 2000.

[41] Hoo, K. S. Information security: why the future belongs to the quants. *IEEE Security & Privacy 1*, 4 (July 2003), 24–32.

[42] Jaatun, M. G. Hunting for aardvarks: Can software security be measured? In *IFIP International Cross-Domain Conference and Workshop (CD-ARES)* (2012), G. Quirchmayr, J. Basl, I. You, L. Xu, and E. Weippl, Eds., vol. 7465 of *Lecture Notes in Computer Science*, Springer, pp. 85–92.

[43] James, J. A. Raising the bar for cybersecurity. Tech. rep., Center for Strategic & International Studies, Technology & Public Policy, Washington, D.C., USA, February 2013.

[44] Kárpáti, P., Opdahl, A. L., and Sindre, G.

Investigating security threats in architectural context: Experimental evaluations of misuse case maps. *Journal of Systems and Software 104* (2015), 90–111.

[45] Kemerer, C. F., and Paulk, M. C. The impact of design and code reviews on software quality: An empirical study based on psp data. *IEEE Transactions on Software Engineering 35*, 4 (July 2009), 534–550.

[46] Kordy, B., Pietre-Cambacedes, L., and Schweitzer, P. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *CoRR* (2013).

[47] Lipner, S. The trustworthy computing security development lifecycle. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC 2004)* (2004), IEEE Computer Society, pp. 2–13.

[48] Manico, J., Bird, J., Harris, D., De Vries, S., and Van der Stock, A. Owasp proactive controls for developers 2014. PDF, 2014. https://www.owasp.org/images/0/07/OWASP_Proactive_Controls_v1.pdf.

[49] Massacci, F., and Nguyen, V. H. An empirical methodology to evaluate vulnerability discovery models. *IEEE Transactions on Software Engineering 40*, 12 (December 2014), 1147–1162.

[50] McDermott, J., and Fox, C. Using abuse case models for security requirements analysis. In *Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC '99)* (1999), IEEE Computer Society, pp. 55–64.

[51] McGraw, G. Software security. *IEEE Security & Privacy 2*, 2 (March 2004), 80–83.

[52] McGraw, G. The role of architectural risk analysis in software security, March 2006. Pearson InformIT article; Accessed: 2016 04 07.

[53] McGraw, G. *Software Security: Building Security In*, 1st edition ed. Addison-Wesley Professional, 2006.

[54] McGraw, G. Cyber war is inevitable (unless we build security in). *Journal of Strategic Studies 36*, 1 (2013), 109 – 119.

[55] McGraw, G., July 2017. Personal communication with the author.

[56] McGraw, G., Migues, S., and West, J. Building security in maturity model (BSIMM). PDF, 2015. https://www.bsimm.com/.

[57] Mercuri, R. T. Analyzing security costs. *Communications of the ACM 46*, 6 (June 2003), 15–18.

[58] Mizzi, A. Return on information security investment - the viability of an anti-spam solution in a wireless environment. *International Journal of Network Security 10*, 1 (January 2010), 18–24.

[59] Mkpong-Ruffin, I., Umphress, ., Hamilton, J., and Gilbert, J. Quantitative software security risk assessment model. In *2007 ACM Workshop on Quality of Protection (QoP 2007)* (2007), ACM, pp. 31–33.

[60] Moore, T. The economics of cybersecurity: Principles and policy options. *IJCIP 3*, 3-4 (2010), 103–117.

[61] Neuhaus, S., and Plattner, B. Software security economics: Theory, in practice. In *11th Annual Workshop on the Economics of Information Security (WEIS 2012)* (2012).

[62] Panaousis, E., Fielder, A., Malacaria, P., Hankin, C., and Smeraldi, F. Cybersecurity games and investments: A decision support approach. In *5th International Conference on Decision and Game Theory for Security (GameSec 2014)* (2014), R. Poovendran and W. Saad, Eds., Springer International Publishing, pp. 266–286.

[63] Pandey, P. "context, content, process" approach to align information security investments with overall organizational strategy. *International Journal of Security, Privacy and Trust Management (IJSPTM) 4*, 3/4 (November 2015), 25–38.

[64] Payment Card Industry (PCI) Data Security Standard: Navigating PCI DSS. Understanding the intent of the requirements. PDF, October 2008. https://www.pcisecuritystandards.org/pdfs/pci_dss_saq_navigating_dss.pdf.

[65] Peeters, J., and Dyson, P. Cost-effective security. *IEEE Security & Privacy Magazine 5*, 3 (2007), 85–87.

[66] Peisert, S., Talbot, E., and Bishop, M. Turtles all the way down: A clean-slate, ground-up, first-principles approach to secure systems. In *Proceedings of the 2012 New Security Paradigms Workshop (NSPW '12)* (2012), ACM, pp. 15–26.

[67] Pfleeger, S. L., and Rue, R. Cybersecurity economic issues: Clearing the path to good practice. *IEEE Software 25*, 1 (January 2008), 35–42.

[68] Poladian, V., Garlan, D., and Shaw, M. Software selection and configuration in mobile environments: A utility-based approach. In *4th Workshop on Economics-Driven Software Engineering Research (EDSER-4), affiliated with the 24th International Conference on Software Engineering (ICSE'02),* (May 2002).

[69] Robertson, S., and Robertson, J. *Mastering the Requirements Process (1st Edition)*. Addison-Wesley Professional, 1999.

[70] Rue, R., and Pfleeger, S. L. Making the best use of cybersecurity economic models. *IEEE Security & Privacy 7*, 4 (July 2009), 52–60.

[71] Rue, R., Pfleeger, S. L., and Ortiz, D. A framework for classifying and comparing models of cyber security investment to support policy and decision-making. In *Proceedings of the 6th Annual Workshop on the Economics of Information Security (WEIS 2007)* (2007).

[72] Salini, P., and Kanmani, S. Survey and analysis on security requirements engineering. *Computers & Electrical Engineering 38*, 6 (2012), 1785–1797.

[73] Scandariato, R., Walden, J., and Joosen, W. Static analysis versus penetration testing: A controlled experiment. In *24th IEEE International Symposium on Software Reliability Engineering (ISSRE 2013)* (2013), IEEE Computer Society, pp. 451–460.

[74] Schechter, S. Quantitatively differentiating system security. In *Proceedings of the 1st Workshop on the Economics of Information Security (WEIS 2002)* (2002).

[75] Schechter, S. E., and Smith, M. D. How much security is enough to stop a thief? the economics of outsider theft via computer systems and networks. In *Financial Cryptography* (2003), Springer-Verlag,

pp. 122–137.

[76] Schneier, B. *Secrets & Lies: Digital Security in a Networked World*, 1st ed. John Wiley & Sons, Inc., New York, NY, USA, 2000.

[77] Sindre, G., and Opdahl, A. L. Capturing security requirements through misuse cases. In *Proceedings of the 14th 14th Norwegian Informatics Conference (NIK 2001)* (2001).

[78] Sindre, G., and Opdahl, A. L. 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.

[79] Stecklein, J. M., Dabney, J., Dick, B., Haskins, B., Lovell, R., and Moroney, G. Error cost escalation through the project life cycle. In *14th Annual International Symposium of INCOSE* (June 2004).

[80] Thomas, R. C., Antkiewicz, M., Florer, P., Widup, S., and Woodyard, M. How bad is it? a branching activity model to estimate the impact of information security breaches. In *Proceedings of the 12th Annual Workshop on the Economics of Information Security (WEIS 2013)* (2013).

[81] Tøndel, I. A., Jaatun, M. G., and Meland, P. H. Security requirements for the rest of us: A survey. *IEEE Software 25*, 1 (January 2008), 20–27.

[82] Tsiakis, T., and Stephanides, G. The economic approach of information security. *Computers & Security 24*, 2 (2005), 105–108.

[83] Verdon, D., and McGraw, G. Risk analysis in software design. *IEEE Security & Privacy 2*, 4 (July 2004), 79–84.

[84] Verendel, V. Quantified security is a weak hypothesis: A critical survey of results and assumptions. In *Proceedings of the 2009 Workshop on New Security Paradigms Workshop (NSPW '09)* (2009), ACM, pp. 37–50.

[85] Voas, J., McGraw, G., Kassab, L., and Voas, L. A 'crystal ball' for software liability. *Computer 30*, 6 (1997), 29–36.

[86] Woody, C., and Alberts, C. Considering operational security risk during system development. *IEEE Security & Privacy 5*, 1 (2007), 30–35.