Software Security Investment Modelling for Decision-Support

Chad D. Heitzenrater

Department of Computer Science
Wolfson Building
Parks Road
Oxford OX1 3QD

Supervisor: Prof. A. C. Simpson

Michaelmas Term 2017

The views expressed are those of the author and do not necessarily reflect the official policy or position of the Air Force, the Department of Defense, or the U.S. Government.

Abstract

While it is widely agreed that contemporary computer security is insufficient to meet the challenges faced, the remedies for its failures are far less obvious. Vast resources have been placed into technical solutions to little effect, prompting some to employ the constructs of economics to frame this problem as one to be 'managed', rather than 'solved'. However, to date economically-inspired decision support approaches have focused disproportionately on post-deployment security investment. With the preponderance of security issues stemming from the introduction of vulnerabilities during design and development, models that span the system development lifecycle are essential to efficiently address the root of many security issues. In addition, the need to impact system security at a fundamental level requires integration with existing security-development processes and standards.

This dissertation presents an approach to secure software development that is derived from an economically-inspired understanding of security. After demonstrating how existing security guidance can give rise to inefficient decisions, models for security investment are developed that incorporate investments made in software security during system inception and development relative to those made during deployment and operations. By employing these models, conditions are identified whereby software security improves the return on (security) investment, and provide theoretical and empirical evidence to support the adoption of software security. This is followed by an exploration of how economic considerations can drive existing secure software engineering processes, culminating in a case study that illustrates the application of these principles to an ongoing system development effort.

Acknowledgements

My heartfelt thanks to everyone who made this dissertation possible:

To Andy for the patience, guidance and wisdom that has led me here;

To my colleagues at Oxford, especially Martin, Katriel, Kevin, Cas, Yang, Emma, Dennis and Daniel, for our exchanges of ideas, insight, culture, and pints;

To my friends Bill, Mark, Tim, Sherry, Jason, Lisa, James, Lauren, Kim, Damien, Karen and Will who did their best to remind me of life outside of this research;

To my coworkers, who enabled me, stuck up for me, and kept me out of trouble, especially Ryan, Tim, Amy, Scott and Warren;

To my OUFC teammates, who allowed me to revisit my youth, if for a bit;

To 'my chair' at the Gardener's Arms, which supported me through so much *reading.*

Most of all, my love and gratitude to Mom and Dad, Julie and Carlo, and my entire family, to whom I owe so much.

To my beautiful wife and amazing children, I dedicate this dissertation:

Beth, this never would have been possible without you. I could not have asked for a better partner on this journey.

Marek, Colin and Owen, if I've accomplished nothing else I hope I have inspired you to continue to challenge yourselves. Never stop.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We will bankrupt ourselves in the vain search for absolute security.

— Dwight D. Eisenhower

1.1 Motivation

Computer security continues to be a vexing problem, undermining our financial stability, diminishing our privacy, and threatening our national security. As cybersecurity failures mount[1], there is a growing realisation that many issues faced in computer security today are rooted in our approach to developing software and systems: computer security has become synonymous with security products. Consumers install anti-virus solutions, struggle to keep applications patched and up to date, and grapple with the configuration of firewalls and routers. Corporations expand on this investment with a range of vendor products and services, often creating a heterogeneous landscape of partial solutions. Security investment has become a reactionary practice — optimising the expenditure of resources to fix problems that are largely preventable, leading some to characterise the current environment as the "dark ages" of security thinking [6].

Security is fundamentally a system design problem [7]. However, building systems 'properly' is expensive, difficult and fraught with pitfalls that are diverse and multi-faceted, requiring technical, legal, and managerial insight. This complexity is perhaps exemplified in the intersection of two emerging fields in computer security: secure software development (otherwise known as 'SwSec' or secure software engineering), which

[1] As evidenced by sites such as `www.breachlevelindex.com`.

seeks to leverage software engineering and risk management practice to raise the quality of security decision making in all phases of software and system development [8]; and information security economics, which seeks to identify the externalities and (potentially misaligned) incentives that drive how those decisions are made [9]. Despite evidence of vulnerability reduction (e.g. [10]), SwSec practices continue to see devitalised adoption[2] — even as cybersecurity expenditure continues to increase[3]. This is no wonder, as the additional processes and tools required can be costly, requiring expertise, resources, and time often unavailable to the development organisation (if available at all) [14]. At the heart of secure software engineering are processes that rely on expert judgement, require specialised expertise, and exhibit variability in their effectiveness [15]. Even when security is 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 introduce potential hazards. Decisions are made at every step that impact the security balance of the programme, and current practices are not well suited to make these trade-offs — or even articulate when they have been made. Achieving higher returns on security investment by building security into the software engineering process would seem intuitive; however, there is little definitive proof to support such an assertion [16]. For all of the guidance on what to do in the name of software security, insight is lacking on how to apply these precepts in practice.

That is not to say that system-level, preventative approaches are not a necessary component to security. Employing the classification of Aslam *et al.* in [17] (and later adopted for economic considerations by Camp and Wolfram in [18]), SwSec largely focuses on only a segment of vulnerability sources: *Coding faults* introduced during development as a result of errors in programming logic, missing or incorrect requirements, or design errors. In practice, disparate systems must come together to form the enterprise, resulting in *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. As a result, traditional approaches to compliance and enterprise defence are unlikely to be devalued anytime soon (nor should they be, necessarily). Rather, an argument can be made that exclusive focus on enterprise security investment is an incomplete approach, and one that fails to reflect the sources of vulnerabilities [19], the inherent

[2]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`.

[3]As cited in surveys by the UK Government [11], Gartner [12], SANS Institute [13], and others.

fallibility of vulnerability discovery [20], and the escalation of costs in the development lifecycle [21; 22]. Given that 'perfect' security (i.e. defence against all possible threats) is known to be cost-prohibitive — if not impossible — developers must carefully weigh pre- and post-deployment security investments, in relation to functional and non-functional requirements, to construct systems that balance these often competing demands [23]. Unfortunately, incorporating SwSec into security planning places a further strain on limited budgets in the face of a seemingly unmanageable task, and models for information security investment do not yet support such a comprehensive treatment.

1.2 Research Problem

This dissertation addresses the lack of decision-support in the area of secure software development, employing an economic approach to motivate, define, and apply the problem of secure software investment. These three goals frame the contribution of this research as follows:

1. *Motivating* software security investment through econometric analysis, resulting in examples of how the current enterprise-focused, compliance-based approach to system security can — if not constructed carefully — result in fiscally irrational and technically inefficient behaviour. This is demonstrated by applying data from a recent information security breaches survey to a current cybersecurity certification scheme.

2. *Defining* a practice that corrects such failures by explicitly considering information security investment throughout the system development lifecycle. This is accomplished through the development and application of economic models that capture pre- and post-deployment security contributions in order to maximise the return on investment over the course of system development and operations.

3. *Applying* economic principles to support 'security contexts' that motivate decision-making relative to the functional and non-functional constraints of the system. This approach employs economic principles in order to allocate resources to security throughout the system lifecycle in a rational and repeatable manner.

These three goals support the overall research problem: *what does economics tell us about the application (or lack thereof) of secure software principles?*

1.3 Publications

In the investigation of this research problem, the following publications were produced (in the order of appearance in this dissertation):

1. "*Policy, statistics, and questions: Reflections on UK cyber security disclosures*" (with A. C. Simpson), published in the Proceedings of the 14th Workshop on the Economics of Information Security (WEIS), June 2015 [24]. An updated version of this paper appeared in the Journal of Cybersecurity, volume 2, issue 1, December 2016 [25].

2. "*Modelling Software Security Investment*" (with A. C. Simpson), under preparation [26].

3. "*The days before zero day: Investment models for secure software engineering*" (with R. Böhme and A. C. Simpson), published in the Proceedings of the 15th Workshop on the Economics of Information Security (WEIS), June 2016 [27]. This work was summarised in "*Software security investment: The right amount of a good thing*", published in the Proceedings of the 1st IEEE Cybersecurity Development Conference (SecDev), November 2016 [28], along with a discussion of the challenges of modelling secure software development.

4. "*When the winning move is not to play: Games of deterrence in cyber security*" (with G. Taylor and A. C. Simpson), published in the Proceedings of the 6th International Conference on Decision and Game Theory for Security (GameSec), November 2015 [29].

5. "*Motivating security engineering with economics: A utility function approach* (with J. King-Lacroix and A. C. Simpson), published in the Proceedings of the 1st IEEE International Workshop on Cyber Resilience Economics (CRE), August 2016 [30].

6. "*Misuse, abuse, and reuse: Economic utility functions for characterising security requirements*" (with A. C. Simpson), published in the Proceedings of the 2nd International Workshop on Agile Secure Software Development (ASSD), August 2016 [31].

In addition to this research, a vision for an over-arching economics of secure software was presented at the 2016 New Security Paradigms Workshop (NSPW): "*A case for the economics of secure software development*" (with A. C. Simpson) [32], September 2016.

This publication outlines a number of research directions, including the contributions above. Elements of this paper pervade this dissertation.

1.4 Dissertation Structure

Following a discussion of the principles and precepts of software security and information security economics in Chapter 2, Chapter 3 proceeds with an analysis of an existing security policy that demonstrates the need for an alternative approach to security planning. This is followed in Chapter 4 with the presentation of modelling approaches that define software security investment in the context of the system development lifecycle. In Chapter 5 the utilisation of economic analysis as part of a secure software development process is explored, with an illustrative example provided in Chapter 6. This dissertation concludes with consideration of future directions and applications in Chapter 7.

Chapter 2

Background

Information security is not a technological problem. It is an economics problem.
— Bruce Schneier

An investigation of the role economics plays in secure software development necessitates an understanding of both software security and information security economics in the context of system security. Due to the broad areas of study encompassed by these terms, a comprehensive literature review would be unhelpful in framing this contribution; instead, focus is placed on the key concepts that capture the salient aspects of these disciplines as they relate to one another. This is accomplished by examining their respective definitions, which are employed as guideposts to the pertinent research and challenges that establish their intersection.

2.1 Software Security

Software security (otherwise known as 'SwSec') has been defined as the application of "processes, principles, and methods to build vulnerability-free software, software that remains in a secure state under attack and continues to provide service to authorised users" [33]. While the practice of software engineering is generally concerned with the development of quality software (with a focus on defect reduction and reliability [21]), the burgeoning software security community seeks to augment specific existing process steps with security-oriented (or security-enhanced) activities. This is distinct from *application security*: 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 [8].

As aspects of the larger goal of system security engineering (SSE), both apply scientific and engineering principles to "specify, predict, and evaluate the vulnerability of the system to security threats" [34]. These goals intersect with software management on the premise that system security fundamentally rests on the elimination of defects that make an application vulnerable to attack [35]. Where software management seeks quality as an inherent attribute, software security recognises the emergent nature of security and elevates security over features and functionality [36].

Following from the definition of Avizienis *et al.* [33], software security can be examined from the perspective of the *processes*, *principles*, and *methods* that comprise its practice. To facilitate that discussion, each of these viewpoints is presented in the context of the NIST Systems Development Lifecycle (SDLC) phases: Initiation, Acquisition/Development, Implementation, Operations/Maintenance and Disposal, as defined by NIST SP 800–64 [37]. Although this taxonomy details *systems* rather than individual *software*, it is within an SDLC that an organisation defines its software development process. Such process models may span SDLC phases, governing the inclusion, frequency, timing and scope of development activities (requirements, architecture, design, coding, testing, and release). Common variants of software processes include the waterfall, evolutionary, iterative/incremental, spiral, prototyping, and agile models; for a primer, we refer the reader to [3]. Application of a process model requires taking into account various organisational and project constraints, which range from the consideration of existing systems, scope of the project (standalone versus part of a product line family), and the constitution of the project team.

2.1.1 Processes

SwSec processes seek to provide guidance to practitioners in order to close the gap between security and development [38], identifying specific steps to be taken in order to develop secure software. While the history of SwSec practice can be viewed as a maturation in the ways in which we plan, develop, and deploy software securely, the concepts are rooted in longstanding efforts such as the Common Criteria [39]. Other approaches have targeted specific sectors or issues, such as the Payment Card Industry Data Security Standard (PCI-DSS) for handling credit card records [40]. While these approaches may have impact on software security, they primarily focus on checklist-oriented evaluation during the SDLC Implementation phase. This has led to criticism for often leading only to very simple solutions [41] and dismissal of this approach as "static", and "expensive

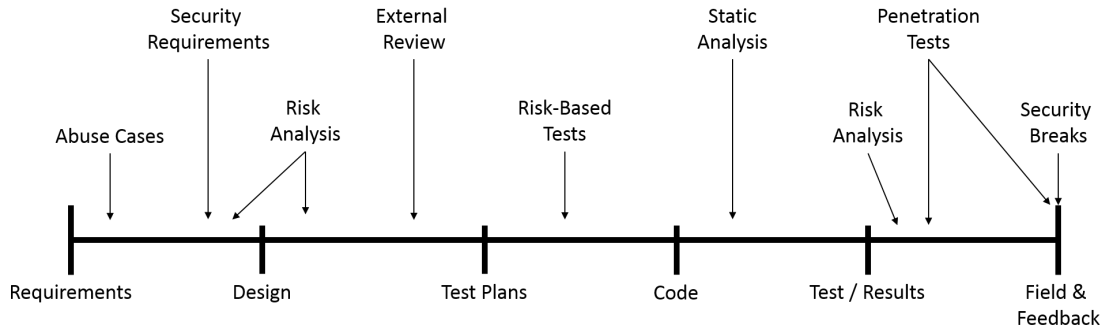


Figure 2.1: The Secure Software Development Lifecycle (SSDL) Touchpoints, adapted from [1]. Permission to reproduce this figure has been granted by Gary McGraw.

without providing equivalent benefits" [42].

These deficiencies have resulted in efforts by collectives such as the Open Web Application Security Project (OWASP)[1], companies such as Microsoft [43] and Adobe [44], and government agencies such as the Department of Homeland Security (DHS) United States Computer Emergency Readiness Team (US-CERT)[2]. Employing the mantra of 'build security in', such processes have been referred to as Secure Software Development Lifecycles (SSDLCs) [45] despite offering limited guidance in later SDLC phases.

Closely related are collections of practice intended to augment existing processes. One of the most recognisable examples of this is the Secure Software Development Lifecycle (SSDL) Touchpoints, often referred to simply as 'the touchpoints' [8; 38; 1]. While the touchpoints are not constructed as a step-wise process, the activities are often visualised within the traditional software development timeline, as in Figure 2.1. Touchpoint practices are often presented according to their effectiveness (as determined by the developer's experience[3]), identified by the 'Touchpoint Rank' column of Table 2.1. Although collectively the touchpoints are widely accepted, individually they are also well-known, if not standard, software engineering practice. Each touchpoint emphasises the security aspects of a particular practice, where code review, architectural analysis, testing and requirements methodologies have long been practised under the guise of software quality and reliability. A comparison of the touchpoints and similar security-oriented processes can

[1]OWASP's project CLASP (Comprehensive, Lightweight Application Security Process) was removed from the OWASP Wiki in June of 2016, due to "very old and often wrong information". See `https://www.owasp.org/index.php?title=Special:Log&page=Category:OWASP_CLASP_Project`.

[2]While no longer updated, US-CERT's website remains an archive for software assurance best practice, knowledge, and tool information. See `https://buildsecurityin.us-cert.gov/`.

[3]Personal communication to the author (e-mail), 14 July 2016.

be found in [46].

In addition to development-focused techniques, processes for assessing the security of overall systems at different points within the SDLC Implementation phase have been developed. Generally risk-focused, these approaches typically require at least a detailed architectural diagram — and at most a functioning system — as a necessary input. Examples of such techniques include the Security Attribute Evaluation Method (SAEM) [47], the Appropriate and Effective Guidance for Information Security (AEGIS) [48], and the domain-specific Quantitative Evaluation of Risk for Investment Efficient Strategies (QuERIES) process [49; 50] for intellectual property protection. While SAEM is notable for introducing a cost-benefit aspect to security considerations, each of these approaches focuses on assessments of risk as the underlying decision mechanism. Transforming such assessments into meaningful and credible qualitative statements is typically beyond the ability of most development teams [47]. While critical to security, such risk-based analyses are but a "yardstick by which we can judge our security design effectiveness" [51].

Recently, the lack of security requirements-focused processes has been identified as a deficiency in secure software development practice [52]. Processes that have attempted to systematically examine security requirements include the Integrating Requirements and Information Security (IRIS) approach (as an extension of the aforementioned AEGIS process) [53] and the System QUAlity Requirements Engineering (SQUARE) project, which builds upon the SAEM approach in an attempt to provide risk estimates for small businesses [54]. This reliance on risk practices and emphasis on system-level considerations underlie common critiques levied at current practice, to include: the inherent difficulty in defining the goals, expectations and objectives of security [55], the need to consider the global security context in any development that is not stand-alone [56], and the lack of quantification [57], specificity [58] and accuracy [59] in risk management approaches to system security.

2.1.2 Principles

Many of the identified processes are built around one or more software principles — fundamental and widely accepted propositions regarding how to secure software and systems. Core to these principles is the address of vulnerabilities in software, to the extent that it has been said that, "preventing the introduction of vulnerabilities prior to release, rather than patching vulnerabilities afterwards is THE challenge SSE [secure software engineering] rises to meet" [19]. SwSec divides focus between the identification and removal of

Domain	Practices	Touchpoint Rank
Governance	Strategy & Metrics (SM) Compliance & Policy (CP) Training (T)	
Intelligence	Attack Models (AM) Security Features & Design (SFD) Standards & Requirements (SR)	5 6 6
SSDL Touchpoints	Architecture Analysis (AA) Code Review (CR) Security Testing (ST)	2 1 4
Deployment	Penetration Testing (PT) Software Environment (SE) Configuration Management & Vulnerability Management (CMVM)	3 7

Table 2.1: Building Security In Maturity Model (BSIMM) [4] domains and associated practices, with the touchpoints activities identified by rank of 'effectiveness'. Note that the touchpoint 'Security requirements' was mapped to BSIMM both 'Security Features & Design' and 'Standards & Requirements (SR)'.

flaws, or errors in design, and *bugs*, which are errors in implementation — with the prevailing wisdom that vulnerability sources are 50% the result of flaws, and 50% the result of bugs [1]. To this end, the software security community has fostered a number of efforts to address flaw and bug removal.

Since 2003, the OWASP Top 10 [60] list has stood as a model for the remediation of software security bugs (until recently, with a focus on web development), becoming a primary means of educating the software development community of the most commonly occurring software security failures. The defects identified, and their remediation, have found their way into a number of static analysis tools as additional filters [35] and are a key component to many processes (notably, the previously-referenced OWASP CLASP). OWASP has since addressed wider implementation security with the 2014 publication of version 1.0 of the OWASP Proactive Controls for Developers [61], although with less specificity in the specific principles and necessary steps to remediation (for example, direction such as "Design and Architect Security In"). This style of addressing security bugs is not unique to OWASP, with others supplying alternative security redress lists focused on implementation security issues in general development (e.g. the "24 deadly sins of software security" [62]).

In an attempt to address the issue of flaws, the IEEE Center for Secure Design (CSD)[4] published their Top 10 Architectural Flaws [63] in 2014. Mirroring the recognisable 'top 10' style, IEEE CSD employed the approach of gathering examples from a collection of software professionals as to the most common errors leading to insecure software designs[5]. With this recent emphasis on addressing vulnerabilities earlier in the process, it remains to be seen what impact such an approach will have on software security.

The Software Assurance Forum for Excellence in Code (SAFECode) consortium — a collection of industry practitioners promoting secure development best practices — have put together a framework to assess secure software practices. SAFECode's Principles for Software Assurance Assessment [64] attempts to standardise the expectations placed on software developers within the supply chain. In doing so, it highlights a number of expected development practices within three tiers of risk assessment.

While readily recognisable, easy to understand, and straightforward to remediate, such prescriptive "collection" approaches have been deemed by many a 'weaker' model of approaching software security [65]. Limited to the Acquisition/Development phase of the SDLC, their focus is confined to artefact analysis and reflection. While incorporation of these elements into processes forms the basis for systematic approaches, it was realised by the community that technical processes alone were insufficient to produce secure software; a holistic approach must address the "business, social and organisational aspects" in addition to technical principles and processes [4].

2.1.3 Methods

Software security methods supply the necessary context to relate individual processes to the other processes (technical and non-technical) that influence their success. A pioneering example of such an approach is the OCTAVE family of processes[6], developed by the Software Engineering Institute (SEI) at Carnegie Mellon University. In its most recent incarnation, OCTAVE-Allegro [66] further develops this risk-centric approach to managing an organisation's cybersecurity posture.

More broadly adopted sources for methodological guidance include various maturity models, including the OWASP Open Software Assurance Maturity Model (OpenSAMM) [67] and the Building Security In Maturity Model (BSIMM) [68] — with BSIMM

[4] `http://cybersecurity.ieee.org/center-for-secure-design/`
[5] Personal communication to the author (phone interview), 24 July 2015.
[6] `http://www.cert.org/resilience/products-services/octave/`

seeing the widest acceptance. These are not methods in the traditional sense, as they focus on the specification of best practices against which others can measure and therefore do not specify technical steps (although each refers to a development process: the touchpoints in the case of BSIMM, and CLASP in the case of OWASP). However, these methods more completely address the range of concerns expressed throughout the SDLC phases, identifying practices that span the Inception, Acquisition/Development, Implementation, and Operations phases.

BSIMM is a study of software security initiatives within different organisations, with the aim of informing the wider SwSec community. It is heralded as supplying "science to security" by way of a systematic survey of security best practice, reflecting the state of SwSec as practised by a number of (primarily corporate) entities [4]. BSIMM version 8 constitutes the most recent iteration of this survey, identifying four domains — each composed of three practices that are further decomposed into 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 2.1.

However, BSIMM is descriptive, not prescriptive. While the tandem of BSIMM and the touchpoints provides a basis to address the global and emergent nature of software security, some will note that collections of 'best practice' do not constitute formally evaluated methodologies [59]. Most notably, BSIMM does not specify criteria for the selection or application of individual practices [69]. That is not to say that these approaches are without value, but instead the inherent goal differs: relative measures are not the same as reasoned process, especially for producing non-obvious results.

2.1.4 Limitations

Despite a rich collection of guidance, there remains a gap in adoption and — in some cases — SwSec understanding. McGraw identifies a number of SwSec 'myths' that often overly simplify the problem to a specific tool, technique, or technology [70]. With a lack of security training for developers, and a focus on scanning and testing to the detriment of other approaches [71], process optimisation remains a challenge. Despite great strides in the past few decades, SwSec remains an area for continued growth and development.

This dissertation focuses on two specific, largely unexplored observations regarding the state of SwSec practice:

1. While each of the approaches identified in this section specifies any number of prac-

tices or activities, little guidance is supplied as to the resources required. Varied implementation, tools and expertise result in varied effectiveness — and, therefore, a varied level of required investment to achieve the desired results. The constructs presented here leave to the practitioner the exercise of identifying how much, or how little, investment to apply to each activity.

2. There remains little to connect activities across individual SDLC phases, with process operation and output largely confined to the phase in which it is performed. This undermines a coherent, systematic approach to addressing security throughout the system lifecycle.

Risk management is often cited as the answer to lifecycle security, forming the basis for much of the NIST and ISO security guidance (e.g. [72; 73]). However, adoption of risk management as a means of analysing system vulnerabilities, threats, impacts and probability faces its own challenges, and can lead to the mis-management of resources. Risk analysis, as it is commonly practised, is difficult to apply directly to software development without deep experience, contextualisation, and tailoring — and may still only offer general results [51]. A lack of risk data accuracy [74], limited insight into security forecasting [75], as well as the unclear and changing nature of many variables [76], renders the application of these principles to individual software artefacts daunting. A common issue faced by practitioners is the inability to distinguish between (and therefore make reasoned decisions about) "high-frequency, low-impact events and low-frequency, high-impact events" [58], the latter of which may dominate decision making [59] while the former may prove more tolerable. Perhaps most problematic is the common practice of combining data types (ordinal and ratio) [74], such as occurs when the qualitative values (e.g. 'high', 'medium' and 'low') are assigned numeric values and used in computation. Applying risk management is a risk itself, demanding unattainable insight during the early lifecycle phases [77]. This is not to say such constructs are flawed; rather, that they are incomplete in their depiction of security practice.

Like risk, economics pervades the system development lifecycle. Current cybersecurity guidance speaks to the need for a contextually-defined notion of 'adequate security', to which the system security is realised and assessed against [7]. An economic approach realises that defending from all credible threats — being 'truly secure' — is prohibitively expensive, with the optimisation of limited resources a central challenge [23]. Missing from a robust SwSec tradition is insight on how to guide the application of security practices

throughout the system lifecycle. Information security economics is offered as a means to begin to address this deficiency.

2.2 Information Security Economics

Recent interest in an economic approach to information security is commonly attributed to the contribution of Ross Anderson *et al.* in the early- and mid-2000s [78; 9]. Economic concepts such as *misaligned incentives* (where the party making the decision to invest in security is distinct from the party benefiting, or suffering, from that investment), *externalities* (where the costs of decision are borne by others) and *information asymmetry* (where one party to a transaction has significantly more information) are foundational concepts [79] supporting a view that "game theory and microeconomic theory are becoming just as important to the security engineer as the mathematics of cryptography" [80]. This line of research is distinct from the economic study of internet-based systems such as two-sided on-line markets [81], user adoption and network effects [82] and the versioning of software and other 'information goods' [83], which emphasise the role of economics in computing more broadly.

Economics is itself a broad field, with various disciplines. With a focus on the application of information security economics to the software lifecycle, this dissertation touches on concepts from both microeconomics and the emerging field of behavioural economics.

- Most of this research focuses on the core microeconomic notion of *utility*, itself building on the idea of *preference* — the ability to identify one outcome as more desirable than another. *Utility functions* follow as representations of preference relationships; that is, identification of preference to 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, it can be assumed (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 bundle B is better than bundle C, then bundle A is better than bundle C) [84]. From this simple construct comes expressive relationships that underlie game theory and optimisation, examining two or more choices based on the provision of benefit, or utility. In the case of cybersecurity, these elements com-

bine to describe 'rational' actions taken by defenders and attackers as a matter of identifying preferences and optimising the utility of potential actions.

- Behavioural economics takes a more data-driven approach, seeking to 'add humans' in economic calculations to improve the accuracy of predictions [85]. Lacking a unified theory like microeconomics, behavioural economics investigates the role heuristics and biases play in decision making. This is primarily achieved through empirical experimentation, employing statistical reasoning to achieve fine-grained predictions. Despite a propensity for counter-intuitive findings, the understanding brought by this approach can be complementary to the rational model. Applied to cybersecurity investment, these approaches offer insights into problems such as the imbalance between cybersecurity expenditure and security results. While much of this research is rooted in utility theory, the implications to cybersecurity decision-making processes necessarily incorporate behavioural economic concepts.

While a definition of the field of information security economics is not as concise as that of SwSec, Anderson and Moore identify four specific economic applications to information security: the economics of privacy, the protection of computer systems from rational adversaries, incentivising the deployment of security mechanisms, and the economics of vulnerabilities [9]. The first is focused on individual valuations of privacy and the impact of externalities; the second, on the impact to the collective of self-interest and free-riding in interconnected systems. Where these are important security attributes, it is the impact of decision making during the engineering process that informs the central thesis of this dissertation. The remainder of this section places focus on the insights provided by the intersection of security deployment and the economics of vulnerabilities.

2.2.1 Deployment of Security Mechanisms

The application of economics in the decision to deploy security mechanisms is often framed as a move away from 'fear, uncertainty, and doubt (FUD)' toward quantified and repeatable approaches [86]. Accepting that movement from "the idea of absolute cybersecurity" fosters a deeper understanding [87], research in this area has produced a number of security investment metrics and models.

Measurement of information security faces many challenges, including the ability to determine the utility and identify the return [88]. Many traditional economic metrics, such as the Annual Loss Expectancy (ALE) [89], Net Present Value (NPV) [90], Inter-

nal Rate of Return (IRR) [91] and Value at Risk (VaR) [92], have been appropriated to measure information security investment — sometimes with alterations to specialise the interpretation (e.g. the transformation of the Return on Investment (ROI) to Return on Security Investment (ROSI) [16; 93; 94]). In addition, alternative metrics such as the Return on Attack (ROA) [95] and the Cost to Break (CTB) [36] have been developed in an attempt to supply a unique security perspective. However, a lack of clarity in cybersecurity measurement has led some to suggest that a business should "measure what it can measure" [86]. In addition to the obvious cost and effort issues of extensive (potentially irrelevant) measurement, others maintain it is essential to first understand how businesses invest, measure effectiveness, and employ their business model. Without careful consideration, even simple questions — such as what an extra dollar buys — go unanswered [96].

Information security investment models build on these metrics to form decision-support constructs for evaluating security decisions. Evaluation can be made *ex ante* (to evaluate the profitability of future investments), or *ex post* (to retrospectively evaluate past investments) [88]. Schatz and Bashroush identified nine categories of investment model: Analytic Hierarchy Process, Decision Support Systems, Game Theory, Net Present Value (NPV), Return on Attack, ROI, a combination of ROI and NPV, Real Options Theory, and Utility Maximisation. Of these, a focus is placed on utility maximisation and game-theoretic approaches, which enjoy the most ongoing research attention and generation of novel ideas [97]:

1. *Utility maximisation* approaches, exemplified by the Gordon-Loeb model [98], seek to maximise the value obtained by a security investment [97]. The Gordon-Loeb model is notable for its finding that a firm should never spend more than $\approx 37\%$ $(1/e)$ of its expected loss on information security — a caveated finding with assumptions that have subsequently been upheld [99] and contested [100]. This model's popularity has led to a number of extensions, to include alternative breach functions [101; 102] and the incorporation of differing risk appetites [103].

2. *Game-theoretic* approaches employ a strategic, mathematical approach to decision making in the face of an adversary [97]. Example game-theoretic models for information security investment include those presented in [104; 105] and [106]. These are distinct from strategic games of engagement — so-called 'cyber war' — such as those found in [107; 108]. This category of investment model can be further sub-divided

into the various game constructs employed, capturing different conceptualisations of security [109]:

a) A **Weakest-Link Game**, in which the attacker's success is reliant upon a single success (the 'weakest link') against a defender with the responsibility for multiple endpoints.

b) A **Total Effort Game**, which describes the situation where attacker success is dependent on the contributions normalised over all of the endpoints.

c) A **Best-Shot Game** presents the attacker success as dependent upon the best effort of all entities within the system.

d) A **Weakest-Target Game**, introduced by Grossklags *et al.* [110; 111], forms a model of defence whereby an attacker is resource constrained and therefore seeks to access as many endpoints as possible with minimal effort. The defender, therefore, seeks to minimise the attacker impact. The authors define 'with mitigation' and 'without mitigation' variants of this game, whereby the weakest player is considered to be minimally secured or entirely un-secured.

These categories are not fully distinct, and overlap with each other as well as the other seven categories identified in [97]. Notably, the Iterated Weakest Link (IWL) model [112] employs a discrete game theory approach that is optimised using a return on investment calculation.

Despite the development of a framework [96] and analysis method [113] for evaluating information security investment models, there remain challenges to implementing such models in practice [114]. Complexity and practicality is commonly cited [115]. Others point to implicit assumptions in the model construction; for example, that investment is made in its entirety (which may not always be the case [116]) and a general lack of temporal considerations. Issues with the application of investment metrics to security investment are discussed in [95] and [117], with some questioning the appropriateness of their use [96] or arguing that the quantification of security is a 'weak hypothesis' [118]. While no single investment model is likely to provide a comprehensive approach to guiding cybersecurity investments, the 'triangulation' of several models in concert offers promise for defining an overall security investment strategy [96]. Frameworks such as that described in [119] attempt to articulate how the merits of different models can be rationalised, but to date such efforts lack the rigour required for effective SwSec decision making. Finally, any mod-

elling approach must overcome the mismatch between the models and the availability of relevant data [120] — an ever-present concern.

2.2.2 Economics of Vulnerabilities

Another area producing novel security insights is the application of economic principles to the understanding of vulnerabilities. With the expansion of initiatives such as the Common Vulnerability and Exposures (CVE) [121] and the National Vulnerability Database (NVD) [122], remediation efforts remain focused on the Operations phase of the SDLC. This approach has been examined through investigations into the driving forces behind 'patch-and-fix' [123], the economics of vulnerability disclosure [124], and the value of vulnerabilities in the white, black and government markets for zero-day attacks [125]. Findings have spurred wider policy discussions regarding the stockpiling of vulnerabilities [108] and the need for vulnerability markets [18].

Software with fewer vulnerabilities is generally considered more secure [126]. With the recognition that the largest source of vulnerabilities is developers [127], economics has also been applied to understanding the introduction of vulnerabilities earlier in the SDLC. Neuhaus and Plattner [128] investigated the introduction and removal of vulnerabilities in the Firefox codebase, seeking insight into the role of diminishing returns — as the codebase is fixed, vulnerabilities should become fewer, leading to a reduced need for investment. Their results indicated that this was not the case, and that the vulnerability rates were stagnant or even increased over time. This finding is somewhat at odds with Ozment and Schechter [129], who found decreasing vulnerability density in the "foundational" (e.g. present from the start of the survey) portion of the OpenBSD codebase. This discrepancy may be the result of ongoing development in the Mozilla codebases.

Such investigations have implications for investment modelling, which often employs the notion of vulnerability — either implicitly or explicitly. An example of the former is application of the aforementioned SQUARE process to the problem of optimal combination of vulnerability mitigation in fixed-budget defence spending, which employs misuse cases to identify architectural and policy changes [130]. In contrast, the Gordon-Loeb model employs an explicit representation of vulnerability, v. While little research has been accomplished on how to best define or establish this parameter, much has been written on alternative models for vulnerability reduction [100; 102] and their impact on the optimal investment result [101]. Hypothetical [131] and empirical [132] examples have sought to demonstrate how such models can be employed in practice at the system level.

One hurdle to the application of models is populating them with specific, relevant data. While software engineering has a tradition of quantified research in quality, reliability and safety (e.g. [21]), security-specific measurement has historically been absent. Recent focus on the challenges of vulnerability mitigation has led to explorations into the effectiveness of code reviews [126], static analysis [133; 134] and penetration testing [20; 35] in eliminating vulnerabilities. Many of these are limited in the depth required to resolve variables central to cost effectiveness, such as management approach, processes employed, technology sector, and other project-specific aspects of security that early lifecycle information security economic modelling will require.

2.3 Software Security Economics

Applying these concepts, one can conceive of how economics may apply to software security: project demands to deliver and the challenges of security result in misaligned incentives and the delivery of buggy software. A heavy-handed approach to security runs the risk of stifling a project, making the "cure worse than the disease" [135]. Conversely, systematic undervaluation of security for the sake of functionality leads to externalities, where the developers — and even most consumers — do not face the consequences of this decision. With vulnerabilities numerous and little motivation for either developers or consumers to change their habits, technological advancements in software security are stymied by the economic forces at play in the software industry.

Unfortunately, this area has seen little research to date [136]. The methodologies surveyed here are largely silent regarding the relative investment into each measure[7], and fail to consider the trade-space between investments into development practices and deployment actions in order to address security concerns. Combating this situation requires a means to value and integrate economic considerations within the secure software development process. Two specific, inter-related deficiencies that benefit from an economic treatment are the focus of this dissertation:

1. From an economic perspective, judicious employment of resources leads to more efficient security outcomes and paves the way for improved decision making. However, pre-deployment practices have largely been ignored as a contributor to both the cost and the security of the resulting system. While the employment of econometric

[7] OpenSAMM includes per-phase costs for each practice; however, focus is placed on the *institution* of the practice, without considering the costs involved in practice *execution*.

constructs to this end is not entirely new to this research [120], current models focus on the enterprise by examining investments in products rather than considering the processes associated with the development and deployment of systems. There remains a lack of models that incorporate software security, and a difficulty in applying standard cybersecurity economic models to software security [128]. This restricts security investment trade-off considerations within a single phase of the software's life, ignoring its creation and post-deployment existence. When the implications of security decisions fail to consider the software lifecycle, risks are ignored, options are limited, and the resulting systems have enlarged attack surfaces and inherent complexity at their interfaces [137].

2. SwSec makes the case that a combination of process, practice and methodology will lead to more secure software. Yet, little guidance is provided by the software security community to guide the implementation of SwSec practices. Security within software and systems engineering is a difficult issue to define and manage [55], with the linkage between enterprise and project security concerns often left to individual project managers. Replacing this disconnect with quantified, repeatable evaluations within the development process is essential for the development of holistic information security investment.

To overcome these challenges, it is necessary to conceive of how software security contributes to the security ecosystem. Relative to the goals established in Section 1.2, this dissertation develops approaches to identify the balance between the benefit and cost of software security. Such an *economics of software security* addresses the need for defined security investment, as well as the application of these principles to engineering practice.

Chapter 3

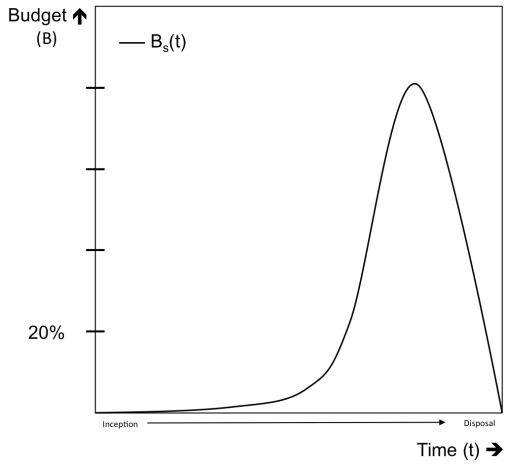
Motivating an Economic Approach

There are only two types of companies: those that have been hacked, and those that will be.
— Robert Mueller

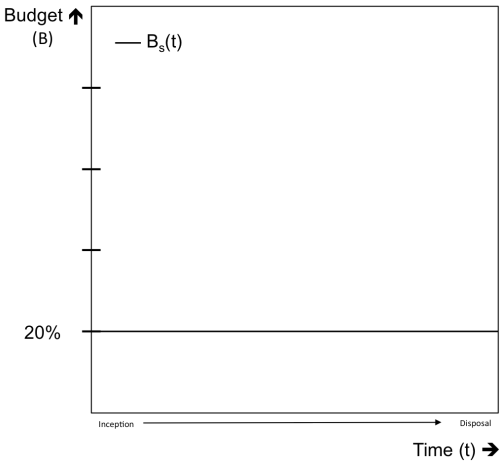
To motivate an economic treatment of software security, it is essential to first understand why — despite the advancements in SwSec and information security economics described in Chapter 2 — failures continue to mount. Section 3.1 frames the development of secure systems in terms of investment throughout the system development lifecycle (SDLC). This is followed by an analysis of an existing post-deployment approach to security, in the form of a UK government cybersecurity standard, in Section 3.2. This analysis illustrates the inefficiencies in the approach to information security commonly employed today.

3.1 Problem Definition

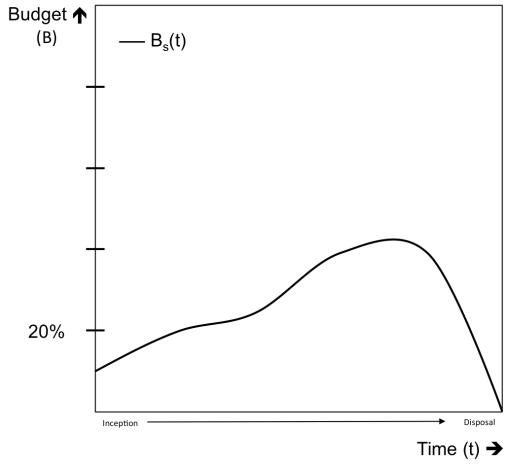
An optimal expenditure of limited, fixed resources that results in the best possible security given technical, legal and business requirements is a critical security enabler [138]. Consider the security investment of a software system as a series of decisions spread over the SDLC phases, starting with Inception and ending with the system's Disposal. Given a fixed budget B, some allocation of B is dedicated to security (denoted B_s) at any point in time t. This can be represented in a Cartesian space, with $B_s(t)$ taking on a variety of forms. Figures 3.1a – 3.1d present conceptualisations of how such security investment might occur, employing a notional 20% budget expenditure.



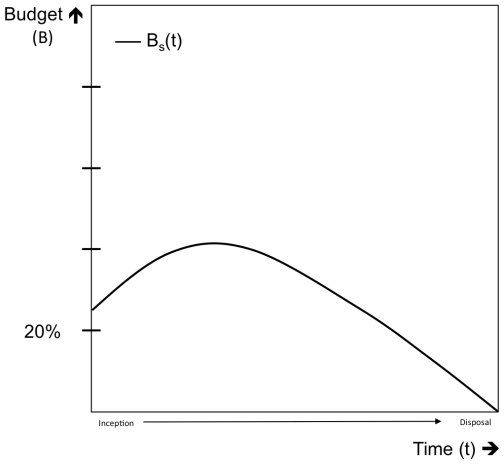
(a) Notional depiction showing the bulk of security investment occurring late in the lifecycle (e.g. post-deployment).



(b) Notional depiction of a constant security budget B_s as a fixed proportion of the per-phase budget B (here, as 20%) over the course of a project.



(c) Notional depiction of security investment that starts early in the lifecycle and slowly increases.



(d) Notional depiction of security investment at the early stages, leading to reduced investment in later phases.

Figure 3.1: Depictions of security investment over the System Development Lifecycle (SDLC).

Figure 3.1a presents what might arguably be considered the current state: a very low security investment effort over the early stages of the process, quickly rising to consume an ever-increasing share of the project costs. Such a profile allocates resources to security to the potential detriment of other investment opportunities (e.g. new functionality, or code maintenance) in later phases. Figure 3.1b, while more controlled, might not depict a more

favourable allocation as it implies that security investment in any given phase is equally beneficial. Many might consider Figure 3.1c to be the most likely scenario to result in effective investment, with security investment ramping up as the security concerns — and system artefacts — become more tangible. However, the SSE community might argue that an investment profile such as that illustrated by Figure 3.1d is the most desirable. This approach reflects the belief that up-front investment yields the best security outcome, with a reduced need for later investment into activities such as accreditation, patching, and breach remediation — a total cost of ownership (TCO) perspective [139]. A logical question to arise from such a construction might be, what is the allocation of resources that provides the most efficient outcome?

To explore this question, this chapter draws from [24; 25] to consider the situation depicted in Figure 3.1a. This is accomplished through an empirical analysis of the UK's Cyber Essentials security standard against hypothetical small to medium enterprise (SME) investments, using the Information Security Breaches Survey (ISBS, or 'Breaches Survey') as source data. Framing the discussion are two questions that elucidate the economic pitfalls of a post-deployment centred approach:

1. *How do the Cyber Essentials controls relate to the reported threat?* As discussed in Section 2.1.1, compliance-focused security approaches often fail to meet security needs due to their prescriptive (and often long-lived) nature. An evaluation of Cyber Essentials controls with respect to the Breaches Survey statistics is used to highlight these limitations.

2. *Is the effort encompassed within the Cyber Essentials controls requisite to the threat?* While the answer to this question is reliant on a number of assumptions regarding the exact implementation and execution of the required controls, this speaks to the heart of the value of a standards-based approach. Employing common econometric measures, this issue of effectiveness is examined to identify the assumptions at play in such approaches.

This chapter concludes with a discussion on how this analysis informs the need for a holistic economic approach that considers investments in software security.

3.2 Standards-based Security: An Example

The Cyber Essentials (CE) scheme, published in April 2014, was created by the UK Department for Business, Innovation & Skills (BIS)[1] to address a perceived lack of existing standards to meet the goals of Her Majesty's Government. While focused on small- and medium-sized organisations in a 'traditional' office setting, CE seeks to influence "organisations of all sizes" and is compatible with established standards (e.g. the ISO 27000 series[2]). At the heart of this policy are five technical controls required for "basic technical cyber protection" in such an enterprise:

1. boundary firewalls and internet gateways;
2. secure configuration;
3. access control;
4. malware protection; and
5. patch management.

These controls are then each sub-divided into between four and seven specific technical measures.

Until 2016, BIS was also responsible for the Information Security Breaches Survey[3], an ongoing series of annual surveys commissioned since the early 1990s [11]. This standardised format (consistent from 2012–2014, and expanded for 2015) provides an executive summary of findings, which is then broken out into: information about the respondents, breakdowns of the data by size and type of business, type of cyber security incident, and loss incurred as a result. In 2015 the report was expanded to include: governance and risk management; 'bring your own device' (BYOD) controls; and incident identification, management and reporting practices.

The common source for these documents offers an opportunity to examine an ISBS-inspired adversarial model to evaluate the controls and effort required by CE. However,

[1] As of July 2016 this scheme now rests with Cyber Aware, a cross-government effort composed of the Home Office, Department of Culture, Media & Sport and National Cyber Security Centre, with funding from the National Cyber Security Programme in the Cabinet Office. See `https://www.cyberaware.gov.uk/about-us`.

[2] `http://cyberessentials.org/background/`

[3] As of July 2016, the Department of Culture, Media & Sport is now responsible for the production of this survey (now known as the Cyber Security Breaches Survey). See `https://www.gov.uk/government/publications/cyber-security-breaches-survey`.

	2015	2014	2013	2012
Infection by viruses or malicious software	63%	45%	41%	40%
Theft or fraud involving computers	6%	10%	16%	12%
Other incidents caused by staff	27%	22%	41%	45%
Attacks by an unauthorised outsider	35%	33%	43%	41%

Table 3.1: Type of breach (for the percentage of respondents suffering a breach). Note that these numbers do not total 100% for a given year, as they are reports of respondents reporting a given breach type and not distribution of breach types.

coordination of these documents does not imply that the participating populations overlap, as only 49% of ISBS respondents are badged or on their way to being badged to Cyber Essentials in 2015 [11]. The relationship between the overall strategy and the resulting policy initiatives prompt questions regarding the impact of this investment in the context of the current cybersecurity environment.

3.2.1 Method

This analysis examines ISBS data for the period 2012–2015 [140; 141; 142; 11]. Given the variability in the size and complexity of corporate defence postures, focus was placed on the category of enterprise defined by the Breaches Survey as 'small businesses'. The ISBS provides data for companies consisting of fewer than 50 employees, with the caveat made by the reports that the data for medium entities (50–249 employees) is "similar to the results for the small ones unless stated otherwise" [11]. As none of the categories investigated state any such caveat, this analysis will employ these statistics to the European Commission definition of a small-to-medium enterprise (0–249 employees).

The surveys provide information regarding both the frequency of a malicious security incident (74% for small businesses in 2015, up from 60% in 2014 and 64% in 2013) and the instances of serious incidents (25% for small businesses in 2015, down from 50% in 2014 but on a par with the response of 23% in 2013). These breaches are then decomposed by type of incident and reported for each of the three years under consideration. Data pertaining to attacks against small businesses is provided in Table 3.1.

Unfortunately (for this analysis) financial data is not reported as overall financial losses, but instead for the largest single loss per entity in a given year. This was compiled into an overall estimate, with *business disruption*, *legal implication*, *incident response*, *financial loss*, and *reputation damage* as contributing costs. The survey combines estimates for these figures into a rolled-up range estimate of £75,200 – £310,800 for worst incident

	2015	2014	2013	2012
Attacks Examined				
Infection by viruses or malicious software	10%	31%	14%	33%
Attack / unauthorised access by outsiders	40%	23%	18%	9%
Attacks Not Examined				
Theft or fraud involving computers	0%	4%	3%	1%
Infringement of laws or regulations	20%	4%	4%	1%
Physical theft of computer equipment	0%	0%	4%	5%
Staff misuse of the internet or email	0%	12%	12%	15%
Systems failure or data corruption	10%	7%	23%	34%
Theft or unauthorised disclosure of confidential information	0%	19%	10%	2%
ISBS Category Differences				
Compromise of internal systems, and subsequent remote access	20%	N/A	N/A	N/A
Other	N/A	N/A	12%	N/A
Largest single loss (£K)	£75.2 – £310.8	£65 – £115	£35 – £65	£15 – £30

Table 3.2: Category of worst attacks suffered and range of largest single loss. Attacks in the first grouping are used for the analysis in Section 3.2.2, while those in the second grouping represent ISBS attack categories not considered. The third grouping depicts year-by-year differences in categorisation.

cost to small businesses in 2015, continuing an upward trend (£65,000 – £115,000 in 2014, £35,000 – £65,000 in 2013, and £15,000 – £30,000 in 2012). This is further broken out into the nature of the worst breach, mapped to categories mirroring (but not equivalent to) the overall incident types (Table 3.2). It is clear from the magnitude of these figures that they are likely skewed toward the upper end of the definition of an SME, an outcome of the granularity of the data reporting approach. As a result, these figures serve the purpose of a worst-case analysis for SMEs.

3.2.2 Analysis

As a starting point for analysis, the individual controls specified by Cyber Essentials were examined for their purpose and approach, and grouped into specific technical or procedural means. The five categories identified by Cyber Essentials serve as the basis upon which specific technical controls are defined. Each control and sub-control in these

Category	Cyber Essentials		ISBS Category
	Control(s)	Identifier(s)	
Boundary firewalls and internet gateways	Firewall Firewall policy	1.1 and 1.5 1.2–1.4	Unauthorised outsiders Unauthorised outsiders
Secure configuration	System administration Personal firewall	2.1–2.4 2.5	Other incidents (staff) Unauthorised outsiders
Access control	Account administration	3.1–3.7	Other incidents (staff)
Malware protection	Antivirus Blacklist	4.1–4.4 4.5	Infection (virus/malware) Infection (virus/malware)
Patch management	Patching	5.1–5.4	Infection (virus/malware)
None	—	—	Computer theft/fraud

Table 3.3: Mapping of computer security controls to Cyber Essentials specified controls (by name and identifier) and Information Security Breaches Survey categories.

categories is stated in a prose form similar to a standard requirement statement, and labelled with a numeric identifier in the [control].[sub-control] format. Note that specified controls do not map directly to specific categories of threat; nor do the sub-controls map to specific technical actions — both of which are necessary for a comprehensive economic analysis. Each sub-control must then be broken into the technical steps for completion, identifying one-time costs versus recurring investment, and providing estimates for items such as time to complete, etc. Malicious incident categories from Table 3.2 were then mapped against the identified controls. With no direct mapping, each control from Cyber Essentials is listed by technology as it relates to the corresponding category of control. The end result is captured in Table 3.3.

The ISBS further describes and decomposes each category into more fine-grained actions [142]. Virus detection and mitigation is the main goal of Control 4 (Malware protection), and is additionally supported by Control 5 (Patching) in the ability of the latter to thwart infection once a virus is present. The threat of outsider attack is more difficult to decompose, as, by definition, it ranges in technical manifestation from penetration, to denial of service, to the impersonation of a company (e.g. phishing) or an individual (e.g. identity theft); however, it is arguably well-covered by the combination of Control 1 (Boundary firewalls and internet gateways), Control 2 (Secure configuration) and Control 5 (Patching). For these categories, focus was placed on technical controls in order to enable economic analysis.

Both 'Threat of incidents caused by staff' and 'Theft or fraud involving computers' are

largely unaddressed within Cyber Essentials. The former is defined in [142] with a range of actions, from unauthorised access to computer systems to breach of data protection law and loss/leakage of confidential information. The latter primarily focuses on the physical aspects of cyber security: the theft of machines, of intellectual property, or of time. As such, they have implications for remedial aspects of cyber security (e.g. backups, restoration and recovery upon the loss of data), which are not part of the Cyber Essentials focus. Since this is a disconnect between the two documents, these two threat classes are not considered in this analysis; instead, focus is placed on the remaining two classes.

3.2.3 Findings

Employing these assumptions, a systematic approach to answering the two questions posed in Section 3.1 can now be undertaken. For the purpose of this analysis, any previous security investment that may have been made by an enterprise is not considered since this information is not readily available from the ISBS data. Due to the updated content and format of the 2015 report, this focus is placed on the 2012–2014 ISBS data.

3.2.3.1 How do the Cyber Essentials controls relate to the reported threat?

Considering a small business' singular worst loss in 2014, the ISBS data is unclear as to if the reported values represent the loss incurred by the worst of *any* malicious breach, or only those considered 'serious'; therefore, for this analysis, the overall probability of breach (60%) is used, rather than the 50% figure representing those who experienced 'serious' breaches. This represents an assumption that any loss results in a worst-case loss, and inevitably strengthens the case for security investment. Loss is calculated using the Bernoulli Loss Assumption, which reduces a probability distribution of losses to a binary realisation of loss with probability p, or no loss at all. This is consistent with the context of a singular breach, and is examined using the Annual Loss Expectancy (ALE), defined as [89][4]:

$$ALE = p \cdot \lambda$$

The ALE_0 (loss with no additional security investment) under these assumptions is presented in Table 3.4, employing the high and low loss event figures for 2012–2014, as

[4]For this analysis a simplified ALE calculation that employs a single probability (p_{breach}) is used, as presented in the ISBS. As such, 'breach probability' can be interpreted as a combination of the more traditional p_{threat} and $p_{\text{vulnerability}}$ commonly employed in risk analysis.

	2012 (£)		2013 (£)		2014 (£)	
	Low	**High**	**Low**	**High**	**Low**	**High**
ALE_{virus}	3,465	6,930	3,724	6,916	12,090	21,390
ALE_{hacker}	9,450	18,900	4,788	8,892	8,970	15,870
ALE_{both}	12,915	25,830	8,512	15,808	21,060	37,260

Table 3.4: Worst case Annual Loss Expectancy for the single worst event (2012–2014). This is calculated using the overall probability of an adverse event conditional on the probability of a single serious event being a virus or hacker.

determined by $p_{\text{breach}} \cdot loss$. Despite the probability of attack dropping in 2014 from 2013 (to 60% from 76% for overall breaches), the ALE continues to rise due to a significant increase in loss incurred; this may be indicative of the increase in 'serious' attacks (66% from 32%), or may simply be tied to escalating costs.

It is worth emphasising that these numbers only consider the loss incurred by attacks in the category 'malicious software' and 'attack by outsiders'. For 2014, this represents 54% of the worst attacks and 78% of the overall attacks. The remaining 46% of worst security incidents (22% overall) fall into categories that either have only partial coverage in Cyber Essentials, such as those identified by Control 3 (account administration), or fall into categories that are not addressed by Cyber Essentials. If the other incident categories identified in Table 3.2 related to staff were to be fully correctable through the implementation of the remaining Cyber Essentials controls, this still leaves 15% of incidents unaddressed by this scheme, and a residual ALE of £9,750 – £17,250 per enterprise in 2014. The implications of this are further explored in Section 3.3.

3.2.3.2 Is the effort encompassed within the Cyber Essentials controls requisite to the threat?

An even bleaker picture of business loss due to cyber breaches can be painted by incorporating additional information from the ISBS. In the absence of total loss numbers beyond the single worst event, overall loss numbers must be extrapolated from the available data.

Looking first at the overall number of attacks resulting in loss, the 2014 report cites the median number of breaches suffered by small businesses as a result of malware infection or attacks by an unauthorised outsider as 3 and 5 respectively, with a median of 6 total incidents overall. Normalising the per-category number of breaches against the median produces an expectation that four of these six attacks will be of one of these two types under consideration (virus and attack). Performing the same analysis on the 2013 and

2012 data produces incidences of 7.9 and 4.5 respectively. This serves as the estimate of the number of attacks resulting in a loss per year.

Since, by definition, these additional attacks are less than the worst reported breach, a worst-case upper bound could be found by multiplying the ALE_0 by the number of incidents; for the 2012–2014 data this results in loss estimates of £116,235, £124,883 and £149,040 respectively (using the upper bound of the ALE for a given year). While rooted in the survey data, it is also an extreme worst case (the median number of incidents, each at the highest end of worst reported loss). As an alternative, the lower estimate for worst loss results in loss estimates of £58,117.50, £67,244.80 and £84,240 per annum respectively. These dire numbers serve as the ALE_0 (loss without security) estimate.

Next, the loss expectancy is compared to the cost and capability to address it. In order to examine these costs, a simplistic cost model is employed for the cost of the Cyber Essentials controls. This model examines costs as a function of:

- the number of machines, n;
- manpower, m_n, per machine;
- wage, w, per unit of manpower;
- one-time costs per machine, o_n, to include licence fees, etc.;
- associated one-time manpower amount, m_o; and
- the fixed cost for investments, I, such as infrastructure (e.g. purchasing a firewall).

This is then calculated, where M represents per-machine costs and O represents one-time costs, as:

$$\begin{aligned} & M + O + I \\ = \; & (m_n \cdot w \cdot n) + [(o_n \cdot n) + (m_o \cdot w)] + I \end{aligned}$$

Using this model, the Expected Net Benefit of Information Security (ENBIS) is employed to examine the rationality of defensive investment. This calculation represents the expected loss without security investment (ALE_0) minus the expected loss with the security achieved by investment s (ALE_s) minus the cost to achieve that security s (assuming

monotonicity in security investment):

$$\begin{aligned} & ENBIS \\ = \; & ALE_0 - ALE_s - s \\ = \; & (p_0 \cdot \lambda) - (p_s \cdot \lambda) - s \end{aligned}$$

In general, one should invest in security at the point that $ENBIS > 0$, representing a positive net benefit. Rewriting to solve for the upper bound of security investment results in $ALE_0 - ALE_s > s$, for which the cost of controls under consideration can be substituted for s. This leaves

$$M + O + I < ALE_0 - ALE_s$$

with an upper bound of

$$M + O + I = ALE_0 - ALE_s$$

Given the bounds placed on the value of ALE_0 and under the assumption that any current security investment is uncounted towards the resolution of the residual probabilities of attack (as presented in the Breaches Survey), estimation of ALE_s follows once the residual probability of loss is known. For now, it is assumed that the implementation of the Cyber Essentials controls will result in a residual probability of 99%; this generous assumption is helpful in investigating the immediate question of resource allocation under analysis. This estimate, along with the loss estimate, provides the information necessary for the right-hand side of the equation.

Turning attention to the left-hand side requires an estimate for the cost of security,

$$\begin{aligned} & s \\ = \; & M + O + I \\ = \; & (m_n \cdot w \cdot n) + [(o_n \cdot n) + (m_o \cdot w)] + I \end{aligned}$$

Fortunately, many of the fixed values can be estimated using publicly available data; Table 3.5 lists representative costs of common cyber security controls based on published surveys, reports and literature. These are employed as estimates for the costs of infrastructure (I) and the fixed costs per machine (o_n).

Providing that wage w can also be estimated from available data, and that the number

Control	Cost (£)	Frequency
Control 1 (Firewall)	222.46 (small) 790.40 (large)	One-time
Control 2.5 (Software firewall)	30.57	Per-machine (One-time)
Control 4 (Antivirus; blacklist)	39.37 24.37	Per-machine (One-time) Per-machine (Yearly)
Control 5 (Patching)	0.00	N/A

Table 3.5: Fixed cost estimates for material investments relative to Cyber Essentials.

of machines n is bounded by the definition of small businesses to be within the range 1–249 — under a simplifying assumption that the number of machines corresponds on a one-to-one basis to the number of employees — most of the values for the model have been identified. The remaining variables m_n and m_o are the most difficult to estimate, as they represent the manpower investment (per-machine and one-time respectively). This includes not only the time to set up and establish the cyber security measure, but also the cost of operation. While the former might be able to be estimated as some percentage of the IT staff budget (or as a bounded time-frame of effort by a smaller organisation), the latter is much more complex. Such costs include not only IT-specific functions such as applying updates, but also the user time spent in the execution of security: time lost to applying and rebooting after a patch; waiting for a virus scan to execute; or in conversation with the help desk upon a (true or false) hit by the antivirus or firewall. Adding to the complexity is that these values are also the most likely to exhibit wide variability, with educated IT staff or competent employees engaging in less time — but also exacting a higher cost per unit of time.

In order to place an estimate on these costs such that the analysis could move forward, relevant literature on this topic was consulted. For ease of use and direct applicability, a model developed by Gartner (and utilised in [143]) was chosen, as it permits estimates of costs based upon the distribution of costs between software (29%), hardware (21%), manpower (40%) and outsourced (10%) costs. The limitations of this model are well documented and acknowledged here; however, for the purposes of providing an analysis of a highly variable quantity for drawing general conclusions, the benefits of this approach outweigh the loss of precision and accuracy in any specific case.

Various SME infrastructure size categories were defined, to include the boundary cases of a single machine, a small company with up to 49 machines and a medium enterprise involving 249 machines. Applying the loss estimates generated previously yields the trend

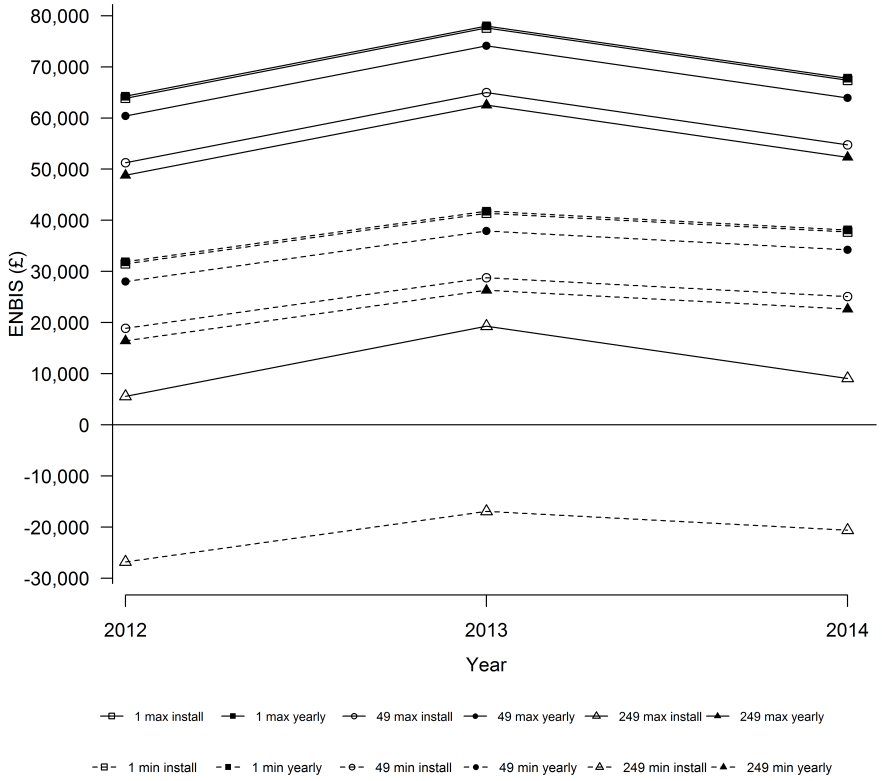


Figure 3.2: ENBIS against loss estimates per year. The lines show upper (solid lines) and lower (dashed lines) bounds, using Gartner assumptions of manpower investment.

lines in Figure 3.2. The first aspect of note is the existence of scenarios in which the security investment may not be a rational choice under the given assumptions: when the lower estimate of loss is applied, the hypothetical organisation at the upper end of SME size experiences a loss in each of the three years. Conversely, the hypothetical single-system organisation exhibits a very high ENBIS.

The relationship between manpower and loss estimates deserves further analysis. Recall that the 'high' loss estimates are calculated for the median number of breaches per year at the maximum reported worst-case loss given the probability of breach and probability of the type of breach being malware or hacker-related. Likewise, the 'low' bounds were set by the same method using the lower-end estimate for the single worst loss. As both use the assumption that each loss would be in the range of the 'worst' single loss, an obvious line of questioning involves this loss assumption: what happens if the loss is

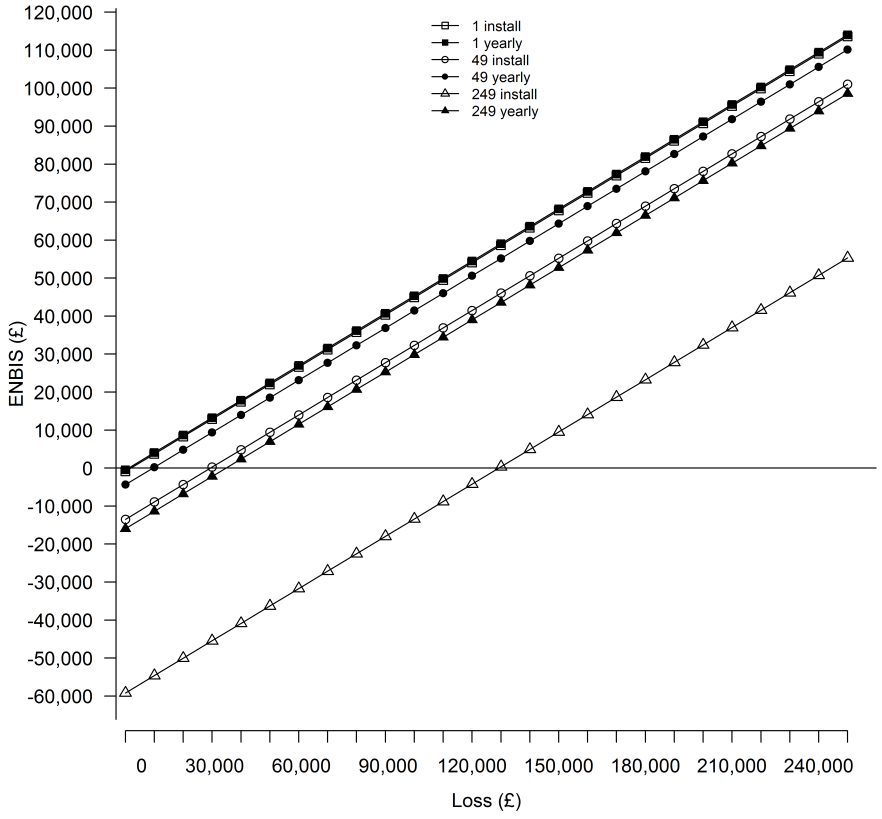


Figure 3.3: ENBIS with varied loss. The lines show yearly and install expenditures, using Gartner assumptions of manpower investment.

lower for a given year (or indeed higher, as the trend in loss values continues to rise)? This scenario is presented in Figure 3.3, using the data for 2014.

Here again, the message that the hypothetical single-machine business should invest in such security is clear: the ENBIS quickly becomes positive in both the installation and annual case, with even small realised losses (above £2,000) providing a positive return on investment. While using the lower cost of the firewall in Table 3.5, it is clear from the estimates for the hypothetical larger company that the scale of the investment is highly dependent on the manpower employed to maintain it. Although the difference in the fixed costs between installation and annual maintenance total £12,137.33 under these assumptions, the overall difference is more than £30,000 of manpower in addition. Since manpower in this analysis is inherently tied to the fixed outlay (as a result of employing the Gartner model), these costs are incurred on a per-machine basis. Therefore, an

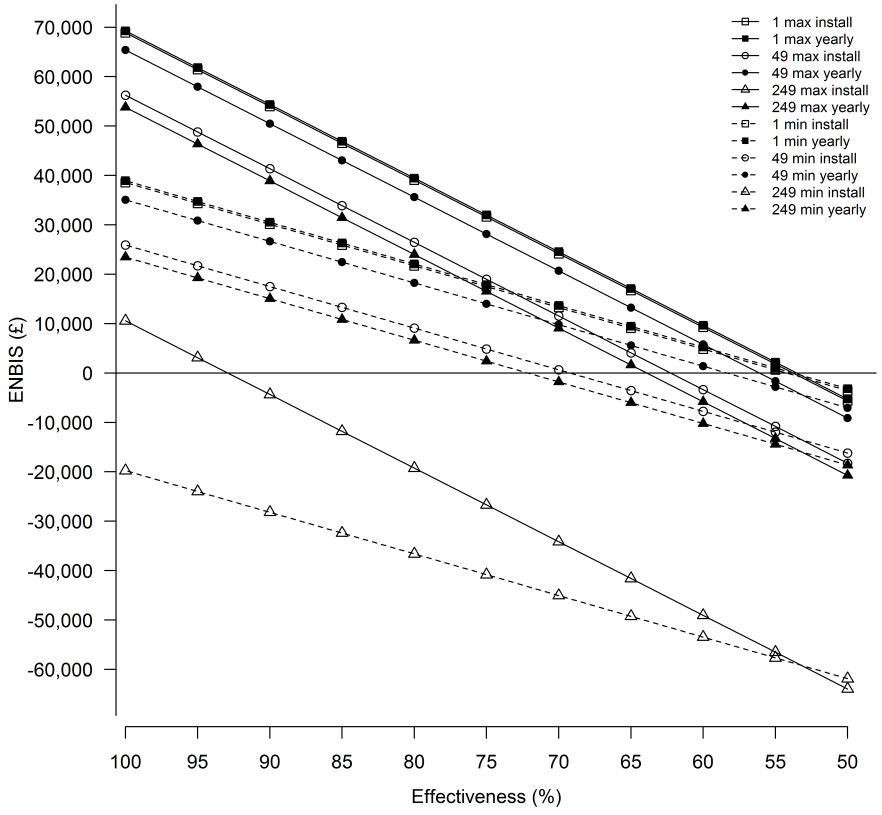


Figure 3.4: ENBIS with varied effectiveness of security controls. The lines show upper (solid) and lower (dashed) loss assumptions for various company sizes (1, 49, and 249), using Gartner assumptions of manpower investment.

increasing amount of loss is necessary to justify the expenditure as the organisation size grows. The resulting effect on the ENBIS supports arguments for automation: the more the manpower can be reduced, the lower the bar for such security measures to constitute a sound investment. However, this must be weighed against any reduction in effectiveness that occurs as a result.

The previous analysis employs the idealistic assumption that the security provided through the implementation of these controls achieves a level of 99%. This implies that the ISBS-reported probability of compromise is reduced from 60% (the incidence of breach for 2014) to just 1%. Clearly, even with the best practices, most IT professionals would be hard pressed to assume their security is so strong. Returning to the high and low estimates of multiple breach loss, the data for 2014 can be examined as security effectiveness varies.

Control	Eff.	Reported ranges / notes
Antivirus	75%	Reported ranges of 5% [144] to 75%[5]
Firewall Firewall policy	60% 80%	Study cited 60% 'out-of-the-box' and 80% only with skilled administration[6]
Blacklists	73.5%	Lowest coverage for a given malware class by all major AV vendors in [145]

Table 3.6: Reported effectiveness (Eff.) for various security controls.

This is shown in Figure 3.4 for effectiveness in the range of 50% to 99%.

From this figure, it is evident that, as the effectiveness of the controls invested in decreases, there is a requisite movement in the point at which the endeavour to deploy cyber defences is no longer a rational investment. For the hypothetical larger small business, this happens quite quickly on the higher loss assumptions for the installation costs: at only around 90% effectiveness these costs overcome the net benefit, as happens at around 64% for the yearly costs. At the lower loss probabilities, the benefit for the install costs is never realised under these assumptions, while the yearly expenditure falls short at around 72%. As before, expenditures at the other end of the spectrum prove quite a good investment, especially at this level of loss; although the upper end of small businesses (49 personnel) calls for closer examination at realistic expectations of effectiveness.

Determining effectiveness of a specific measure can be a difficult exercise; much depends on the specific configuration and deployment scenario, and, to deploy a well-worn cliché, 'the devil is in the detail'. Paywall-protected consultancies often perform analyses of specific software or hardware in order to use that data as part of their competitive edge, leaving only the 'talking-points' version reported by popular trade magazines as a common source. Some of the best openly published estimates are presented in Table 3.6.

The effectiveness of the remaining control — patching — is notably hard to estimate. An initial line of thought would seem to suggest that regular, automated patch application would by definition secure one against all known threats, resulting in an effectiveness close to 100%. However, research, literature and trade publications on the topic seem to suggest that this is almost never accomplished, and the bigger the organisation (thus, the bigger the target), the longer it takes for the company to roll out patches. This is often due to additional testing to ensure non-interference with home-grown applications [146]. Due to

[5] http://www.forbes.com/sites/andygreenberg/2010/10/19/study-finds-microsofts-free-antivirus-as-effective-as-symantecs-norton/

[6] http://www.itproportal.com/2011/04/19/firewalls-only-60-cent-effective-against-malware/

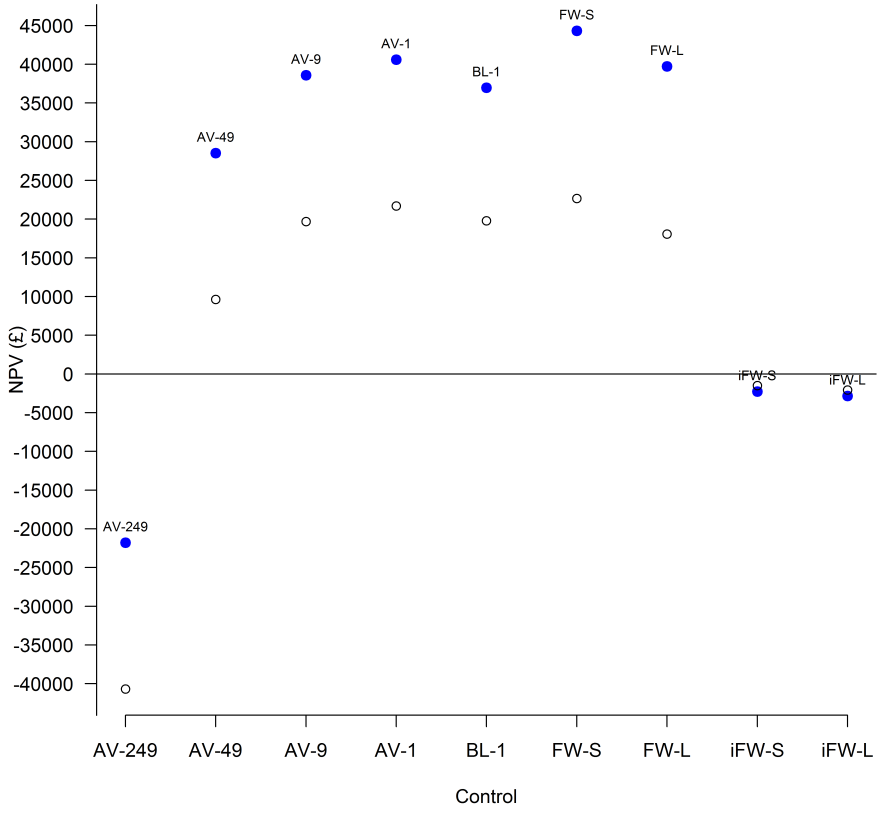


Figure 3.5: Net Present Value (NPV) of security controls as calculated using the assumptions contained in this analysis. Solid dots are calculated using the high-bound loss estimate, while circles employ the lower-bound loss estimate.

the variability inherent in this function, this control is not considered here; although it is worth noting that, under the assumption of high effectiveness, the values cited in the previous subsection (efficiency of 99%) serve as a guide as to what such an analysis might yield.

The aggregate benefit over multiple future periods (n) can be computed using the Net Present Value (NPV), taking into consideration both one-off and recurring costs:

$$NPV = -c_0 + \sum_{t=1}^{n} \frac{ALE_{0,t} - ALE_{s,t} - c_t}{(1+r)^t}$$

Employing the same estimates as in the previous subsection, along with an estimated rate of return of 5%, yields the values in Figure 3.5 for the 2012–2014 data. Specific

comparisons of note include:

- Host-based antivirus and blacklists, mapped to the probability of loss due to malware, with a fixed cost plus an annual fee. These controls illustrate the trade-space of effectiveness with measures of 75% and 73.8% respectively. Values are calculated at the boundary of each class of enterprise that comprise the definition of SME: micro (9), small (49) and medium (249), in addition to a single machine[7].
- A firewall and an 'ineffective firewall' employ the estimates for hardware costs identified in Table 3.5, for small (S) and large (L) businesses. Difference in these controls correspond to ongoing investment, with the 'ineffective firewall' an 'out-of-the-box' configuration with no specialisation in policy or rule set. Absent maintenance it does not incur manpower costs, but operates at an effectiveness of 60% rather than the managed firewall effectiveness efficiency of 80%.

From this graph, it is readily apparent (and rather unsurprising) that the previously identified conditions hold: cost increases with organisational size, yielding higher returns for controls at the lower size estimate. Likewise, in all cases except the inefficient firewall, the lower loss estimate yields a lower NPV. However, the range of results from high to low loss estimates, as well as the estimates resulting in negative NPV, are notable. This underscores a central point of this analysis: effectiveness matters. Deploying just any defence will not yield benefits unless efforts are made to ensure maintained effectiveness. Unfortunately for the case of the technical controls under consideration, the effectiveness will be dependent on the recurring updates and increased manpower costs. While enterprise security mechanisms show some benefit above host-based, manpower-intensive solutions (under the given assumptions and for the same loss expectancy), the margin in many cases may be small. The criticality of this point is owed to the difficulty in ascertaining post-deployment cybersecurity effectiveness, especially against the backdrop of a heterogeneous software environment concealing an unknown number of vulnerabilities.

Comparison of these findings against information security investment models, such as the Gordon-Loeb model [98], show agreement — with caveats. Gordon and Loeb famously found that such investments should never exceed $1/e$ (37%) of the potential loss, under specific assumptions regarding risk posture and security effectiveness. Increased investment in the maintained firewall yields benefit over the alternative, and largely follow this

[7] The values for blacklists track those of antivirus with the same delta as the single machine case, and therefore are omitted from the graph.

rule. However, in the case of the per-host investment, estimates up to 91.1% (install) and 56.4% (annual maintenance) for antivirus in a (hypothetical) upper-end small business violates this guideline when compared against the ISBS potential loss from malware. All else being equal, lower expenditures make for better investments; however, other factors necessitate further consideration. Effectiveness and maintenance must be considered in security decisions, with implications to Gordon and Loeb's rule further explored in Chapter 4.

Another way to interpret Figure 3.5 is in the following postulation: if I had only one dollar / pound / euro to spend on security, which technology is the 'best-bet' for application? While this data is based on a number of assumptions unlikely to hold in totality for any real organisation, these assumptions were held constant throughout and thus provide a basis for investigation of relative merit. From this perspective some indications emerge: as noted, antivirus is preferable to blacklisting, strictly based on effectiveness. Firewalls offer the best single investment when maintained, and prove beneficial in some instances even when little care is given to configuration and administration. This is not to advocate for failing to administer, as the NPV of this control is still negative, and the value increases dramatically with the increased effectiveness that manpower investment brings; rather, it is a good case for investing in controls that secure the network overall, and to push for automation in those that must be host-based. Naturally, the compounding of estimates throughout this analysis has an impact on these conclusions, as does the ability to compose protections (were effectiveness measures for such defensive structures known). A far more interesting question is the relative merits of the individual sub-controls, as their cost and residual impact vary widely across the set.

3.2.3.3 Caveats

The use of ISBS data to perform an analysis of policies such as Cyber Essentials presents challenges that must be considered when interpreting results. These are summarised below; for a full treatment of these issues, the reader is referred to [25].

1. *Survey Consistency.* Variations in the four Information Security Breaches Survey reports employed have potential impact on the analysis. Differences in the presentation of data for small and large businesses, variations in categorisation, and definition of categories resulted in the need to extrapolate and infer on the data. A few inconsistencies, as well as the lack of raw data for any year other than 2014,

challenge the ability to perform some analyses. In particular, changes to the 2015 report necessitates that comparisons to prior data are performed with care.

2. *Methodological Challenges.* The self-selection survey methodology of the data risks gross inflation or domination by the responses of a few participants, in addition to various response biases. As argued in [147], cybercrime surveys — as with any survey of a population where the characteristic of interest is highly concentrated among a small segment of the population — are ripe with errors regarding robustness and statistical validity. Such issues are recognised by the ISBS authors, and despite the care taken by the survey authors (to include the use of pilot studies, 'sticky sessions', and independent reviews to increase data quality)[8] they warn that: "As with any self-select survey of this nature, extrapolation to the wider population should be treated with caution" [11].

3. *Data ambiguity.* The sources employed lack in-depth data in many areas, resulting in ambiguity. Examples include effective security (e.g. the 'detection rate') and the time (manpower) invested, which both vary based on the skill, complexity, institutional size and level of automation assumed. Where possible, such aspects were treated as variable; in other instances, one of two approaches was taken: use of the best/worst case to bound the analysis, or the use of an average to represent a reasonable estimate.

Built upon hypothetical (albeit realistic) scenarios, some specific assumptions were required in order to fill in holes unaddressed by the ISBS:

- Probability of loss was calculated as the overall probability of a breach multiplied by the probability of the breach being of the type (i.e. malware; hacker). The probabilities for these types of breach were treated as exclusive, due to a lack of insight into the underlying probability distributions. As noted elsewhere (e.g. [148]), this will often not be the case.

- Calculated loss was based on the ISBS-reported worst loss in the year considered, and (as with the probability of loss) this was treated independently.

- The expected number of breaches was based on overall probabilities and not the occurrence of 'serious' events the worst loss (presumably) represents.

[8] Chris Potter of PwC, email message to the author, 18 May 2016.

- In the absence of total loss numbers, calculated financial losses are a generalisation based upon the worst-loss event. This was accomplished by multiplying the upper and lower bounds by the worst-loss numbers. It is reasonable to expect additional events to incur a lesser (perhaps far less) loss than the worst loss event, rendering these calculations an upper bound. ISBS indicates that events exhibit varied cost; this analysis shows that as overall cost of events fall, the net benefit is quickly driven lower — especially as manpower remains a primary cost driver.
- Effectiveness measures chosen represented the 'best' (read: most beneficial) estimate of effectiveness across a range of reported values. Naturally, true effectiveness of any such control is dependent on many aspects not considered here, from the vendor technology to the skill of the administrator.

By virtue of the 'worst-case' approach employed, this analysis should be treated as a boundary case rather than a representative example that highlights the 'push-and-pull' between various considerations. Out of context, these results could be interpreted as advocating limited, or even no, investment in security. This topic has been approached by others (e.g. [149], based on improved estimates of breach cost). Others are quick to point out such estimates fail to consider many costs; specifically, externalities created by breaches [150]. Balancing richness with complexity is critical, as overly-simplistic analyses may be misleading or harmful [151]. Ultimately, the decision to implement specific cyber defences relies on much more than an economic analysis, and by necessity must take into effect regulatory, reputational and ethical concerns.

3.3 Informing Security

Each aspect of the presented analysis contributes insights necessary to successfully undertake or expand a cyber defence programme. The effectiveness against the threat, the role of manpower, the pitfalls of scalability, and the expected business loss are all key aspects of the trade-space that enables cyber defence to be a meaningful and beneficial undertaking. The importance and complexity of this trade-off continues to intensify, as spending on security continues to increase (reflected in the 2015 ISBS by 44% of small businesses increasing security spending in the last year, up by 27% over 2014 [11]).

3.3.1 Post-deployment security

The analysis above demonstrates that the potential loss resulting from a breach continues to rise, propelled by increases in the probability of serious events and the costs associated with their resolution. While this bolsters the case for investments in cyber security technology — as advocated by schemes such as Cyber Essentials — there remain significant threat gaps that are unaddressed or left open-ended. Implementation of Cyber Essentials lends itself to a wide range of possible costs, primarily driven by the amount of time invested and scope of deployment. While it is unsurprising that higher effectiveness translates to better value (all else being equal), the interplay between deployment scope and effectiveness results in nuance when considering where to spend the first 'security dollar'. In some cases, for a (hypothetical) larger small business it may prove less beneficial to deploy manpower-intensive security across an enterprise than to deploy less effective, but less manpower-consuming, technologies. This echoes other findings regarding the role of size and industry sector when setting compliance goals (for example, [93]). While the overall value proposition of cyber defence was largely supported by this analysis, the margin between a positive and negative return of investment is perilously narrow — even under the most ideal of circumstances, and with flawless implementation.

3.3.2 Pre-deployment security

Even under a 'worst-case' treatment, the limitations of policy-driven security approaches (such as Cyber Essentials) are evident. Indirectly, the results of Section 3.2.3 are informative to how investments into pre-deployment security might positively impact these findings.

1. The mismatch between the CE controls and ISBS threat information is an inherent limitation of compliance approaches, which some see as static and expensive while lacking equivalent benefits [42]. Worse, there is anecdotal evidence that such approaches may actually undermine security by taking resources from other security activities [152], as depicted in Figure 3.1a. Some have gone as far as to label compliance-driven security as "at best a qualified failure" [153], although the value of compliance in other aspects (e.g. liability mitigation) must also be considered [58].

2. Hoo notes that the viability of compliance approaches depends on reasonableness and effectiveness [58]. While the CE producers took efforts to address the former,

this analysis demonstrates the impact of the latter on the economic viability of compliance. The effectiveness measurements derived from the literature elucidate the inherent, and often unwarranted, assumption of highly effective implementation. As vulnerabilities stem from both coding and emergent faults [17], it can be argued that it is unreasonable to expect post-deployment mechanisms alone to provide high levels of assurance — and as unreasonable not to expect them to play a part in an effective holistic solution. Identifying this balance is critical to system security.

3. This analysis highlighted manpower as a driver for cost. As security measures consume ever greater resources — either through additional staff or encroachment on the user — costs can be expected to escalate. As noted by Herley, user time is not free — even if that time is employed in the name of security [154]. Applying the processes, principles and methods of SwSec offers the promise of software with minimal vulnerabilities that remains secure in the face of attack [19], potentially reducing the need for post-deployment investments. Such practices represent a shift in cybersecurity expenditure, with success depending on the ability to control and optimise this investment.

With this example serving as motivation, Chapter 4 presents methods that incorporate software security investment into information security investment planning. This approach establishes how these issues can be rectified through early security investment, resulting in the consideration of security throughout the system lifecycle, a reduction of the reliance on post-deployment security compliance, and a balancing of pre- and post-security investment.

Chapter 4

Software Security Investment

Essentially, all models are wrong, but some are useful.

— George Box

Meeting the challenges presented in Chapter 3 requires re-evaluating existing approaches to information security investment. Recalling Figures 3.1a – 3.1d, Section 3.3 examined how over-reliance on post-deployment, compliance-oriented security (as depicted in Figure 3.1a) can be inefficient and result in lowered return on investment. Switching to an *ex ante* perspective, this chapter considers the effect of earlier intervention that leads to investment profiles such as that portrayed in Figures 3.1c and 3.1d. This will be accomplished through the development and application of economic models.

In the early phases of the SDLC, security effort is focused on execution of secure software methodology; however, as discussed in Chapter 2 current information security investment models fail to consider such pre-deployment actions. One approach to addressing this deficiency is within the framing provided by existing models, whose structure and assumptions capture critical aspects of various security scenarios. Rather than replacing these established approaches, augmenting these constructs provides an opportunity to consider the value of secure software engineering within the context of the entire system lifecycle.

This chapter introduces models for lifecycle security investment, based on existing information security investment principles. An approach to modelling SwSec is established in Section 4.1, along with assumptions and constraints influencing the development of such models. Using these principles, Section 4.2 constructs a novel, single-period approach, based in empirical software engineering practice. In Section 4.3, a multi-period model

is formed that introduces attacker considerations into SwSec investment planning. The developed techniques are then analysed in Section 4.4.

4.1 Modelling SwSec Investment

Calculating the return on investment (ROI) for SwSec requires the consideration of the costs and benefits derived from the wide array of software security practices; unfortunately, no definitive source of such data exists. Despite a plethora of partial sources, including industry reports, academic investigations and technical reviews, empirical data remains a challenge for the security community, and software security in particular [69]. None of these sources are comprehensive or complete, and no modelling technique can completely overcome a lack of good data [58]. Tackling complex questions requires us to start with what is available, and make progress through analysis and sharing [86]. To address these challenges requires the synthesis of theory and practice in both domains, as comprehensive data collection supporting economic analysis of software security remains a rich area for future empirical research.

4.1.1 The COSECMO Model

One of the richest sets of software engineering data resulted in the development of the Constructive Cost Model (COCOMO) [21; 155], proposed by Barry Boehm in 1981 and revised in 1995[1]. COCOMO subsequently served as the basis for the development of the Constructive Security Cost Model (COSECMO) [156], which provides estimates of the effort required for incorporating security into accredited software. COSECMO factors capture the increased costs of development throughout the SDLC, related to progressively higher assurance software. While its use of cost factors is too explicit for use as a general estimate (since a definitive set of secure software activities and their effectiveness is not generally agreed upon [157]), this model serves as one of the few resources for secure software cost estimation.

While the overall COSECMO model expands on the COCOMO model to provide an overall effort estimate for software development, for the purposes of modelling software security investment, only model aspects directly related to software security are relevant to this discussion. Security is captured in COSECMO by the parameter `%Effort(AL)`, corresponding to the assurance level multiplicative factor (i.e. the effort required to develop

[1] `http://sunset.usc.edu/csse/research/cocomoii/cocomo_main.html`

to a given assurance level). For the popular Common Criteria[2] Evaluation Assurance Levels (EALs), this is calculated as:

$$\begin{aligned} \texttt{\%Effort(AL)} &= \texttt{\%Effort}_3 \cdot \texttt{SECU}^{(\texttt{EAL}-3)} && \text{for } \texttt{EAL} \geq 3 \\ &= 0 && \text{for } \texttt{EAL} < 3 \end{aligned} \tag{4.1}$$

COSECMO applies the `%Effort(AL)` factors to the base effort for elaboration and construction `Effort(EC)`, yielding the additional effort for internal assurance:

$$\texttt{Effort(Int.Assure)} = \texttt{Effort(EC)} \cdot \texttt{\%Effort(AL)} \tag{4.2}$$

As noted in [157], COSECMO does not equate to individual investments into secure software. However, these equations supply calculations for identifying the magnitude of the incurred cost over the system lifecycle (Equation 4.1), as well the additional cost of secure software relative to a specific development effort (Equation 4.2). This enables an analysis of security costs relative to different timeframes: the entirety of the system lifecycle, within a given phase, and for a specific investment.

4.1.2 Assumptions and Constraints

The inherent limitations of models such as COCOMO [158], and of Lines of Code (LOC)-based estimation generally [159], require care when interpreting their output. Software development is a complex endeavour, with numerous influences and constraints that impact programme execution. Regardless the functional goals, each software development generates a set of actions (design, coding, test) that results in a set of costs (time, manpower, tools), producing a system with a requisite security baseline. Adherence to various processes, principles and methods will positively or negatively impact this baseline and, subsequently, system cost. Secure software engineering augments standard software development by introducing a security focus to these activities (presumably at a higher cost), with the goal of raising the security baseline by reducing the incidence of vulnerability introduction. Thus, an economic exploration of secure development must consider various development approaches within the context of cost and security.

Impacting this trade-space are a number of external constraints that affect process execution, development costs, and vulnerability reduction effectiveness. To model secure software development, it is important to identify these constraints and their influence.

[2] `https://www.commoncriteriaportal.org/`

1. Software is often built as an extension to, or a replacement for, an existing system, perhaps as part of a software product line (SPL) with a shared codebase. This may result in requirements to conform to specific features or operational control flows. These conditions impact security decisions by requiring the use of legacy components and code, or supporting features with complex security requirements. Software that is built absent any existing system is known as *greenfield development*.

2. Most software is developed to operate in an integrated environment, interacting with other software. Security can be impacted by the interaction with systems via legacy interfaces, incorporation of certain frameworks, or the support of specific data formats and communication channels. Systems that operate independent of other systems are known as *standalone systems*.

3. Traditionally, software was developed under the direction of a manager or project leader, as a concerted effort of a small group of developers for the benefit of a single entity — and the code tightly controlled. Increasingly, software is constructed by geographically dispersed teams in an asynchronous manner, with the result (and the artefacts) widely available for inspection by the entire community. The impact of *closed source* versus *open source* development on vulnerability introduction is a long-running debate [79], contrasting the merits of diverse, potentially numerous reviews resulting from transparency with the motivated, professional examination resulting from opacity. This dilemma has led some to advocate a 'closed-to-open' model [160].

4. Approaches to software engineering vary, and constitute an active research topic. The classic approach to software development is the waterfall model, in which each phase of development (i.e. requirements, design, code, etc.) is distinguishable and sequential. While there is evidence that such models remain a dominant approach in many development environments [161; 162], increasingly, other models have been adopted in order to overcome the inherent challenges of taking such an approach. Evolutionary, iterative, prototyping, and agile process variants are now commonplace, each offering a different perspective on the execution development activities. Subsequently, each process variant has the potential to impact security in different ways, at different times, and with different costs.

To examine how these aspects of software development impact investment into secure software, both single-period and multi-period models are considered.

4.2 Single-Period Models

The majority of economic modelling employs a singular time horizon, applying to a wider variety of business cases and requiring few assumptions. As previously identified, the model introduced by Gordon and Loeb for information security investment [98] has stood as the 'gold standard' in this area since its publication in 2002 [114]. This section examines SwSec investment principles in a single-period model, using the Gordon-Loeb model as the basis, to examine the applicability of such an approach to SwSec.

4.2.1 The Gordon-Loeb Model

The utility maximisation approach is exemplified by the Gordon-Loeb model [98], whose finding that a firm should never spend more than $\approx 37\%$ $(1/e)$ of its expected loss on information security contains assumptions that have subsequently been upheld [99] and contested [100]. This model's popularity has led to a number of extensions, including alternative breach functions [101; 102] and the incorporation of various risk appetites [103].

The Gordon-Loeb model characterises the vulnerability of a system (an "information set") as a function of initial *vulnerability* (v), expressed as a probability, and an *investment into security* (z). The reduction in vulnerability provided by z is expressed by a *security breach probability function* (SBPF, denoted $S()$), governing the return per unit z. The resulting Expected Net Benefit of Information Security (ENBIS) for a given investment z is calculated as [98]:

$$\text{ENBIS}(z) = [v - S(z, v)]L - z \tag{4.3}$$

where

$$L = t\lambda \tag{4.4}$$

Here, the *expected loss* L (in the same units as z) is a combination of the single-threat probability t (fixed at $t = 1$ for the evaluation in [98]), and the *loss incurred when the information set is breached*, λ. The loss λ is finite and assumed to be bounded, and so "...the model is not intended to cover protection of national/public assets or other circumstances where a loss could be catastrophic" [98].

Key to the role of SwSec is the nature of the vulnerability v and security breach probability function $S()$, which warrant a more thorough analysis.

4.2.1.1 Vulnerability

In the original Gordon-Loeb model, the vulnerability v ($v \in [0, 1]$) of a system (or 'information set') is defined as [98]:

> "... the probability that without additional security, a threat that is realised will result in the information set being breached and the loss, λ, occurring."

Various interpretations of this statement have emerged in the literature. Willemson [100; 102] relates v to the initiation of action by the attacker, while Farrow and Szanton [163] appear to equate the probability of a breach to the outcome of such a breach. These definitions fail to disassociate the probability of an attack, represented by Gordon and Loeb as t, from the success of such an attack. Huang, Hu and Behara [103] identify v as being intrinsic (as opposed to the extrinsic threat value, t), but attribute this solely to the topology and connectivity of the system — by no means the only intrinsic characteristic.

An expanded definition is required that reflects the correspondence between v and system characteristics, as well as the independence of v and t.

> The *vulnerability* v is an intrinsic state of a system (information set), determined by the inherent characteristics of that system. It is represented as the probability that, without additional security, a threat that is realised will result in the information set being breached and the loss, λ, occurring.

This definition clarifies that the system vulnerability v has a direct correspondence to system characteristics as determined by its construction, and is differentiated from the external threat, t. It not only supports the application of the Gordon-Loeb model throughout the SDLC, but also communicates the ever-changing nature of vulnerability throughout the lifecycle.

Although the effect of various v values on the outcome of the Gordon-Loeb model is well studied, the assignment of v for a specific system has been an unexplored topic in the literature. While the definition in [98] is consistent with probability-based frameworks for operational security measurement, the basis for how this probability is to be established or interpreted is never discussed.

4.2.1.2 Security Breach Probability Function

In much the same way, the security breach probability function $S(z, v)$ has been interpreted differently in various Gordon-Loeb model-related works. The definition in the the

original paper states [98]:

> "... the probability that an information set with vulnerability v will be breached, conditional on the realisation of a threat and given that the firm has made an information security investment of z to protect that information. We refer to the function $S(z, v)$ as the security breach probability function and to its value at a particular level of z and v as the security breach probability."

As before there are discrepancies in the terminology, starting with the conflation of the probability of a breach occurring and the success of the attacker [163; 101]. In addition, and more specific to the security breach probability function definition, there appear to be two prevailing conceptions: that of a 'residual' probability (implying investment z has reduced the breach probability to this level) [100; 102; 163], and that of a 'revised' probability (implying that investment z results in a new breach probability) [131]. Clearly, the formulation of the model supports the latter definition, as the first would negate the need to subtract $S(z, v)$ from the original v. Coherent with [98] and the vulnerability definition above, the following definition is employed:

> The *Security Breach Probability Function* ($S(z, v)$) is a function that results in the revised measure of the probability that a system (information set) with vulnerability v will be breached, conditioned on the realisation of a threat and given an information security investment of z.

Unlike v, much has been written regarding the definition of $S(z, v)$ and underlying assumptions; this work is summarised here, and presented in full in Appendix A.

Gordon and Loeb define three assumptions for the construction of probability breach functions, denoted $A1$–$A3$, summarised below:

- $(A1)$: If $v = 0$, z has no impact. The system is invulnerable in the absence of vulnerability $(\forall z.\ S(z, 0) = 0)$.
- $(A2)$: If $z = 0$, then v is unchanged. The vulnerability does not change in the absence of investment $(\forall v.\ S(0, v) = v)$.
- $(A3)$: The reduction of v is monotonically increasing with z, but at a diminishing rate. Therefore, returns decrease as a result of increased investment, with an asymptotic limit at 0 as $z \to \infty$.

Following from these assumptions, Gordon and Loeb identify two security breach probability functions known as S^I and S^{II}:

$$\begin{aligned} S^I(z,v) &= \frac{v}{(\alpha z+1)^\beta} \\ S^{II}(z,v) &= v^{\alpha z+1} \end{aligned} \tag{4.5}$$

These equations provide alternative depictions of the effect of security investment, with S^I providing a greater reduction in v at low investment amounts while S^{II} results in larger reductions of v overall (for equivalent parameter values).

Others have followed S^I and S^{II} with additional assumptions (as well as variations on $A3$), producing a number of security breach probability functions that describe various potential situations. Relevant to this discussion is assumption $A4$ proposed by Hausken [101], which states that for all investment under some intermediate amount security increases at an increasing rate, beyond which the security continues to increase but at a decreasing rate (resulting in a sigmoid curve). This is used to develop a new security breach probability function, denoted $S^{III\text{-}H}$:

$$S^{III\text{-}H}(z,v) = \frac{v}{(1+\gamma(e^{\theta z}-1))} \tag{4.6}$$

Few attempts have been made to provide meaning to SBPF parameters, rendering the application of these functions more an art than a science. For S^I and S^{II}, Gordon and Loeb indicate only that α and β are measures of the "productivity" of investment [98], with the probability of a breach decreasing in these parameters. In [131] further insight is provided as progressively larger values of β are applied as v increases, indicating a correspondence to productivity relative to the vulnerability. Hausken provides an interpretation only for his proposed $S^{VI\text{-}H}$form, describing the function as representing "...a hole with a given size [λ] through which cyber attacks flow. The success of cyber attacks is proportional to the size of the hole" [101]. Willemson is the most specific, proposing parameters that capture "maximal amount of investment [b] required to completely secure our information set" in addition to parameters that play a role in investment productivity (k, which is employed in a similar manner to α in S^{II}) [100; 102]. However, the precise meaning and proper usage is left to interpretation.

Further insights can be gained in the purpose of productivity parameters relative to the investment, z. Little is stated as to the nature of z, with Gordon and Loeb only stipulating its measurement be "in the same units (i.e. dollars) as L" [98]. However, within

each proposed SBPF, z is either directly or inversely proportional to the vulnerability, rendering appropriate parameter values dependent on the magnitude of z — rather than purely absolute values. A simple example can be created from [131], in which the authors demonstrate the application of the Gordon-Loeb model to a security investment decision. Units of \$1 million USD are employed for calculations, such that $z = \{20, 40, 60, 80, 100\}$, $\alpha = 1$ and $\beta = \{1, 2, 3\}$ (the latter corresponding to high, medium and low vulnerability). It is simple to show that changing $z = \{20000000, ...100000000\}$ (i.e. a unit of \$1 USD) without changing the other parameters yields drastically different results. Dividing α by the base unit value rectifies the calculation, leading to an interpretation of α as 'productivity per unit investment'.

To date, the literature has failed to provide standards for Gordon-Loeb model parameter values [164], relying instead on interpretation by the model user. This is further compounded by the fact that, despite the numerous proposed functional classes, none have been properly investigated for real-world applicability. Some empirical evidence for the function S^{II} exists from a study completed by Tanaka and Matsuura [132]; however, Willemson points out that, comparing function classes S^{I} and S^{II}, the real claim is that the functional family S^{II} "fits more" for the case of computer viruses spread by email — with supporting evidence that is "rather remote" [102]. In [100] Willemson further asserts that there is "no reason to assume that any function in any of these families corresponds to any real vulnerability decrease scenario", while recognising that any such mapping would be heavily application-specific and could consider other effects of investment in addition to vulnerability reduction.

4.2.2 Applying the Gordon-Loeb Model

Establishing such a relationship between information economic representations and software engineering practice is critical to forming a basis for SwSec investment modelling. The following sections evaluate the connection between information security investment model parameters and empirical software engineering data and models. Using the Gordon-Loeb model, the investment into software security across a system's development lifecycle is considered, employing the COSECMO model as an approximation of lifecycle software security investment. Starting with an examination of software security breach probability functions, this relationship is then examined at the lifecycle, phase and practice levels of SwSec. This analysis assumes a phased development (coherent with the COCOMO model), and that COSECMO cost factors accurately portray the escalation of SwSec

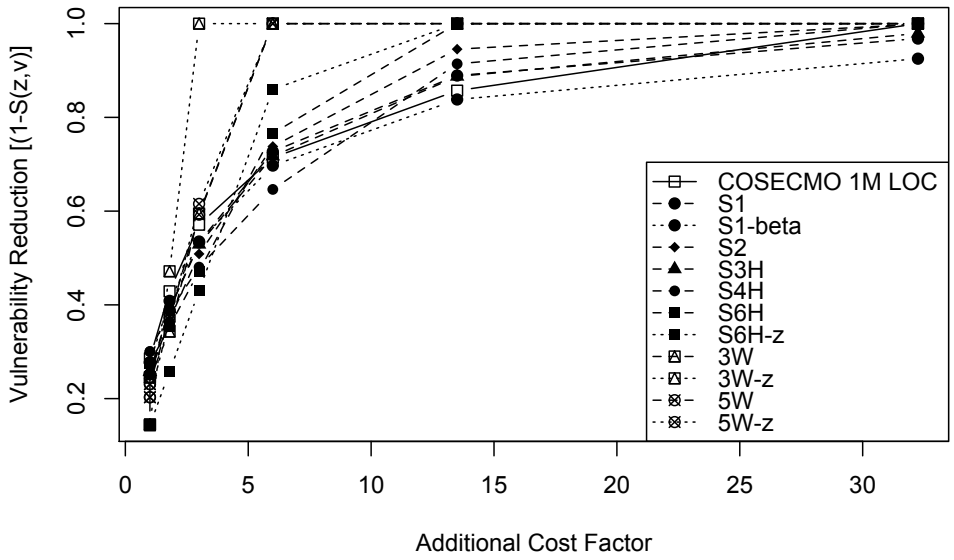


Figure 4.1: Best fit to the COSECMO 1 million LOC investment factor (solid line), for the security breach probability functions proposed by Gordon and Loeb (S^{I}, S^{II}), Hausken ($S^{III\text{-}H}$–$S^{VI\text{-}H}$) and Willemson ($S^{III\text{-}W}$, $S^{IV\text{-}W}$).

investment required for a given vulnerability probability.

4.2.2.1 Software Lifecycle

Recall that, in the COSECMO model, security is described relative to an assurance level; the most widely recognised being the Common Criteria EALs. COSECMO defines additional effort beyond an EAL level 3 ($\texttt{\%Effort}_3$) system as 20%, 40%, 60%, and 80% of the development effort for codebase sizes 5 KLOC, 20 KLOC, 100 KLOC, and 1 MLOC (where 'KLOC' and 'MLOC' refers to thousands (K) and millions (M) of LOC) was employed, as identified in [156].

In order to enable a quantitative examination of security, EAL levels were mapped to vulnerability by dividing the vulnerability interval $[0, 1]$ by seven (0.143, 0.286, 0.429, 0.571, 0.714, 0.857, and 1.0) such that each EAL level approximates a vulnerability value. This transformation is performed under the assumption that EAL 7 represents software that has achieved arbitrary closeness to formally verified, ideally secure software — an assumption deemed reasonable given that the Common Criteria Certified Products List[3] contains only six EAL7/EAL7+ certified products, out of 2206 overall. It is a simplification and generalisation, but within the spirit of the EAL levels.

[3] `http://www.commoncriteriaportal.org/products_OS.html`, accessed 13 February 2017

Best-fit parameters for a given SBPF were identified using the nonlinear least squares (`nls()`) function in the statistical package R[4]. This algorithm was chosen over other approaches (e.g. maximum likelihood estimation) due to its simplicity and lack of assumptions regarding the distribution of the underlying data. For each SBPF, the regression first executed with all parameters free, followed by a second estimation with specific values set in accordance to published analyses. Each SBPF is started with a vulnerability probability of $v = 1$[5], reflecting an implicit assertion that any codebase of significant size will contain an exploitable vulnerability in the absence of security efforts; this is consistent with current secure software engineering thought [1].

From these calculations, the correlation between the EAL estimation of vulnerability and the nonlinear least squares approximation for a given SBPF were recorded, along with the residuals. These values were calculated using the correlation (`cor()`) function in R. This information is captured in Table 4.1 and Table 4.2, with the best-fit for 1 MLOC presented in Figure 4.1. From the data in these tables, certain trends can be identified.

Gordon and Loeb: S^I and S^{II}. Starting with the Gordon and Loeb functions, the SBPF S^I exhibited the highest correlations of all examined functions. In this analysis, when β is held to 1 (indicating a lowered threat environment [131]), correlation values increase (with a small increase in residual values, which `nls()` seeks to minimise). Additionally, the value α remains well below 1, indicating a comparatively inefficient investment. Together, these results fit the narrative of SwSec: an environment characterised by high vulnerability, but in which the effectiveness of the investment is indirect. In other words, not every dollar spent on reviews and tests addresses vulnerability reduction, but these investments combine to contribute to better, more secure software overall. As commonly conceived, the benefit of SwSec investment lies in the long-term returns (the 'total cost of ownership'), rather than in immediate savings [139].

Function S^{II} only utilises one parameter, α. This regression results in correlation values that are highest in a high vulnerability environment (i.e. when v is also high). However, this only occurs with commensurately high values of α, indicating security productivity 8-25 times the values employed in [131]. Such an increase is well outside commonly employed parameters, and correlation is reduced from S^I in all cases.

[4]R version 3.3.2 GUI 1.68 Mavericks build (7288). The `port` variant of the algorithm was employed, as the only algorithm supported by `nls()` that permitted the specification of start, upper and lower values for the parameters.

[5]As previously described, one exception is the S^{II}function due to its failure to converge when $v = 1$. Instead, a value of $v = 0.99$ was employed.

	5K LOC			
S^{I}	1	$\alpha = 0.04541$, $\beta = 10$	0.9524087	0.07084
	1	$\alpha = 0.6575$, $\beta = 1$	0.9561801	0.1336
S^{II}	0.9599	$\alpha = 10$	0.9483162	0.07359
	0.99	$\alpha = 42.79$	0.9511346	0.06809
$S^{III\text{-}H}$	1	$\phi = 1.1987$, $\gamma = 0.1811$	0.9600694	0.04694
$S^{IV\text{-}H}$,$S^{V\text{-}H}$	1	$\mu/\omega = 0.3379$, $k = 0.7056$	0.9331026	0.0745
$S^{VI\text{-}H}$	0.8534	$\lambda = 0.2242$	0.9202003	0.08747
	1	$\lambda = 0.3168$	0.9492328	0.06317
$S^{III\text{-}W}$	0.656	$b = 4.406$, $k = 2.010$	0.8771606	0.1316
	1	$b = 1.908$, $k = 2.194$	0.9441335	0.1189
$S^{III\text{-}W}$	0.8492	$b = 4.4582$, $k = 1.0100$	0.91975	0.08795
	1	$b = 2.860$, $k = 1.179$	0.9535769	0.05919
20K LOC				
Function	v	Parameters	Correlation	Residual
S^{I}	1	$\alpha = 0.03544$, $\beta = 10$	0.9719699	0.033
	1	$\alpha = 0.5142$, $\beta = 1$	0.976404	0.06644
S^{II}	0.9684	$\alpha = 10$	0.969124	0.0359
	0.99	$\alpha = 33.29$	0.9704387	0.578
$S^{III\text{-}H}$	1	$\phi = 0.3395$, $\gamma = 1.0031$	0.9709638	0.03335
$S^{IV\text{-}H}$,$S^{V\text{-}H}$	1	$\mu/\omega = 0.3204$, $k = 0.5315$	0.9481614	0.05833
$S^{VI\text{-}H}$	0.7580	$\lambda = 0.1265$	0.920346	0.08732
	1	$\lambda = 0.2311$	0.9587874	0.06318
$S^{III\text{-}W}$	0.7336	$b = 4.2648$, $k = 2.0100$	0.9375545	0.07755
	1	$b = 3.738$, $k = 2.010$	0.934044	0.2646
$S^{III\text{-}W}$	0.9164	$b = 4.6885$, $k = 1.0100$	0.9568297	0.05652
	1	$b = 4.306$, $k = 1.010$	0.9585885	0.0642

Table 4.1: Security Breach Probability Function to EAL comparison for 5 and 20 KLOC.

Hausken: $S^{III\text{-}H}$, $S^{IV\text{-}H}$, $S^{V\text{-}H}$, and $S^{VI\text{-}H}$. The Hausken SBPFs are characterised by the relaxation of a specific assumption underlying the functions defined by Gordon and Loeb: that of increasing, but diminishing, returns to security investment.

In the case of $S^{III\text{-}H}$, ϕ and γ affect the steepness and sigmoid midpoint of the logistic function. As ϕ directly impacts z, it can be inferred this is similar to α in Gordon and Loeb's SBPFs (a view supported by the values assigned to ϕ in [101] and α in [131]). Taking this view, we again see high vulnerability ($v = 1$) with either ϕ or γ less than 1 in each case (and the other value within an order of magnitude). This combination of values results in a muted bend in the characteristic logistic 'S' shape, approximating

100K LOC				
Function	v	Parameters	Correlation	Residual
S^{I}	1	$\alpha = 0.1167$, $\beta = 2.8397$	0.9803488	0.02299
	1	$\alpha = 0.4361$, $\beta = 1$	0.9836491	0.03926
S^{II}	0.9732	$\alpha = 10$	0.975312	0.02927
	0.99	$\alpha = 27.93$	0.9758913	0.02919
$S^{III\text{-}H}$	1	$\phi = 0.1474$, $\gamma = 2.3619$	0.9794545	0.02395
$S^{IV\text{-}H}$,$S^{V\text{-}H}$	1	$\mu/\omega = 0.3089$, $k = 0.4646$	0.956637	0.04909
$S^{VI\text{-}H}$	0.8549	$\lambda = 0.1528$	0.957021	0.05631
	1	$\lambda = 0.1778$	0.9576561	0.0873
$S^{III\text{-}W}$	0.7087	$b = 5.8545$, $k = 2.0100$	0.9337535	0.08165
	1	$b = 5.123$, $k = 2.010$	0.9307594	0.3369
$S^{III\text{-}W}$	0.8519	$b = 6.5312$, $k = 1.0100$	0.9567864	0.05657
	1	$b = 5.603$, $k = 1.010$	0.9573196	0.08934
1M LOC				
Function	v	Parameters	Correlation	Residual
S^{I}	1	$\alpha = 0.1711$, $\beta = 1.8385$	0.9843189	0.01826
	1	$\alpha = 0.3837$, $\beta = 1$	0.9865882	0.02605
S^{II}	0.921	$\alpha = 2.545$	0.9758226	0.02917
	0.99	$\alpha = 24.25$	0.9775927	0.03168
$S^{III\text{-}H}$	1	$\phi = 0.07637$, $\gamma = 4.36077$	0.9842629	0.01841
$S^{IV\text{-}H}$,$S^{V\text{-}H}$	1	$\mu/\omega = 0.3004$, $k = 0.4276$	0.9623073	0.04285
$S^{VI\text{-}H}$	0.8223	$\lambda = 0.1192$	0.957021	0.05631
	1	$\lambda = 0.1433$	0.9565412	0.1112
$S^{III\text{-}W}$	0.7976	$b = 4.2684$, $k = 2.0100$	0.9416784	0.1224
	1	$b = 2.616$, $k = 2.010$	0.8960806	0.3081
$S^{V\text{-}W}$	0.9472	$b = 5.2355$, $k = 1.0100$	0.9470132	0.1164
	1	$b = 4.851$, $k = 1.010$	0.9494476	0.1185

Table 4.2: Security Breach Probability Function to EAL comparison for 100 KLOC and 1 MLOC.

the S^{I} function (the 1 MLOC best fit lines for S^{I} and $S^{III\text{-}H}$ have a correlation value of 0.9997226). This indicates that, under these assumptions, SwSec investment exhibits limited convexity.

Functions $S^{IV\text{-}H}$ and $S^{V\text{-}H}$ offer a distinctively different effect: the potential for total elimination of vulnerability [101]. Although these cases differ in assumptions, they are differentiated by the bounds placed on k. As seen in Table 4.1 and Table 4.2, the nonlinear least square approximation on all parameters finds values less than 1 for both the μ/ω and k parameters. In fact, for $k > 1$ the `nls()` function fails to converge. Since k controls

Size	COCOMO Estimate (Person-Months)					Correlation	
	Req.	Des.	Imp.	All	Per KLOC	S^{I}	$S^{III\text{-}H}$
5 KLOC	2.2	3.9	5.5	11.6	2.32	0.9876	0.9555
20KLOC	10.4	17.7	25.2	53.3	2.67	0.9762	0.9679
100 KLOC	61.2	104.3	147.5	313.0	3.13	0.9587	0.9559
1 MLOC	770.4	1312.4	1856.9	2939.7	2.94	0.9534	0.9541

Table 4.3: COCOMO effort estimates per phase and COSECMO correlation for S^{I} and $S^{III\text{-}H}$ security investment estimates.

the convexity ($k < 1$) or concavity ($k > 1$) of the function, it can be concluded that the vulnerability reduction exhibited by the COSECMO model approximation is convexly decreasing (or concave in residual vulnerability, as shown in Figure 4.1). Under $k = 1$, high vulnerability and low investment productivity are again observed (albeit with lower correlation than demonstrated previously).

Finally, function $S^{VI\text{-}H}$ is linearly decreasing with a single parameter, λ. As with $S^{IV\text{-}H}$ and $S^{V\text{-}H}$ this 'hole' can be fully patched, resulting in a residual vulnerability of 0. The low values of λ again evoke low investment productivity, and we see that best-fit is found with a lowered vulnerability (although competitive values are found when $v = 1$). This dataset also exhibits lowered correlation values.

Willemson: $S^{III\text{-}W}$ and $S^{V\text{-}W}$. The Willemson functions are likewise characterised by their ability to eliminate vulnerability. By definition, $S^{III\text{-}W}$is constrained by $k > 2$, while for $S^{V\text{-}W}$ this is relaxed to $k > 1$. Tables 4.1 and 4.2 show that the nonlinear least square fit suppresses k to its lowest possible value in almost every case, while also lowering v. When $v = 1$, b is further reduced, saturating at maximal investment. The resulting correlations are the lowest of all SBPFs studied, indicating this is not a good fit.

The data in Table 4.1 and Table 4.2 demonstrates a relationship between various security breach probability functions and the expenditure required over the SDLC to construct secure software (as modelled by COSECMO). With a relationship between the Gordon and Loeb-defined SBPFs and software development effort established for the overall SDLC, the best-performing functions and parameters (S^{I} and $S^{III\text{-}H}$) are now examined for their explanatory power in phase-specific security investments.

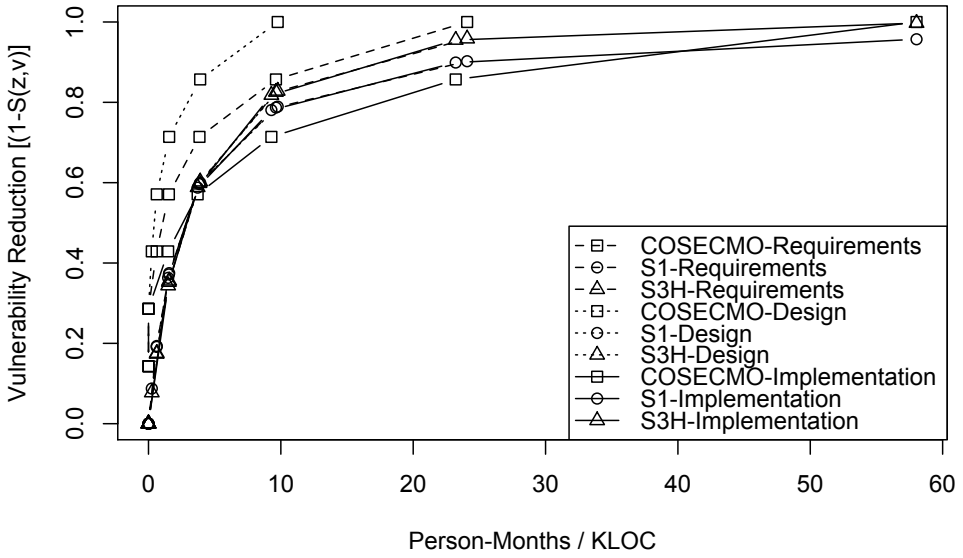


Figure 4.2: Effort investment per KLOC at 1MLOC in the requirements, design, and implementation phases, for the COSECMO, S^I and $S^{III\text{-}H}$ models.

4.2.2.2 Software Phases

Investments into secure software engineering processes are the cornerstone of software security [1], focused in the elaboration and construction phases: requirements (e.g. security requirements), design (e.g. architectural analysis), and implementation (e.g. code review and test). Employing the same relative project sizes used previously (5, 20 and 100 KLOC, and 1 MLOC), estimates of the required base development effort were produced using the COCOMO Suite Tool[6]. The 'COCOMO only' model was employed with default parameters ('nominal' setting), using the 'Source Lines of Code' sizing method. These were aggregated and normalised by size to generate a per-KLOC security effort estimate. These base estimates were employed in the COSECMO `Effort(Int.Assure)` (Equation 4.2), S^I and $S^{III\text{-}H}$ calculations, using the parameters identified in Section 4.2.2.1, to model vulnerability reduction. The estimates and correlation values are shown in Table 4.3, with the results depicted in Figure 4.2 for 1 MLOC.

While the overall findings of Section 4.2.2.1 appear to hold for software phase investment, there are some interesting deviations. The Gordon and Loeb SBPFs under-value smaller investments, with COSECMO more optimistic at lower EALs and S^I providing the least reduction for all security levels at 5 KLOC. This may be due to the structure of COSECMO, which does not differentiate below EAL 3. By 20 KLOC, the Gordon

[6] `http://csse.usc.edu/tools/COCOMOSuite.php`

Tool/Process	TP/FP	Cost/KLOC	Notes
seL4 Security Proof [166]	99%	$78,000–$192,000	
Synopsis Coverity [167]	45%/15%	$5,402.18	TP rate assumed[7]
"Tool A" [133]	37.5%/22.1%	$3,532.44–$7,634.16	Assume no tool costs
"Tool B" [133]	28.6%/5.3%	$907.77–$1,961.83	
"Tool C" [133]	25%/6%	$1,004.85–$2,171.64	
"Tools A, B, C" [134]	0–21%/1–35%	$334.10–$11,833.79	Assume no tool costs

Table 4.4: Reported SwSec cost effectiveness.

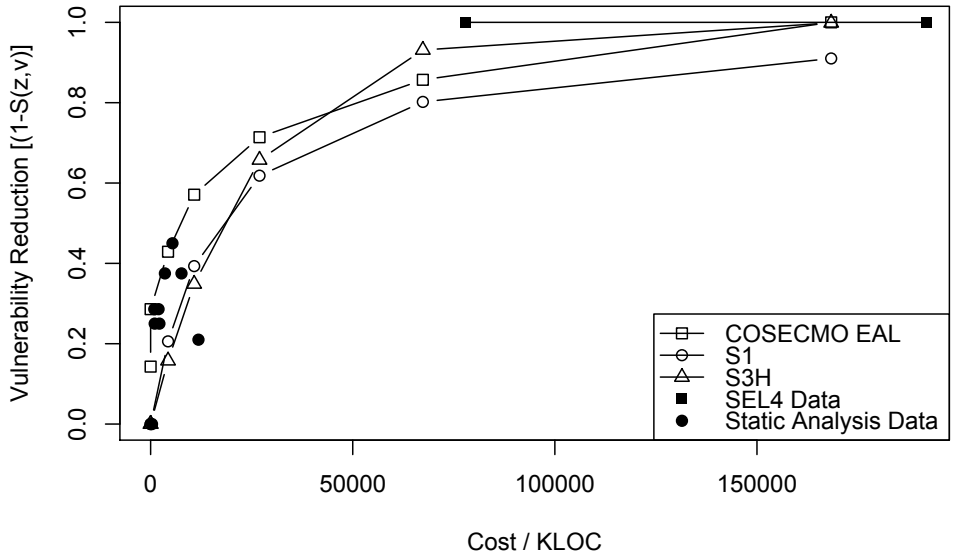


Figure 4.3: SwSec costs per KLOC against S^I and $S^{III\text{-}H}$SBPFs.

and Loeb-defined functions yield higher values for all EALs above 3. At 1 MLOC these functions overtake COSECMO, with $S^{III\text{-}H}$ consistently providing higher vulnerability reduction than S^I; this appears to reflect criticisms of COCOMO over-estimation [165]. Although the EAL mapping employed is a simplification, the consistency and correlation between models (> 0.953) indicate descriptive power. Overall, this suggests that, at least for projects well-modelled by COCOMO, SwSec investment is representable by the Gordon-Loeb model with appropriate parameters.

4.2.2.3 Software Practices

Published evidence suggests the findings in Section 4.2.2.2 hold generally: Edmundson et al.'s [126] investigation of multiple reviewers on the probability of finding vulnerabilities produced findings closely representing the sigmoid function of $S^{III\text{-}H}$ with low b value ([126], Figure 4, page 11)[8]. For further evidence, a variety of industry reports, academic investigations, and technical reviews regarding SwSec investment are examined against Gordon-Loeb model estimates. The results are captured in Table 4.4 and Figure 4.3.

A basic sanity check of the model can be derived by examining extremes of the security development spectrum. First, data from the seL4 project[9] is employed, representing a high-performance, formally verified microkernel project. The seL4 kernel was developed in part to demonstrate the effectiveness of formal development, resulting in the costs for development of code and various proofs tracked [166]. An effort of 4.1 person-years (py) for the approximately 10,000 source lines of code (SLOC) is stated as having a cost of approximately $78 USD per SLOC for the Security Proof (a cost of approximately $190K/py). However, nearly all bugs discovered in the effort were the result of the Correctness Proof, indicating a significant impact of this effort on overall system security. Of the 20 py of effort dedicated to this proof, it is estimated that 9 py were involved with developing the tooling required; furthermore, the remaining 11 py of effort is estimated to be reducible to 6 py re-using the methodology. While this effort is not entirely security-related, it provides a tangible upper-bound. This range in costs is reflected by the filled-in squares in Figure 4.3 (from left to right: Security Proof only, estimated Correctness Proof plus Security Proof).

At the other end of the security scale, a number of publications for data regarding the application of static analysis tools (SATs) were examined. Employment of SATs is cited by numerous secure development models as a recommended practice, with many basic open source tools available. However, the limited true positive rates exhibited by some tools [14; 134], coupled with the potential for high false positive rates [170], place the value of SATs into question. Employing academically published results from [133] and [134], in addition to a commercial tool industry report [167], the cost of execution for a 10 KLOC hypothetical codebase was estimated using the following assumptions:

- An average defect density of 100 per KLOC [15] is assumed, for a total of 1,000

[7] Estimated effectiveness limited to bugs (implementation defects) [14; 168; 169]

[8] Efforts to contact the authors of this report and obtain the underlying data went unanswered.

[9] Secure Embedded Level 4 microkernel: `https://sel4.systems/`

defects. Of those, 2% of total defects are deemed 'security-relevant' [171], for a total of 20 vulnerabilities in the 10 KLOC codebase.

- An average in-phase fix time of 5 hours was employed [167], with all fixes made in-phase. False positives are treated as additional fixes, calculated as a set of defects erroneously deemed security-relevant.

- The same labor rate range as employed in the seL4 analysis was employed ($64,527–$139,453 per person-year[10]). Costs were computed using the COSECMO effort-cost equation [156]:

$$\texttt{Cost} = \texttt{Effort} \cdot \texttt{Labor Rate} \tag{4.7}$$

- The total cost for vulnerability reduction during implementation is represented as:

$$\texttt{Cost}_{\texttt{tool}} + \texttt{Cost}_{\texttt{process}} + \texttt{Cost}_{\texttt{fix}} + \texttt{Cost}_{\texttt{FPs}} \tag{4.8}$$

While the data in Figure 4.3 appears to conform to the established model, this aspect of the analysis should be interpreted with care. Effectiveness of tools and processes has been shown to vary on the mixture of security and SAT experience [172], tool and weakness category [170], and even code complexity [173]. Additionally, it is evident that the different studies cited employed varied criteria for false positive rates, and some may have considered a wide variety of errors in the true positive rate. In the case of seL4, the additional proofs (e.g. binary verification, optimisation) are likely to have also affected the system security. The full, initial correctness proof effort alone resulted in an investment roughly double that deemed efficient by the Gordon-Loeb model (20.5 py, $389,000 per KLOC), which still pales in comparison to the cited industry 'rule-of-thumb' of $1K/LOC ($1,000,000 per KLOC) for lower-assurance EAL6 code using conventional methods [166]. Depending on scope and viewpoint, these findings either confirm or refute the narrative that formal methods are an uneconomical investment [174].

4.2.2.4 Analysis Summary

This analysis has identified a relationship between SwSec for programmes that are well-represented by the COCOMO/COSECMO construct (e.g. software development processes with well-defined phases), and information security economic models employing specific

[10] As obtained from www.salary.com for U.S. national averages, accessed 6 July 2017

breach probability functional forms. Those showing the highest correlation — S^{I} and $S^{III\text{-}H}$ at low values of ϕ and γ — appear to well-model software security investment under the stated cost and effectiveness assumptions. The values in Tables 4.1 and 4.2 serve as a reference when applying SPBF models to various software developments, while the methodology can be employed to identify functions and values corresponding to other software development paradigms (e.g. agile development) — given sufficient data. While the analysis is necessarily a generalisation, it serves as a first to relate information security models to software security investment.

Calibrated modelling provides the decision maker (e.g. the programme manager) with grounded estimates that can be used to direct and substantiate investment decisions. Evidence of the Gordon-Loeb model's potential to represent varied, 'effective' software security investments supports its employment as part of a robust security analysis (further described in Chapter 5). In order to frame such decisions, it is necessary to place SwSec investments in the context of overall lifecycle security expenditures. Armed with the presented analysis, such a model will now be explored.

4.2.3 Extending the Gordon-Loeb Model

The findings of Section 4.1 stand in contrast to the conclusions of Neuhaus and Plattner, who question the applicability of standard cybersecurity economic models to SwSec [128]. However, their findings — along with Tanaka *et al.*'s [132] correlation of S^{II} to email-borne computer viruses — support the intuition that SwSec investments are likely to *exhibit differently* than post-deployment security investments. Representing this reality requires an expansion of the base GL model, considering three distinct points within the software lifecycle: pre-deployment (PRE), at deployment (DEP), and post-deployment (PST).

4.2.3.1 Model

As in the standard model, focus is placed on system vulnerability, denoted v. For simplicity, this is assumed to be a non-overlapping combination of errors in design (flaws), implementation (bugs), and configuration, with the first two constituting the target of SwSec investment. The variable δ captures the component of vulnerability not addressable by SwSec (where $0 \leq \delta \leq v$), such as unchanged defaults and emergent system behaviour. Pre-deployment security investment (z_{PRE}) is then modelled as the revised

vulnerability at the time of deployment given investment z_{PRE}, modulo the introduced misconfiguration error introduced during configuration:

$$\begin{aligned} v_{\text{PRE}} &= v - \delta \\ v_{\text{DEP}} &= S_{\text{PRE}}(z_{\text{PRE}}, v_{\text{PRE}}) + \delta \end{aligned} \tag{4.9}$$

Additional security investment at post-deployment further reduces this vulnerability. Presumably, this follows a different security breach probability function S_{PST}:

$$v_{\text{PST}} = S_{\text{PST}}(z_{\text{PST}}, v_{\text{DEP}}) \tag{4.10}$$

The overall ENBIS at post-deployment of the total security investment is then the application of the Gordon-Loeb model utilising the values derived from periods PRE and DEP:

$$\text{ENBIS}(z_{\text{PRE}}, z_{\text{PST}}) = [v - v_{\text{PST}}] \cdot L - (z_{\text{PRE}} + z_{\text{PST}}) \tag{4.11}$$

While this form supports the potential for an additional investment (and subsequent reduction) at period DEP, only pre-deployment (e.g. SwSec) and post-deployment (e.g. enterprise security) investments are considered. The extended overall model takes the form:

$$\begin{aligned} v_{\text{PRE}} &= v - \delta \\ v_{\text{DEP}} &= S_{\text{PRE}}(z_{\text{PRE}}, v_{\text{PRE}}) + \delta \\ v_{\text{PST}} &= S_{\text{PST}}(z_{\text{PST}}, v_{\text{DEP}}) \\ \text{ENBIS}(z_{\text{PRE}}, z_{\text{PST}}) &= [v - v_{\text{PST}}] \cdot L - (z_{\text{PRE}} + z_{\text{PST}}) \end{aligned} \tag{4.12}$$

The maximisation of this equation is therefore dependent on the form of S_{PRE} and S_{PST}. We refer to the resulting composition as Gordon-Loeb for Secure Software Engineering (GL-SSE).

4.2.3.2 Consistency

In considering such a model, an examination of the model parameters provides an initial approximation of the model validity. Figure 4.4 provides an approximation of the model effects on the ENBIS calculation.

As depicted, the model responds to changes in the parameter space as would be expected (at least in the general case). The overall ENBIS rises sharply, then falls off gradually as z_{PRE} increases. Peak ENBIS and the space of positive pre-deployment investment decrease as vulnerability v, threat t or asset value at risk λ decrease, or the

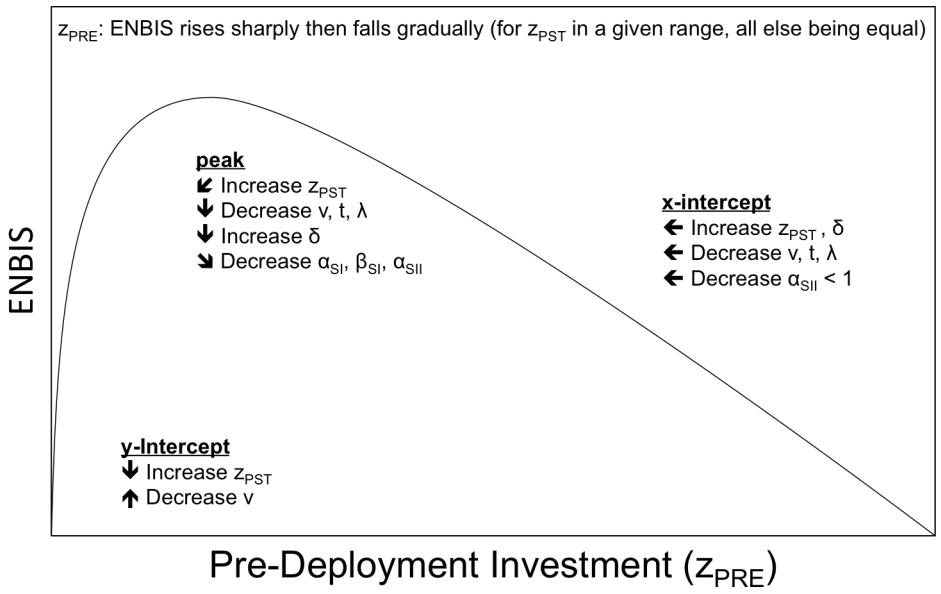


Figure 4.4: Summary of the effect of parameters on the ENBIS calculation for the Gordon-Loeb for Secure Software Engineering (GL-SSE) construction ($S_{\text{PRE}}= S^I$, $S_{\text{PST}}= S^{II}$).

vulnerability attributable to software decreases (δ increases). In addition, the ENBIS peak benefit decreases and moves toward lower z_{PRE} with increases in z_{PST} (indicating a lessened need for pre-deployment investment), with a diminished maximum positive return indicated by a flattened and lowered curve. Alternatively, peak ENBIS lowers and moves to the right as productivity parameters α_{S^I}, β_{S^I}, and $\alpha_{S^{II}}$ decrease — less productive investments yielding a lower return reliant upon greater investment (and in the case of post-deployment productivity $\alpha_{S^{II}} < 1$, decreasing the effective pre-deployment investment range). Finally, reducing post-deployment investment drives the ENBIS at $z_{\text{PRE}} = 0$ toward lowered benefit of minimal pre-deployment investment, while reductions in initial vulnerability increase the net benefit at low pre-deployment investment.

If $v_{\text{DEP}} = 1$ the efficient investment for S^{II} will remain at 0. This supports the narrative of the software security community, who view cybersecurity as risk mitigation that addresses the symptoms but not the problem [175]. In this case, there is no amount of security investment (as modelled by these functions) that compensates for truly flawed software.

4.3 Multi-Period Models

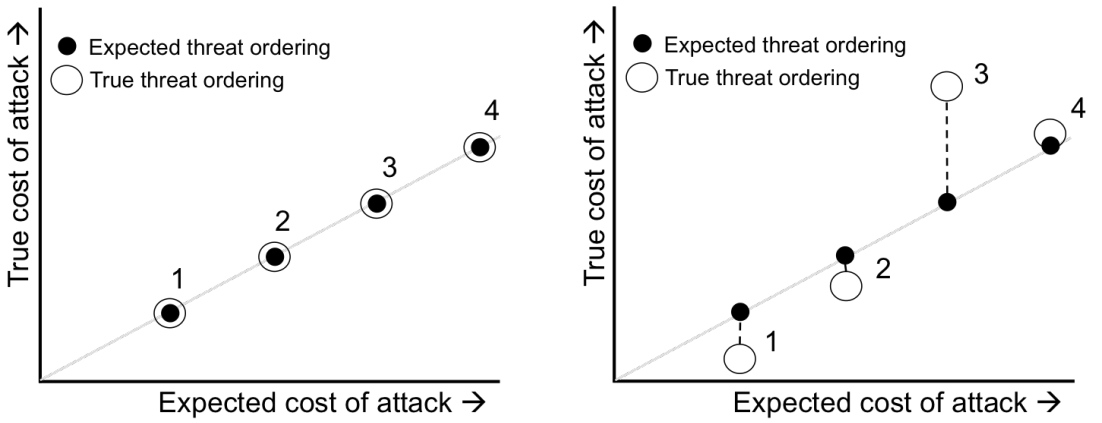
Multi-period models offer another perspective on the challenge of software security investment. Due to their nature, such models offer the ability to identify specific points within the system lifecycle where security investment might occur and understand the impact of specific investments in the context of a potential attack. The Iterated Weakest Link model [112; 176] is a discrete-time information security investment model that incorporates concepts from the game-theoretic weakest link attacker model (discussed in Section 2.2.1), and serves as another basis for examining SwSec investment.

4.3.1 The Iterated Weakest Link Model

The IWL focuses on optimising defender decisions based on starting conditions that lie at the heart of software engineering: quality (in the form of cost to attacker) and uncertainty regarding vulnerabilities. SwSec practices exhibit indications of weakest link behaviour as well, as reflected in the motivation for the IWL: "The most careless programmer in a software firm can introduce a critical vulnerability" [112]. Common practices such as 'top 10 list'–driven reviews and code analysis rulesets evolve with the realisation of 'yesterday's attack'; they are intended to place focus on the most common errors, with increasingly deeper review and analysis driving both security and cost. Central to the IWL is the premise that attacks are "unknowable and hence innumerable in advance" [177]. This leads to the definition of a model that emphasises dynamic, adaptive investments over time. The authors attempt to capture the iterative nature of security investment through a focus on the following three key characteristics:

- The model plays out as an iterative process of attacking and defending successive weakest links.
- Defender uncertainty regarding which components are weakest is a key consideration in investment decisions.
- Countermeasures are represented as interdependent (rendering the diminishing marginal return of information security investment endogenous in this model).

The model demonstrates conditions where under-investment in security can be a rational action, given that: a) reactive investment is possible; b) uncertainty exists about the attacker's relative capability to exploit different threats; c) successful attacks are not



(a) Attack profile with certainty ($\sigma = 0$). (b) Attack profile with uncertainty ($\sigma > 0$).

Figure 4.5: Attack profile under certainty (left) and uncertainty (right).

catastrophic; and d) the sunk cost to upgrade the defence configuration is relatively small [112]. This model is applied to phishing in online crime and payment card security, with subsequent research [177] examining the use of penetration testing as a means to conduct such uncertainty reduction.

The IWL seeks to model the protection of a set of *assets* (a), from which a *return* of r per-period is enjoyed by the defender. This is jeopardised by n possible components threatened by attack (e.g. *attack vectors*), each with an associated cost of attack. In the IWL, each successive threat ($1, \ldots, n$) has an increase in cost of Δx over the previous threat, forming an *attack gradient*. In the original IWL, this was set at $\Delta x = 1$, specifying that each defender investment increased the cost to the attacker linearly.

While the true ordering of n (the actual cost, x) is presumed to be known by the attacker, this information is unknown to the defender who must form an expected ordering of likely attacks (e.g. through attack modelling, as specified in BSIMM's Intelligence domain [4]). The defender's *expected cost ordering* is denoted $\overline{x}$ (where $\overline{x}_1 \leq \overline{x}_i \leq \overline{x}_n$). If the defender's attack modelling is perfect, they will generate an attack profile that matches the attacker's profile; this is the case depicted in Figure 4.5a. In this instance the defender (dots) has absolute certainty regarding the attacks (rings). However, a much more likely scenario is one where the defender's estimate does not fully align with the attacker's profile. Such a case is illustrated in Figure 4.5b where defender uncertainty has led to an incorrect ordering of threats 3 and 4, resulting in misplaced investment by the defender. This potential for a misalignment of threats and defences by the defender is captured by the IWL as σ, specifying the degree of the defender's *uncertainty*.

Defender investment in IWL plays out over discrete time $t = (1, \ldots, t_{\max})$. At each t, the defender forms a defensive configuration represented by a vector $\mathbf{d}_t$. The elements d_i of $\mathbf{d}_t \in \{0,1\}^n$ indicate that defence against the i-th threat is implemented ($d_i = 1$) or not ($d_i = 0$). The summation of these defences is represented by k, guarded at a unit cost (1) per protection, per round.

The knowledge employed by the defender to make investment decisions is rooted in the expected costs for the i-th threat, represented as:

$$\bar{x}_i = \bar{x}_1 + (i-1) \cdot \Delta x$$

with

$$\Delta x > 0 \tag{4.13}$$

The unknown true costs are modelled as a Gaussian random variable $\mathcal{N}$, with mean x_i and standard deviation $\sigma/\Delta x$ (censored to values $x_i > 0$):

$$x_i = \sup(0, \chi_i)$$

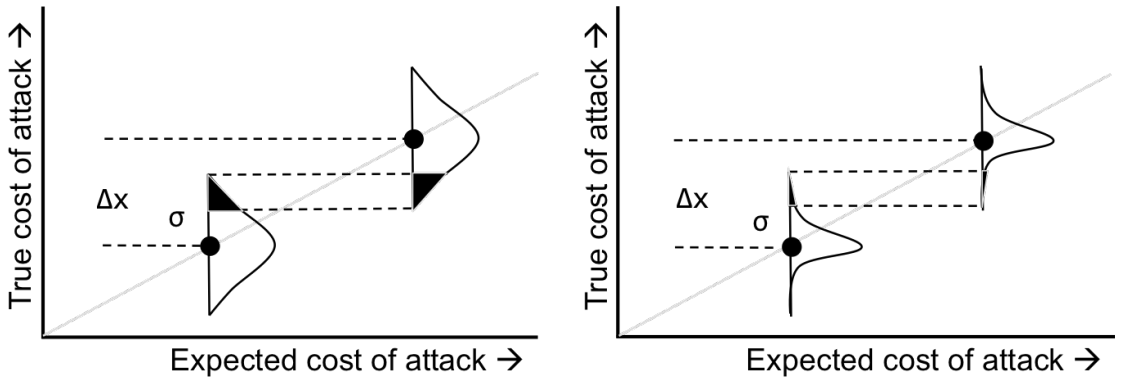
with

$$\chi_i \sim \mathcal{N}(\bar{x}_i, \sigma/\Delta x) \tag{4.14}$$

The role of σ in the IWL is visualised in Figure 4.6: a higher value of σ leads to a wider distribution, increasing the probability that the defender's ordering is incorrect and results in misplaced security investment (Figure 4.6a). However, with lower uncertainty (i.e. a smaller standard deviation) this probability is decreased (Figure 4.6b).

The value of the attack gradient Δx also contributes to the overall security investment decision: the larger Δx, the less overlap in expected threat costs for a given σ. This is depicted in Figures 4.5 and 4.6 as the slope of the line on which the actual attack costs lie relative to the distribution of defender expectations. Therefore, the effect of these parameters on the security investment is best thought of as a ratio, $\sigma/\Delta x$ (representing an increase in attacker cost for a given amount of uncertainty). This distinction is made as an extension of the exposition in [112], with the original IWL employing a unit cost for Δx rendering this distinction moot ($\sigma/\Delta x = \sigma/1 = \sigma$).

The model operation starts with the defender specifying an initial k at $t = 0$. In each subsequent round ($t = 1, \ldots, t_{\max}$) the attacker may choose to exploit the component with the least true cost not covered by $\mathbf{d}_t$, looting a fraction z from asset a — but only if



(a) Defender attack ordering with high uncertainty, potentially leading to incorrect ordering of attack vectors.

(b) Defender attack ordering with low uncertainty, reducing the potential for incorrect attack vector ordering.

Figure 4.6: The relationship between uncertainty σ and attack gradient Δx in the establishment of defender attack order.

the benefits exceed the cost to attack. In the face of uncertainty regarding the economic viability of the attacker exploiting the next weakest link, the defender may carry out the following actions:

- Ignore the attack and absorb the potential losses. The model does not consider public costs, so such a private loss can be seen as rational when the loss is less than the security investment demands.
- Disinvest from the enterprise, which occurs when the defender's position becomes non-viable due to costs and uncertainty.
- Specify a new defensive configuration $\mathbf{d}_t$ as each part of the true ordering is revealed by the weakest link. This incurs a loss of the current defence sunk cost, λ.

The process then repeats at each step with the attacker decision. A secure state is reached when the remaining vectors below the reservation cost of attack are protected.

4.3.2 Extending the Iterated Weakest Link Model

The secure software engineering variant of the IWL (referred to as IWL-SSE) is concerned with the genesis of the vulnerability set addressed by IWL, as well as the defender's uncertainty regarding this set. Pre-deployment practices to address vulnerabilities (in the form of flaws and bugs) are considered relative to the cost and value of flaw and

bug identification prior to system integration and deployment, i.e. within the Acquisition / Development phase of the SDLC. These security practices are characterised in the context of the SSDL-Touchpoints domain of the BSIMM: Architecture Analysis (AA), Code Review (CR), and Security Testing (ST). Defender investment removes potential vulnerabilities at a greatly discounted rate (in comparison to the costs involved at later stages), reducing the defender's uncertainty (σ) while raising the initial security condition of the software itself (represented by the attack gradient, Δx). The overall goal of the model is to minimise costs over the entirety of the system's lifecycle.

The original IWL-SSE model, introduced in [27], employs a number of assumptions regarding the software project under consideration in order to control for variability in the development process. Specific assumptions include:

1. Standalone, greenfield development, where all security investment decisions are left to the programme manager and the implications are confined to the individual system. This removes any concern over interfaces, integration, or existing frameworks altering investment decisions.

2. Sequential development lifecycle with distinguishable phases, where the development process is well-defined, controlled, and guided by a programme manager. This is easily thought of as a waterfall or incremental model, with defined Architecture/Design and Code/Test phases (and extendable to other development approaches, as discussed in Chapter 7).

3. The developer is involved in the fielding and operation of the development, or information relative to development decisions is reliably and truthfully conveyed. The conveyance of information is directly related to the uncertainty in the process artefacts upon which security decisions are based — a key element to this model.

4. Once a flaw or bug is found, it is fixed; the fix enacted is correct; and repeated iterations of SwSec activities occur with the same effectiveness each time. It should be noted that there are indications that the potential for mistakes when correcting bugs can be significant [178], and the negative impact of delays, errors and consistency when dealing with vulnerabilities will negatively impact the outcome.

The model follows the strategy employed by [177] of assuming a risk neutral actor, with no sunk costs ($\lambda = 0$) or interdependency of defences ($\rho = 0$).

4.3.2.1 Model

Consider a software project managed by a system developer, whose goal is the most efficient investment of a security budget. Each unit of potential investment can be seen as providing two benefits. The first is a reduction in economically viable vulnerabilities — vulnerabilities for which the attacker sees a return on an investment of resources. This is accomplished either through remediation (e.g. deployment of a defence) or removal (e.g. a correction in implementation) of vulnerabilities, and is generally driven by a risk-based attacker model (such as in SSDL-Touchpoints [1]). The second is a reduction in uncertainty regarding the identification and ordering of vulnerabilities, under the assumption that security expertise is more adept at the former than at the latter [112]. Under the IWL model the attacker is always concentrated on the economically most viable vulnerability, requiring that a determination on the extent and relative weakness of vulnerabilities is known with as little uncertainty as possible in order to properly place defensive resources. This uncertainty is tightly coupled to the cost of successive attacks, and so must be considered relative to the attack varying gradient Δx ($\sigma/\Delta x$).

The secure software engineering process offers an investment opportunity for the defender via processes such as architectural and design reviews, code reviews, static and dynamic code analysis, and risk-driven testing. As such, it is complementary to the current secure software engineering literature, which places a heavy emphasis on vulnerability reduction when considering return on security investment. While this is an essential result of a secure software process, the complementary benefit — uncertainty reduction and raising the bar to attack — is just as vital to warding off security over-investment.

4.3.2.2 Investment

IWL-SSE unfolds in discrete time, using negative time slices to denote pre-IWL steps for consistency of representation. A high-level depiction of the overall process is shown in Figure 4.7.

Investment by the defender proceeds according to two phases: Architecture and Design ($t = -2$, denoted AD), and Implementation and Test ($t = -1$, denoted IT). This investment occurs through the choice of the number of review or test iterations (i), which succeed with a probability α (finding a flaw via review) or β (finding a bug or flaw via test), respectively. Therefore, the overall investment by phase is defined as

$$I_{\{\mathrm{AD,IT}\}} = (i \cdot c) + i \cdot (\mathit{eff} \cdot e)$$

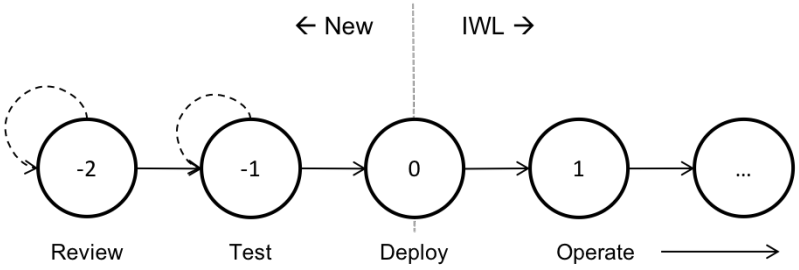


Figure 4.7: High level depiction of the overall integrated process, with the introduction of additional process investment steps ($t = -2, -1$) that complement the system-level security investment ($t = 1, \ldots, t_{\max}$). This is anchored by the deployment point ($t = 0$), whereby the values set by the software process are fed into the system level model.

where

$$\begin{aligned} i \in & \{i_{\mathrm{AD}}, i_{\mathrm{IT}}\} \\ c \in & \{c_{\mathrm{AD}}, c_{\mathrm{IT}}\} \\ \mathit{eff} \in & \{\alpha, \beta\} \end{aligned} \tag{4.15}$$

Here, I represents the cost of the secure software engineering activity conducted at time t for a given set of iterations within that activity, i. The value *eff* is the effectiveness of the security activity; in this case, α for reviews and β for tests. In this context, 'effectiveness' refers generally to the benefit derived from execution of the process. This is related, but not equivalent, to the effectiveness of a particular tool or activity (e.g. static analysis software or formal specification reviews) and cost c invested. Finally, e is the cost of conducting the identified fixes; for this model, it is assumed that all identified flaws or bugs are fixed. This parameter deserves further explanation, as it varies between phases.

The field of software engineering has long been concerned with the costs of fixes in different phases. Surveys such as [22], along with work by those such as Boehm [21; 179], have empirically shown that costs escalate by roughly a factor of 10 with each phase of development as the project moves from requirements, to architecture and design, to implementation, and finally to operations. This is reflected by setting the in-phase cost of a fix as 0.01 (one one-hundredth of the cost of fixing a flaw in operations), consistent with the software engineering literature and the IWL unit definition for cost of defence. An order of magnitude increase is exacted in each future phase for the cost of flaws (c_f) and bugs (c_b), starting from the phase they are introduced — at $t = -2$ for flaws, and $t = -1$ for bugs. Using the rule of thumb that vulnerabilities are 50% flaws and 50%

bugs, investment in previous process phases is rewarded by a reduction in the proportion of higher cost fixes to in-phase fixes within the current phase. This reduction should be related to the number of iterations undertaken in the previous round, reflecting current SwSec thinking towards such investments [8; 90]. This function should therefore exhibit convex behaviour, reflecting a decreasing return on security investment that is asymptotic at zero — under the assumption that both categories of error will never be fully removed. The resulting definition for e in each phase is then:

$$\begin{aligned} t = -2: &\quad e_{\mathrm{fAD}} &&= 0.01 \\ t = -1: &\quad \tfrac{e_{\mathrm{flT}}}{2^{\alpha i_{\mathrm{AD}}}} + e_{\mathrm{bIT}} &&= \tfrac{0.1}{2^{\alpha i_{\mathrm{AD}}}} + 0.01 \end{aligned} \tag{4.16}$$

Combining equations 4.15 and 4.16 yields the following definitions for cost at Architecture and Design ($t = -2$, AD) and Implementation and Test ($t = -1$, IT):

$$\begin{aligned} t = -2: &\quad I_{\mathrm{AD}} = &&(i_{\mathrm{AD}} \cdot c_{\mathrm{AD}}) + i_{\mathrm{AD}} \cdot (\alpha \cdot e_{\mathrm{fAD}}) \\ t = -1: &\quad I_{\mathrm{IT}} = &&(i_{\mathrm{IT}} \cdot c_{\mathrm{IT}}) + i_{\mathrm{IT}} \cdot \left(\beta \cdot \left[\tfrac{e_{\mathrm{flT}}}{2^{\alpha i_{\mathrm{AD}}}} + e_{\mathrm{bIT}}\right]\right) \end{aligned} \tag{4.17}$$

Here, c_{AD} and c_{IT} represent the cost of conducting a review or test, respectively. The overall cost for the software process I_{P} is then simply the linear combination of the phase costs:

$$I_{\mathrm{P}} = I_{\mathrm{AD}} + I_{\mathrm{IT}} \tag{4.18}$$

4.3.2.3 Payoff

The benefits to the defender that fuel investment into these secure software engineering stages is twofold. Pursuant to the prevailing wisdom of the information security economics literature [75; 180] these benefits operate under similar assumptions and functional representation as those identified in Section 4.2.1.

The first benefit is the reduction in the overall uncertainty faced by the defender, relative to the overall amount invested in the respective phases. System vulnerability uncertainty σ is defined as the equal combination of the uncertainty resulting from each phase of the software process. For the purposes of this model this is represented as an equal amount of uncertainty regarding both bugs and flaws, with equal weight given to each:

$$\sigma = \sigma_{\mathrm{AD}} + \sigma_{\mathrm{IT}} \tag{4.19}$$

To determine the uncertainty reduction provided by the process elements per phase, a model that correlates the number of iterations with the effectiveness per iteration is required. Assuming independent and equally effective iterations, this should reflect a reduction in uncertainty that is exponentially decreasing asymptotically to 0 (just as one will never reach full certainty in practice, one will never reach fully impervious software). Additionally, neglecting to undertake any process elements within a given phase should not result in any uncertainty reduction. This is modelled as follows:

$$\sigma_t = \frac{\sigma_{max}}{2} \cdot \mathit{eff}^{\frac{1}{i}}$$

with

$$\begin{aligned} \mathit{eff} \in &\ \{\alpha, \beta\} \\ i \in &\ \{i_{\mathrm{AD}}, i_{\mathrm{IT}}\} \\ t \in &\ \{\mathrm{AD}, \mathrm{IT}\} \end{aligned} \tag{4.20}$$

Here, σ_{max} refers to the starting level of uncertainty, while *eff* and i correspond to the values relative to phases $t = (-2, -1)$ for the effectiveness and iteration counts, respectively.

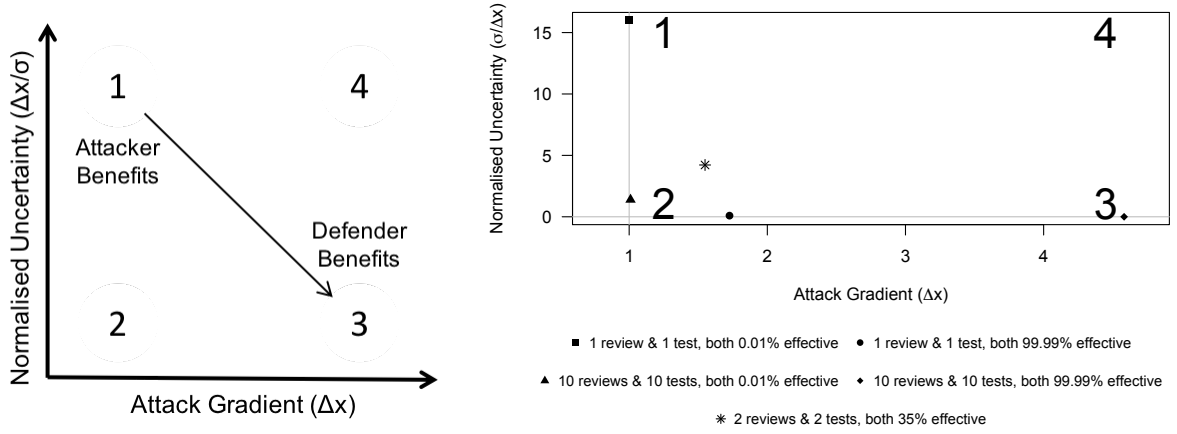
A second SwSec payoff is an increase of the gradient of attack, Δx. This can be expected to increase with diminishing returns, exhibiting sub-linear growth and concavity. In addition, it must also reflect the unit definition for defensive costs when no investment into process is made (reducing to the IWL definition). This is captured as:

$$\Delta x = \sqrt{(1 + \alpha i_{\mathrm{AD}} + \beta i_{\mathrm{IT}})} \tag{4.21}$$

4.3.2.4 Consistency

The operation of the model can be considered within the trade-space depicted in Figure 4.8a, with four differentiating points readily identifiable (specified on the graph by positions 1 to 4):

1. The situation where a number of viable vulnerabilities are present, and the system developer has a great deal of uncertainty regarding the exposure resulting from these vulnerabilities (low Δx, high σ). This point arguably represents the most likely result of a modern software development effort that lacks investment into software security processes.



(a) (b)

Figure 4.8: Investment in secure software relative to the number of economically viable vulnerabilities (vulnerabilities for which an adversary attack results in a return after their costs), as defined by gradient of increasing attack costs (Δx) and the uncertainty regarding the nature of the vulnerabilities present (σ). Figure 4.8a shows the goal of secure software development investment in the movement toward the lower right of the graph: fewer economically viable vulnerabilities with less uncertainty. Figure 4.8b depicts IWL-SSE outputs for various parameter choices.

2. The situation in which the system contains a number of economically viable vulnerabilities, with the system developer well aware of their existence, relative exposure, and rank order (low Δx, low σ). This corresponds to some combination of inadequate process and a lack of effectiveness in any process undertaken; in the worst case this is the 'snake-oil salesman' of commercial software security [153].

3. The ideal situation where there are few (if any) economically viable vulnerabilities in the system, and there is relative certainty regarding this fact (high Δx, low σ). This can be considered the goal of secure software engineering investment.

4. The situation where there are few economically viable vulnerabilities, but this fact is not known to the system developer with any certainty (high Δx, high σ). Such a point could be a very 'lucky' development result, or perhaps a very good process leading to 'hardened' software (i.e. a high actual initial attack cost x_1) with no means to measure the result (e.g. no process artefacts, or a lack of metrics). In either case, this state is likely to result in security over-investment, as the system developer seeks assurance that the resulting system is secure.

This depiction is useful in conceptualising relevant security outcomes to result from investment in the software process. As an initial validation of this model and its faithfulness to secure software development principles, the IWL-SSE was exercised along the following parameters:

- The number of iterations of each process step (review and design), ranging from 0–25 each.
- The effectiveness of each phase α and β, in the interval $[0, 1]$.

Absent SwSec process, it is assumed that there is a high degree of uncertainty regarding vulnerabilities (σ) and a low attack gradient (Δx). Therefore an initial attack gradient $\Delta x = 1$ and uncertainty $\sigma = 16$ was employed for this examination, consistent with [112]. Solid lines on the graph represent the minimum attack gradient (i.e. the increase in attack costs without any process investment), and the asymptotic point of absolute certainty, $\sigma = 0$. Figure 4.8b depicts combinations of input parameters to IWL-SSE under the following conditions:

1. Region 1 is the result of a failure in the investment of process, either through a lack of iterations (low i_{AD} and/or i_{IT}), or through a lack of effectiveness in the process (α and β at or near 0%). Coupled with a high cost of review or test, this becomes the worst-case scenario of investment for no gain in protection or understanding (2 iterations, 0.01% effective, depicted by ■).
2. Region 2 is occupied by two scenarios: few iterations with high effectiveness (2 iterations, 99.99% effective, depicted by •) and many iterations with low effectiveness (10 iterations, 0.01% effective, depicted by ▲). These points both demonstrate an undesirable (but not catastrophic) state.
3. Region 3 is approached through continued investment in the software engineering process, coupled with high effectiveness (driving down σ, while driving up Δx): the lower the effectiveness, the more iterations required to reach this state. Investment into more effective measures, predictably, results in the need for fewer iterations to reach the same return (10 iterations, 99.99% effective, depicted by ♦).
4. Conceptually, Region 4 represents the lucky situation where the defender has a high attack gradient (i.e. there are few economically viable attacks), yet there is a high degree of uncertainty in this. Therefore, this value is driven by the initial Δx — as

this line shifts right, the starting point for attack is increased and the points at (1) and (2) shift to the right.

5. The final point (4 iterations each, both 35% effective, depicted by $*$) is intended to show a 'realistic' set of parameters, indicative of common development practice.

This particular form provides the general functional form desired by this model: modest return in process quickly provides returns with respect to reducing the uncertainty of the software's state, with diminishing returns — especially at low rates of effectiveness. It also reflects the need for significant, repeated investment in order to drive up the base level of economically viable attacks. More expressive modelling of the relationship between uncertainty and attack costs is left to future work, as discussed in Chapter 7.

4.4 Analysis

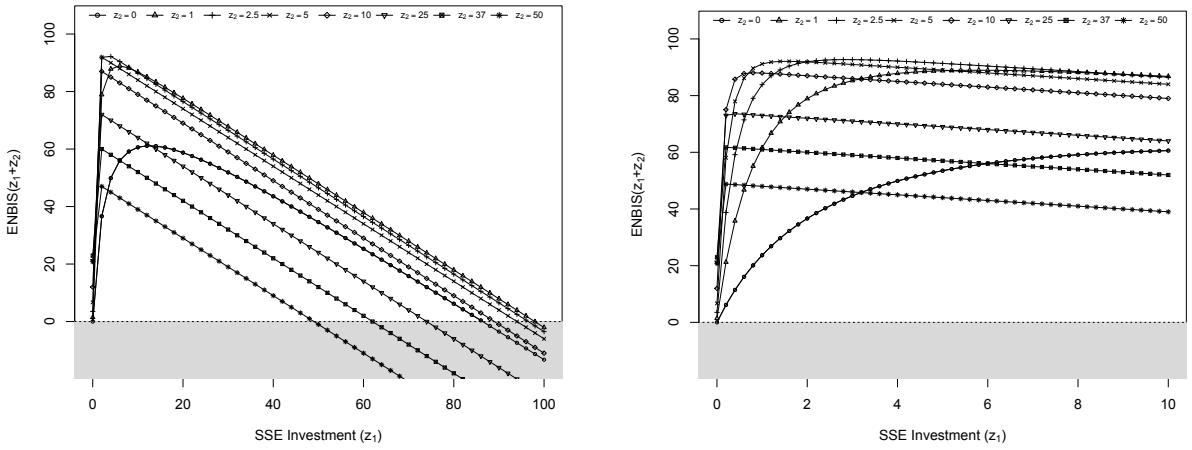
The benefit of secure software engineering investment on the overall SDLC is now examined through the application of these models. This is accomplished using econometric indicators, as introduced in Chapter 2 and employed in Chapter 3. Focus is placed on insight into the distribution of security investment between pre- and post-deployment measures.

4.4.1 Analysis of GL-SSE

These constructions permit those making security investments a straightforward comparison between potential solutions in order to find the optimal investment strategy for a given set of initial conditions. While disagreement exists regarding the validity of annualised security benefit metrics, these calculations have emerged as a means for evaluating and justifying security expenses within an organisation [88]. Such metrics have the benefit of wide acceptance among managers and accountants, permitting security engineers and project owners to present a case for security expenditure in a non-technical, business-accessible fashion.

4.4.1.1 ENBIS in GL-SSE

The GL-SSE construct lends itself to analysis using a variety of information security economic metrics, which are often defined over a single period. Comparing Annual Loss



(a) ENBIS for GL-SSE (Overall) (b) ENBIS for GL-SSE (at low z_1)

Figure 4.9: Expected Net Benefit of Information Security (ENBIS) in the GL-SSE model ($\lambda = 100$, $t = 1$, $v = 0.99$, $\delta = 0.1$, $S_{\text{PRE}} = S^I$ ($\alpha = 0.3837$, $\beta = 1$), $S_{\text{PST}} = S^{II}(\alpha = 2.5)$).

Expectancy (ALE) [88] under different investments provides the Expected Net Benefit of Information Security (ENBIS), as defined in Section 3.2.3.1:

$$\text{ENBIS} = \text{ALE}_0 - \text{ALE}_1 - \text{average security investment} \tag{4.22}$$

Here, ALE_0 is the expected loss without security investment, and ALE_1 the expected loss with security investment. The parameter 'average security investment' captures the security investment; for GL-SSE, this is the combined pre- and post-deployment investment $(z_1 + z_2)$.

Figure 4.9 illustrates ENBIS for a notional security investment scenario with a potential loss $\lambda = 100$, threat $t = 1$, and high starting vulnerability $v = 0.99$ commensurate with software development. Various levels of post-deployment investment $z_{\text{PST}} \in \{0, 1, 2.5, 5, 10, 25, 37, 50\}$ are depicted, with increasing pre-deployment investment along the x-axis $z_{\text{PRE}} \in [0, \lambda]$. The function S^I ($\alpha = 0.3837$, $\beta = 1$) is employed for S_{PRE}, corresponding to the identified best-fit values in Table 4.2 for 1 MLOC. For post-deployment, $S^{II}(\alpha = 2.5)$ is employed for S_{PST} to reflect the (tenuously) identified post-deployment function in [132]. Consistent with [181], 10% of the vulnerability is attributed to configuration issues ($\delta = 0.1$).

Figure 4.9a provides an overall view, while Figure 4.9b expands the leftmost portion of the graph for better viewing at low values of z_1. Easily identifiable is the characteristic

S^I ENBIS curve at $z_2 = 0$, with subsequent post-deployment investment driving the ENBIS value higher at lower pre-deployment investment — to a point. At $z_2 > 2.5$ ENBIS values are decreasing from their highest point with increasing investment. Various effects depicted in Figure 4.4 can be identified, with a few items worth noting:

- *Failure to invest in pre-deployment security results in low ENBIS*, even for significant investments in post-deployment security. This finding is consistent with current security thought and observation.
- The highest overall ENBIS occurs with *moderate pre- and post-deployment investments*, in the range of $(0, 5]\%$ of the potential loss. This supports the assertion that software security is especially important in less secure deployment environments. Positive ENBIS exists over a broad range of pre-deployment investment when complemented by post-deployment investment. However, this range is diminished with too little (or too much) investment following deployment.

For the scenario depicted, the values $z_1 = 2.8$, $z_2 = 2.5$ represent the greatest return at 92.712. While this would appear to indicate a significant benefit, this value will vary according to the parameters for any given scenario.

4.4.1.2 ROSI in GL-SSE

While ENBIS is useful when pre- and post-deployment investments are centrally controlled, this may often not be the case (as discussed in Chapter 5). With information security reliant upon accreditation and compliance processes for the foreseeable future, the security environment (and subsequent expenditure) is often established independently of the system under development. In these cases, it may more critical to establish the range of beneficial pre-deployment investment in light of a fixed, limited budget and foregone post-deployment expenditure.

One tool commonly applied to analyse information security investment is the Return on Security Investment (ROSI), which has multiple equivalent definitions. Most commonly, ROSI is formulated in terms of ALE, as described in Section 3.2.3.1:

$$\text{ROSI} = \frac{\text{ALE}_0 - \text{ALE}_1 - \text{average security investment}}{\text{average security investment}} \tag{4.23}$$

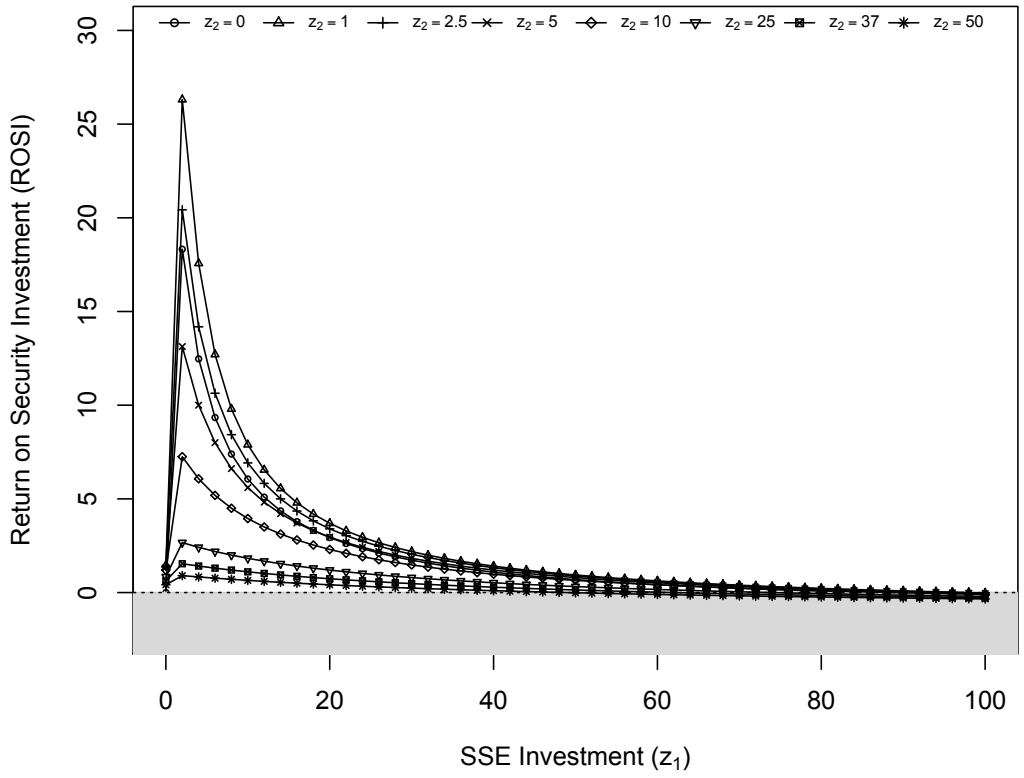


Figure 4.10: Return on Security Investment (ROSI) in the GL-SSE model ($\lambda = 100$, $t = 1$, $v = 0.99$, $\delta = 0.1$, $S_{\text{PRE}} = S^I$ ($\alpha = 0.3837$, $\beta = 1$), $S_{\text{PST}} = S^{II}(\alpha = 2.5)$).

Substituting Equation 4.22 for the numerator yields an alternative construction useful for the analysis of GL-SSE (which is defined in terms of ENBIS):

$$\text{ROSI} = \frac{\text{ENBIS}}{\text{average security investment}} \tag{4.24}$$

Figure 4.10 depicts ROSI in the GL-SSE model, employing the same parameters as used in Figure 4.9. From this graph, a few observations can be made:

- Unsurprisingly, *the greatest ROSI is in the initial investment*, supporting the narrative that the most impactful security dollar is the first. This behaviour follows from the diminishing returns assumption of Gordon and Loeb, and is demonstrated in the respective SBPF curves. However, peak ROSI does not correspond to the lowest v,

underscoring the need to balance returns and reduction in vulnerability. The long tails point to a wide range of potentially beneficial investment, to be adjudicated against the security needs of the system.

- As before, *positive values exist for a broad range of software security investment* when complemented by post-deployment investment, spanning a large portion of the z_{PRE} range. While in part an artefact of the high threat ($t = 1$) and high initial vulnerability ($v = 0.99$) values, this result indicates that combined pre- and post-deployment investments may rationally exceed the upper bound for information security investment as defined by Gordon and Loeb in [98].

Rather than a retort, this model expands on Gordon-Loeb model to consider two aspects previously unsupported by the original model: first, a multi-staged treatment of vulnerability that supports introduction following initial investment; second, a multiple investment scenario modelled by separate functions. Such a construct more accurately and comprehensively captures the nuanced nature of information security investment over the system lifecycle, reflecting current information security realities.

4.4.2 Analysis of IWL-SSE

Analysis of models such as IWL-SSE is less straightforward than that of models such as GL-SSE, with direct application of standard econometrics rendered difficult due to their construction. Investments impact the nature of the calculation in each period, altering the value of parameters within each round. In addition, traditional econometrics fail to capture non-monetary benefits, such as a reduction in post-deployment security outlay. As a result, an uninformed analysis can yield ill-defined results (e.g. a more secure outcome with lower expenditure, resulting in an undefined ROSI). These issues are discussed in [177], where they were partially overcome by calculating metrics against a constant parameter (e.g. uncertainty).

4.4.2.1 Benefits in IWL-SSE

In examining IWL-SSE, it is informative to consider benefits beyond those conveyed by traditional econometrics. Table 4.5 provides a comparison of model parameters under the same security scenario as presented in [112]. A starting uncertainty of $\sigma = 16$ was utilised (the maximum used in the IWL), along with a starting attack gradient $\Delta x = 1$ (which is fixed in the original IWL). Effectiveness values of $\alpha = 60\%$ and $\beta = 30\%$ were employed

Reviews and Tests	σ	σ_{max}	Δx	k	Process Cost	Return
0 reviews, 0 tests	0	0	1	11	0	33.50
0 reviews, 0 tests	16	16	1	0	0	25.13
1 review, 1 test	8.8	16	1.38	5	4.03	34.50
5 reviews, 1 test	6.38	16	2.07	4	16.04	40.57
1 review, 5 tests	4.91	16	1.76	4	8.12	39.39
8 reviews, 24 tests	0.89	16	3.61	3	48.15	44.64
25 reviews, 25 tests	0.54	16	4.85	3	100.23	42.56

Table 4.5: Pre-deployment investment benefits as in the IWL-SSE model ($a = 1000,\ r = 5\%,\ z = 2.5\%,\ x_1 = 15,\ n = t_{\max} = 25$).

to approximately correspond to reported effectiveness of architectural reviews [179] and singular static analysis tools [133]. However, as noted in Section 4.3.2.4, the relationship between the effectiveness of the SwSec activity represented by the model and the measures of effectiveness reported in the literature for specific practices is complex; these values merely provide a reasonable starting point for analysis of the model's operation.

From Table 4.5, a number of projected benefits arising from SwSec security investment can be identified:

- An *increase in the attack gradient* Δx, resulting in fewer economically viable attacks on the system.
- A *decrease in the uncertainty* σ, providing better insight into the nature of vulnerabilities within the system.
- As a result of these changes, the optimal *initial defence* k *decreases* (from 11 to 3 for these parameters). Less security expenditure is required in subsequent periods, increasing returns for the defender.
- Importantly, this reduction in initial defence occurs while *returns increase.* A reduction in defences that requires extensive post-deployment investment would result in increased expenditures and more attacks in the long run (as happens when $\sigma = 16$).

4.4.2.2 Returns in IWL-SSE

To more fully illustrate the benefits of pre-deployment security investment when compared to the IWL alone, Figure 4.11 depicts the investment scenarios identified in Table 4.5. The unfilled boxes represent the case of no process: with high uncertainty (crossed circles),

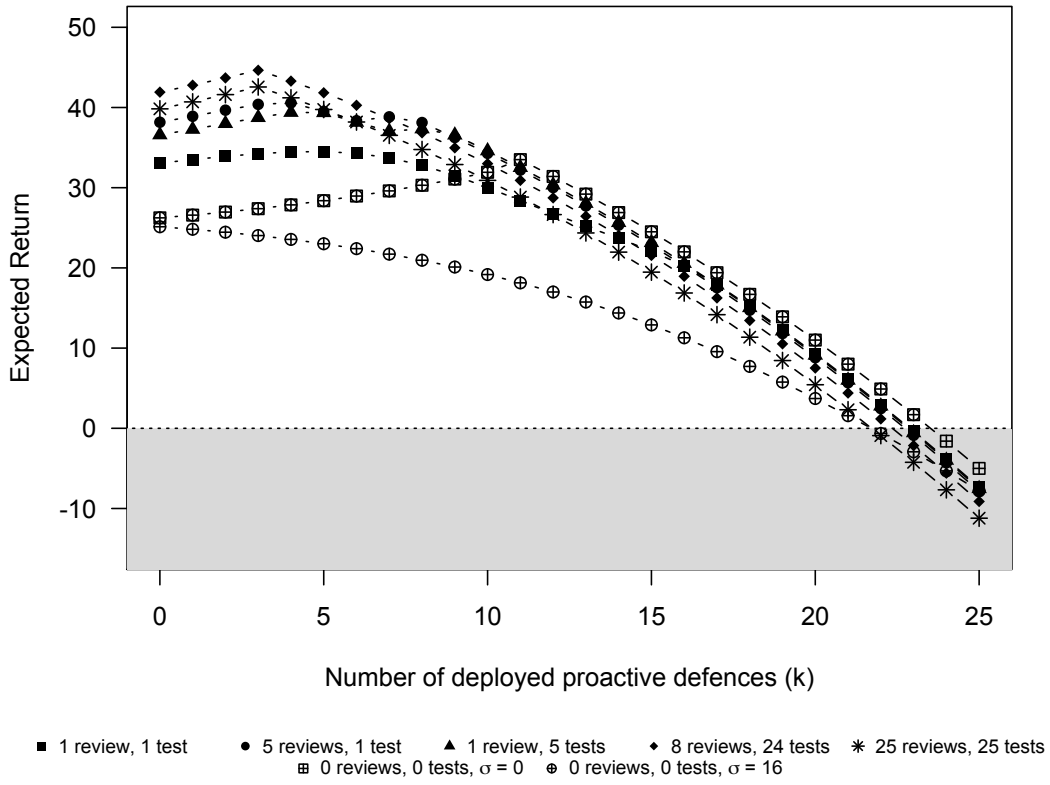


Figure 4.11: The per-period expected return in the IWL-SSE model ($a = 1000,\ r = 5\%,\ z = 2.5\%,\ x_1 = 15,\ n = t_{\max} = 25$).

and with absolute certainty (crossed squares). The latter serves as the 'best case' scenario without pre-deployment investment. For this particular scenario:

- *The overall return has increased in all instances employing secure software process.* This indicates that the investment toward reducing vulnerabilities through software process has resulted in a more favourable position for the defender, who is able to retain a greater portion of the value of the asset (a).

- *The optimal number of proactive post-deployment defences has decreased in all instances* employing secure software process. This would imply that investment into SSE process reduces the burden on proactive post-deployment defence. Such a result undoubtedly contributes to the increased return demonstrated, as the one-time

costs borne prior to deployment do not impart recurring costs.

These effects are attributable to the removal of 'weaker links' prior to deployment, which has increased the value of Δx. Importantly this is a reduction, rather than removal, of proactive post-deployment defences (from $k = 11$ in the optimal case of IWL under complete certainty, to $k = 3$ in the case of IWL-SSE with 8 reviews and 24 tests). Figure 4.11 also highlights bounds on the return that SSE can provide, with the highest point (for this set of parameters) falling below the maximum number of reviews and tests. This would indicate a 'diminishing returns' scenario that requires well-constructed process investments, rather than perpetual and unbounded expenditure. Both of these findings coincide with accepted security thinking regarding the role of SwSec, and further reinforces the lack of a 'silver bullet' in information security necessitating broad and balanced approaches [8].

4.4.3 Return on Secure Software Process

The acceptance that metrics such as ROSI have experienced has led to adaptations intended to describe a number of specific security investments, notably the Return on Penetration Testing (ROPT) metric employed in [177]. Following this convention, a new metric is derived: the *Return on Secure Software Process (ROSSP)*. Utilising standard econometrics, this can be defined as:

$$\text{ROSSP} = \text{ROSI}_{\text{SSE}} - \text{ROSI}_{\text{NoSSE}} \tag{4.25}$$

Here, ROSI_{SSE} represents the return realised after pre-deployment investment, while $\text{ROSI}_{\text{NoSSE}}$ is the return without such investment. Although defined as a comparison of opposing states (with and without secure software process), this measure can also be applied to the examination of pre-deployment security investment alternatives more broadly. The comparison of various security investment levels or the employment of other measures of return are equally valid applications of this concept.

4.4.3.1 ROSSP in GL-SSE

An example of ROSSP calculated using GL-SSE where $S_{\text{PRE}} = S^I$ and $S_{\text{PST}} = S^{II}$is provided in Figure 4.12. Employing values of $z_{\text{PRE}} = 10$ and $z_{\text{PST}} = 25$, a representative

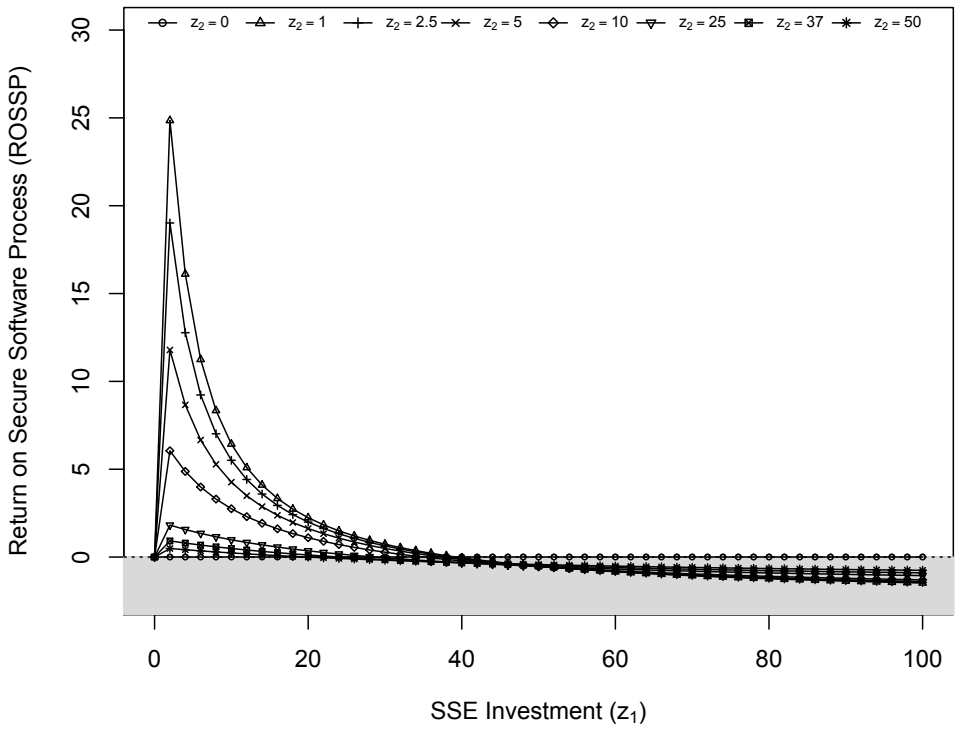


Figure 4.12: The Return on Secure Software Process (ROSSP) in the GL-SSE model ($\lambda = 100$, $t = 1$, $v = 0.99$, $\delta = 0.1$, $S_{\text{PRE}} = S^I$ ($\alpha = 0.3837$, $\beta = 1$), $S_{\text{PST}} = S^{II}(\alpha = 2.5)$).

ROSSP can be calculated as:

$$\begin{aligned} \text{ROSSP} &= \text{ROSI}(10, 25) - \text{ROSI}(0, 25) \\ \text{ROSSP} &= 27.286 - 17.470 \\ &= 9.816 \end{aligned} \tag{4.26}$$

Examining the ROSSP in GL-SSE clarifies the contribution of pre-deployment investment to overall return on security investment. It also highlights the bounds on positive return: larger investments in effective post-deployment security reduce pre-deployment benefit, which is limited to a small range (for this example, below approximately 4% of the loss).

The relative effectiveness of security measures alters the range of ROSSP; for example, setting $\alpha = 1$ for S_{PST} results in positive ROSSP beyond 8% for $z_{\text{PST}} = 25$. In addition, higher initial vulnerability scores increase the productive range for z_{PRE} and further the

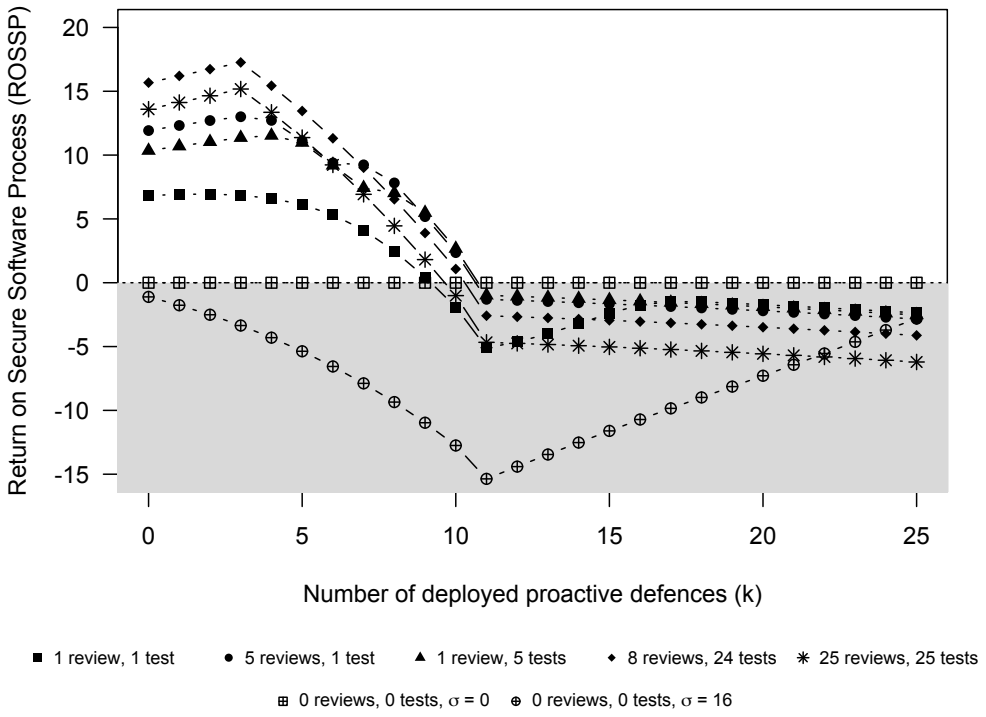


Figure 4.13: The Return on Secure Software Process (ROSSP) in the IWL-SSE model ($a = 1000,\ r = 5\%,\ z = 2.5\%,\ x_1 = 15,\ n = t_{\max} = 25$).

assertion that SwSec is essential in vulnerability-prone systems.

4.4.3.2 ROSSP in IWL-SSE

While informative to pre-deployment security process planning, ROSSP proves most beneficial when applied to models such as IWL-SSE. Just as the difficulties of applying traditional econometrics to the IWL Penetration Testing (IWL-PT) model prompted the development of the Return on Penetration Test (ROPT) metric in [177], ROSSP provides a mechanism to isolate the benefit of SwSec and other pre-deployment investments.

Figure 4.13 presents a ROSSP comparison between the per-period returns for the previous best-case scenario ($\sigma = 0$) and identified secure software process scenarios, with all other values held the same as per Figure 4.11. As before, the unfilled boxes represent the case of no process: with high uncertainty (crossed circles), and with absolute certainty

(crossed squares). The latter serves as the 'best case' scenario without pre-deployment investment, and serves as the baseline for comparison. Under this security scenario, at the optimal point of investment the ROSSP calculation between the IWL (with absolute certainty) and the identified best-case IWL-SSE scenario of 8 reviews and 24 tests is:

$$\text{ROSSP} \quad = 44.6 - 33.5 = 11.1 \tag{4.27}$$

Graphical results for ROSSP in IWL-SSE across a variety of scenarios are depicted in Figure 4.13. This model, and the associated ROSSP measure, is inherently conservative in that it assumes a single system deployment. Multiple deployed instances of the same software could result in economies of scale resulting from SwSec investment.

Proper use of such calculations can aid the project owner in evaluating alternative development approaches, and in justifying the process investment required for secure software engineering. The comparative nature of the ROSSP metric isolates the pertinent investment (e.g. SwSec) without the need to re-cast values into traditional econometrics. The example demonstrates that, while significant returns are possible, SwSec does not have a 'free pass' for 'software security at all costs'. The diminishing ROSSP of the 25 review and 25 test values (asterisk line in Figure 4.13) shows that, like any other security investment, software security must be weighed against resource commitment. While specific maxima will vary based on the costs and effectiveness of particular SwSec approaches, undoubtedly the benefits wane as it becomes harder — and therefore more costly — to address the next weakest link through process alone. Where BSIMM and others have sought to identify *what* must occur to improve software security, this contribution is the first to attempt to address *how much* is reasonable. The answer to both these questions is crucial if such practices are to be adopted and rise to the importance their advocates feel is necessary to exact real change in security.

4.5 Summary

This chapter has explored two approaches to modelling secure software practices (more generally, any pre-deployment security) as an information security investment. The goal of this exercise is to better understand the role such processes play in the overall problems current information security practices face. Only by considering investments in the context of the entire system lifecycle can such investments be balanced against the competing requirements and constraints of engineering secure systems: functionality, security, and

resources must each be addressed in the development of complex, resilient systems that can operate in the modern inter-connected technology environment.

While these models were discussed as continuous (single-period) and discrete time (multi-period) approaches, they could have easily been divided in other ways: models that include implicit versus explicit threat constructs; models focused on process versus monetary investment; or models of probabilistic versus deterministic attack. The presented research represents only an initial foray into the problem of lifecycle security research management. Further extension of existing models — or construction of software security centric models from scratch — represent viable paths to more accurate, encompassing, or situationally diverse means to address emerging security needs. This chapter has attempted to establish the viability of modelling to the understanding of security. While stylised, the results are supportive of current beliefs: investments in software security offer the ability to improve the efficiency of security, offering higher benefit (i.e. return for unit investment) than post-deployment security alone. This benefit is captured by the introduced ROSSP metric.

However, it is important to note that *most efficient* investment does not equate to *most secure* investment for a particular scenario — especially as the modelling approach remains abstract from the scenario it emulates. In addition to exploration of assumptions underlying the models (e.g. risk appetite, process approach, or security capacity), a fundamental principle has emerged from this research regarding the interpretation of outcomes. Metrics such as ENBIS may tempt a manager to invest only where it is most efficient; however, there is no guarantee that such investments will produce the desired security outcomes — only the most benefit per unit investment. Security must still be weighed against competing system constraints, and the investment itself allocated to activities that achieve security needs. Ideal employment of these models may not lie in the identification of highest efficiency, but in the identification of net positive outcomes for given security goals and functional requirements. This allows managers to explore technical aspects of security and find better solutions in light of managerial constraints: resources, capability, and process.

Ultimately, the utility of such metrics and models is in their employment as guiding forces, informing software development programme stakeholders and enabling rational decision making based upon sound reasoning and explicit assumptions. The next chapter will examine how to integrate economic concerns into the secure software development lifecycle in order to achieve "adequate security" as defined for a given system.

Chapter 5

Application to Secure Development

This whole economic boom in cybersecurity seems largely to be a consequence of poor engineering.

— Carl Landwehr

Once the allocation of a security budget has been established, there remains the challenge of mapping expenditure to effective action. As discussed in Chapter 2, SwSec comprises a number of practices, captured in different methodologies intended to achieve vulnerability reduction. As in the case of post-deployment security compliance described in Chapter 3, expenditure absent an understanding of effectiveness, cost drivers and knock-on effects has the potential to result in economically irrational behaviour. Where post-deployment security expenditure has been considered through the lens of economics, this has accentuated the need for models that capture SwSec investment (such as those described in Chapter 4). These challenges suggest a need to place economic considerations within the context of security-focused systems and software engineering.

To meet these challenges, this chapter explores the application of economic principles to decision making within secure software engineering activities. The following sections present approaches to economically-driven security engineering, summarising the findings of [29; 31; 30]. This research proposes techniques for translating the outcome of software security investment models into a set of security objectives and requirements, leading to designs, implementations and operational systems that reflect the complexities of engineering secure systems under resource constraints.

While a complete treatment of economics within the SDLC is out of scope for this dissertation, this contribution is presented within the system security engineering (SSE)

constructs for security objectives and requirements as identified within the National Institute of Standards and Technology (NIST) Special Publication (SP) 800-160, "Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems", volume 1 [7][1]. Following a discussion of this guidance and its relationship to SwSec, the remainder of this chapter is focused on the intersection of economic reasoning and engineering practice. The chapter closes with considerations for future directions.

5.1 Secure Systems Engineering

Published in 2016, SP 800–160 is the product of a five-year collaboration between NIST, the US National Security Agency (NSA) and industry partners to provide comprehensive, foundational guidance that addresses the growing gap between SSE and our ability to understand and manage the complex — and increasingly contested — environment in which systems operate [182]. As a compendium to International Standard ISO/IEC/IEEE 15288, "Systems and Software Engineering — System Life Cycle Processes" [183], SP 800–160 seeks to provide flexible, high-level guidance for applying the 111 system security engineering activities and 428 tasks within the 30 systems engineering life-cycle processes specified in ISO/IEC/IEEE 15288 [182]. As such, SP 800–160 does not seek to provide authoritative statements regarding the nature of SSE or mandate processes and activities, but instead provides a framework to reason about, structure, and conduct SSE. Two key aspects of this framework include the roles supported by SSE, and the contexts in which SSE decisions are made.

5.1.1 Security Engineering Roles

Modern software development teams are comprised of various stakeholders, each influenced by technical and non-technical factors [115]. As discussed in [46], security processes require involvement from those best positioned to contribute, to include: customers (asset values), project managers (cost of development), and users (functionality). This raises an important question when considering methods for decision support: whose decision is being supported?

[1] In March 2018, this publication underwent minor updates ahead of the publication of volume 2 of the series, "Systems Security Engineering: Cyber Resiliency Considerations for the Engineering of Trustworthy Secure Systems" (currently in draft). See `https://www.nist.gov/news-events/news/2018/01/update-nist-special-publication-800-160-systems-security-engineering`.

While many team compositions have been proposed, one common aspect of security-focused engineering processes is the existence of a person (e.g. a Security Advisor [43]) or team (such as the Software Security Group, or SSG [4]) tasked with leading the charge for security practices. This role may be undertaken as a key role to the larger engineering effort, contributing to SSE practices and activities in order to achieve programme security goals. NIST SP 800–160 Appendix E defines the roles, responsibilities, and skills required of a system security engineer, citing business, managerial, and technical stakeholders as benefactors of security insight [7]. Central to this role are analyses that support suitability, risk management, and cost-effectiveness claims of the security solution. This role may be accomplished in a leading or supporting position, often in conjunction with traditional functional roles (such as those defined in the Microsoft Team Model[2]).

For the remainder of this chapter, each concept will be presented from the point of view of a system security engineer charged with supporting security decision-making in collaboration with other project roles (e.g. business manager, project manager, or developer).

5.1.2 Security Engineering Contexts

The Systems Security Engineering Framework forms the central proposition for this ratiocinating, defining three contexts within which system engineering activities are conducted: the *Problem Context*, the *Solution Context*, and the *Trustworthiness Context* [184]. These contexts integrate technical and non-technical activities to develop a "coherent, well-formed, evidence-based case" that articulates various security considerations [7]:

1. The **Problem Context** captures the stakeholder definition of "an acceptably and adequately secure system", as defined by needs, concerns, and constraints placed on the system. Aspects covered include the lifecycle, security objectives, security requirements, and measures of success.

2. The **Solution Context** transforms the stakeholder security definition into the requirements, design, and realisation of the system. This includes establishment of the security aspects of the system, achieving those aspects and providing the necessary supporting evidence.

[2] https://msdn.microsoft.com/en-us/library/aa266916(v=vs.60).aspx

3. The **Trustworthiness Context** is the evidence-based affirmation that the security definition has been met, involving the development, maintenance, and demonstration of the assurance case.

These contexts are supported by *system security analyses*, generated by the system security engineering to provide supporting data and technical interpretations to the process of defining security requirements, architecture, and design. Analyses are differentiated for application within each context, relative to the scope and objectives of the engineering process as well as attributes such as security performance, behaviour, risk, uncertainty, and cost (among other considerations). Underlying this framework is the organising principle of *context-sensitive security* in which the contextual establishment of asset value drives the application of adaptable and tailorable processes to SSE [7].

The identification of analyses that support the balance of SSE processes, activities and tasks, inform security trade-offs and resource allocations, and enable successful SSE execution is an open problem [182]. The economics of SwSec supplies context-sensitive security analysis inside a consistent basis of comparison. The following sections describe three economically-inspired security engineering concepts, each supporting an "integrated, holistic security perspective across all stages" of the SDLC [7].

5.2 Establishing Security Objectives

In eschewing the notion of 'absolute security' for one that is contextually based, SP 800–160 establishes the principles, constraints, and objectives of system security engineering. The definition of security objectives features prominently within the Problem Context, requiring the engineer to "establish and scope what it means to be *adequately secure* in terms of protection against asset loss and the consequences of such asset loss" [7]. The ISO/IEC/IEEE 15288 Technical Process and Technical Management Process activities feature prominently, as does the notion of identifying and defining security aspects (as part of preparation, definition, and planning activities[3]). Where ISO/IEC/IEEE 15288 activities supply direction, in the fashion of SP 800–160 the means for defining and planning 'adequate security' is left to the engineer.

[3]Activities XX-1, where XX represents the 15288 process designator.

5.2.1 Planning

Technical Management Processes, such as *Project Planning* (PL), *Project Assessment and Control* (PA), and *Decision Management* (DM), specify tasks necessary to resource planning for adequate security: Defining the security-related costs for the project and plan the budget informed by those projected costs (PL-2.3), Assess the adequacy and availability of resources allocated to the security aspects of the project (PA-2.5), and Evaluate each alternative against the security evaluation criteria (DM-2.4), for example. Such planning and assessment forms the core of previous chapters, with Chapter 3 considering security expenditure confined to post-deployment security while Chapter 4 explores the need to conduct such analysis prior to the definition of engineering artefacts. Through these activities the systems security engineer supports decision making by business manager (for setting budgets) and project manager (for establishing the portion of the budget dedicated to security).

In particular, Chapter 4 focuses on the establishment of security objectives through the representation and rational analysis of pre- and post-deployment security investments in the context of the overall programme resource allocation. Under an assumption that economic efficiency is a security objective, the GL-SSE (Section 4.2.3) and IWL-SSE (Section 4.3.2) models permit the analysis of resource allocation to the engineering process under varied assumptions. In conjunction with the base models (the IWL and Gordon-Loeb), such constructs support the comparison of SwSec investment under varied attacker assumptions, differing investment horizons, and the manager's understanding of process effectiveness, cost, and time. Ultimately, the evaluation of several models in concert to define an overall security strategy is likely the best approach [96] — with rational investment demanding better definition and orchestration of models to paint a broader picture inclusive of SwSec. This expressiveness not only supports the Technical Management Process activities identified above, but supplies a necessary stakeholder constraint on the Technical Process activities prominent in the Solution Context. An example of such an analysis is presented in Chapter 6.

5.2.2 Defining

The identification of security investment is critical to the definition of security in context, but is not the only useful analysis of resources. Technical Processes, such as *Business or Mission Analysis* (BA) and *Stakeholder Needs and Requirements Definition* (SN), call

for the identification of stakeholder protection needs (SN-2.5) and the analysis of security objectives in light of limitations, constraints, and other relevant security considerations (BA-4.1). These activities allow the system security engineer inform the negotiation between stakeholders regarding the business, managerial and technical security goals of the system.

Economically-driven analyses have proven useful as a means to reason about context-sensitive security goals. Alternative security goals focused on optimising defender investment against adversary expenditure, such as the notion of an 'attacker economist'[4], can aid in the identification of asymmetries and can inform notions like cyber resilience. Another context-sensitive conceptualisation of security that has benefited from an economic treatment is that of deterrence.

5.2.3 Example: Deterrence

Traditionally confined to notions of national security (e.g. the Cold War tenet of Mutually Assured Destruction (MAD)), research on deterrence in cyberspace has primarily focused on strategic treatments (e.g. [185]). However, an alternative approach considers deterrence at the strategic, operational and tactical levels as a reduction of the net expected utility of a particular means (e.g. attack) for a group to achieve their objective against another group [186]. Employing this more nuanced view, [29] focuses on deterrence in cyberspace at the operational and tactical levels using an approach rooted in security as an information asymmetry [9].

Expressed as screening and signalling games, this research identifies the conditions necessary to deter "low-skill and low-focus" attacks, employing commodity tools against "thousands of networks world-wide"[5]. The approach was first demonstrated as a screening game representing the actions of spammers (as described in [187]), in which a defender takes action(s) to deter further attacks by attempting to shift adversary beliefs. An example reconceptualising the findings of [187] illustrates the concept. The same premise could be applied to more complex scenarios involving multiple rounds with both positive and negative observables, providing a basis for characterising deterrence against activities such as attack reconnaissance: port probing (reporting open ports or services running on those ports), DNS response information (leading the attacker to believe the system is

[4]As defined by researcher Daniel Bernstein in `http://ecrypt-eu.blogspot.de/2015/11/break-dozen-secret-keys-get-million.html`.

[5]As described in Bruce Schneier's December 2014 blog post, "Lessons from the Sony Hack": `http://www.schneier.com/blog/archives/2014/12/lessons_from_th_4.html`.

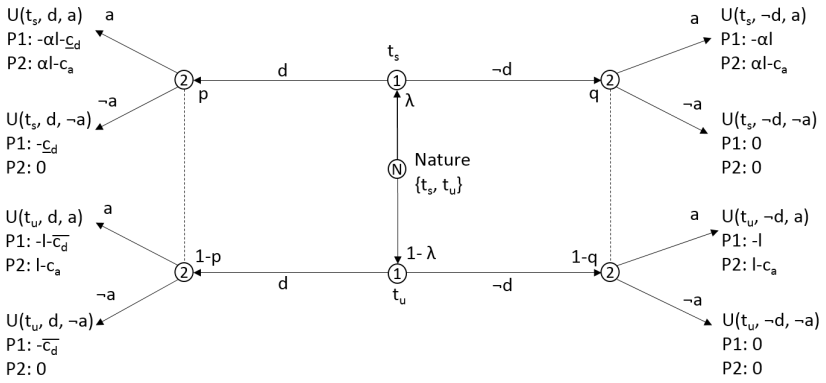


Figure 5.1: Extensive form of the deterrence signalling game.

up to date or of a specific type), or system fingerprinting (reporting specific patch levels, installed applications, etc.).

In a second conceptualisation, a signalling game is employed to identify conditions related to deterrence of wireless network attacks. This is depicted in Figure 5.1. Based upon the concept of actions signalling a particular security level (secure or unsecure), the game proceeds as follows:

1. Player 'Nature' moves, allocating the distribution of the types of users. As the real distribution of secure versus unsecure users is scenario-specific, this is represented as the probability λ of being secure (t_s), and a corresponding probability of $1 - \lambda$ of being unsecure (t_u).

2. Player 1 then chooses to send (d) or not to send ($\neg d$) a 'message' — that is, chooses to deter or not — with the former action implying a cost that differs between types of user. Thus, the action costs secure users a low amount $\underline{c_d}$, while unsecure users will incur a higher cost of $\overline{c_d}$. In this model, messages have no meaningful effect on security; the question to be addressed is whether they can nevertheless deter attacks.

3. Player 2 observes the message (deterrent) and subsequently chooses to attack or not attack, a or $\neg a$. Attacking incurs a cost of attack, c_a. Attacking a user of type t_u is assumed to succeed, resulting in a gain of l (Player 1's loss); whereas attacking a user of type t_s is assumed to succeed only with some small probability α, resulting in a gain of αl. At any point the attacker chooses not to attack ($\neg a$), the resulting cost is 0.

Under the assumptions of a significant cost difference between the cost to the secure user ($\underline{c_d}$) versus the cost to the unsecure user ($\overline{c_d}$) to send this signal (i.e., $\overline{c_d} \gg \underline{c_d}$) and Player 2's inability to discern the type of the user, the best that Player 2 can do is to form a set of beliefs as to which type of agent (t_s or t_u) they are playing. This is represented by the value p, corresponding to the belief probability that a message d (the deterrent action) corresponds to a user of type t_s, and the corresponding belief probability $1-p$ indicates a player of type t_u. The belief probability q (and $1-q$) serves the same function for $\neg d$.

Analysis of this game finds multiple equilibria; of particular interest is a separating equilibrium (where the deterrence action perfectly separates the types of users), found when $\frac{c_d}{\alpha} < l < \overline{c_d}$, with

$$(P1_s(d), P1_u(\neg d), P2_d(\neg a), P2_{\neg d}(a), p = 1, q = 0).$$

This equilibrium signifies a benefit for defenders to deter, conveying a belief that the user's type is t_s[6]. Within the example of wireless network attack security, this may correspond to the employment of easily-overcome security measures (e.g. SSID hiding or MAC filtering). While the existence of such measures is unlikely to thwart a determined attacker, their mere presence might prove a deterrent to the low-focus, low-skill economist attacker.

While simplified, this model demonstrates how a security objective such as deterrence (as previously defined) can be examined through the analysis of economic conditions. In this model $c_a \geq \alpha l$ must hold such that the expected result of attacking a secure player is less than the cost to attack; this is in line with accepted notions of security. More notably, the deterrent must meet the requirement that $\frac{c_d}{\alpha} < l < \overline{c_d}$, so that the cost of deterring for an unsecure user is higher than the expected loss. In addition, effective deterrence outcomes rely on the attacker incurring a sufficient cost $c_a \geq \alpha l$, as with a small c_a the attacker becomes indifferent to various plays (since they incur little or no cost). These conditions can be explored within the context of the game to identify boundary conditions; for example, an increase in attacks that may result from convergence in deterrence costs ($\overline{c_d} \to \underline{c_d}$) or a reduction in costs to the attacker ($c_a \to 0$). More broadly, these results may also inform the relationship between security and deterrence in general. In relying on the existence of both secure and unsecure participants these findings do not constitute an argument for deterrence to replace security, but instead demonstrate how deterrence may

[6]Note that for a user of type t_u playing d is off the equilibrium path, and so no information can be ascertained. In fact, due to this equilibrium, such a move is likely to swing the belief of the attacker towards inferring that the user is of type t_s and refrain from attack, thereby providing a type t_u player the best outcome.

contribute to an overall security strategy — at least for filtering the 'background noise' of low-focus, low-skill attacks. These findings are closely related to the views of [117], which focused on the role of defender investments in changing adversary behaviour outside of the construct of deterrence.

Fulfilling the goal of holistic systems security engineering driven by a closed loop feedback among and between the framework contexts and security analyses, as outlined in NIST SP 800–160, requires techniques such as these to convey and analyse these conditions relative to system concerns within the Solution Context. The next section presents an approach by which economically defined security objectives can be employed to define and realise a security solution.

5.3 Security Requirements

The specification of security requirements marks a shift from the security engineer's focus on business and management decisions toward supporting the technical decisions of the development team. This process originates within the Problem Context with the definition of the stakeholder security requirements for the overall system. These requirements are then "transformed" into Solution Context system design requirements, addressing "all security architecture, design, and related aspects necessary to realise that a system satisfies those requirements" as well as supplying "sufficient evidence to demonstrate that those requirements have been satisfied" [7]. This dichotomy is reflected in the respective ISO/IEC/IEEE 15288 activities, which are separated into *Stakeholder Needs and Requirements Definition* (SN) and *System Requirements Definition* (SR). A natural question arises as to the means and methods by which this transformation, and the necessary security analysis, occurs. Addressing this gap requires the incorporation of security-specific activities relative to the approach identified within the Security Lifecycle Concept. It is here that the intersection between the duo of NIST SP 800-160–ISO/IEC/IEEE 15288 and the greater security community is made apparent, with applicable contributory processes generally chosen from the collection of SwSec practices and SDLs (such as those discussed in Chapter 2) to meet the activities prescribed by the framework.

However, there is no guarantee that contributory processes exist, or are applicable within the system security framing. There is a recognised difficulty in defining the goals, expectations and objectives of security [55] that is tied to this lack of a unified definition for security requirements, or even how concrete and verifiable they should be [188]. It

is very rare for organisations to provide developers with requirements that guide them down the path towards secure software, leading to increased potential for developers being "set up for failure" [61]. Of particular concern is the practice of specifying a solution in the requirements phase [189], often in the form of a specific technology [190], as this undermines the engineering process.

One approach to meeting this challenge is the specification of security requirements as constraints on functional concerns [189], either as requirements in their own right or (more likely) in the vein of 'fit criteria' [191][7]. The latter necessitates that security requirements are measurable and testable — an explicit goal of the Problem Context, but challenging absent standard measures of security [41]. Economically-derived security objectives provide a means by which such requirements can be conveyed and analysed. This can be illustrated using a common mechanism for defining security requirements, in the form of negative use cases.

5.3.1 Example: Requirements Conveyance

Threat modelling has become a critical element within secure software engineering practice, reflected by the emergence of *negative use cases* (in the form of misuse and abuse cases) as a common practice among security professionals in order to elicit security requirements [188]. 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. With cited benefits that include early security focus [192], requirements traceability [193] and promotion of customer awareness [192], negative use cases are widely seen as a software security 'best practice' [194].

A basis for incorporating economic considerations within this practice is rooted in the defining, original works. Sindre and Opdahl propose renaming the standard use case field *Stakeholders and Interests* as *Stakeholders and Risks* in their original paper on misuse cases [193], in order to quantify costs and likelihoods "with more ambition" than allowed by textual descriptions. In related work, the duo cite the integration of misuse case analysis with risk analysis, costing and formal methods as a potentially interesting direction to pursue [195]. 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

[7] As employed in the Atlantic Systems Guild Volerè Process; see `http://www.volere.co.uk/`.

with their definition of abuse cases, key elements of an abuse case are the "resources, skills and objectives" [196] — the first two (arguably) being 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 malactor effort [197]. Unfortunately, the authors do not discuss how this would be done, leaving the method of ranking, matching, and estimation to the reader.

Building on these initial ideas through extensions to the negative use case construct provides a means for economic concerns to be directly incorporated into the software process. The addition of quantifiable utility relationships provides a means to convey and analyse attacker and defender decisions and constraints, as identified within the Problem Context, throughout the SDLC. In addition to supporting a richer basis for analysis, the expressiveness provided addresses specific criticisms and deficiencies of negative use case practice in general.

5.3.1.1 Economic Misuse Cases

In [31], the following augmentations to textual misuse case description in [193; 2] are proposed as a means of addressing these challenges.

1. *Template Separation.* As in [2], specification of misusers (malactors) in a separate template — along with the additional step of generalising malactors — provides an approach to abstracting system details from those arising in the application environment. Abstracted misuse cases are then linked to specific actors through the *Potential Misuser Profile.*

2. *Utility Incorporation.* Minor changes and extensions to current template descriptions facilitate the expression of attributes as economic utility functions.

    a) *Potential Misuser Profile.* Generalised misuser templates permit the misuser profile to be specified in richer, quantifiable ways. Examples of a quantified misuser attribute might include: likelihood of attack (p_a), probability of success given an attack (p_s)[8], or attack costs $a \in \{a_1, a_2 \dots a_x\}$. Using even simple

[8] As in Chapter 3 these two events are considered independently: an attack on the system, p_a, and the state of the system against attack, p_s. This allows 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.

constructs provides the system designer the ability to specify constructions mapped to existing data sources on attacks, and explicitly bound qualitative statements often employed in attack models. One example might be to specify a probability of loss (what is commonly called 'likelihood' or 'attack' in other models) as $p_{s|a} = p_s \cdot p_a$. The result is a set of implicitly created secondary classes of misusers, each with explicit requirements regarding the expectations placed on the system security — 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 for system-level risk assessments.

b) *Stakeholders and Threats.* This field can be expanded to supply similar utilities for the consequence portion of the calculation. While likely to be best conceptualised in monetary terms, this may also take the form of any other factor the designer deems important: website accessibility/connectivity (in the case of an attack on availability) or data record loss (in the case of an attack on integrity). Expressions relating terms such as the cost of data loss (d) relative to the liability of the company in the event of data loss (l) can be developed, leading to relationships that can drive investment decisions (e.g. $0 < d < l$).

c) *Mitigation Points.* Capturing the estimated resource investment (i.e. costs) permits the designer to examine the projected effects of the investment on other parameters in light of the cost of achieving a particular mitigation approach. 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 as discussed in Section 5.2.1.

3. *Cohesion.* To provide cohesion, the 'Scope', 'Abstraction' and 'Precision Level' entries of the template 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.

5.3.1.2 Illustration

To illustrate these concepts, an example based upon a subset of the running example employed by Sindre *et al.* in a series of papers [193; 195; 2; 192] will be utilised. Figure 5.2

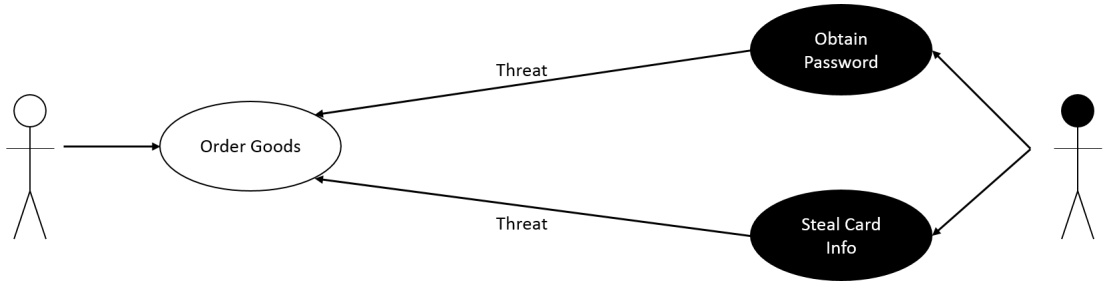


Figure 5.2: Base use and misuse cases for this example. The construction is a subset of the example used by Sindre, Opdahl, and Brevik [2].

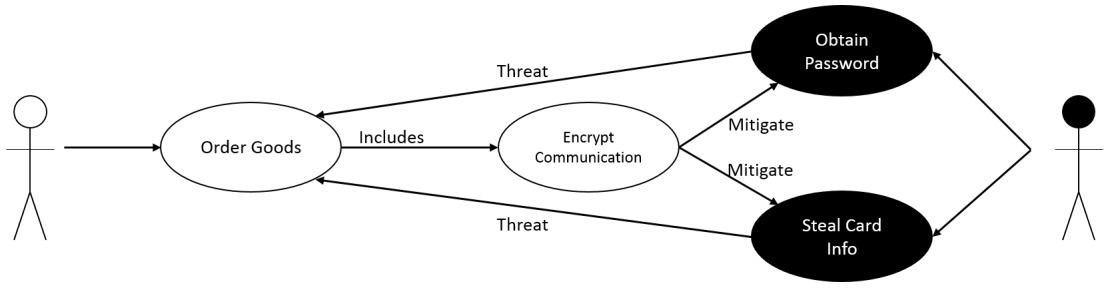


Figure 5.3: Updated use and misuse cases for this example. The construction is a subset of the example used by Sindre, Opdahl, and Brevik [2].

illustrates the use case ('Order Goods'), which is threatened by two identified misuse cases ('Obtain Password' and 'Steal Credit Card Info'), while Figure 5.3 demonstrates a proposed design decision based on an analysis of the identified use cases.

Table 5.1 depicts a misuser template for threat actors seeking a password. It specifies a parameterised malactor/misuser, employing economic utility relationships to describe, separate and specify the relative attributes of the attacker. This template is then employed to define a class of misuse cases through the process of pairing the misuser specification of Table 5.1 with the abstract misuse cases identified in Table 5.2 (modelled on Table 3 of [2]). Such definitions not only enable reuse, but also segment the knowledge required at various stages of design and development: the abstracted attacker specification is populated with knowledge resulting from threat intelligence, while the abstract misuse cases rest on security engineering insight. Rather than over-specified misuse cases for each project, various utility relationships can be defined independently and applied to a given project as needed.

Valuations — monetary or otherwise — are captured in the 'Stakeholders and Threats'

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):** Dependent on location, skills. Low likelihood of capture. **Attacker cost (a):** Dependent 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 5.1: Parameterised misuser template. Fields not relevant to this discussion have been omitted for this example.

relative to system and business constraints. 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 decisions. Integration with the NIST SP 800–160 processes such as business operations and risk management practices provides critical security considerations, and supplies an explicit link to the defined security objectives or broader analyses (such as risk-based software engineering approaches, e.g. OCTAVE [198; 199; 66] or AEGIS [48] described in Chapter 2).

From this set of abstract considerations specialisations for attacker models can be created, reflecting specific system threats, industry knowledge, risk attitude, and known data. This is exemplified in Tables 5.3–5.5 for the 'Obtain Password' general misuse case, with notional quantitative values for the negative use case actor profile to illustrate the concept. Parameterisation provides the ability to specify a virtually unlimited set of potential adversaries, without relying on vague classifications such as 'vandalism', 'criminal' or 'military'. The disparities between attackers and defenders, along with the increasing value of information held in the public sector, have mandated a wider scope of threat and attacker considerations for security professionals[9].

[9]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://`

Name: Obtain Password (Misuser target: User Password) **Summary:** An attacker obtains passwords(s) for user accounts... **Known Specialisations:** Obtain Password by Phone, Obtain Password by Sniffing, Obtain Password by Phone. ...
Assumptions: ... **Mitigation Guarantee:** The attacker is disincentivized to attack ($a > p_{s\|a} \cdot M$), or is unsuccessful $p_{s\|a} \to 0$ **Related Business Rule(s):** Only authorised users shall access restricted services.
Stakeholders and threats: The acquisition of a user password carries an average liability of £1,000. ...

Table 5.2: Generalised (abstract) misuse case for 'Obtain Password' based on Table 3 of [2]. 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 ($1 < a_i <$ 10ph). Requires simple, readily available tools.

Table 5.3: Misuse case specialisation for Obtain Password by Sniffing. In this example, 'ph' refers to 'person-hours'.

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 ($1 < a_i <$ 10ph). Requires simple, readily available tools.

Table 5.4: Misuse case specialisation for Obtain Password by Race Condition. In this example, 'ph' refers to 'person-hours'.

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 ($10 < a_i < 40$ph). Requires some research.

Table 5.5: Misuse case specialisation for Obtain Password by Phone. In this example, 'ph' refers to 'person-hours'.

	High			Low		
Target	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 5.6: Notional misuser specification (no protection). The values for p_a and p_s are probabilities, whereas a_i is defined in terms of person-hours for this example. The value of £1,000 is employed for the attacker monetisation as an exemplar.

Broader views of requirements are necessary to enable creative solutions to security problems, as the choice of requirements can constrain the architecture approach — which in turn will constrain system designers [200]. As demonstrated in previous chapters, careful consideration of cost, loss, and effectiveness of potential solutions is key to the development of an efficient security design. Table 5.6 illustrates a utility-based attacker profile populated with notional values that correspond to risk-based analysis, which can in turn be employed to prioritise negative use cases (for example, ranking each based upon their relative values, or by calculations such as ALE or ROA). This process is similar in concept to McDermott and Fox's 'refutation' process [196], with an added element of formality and justification to the analysis. While such refutation mechanisms cannot

`www.theguardian.com/technology/2015/jun/04/us-government-massive-data-breach-employee-records-security-clearances`) attributed to China.

provide absolute guarantees that no exploitable vulnerability remains [180], they enable a more calculated consideration of various design alternatives.

Finally, the incorporation of economic considerations offers an opportunity to holistically consider various system and software security choices across the range of design possibilities, to include prevention, detection, and response. As an example, a designer may be able to control the value of a password through the employment of database encryption, or the addition of an efficient password monitoring or revocation framework. Expanded misuse case descriptions aid in these considerations by allowing the designer to explicitly consider cost and effectiveness. This contributes to mechanisms for countermeasure selection wherein design choices can update parameter values in an iterative process. As a result, the designer can execute 'what-if' scenarios to satisfy emerging concerns and trends. These methods provide a means to convey and deal with the inherent trade-offs in security design; for instance, where a 'better' security option (in terms of lower p_s) may be more expensive but also incur a higher instance of error — possibly due to the difficulty to properly implement the protocol, or utilise the API. Ultimately, such malleability is 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 [180].

5.3.2 Example: Requirements Analysis

When translating security requirements from the Problem Context to the Security Context, engineers are confronted with the challenge of assessing these concerns against myriad functional and non-functional considerations. Relative to the NIST SP 800-160 and ISO/IEC/IEEE 15288 requirements activities, both the SN and SR processes call for analyses that examine the broader set of requirements for security concerns (SN-5.1, SN-5.3, SR-3.1, and SR-3.3), identify measures to assess performance (SN-5.2 and SR-3.2), and identify (SR-3.4), validate (SN-5.4) and resolve (SR-3.5 and SN-5.5) security requirements and constraints. Employing economic considerations in these analyses can provide a common basis for considering technical and non-technical concerns. Relative to security, this added dimension facilitates the evaluation of benefit, supports conscientious requirements trade-offs, and enables decision traceability. Nowhere is this more apparent than in resource-constrained systems, such as the emerging Internet of Things (IoT).

Suite Name	Security Concern	Description
Null	None	No encryption or integrity
AES-CTR	C	Encryption only, counter mode
AES-CBC-MAC-128	I	128 bit CBC-MAC
AES-CBC-MAC-64	I	64 bit CBC-MAC
AES-CBC-MAC-32	I	32 bit CBC-MAC
AES-CCM-128	C and I	128-bit CBC-MAC, AES-CTR
AES-CCM-64	C and I	64-bit CBC-MAC, AES-CTR
AES-CCM-32	C and I	32-bit CBC-MAC AES-CTR

Table 5.7: IEEE 802.15.4 security suites (adopted from [5]).

5.3.2.1 Application Domain

Described as "the interconnection of highly heterogeneous networked entities and networks" [201], IoT has focused on the pervasive instrumentation of physical objects with sensors and actuators, and the connection of those sensors and actuators to the Internet [202]. These devices come in many flavours, but are often differentiated from the general Internet by low-power, low-cost electronics with concomitantly low performance.

The IEEE 802.15.4 specification [203] (updated in 2006 [204]), known commonly as Low-Rate Wireless Personal Area Networks (LR-WPANs), serves as the primary means of IoT device connectivity. The standard specifies a complement of cryptographic suites to meet varied security needs, most employing the Advanced Encryption Standard (AES) cryptographic cipher to addresses confidentiality (AES-CTR and AES-CCM), integrity (AES-CBC-MAC and AES-CCM), neither (NULL), or both (Table 5.7). This is not to say that the security provided by IEEE 802.15.4 is infallible, with an analysis of the 2003 version of the security specification identifying a number of "defects" [5]. Despite these concerns remaining largely unaddressed, IEEE 802.15.4 has seen wide adoption in a range of IoT applications.

A reasonable stakeholder requirement for such devices is compliance with the IEEE 802.15.4 specification. In such cases, the engineer is left with a number of security-relevant decisions when defining the solution; for instance, the specific cryptographic method, bit length, and implementation medium (hardware or software). Absent methods to analyse varied concerns, competing functional requirements regarding availability, throughput, and product lifetime may result in sub-optimal security decisions that meet the letter of the stated requirement, but fail to provide the desired security outcomes. Three example

trade-offs will be explored to demonstrate this concept.

- Application of any cryptographic mode will result in varying impacts to the system's functionality and lifespan. However, choosing to maximise availability through the use of the NULL mode has an obvious negative effect on security and resiliency, leaving the system susceptible to any verification failure resulting from malfeasance or transmission error.
- When realising the security construct, choosing a hardware or software implementation encompasses a series of trade-offs. Commodity hardware often fixes security-related parameters such as key length and Message Authentication Code (MAC) generation, resulting in limitations that must be weighed against the increased resource demands and fallibility of more flexible software implementations.
- The current practice of applying any and all security measures would suggest encryption and authentication of all communications as a matter of course (e.g. [205]). This may lead to security solutions that are inappropriate for the threat space, unnecessarily complex, or result in resource, maintenance or performance issues.

These outcomes serve as examples of pitfalls that must be overcome absent specific guidance to render security requirements in a form that is sufficiently specific and verifiable to guide designers [206]. As demonstrated in Section 5.3.1 the incorporation of economic considerations permits defender and attacker concerns to be represented, supporting reasoned judgements regarding the completeness and coverage of stakeholder security requirements. Likewise, this information can be employed in the translation of security needs into the system design.

5.3.2.2 Illustration

Consider a simplified WPAN device (D) that is defined as a combination of hardware (HW) and software (SW):

$$D = HW + SW \tag{5.1}$$

Such a device may be considered relative to its business and functional aspects in order to examine the security, resiliency and functionality trade-space. This entails the consideration of additional functional requirements that are dependent on the specific application: the number of years the device is intended to operate (Y) — with a finer grained specification considering time (t) as a subdivision of Y; the frequency with which

the device transmits (f); and the size of the data payload to be transmitted (m), for example. Such stipulations are often the result of a requirements analysis, derived independently of security analyses or established by customer edict. In many instances, such as in research and development or early product development, specific values may be less important than acceptable ranges and limits. Parameterisation of these values permits a broader consideration of potential solutions.

Business Requirements A typical business concern is the minimisation of cost (C), relative to a target budget α.

$$\begin{aligned} &C = c_p + c_d + c_m \quad \text{and} \\ &min(C) < \alpha \end{aligned} \tag{5.2}$$

Here, c_p refers to the cost of power, c_d refers to the per device cost,

$$c_d = c_{HW} + c_{SW} \tag{5.3}$$

and c_m refers to the device maintenance cost over the system's lifetime of Y years. Therefore, minimisation of C directly relates to the minimisation of c_p, c_d, and/or c_m relative to Y.

Functional Requirements Security requirements may be constrained by provisioned resources, and informed by functional and environmental considerations.

Output (O) includes network traffic with packets of size x produced by a device at a rate r, in turn expressed in terms of a period of time t and frequency f.

$$\begin{aligned} &O(x, r) = x \cdot r \qquad \text{where} \\ &x = m + m_o + m_s \qquad \text{and} \\ &r = \tfrac{f}{t} \end{aligned} \tag{5.4}$$

The values m_o and m_s relate to the message overhead incurred by the choice of protocol (overhead) and security, respectively, for a given message size m. These relationships are naturally bounded by the constraints of any chosen hardware or software, such that:

$$\begin{aligned} &O(x, r) < O_{max} \quad \text{and} \\ &x < x_{max} \end{aligned} \tag{5.5}$$

Here, O_{max} is the capacity of the network in time frame t. The packet capacity x_{max} and the network O_{max} is established by the choice of hardware, software and protocol.

Power consumption (P) is considered relative to the energy afforded to the system. Judicious use of resources directly affects the maintainability, user experience and the direct costs to manufacture, produce and sell the product — not to mention the ability of the device to sustain attack and execute its functional purpose.

Typically, the minimisation of power consumption to maximise operating capacity and lifetime is a design goal for constrained systems. This can be accomplished either by the power allocation being sufficient to meet the lifetime expectancy of the product, or with a sufficiently low service cost to maintain the necessary power source. The total power required $P_{required}$ is defined as the combination of the power required for security ($P_s(x)$) and transmission ($P_t(x)$) operations, for a given packet size x. This must then be accounted for given the output of the device O over its lifetime, Y.

$$\begin{aligned} &min(P_{required}) && \text{and} \\ &P_{required} = (P_s(x) + P_t(x)) \cdot r \cdot Y \end{aligned} \tag{5.6}$$

The required power is, then, ideally lower than the total power available to the device (P_{total}):

$$P_{required} < P_{total} \tag{5.7}$$

5.3.2.3 Analysis

Examining each of these utility-driven constraints relative to the stakeholder security concerns yields insight into the various design considerations, such as those identified in Section 5.3.2.1.

Hardware versus Software Recognising that price, resources and efficiency are factors that must be considered at the design stage [207], the relationship between functional constraints supplies insight into efficient security design. Considering only power, the constraints defined in Equation 5.6 can be combined to derive a single utility relationship:

$$min(P_{required}) = min((P_s(x) + P_t(x)) \cdot r \cdot Y) \tag{5.8}$$

Operation	Hardware Accel. Time	Hardware Accel. Energy	w/o Hardware Accel. Time	w/o Hardware Accel. Energy
AES-128 encryption (single block)	0.09 ms	2.43 μJ	4.89 ms	131.89 μJ
AES-CCM encryption/decryption (96 bytes)	0.81 ms	21.85 μJ	37.24 ms	1.01 mJ
AES-CCM MAC generation (96 bytes data + 11 bytes AD)	0.61 ms	16.45 μJ	54.77 ms	1.48 mJ
802.15.4 frame transmission (115-byte payload + 11-byte header)	4.93 ms	424.47 μJ		

Table 5.8: Measurements of cryptographic operations on a CC2530 system-on-a-chip (96 byte payload).

Comparing the power devoted to security P_s against the overall power allowance $P_{required}$ on a CC2530[10] system-on-a-chip yields the measurements shown in Table 5.8[11]. This data demonstrates the impact of a range of IEEE 802.15.4 cryptographic modes on latency and power consumption, providing a basis for security requirements analysis. As an example, the software-based AES-CCM energy consumption — *65 times* that of hardware — could be weighed against the 128-bit AES security provided by the hardware solution, which is projected to be 'adequate' for only another 10 to 15 years[12]. Deployment scenarios with a confidentiality requirement or system lifetime that exceeds this date require the consideration of security against a potential software-based implementation (or at least alternative hardware), affecting business and functional requirements of the device that include battery size, replacement rate, and operational environment.

Choosing MAC size Most hardware generates 128-bit hashes for all AES-CBC-MAC and AES-CCM modes, reducing the execution difference between mode variants (32-bit, 64-bit, and 128-bit) to the amount of the MAC transmitted. Given the resource investment in this operation, a designer may seek to understand if an alternative mode supplies adequate security to meet stakeholder needs.

All IEEE 802.15.4 radios have a theoretical maximum raw data rate of 250kb/s on network by specification [203]. Combining the minimum packet size for each mode resulting from a 1-byte message, a theoretical upper bound for an adversary to brute-force the

[10] `http://www.ti.com/product/CC2530`

[11] Message length $m = 96B$, 1 message per second, over 1 year: $r = \frac{1}{1s}$ and $Y = 31536000s$.

[12] Based on the projection for 128-bit symmetric key lengths: `http://www.keylength.com/en/compare/`.

	Ave time to success (Years)		
Data Rate	**32** (2^{31})	**64** (2^{63})	**128** (2^{127})
250kb/s	0.0349	$1.872e+8$	$4.834e+27$
7Gb/s	$1.245e-6$	6685.056	$1.726e+23$

Table 5.9: Theoretical minimum average time (in years) to brute-force the submission of a valid 1-byte packet.

submission of a valid packet can be derived — ignoring the obvious service denial that would result. The top row of Table 5.9 provides a lower bound estimate of the minimum average time for an adversary to theoretically spoof a single packet in such a configuration. An alternative (and perhaps more likely scenario) is the submission of a packet to a collection node receiving data from such devices. Similar calculations performed for an IEEE 802.11ad device are given on the bottom row of Table 5.9[13].

Clearly, the 64-bit version (the only AES-CCM mode required by the specification) is sufficient to withstand such a hypothetical attack. However, the 32-bit mode, at first appearing inadequate, may deserve further consideration. A single maligned 1-byte packet is likely to be insufficient to meaningfully undermine many applications, and could easily be mitigated by time averaging, outlier removal, or other processing on the receiver end. The difference in power consumed per operation in comparable 32-bit and 64-bit modes may result in a significant reduction in the power required over the system's lifetime. Such refinements provide a basis for the development of a security approach coherent with the goal and nature of the processing requirements, the application scenario, and functional constraints such as the radio bandwidth, message packet size and power. When done properly, resultant system designs have the potential to exhibit greater system resiliency, lifetime and performance. This approach is complementary to that of Camp and Wolfram [208], who propose computing resources as the units by which to measure the 'cost-to-break' (CTB) metric [36].

Choice of Cryptographic Mode This analysis builds to the primary security decision facing the designer: choice of cryptographic mode. With 64-bit AES-CCM a common default, the incurred effect of overhead becomes salient. Consider the cryptographic over-

[13]With a maximum theoretical speed of 7 Gb/s, according to `http://standards.ieee.org/news/2013/802.11ad.html`.

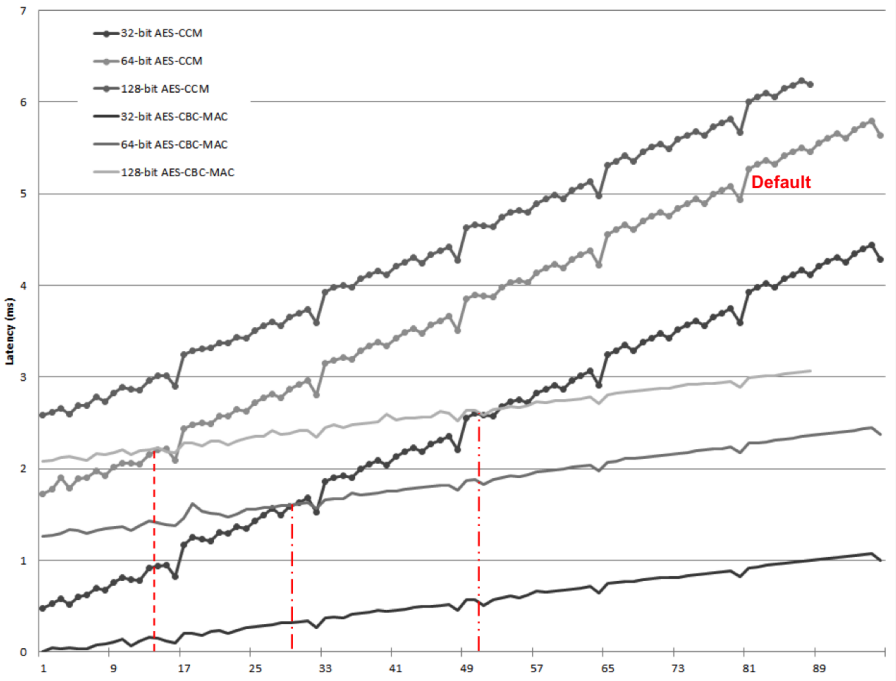


Figure 5.4: Latencies incurred by cryptographic operations at different packet sizes. This graph shows round-trip latency, with the cryptographic functions performed four times between two nodes, A and B: encrypt request at A, decrypt request at B, encrypt reply at B, and decrypt reply at A.

head m_s for a payload size m bytes and m_o byte header, $p = m + m_o + m_s$ (Equation 5.4):

$$\begin{aligned} \text{NULL, AES-CTR} &= m + m_o \\ \text{AES-CBC-MAC, AES-CCM} &= m + m_o + \{4, 8, 16\} \end{aligned}$$

Employing latency as a proxy for power consumption permits these modes to be more clearly investigated. Figure 5.4 shows the variance in the latency for a single-hop, round-trip packet transmission between two CC2530 nodes within a test environment over a series of message sizes (m)[14]. This data was generated by measurement of the overall latency (averaged over multiple runs), and subtracting from that the time incurred by the initialisation vector generation, the software stack, and the transmission (also averaged over multiple runs) — thereby representing only the latency incurred by the cryptographic process. As the packet size increases, the AES-CBC-MAC (integrity) modes show slow

[14]The header size (m_o) is held constant at 11 bytes. The 128-bit measurements are not measured beyond 88 bytes of payload, as the maximum size for the packet x_{max} is reached.

growth consistent with the packet size, as does AES-CCM (although at a higher rate of increase). Given the additional encryption overhead, it is unsurprising to find that the AES-CCM places a larger demand on system resources. However, when analysed against packet size, there exist points where CCM modes are more efficient than alternative modes — likely attributable to shared cryptographic operations that are accelerated on certain devices. With this insight, design spaces can be identified where specific cryptographic modes represent a more efficient choice:

1. 64-bit AES-CCM, compared to 128-bit AES-CBC-MAC for messages under 16 bytes (dotted line).

2. 32-bit AES-CCM, compared to 64-bit AES-CBC-MAC for messages under 32 bytes (dot-dash line), and 128-bit AES-CBC-MAC for messages under 48 bytes (dot-dot-dash line).

From these numbers, it is apparent that for integrity-focused applications AES-CCM may be overkill. Absent a confidentiality requirement, resource savings can be found in most cases by choosing a CBC-MAC mode over CCM (with the exception of 128-bit AES-CBC-MAC for short messages). Conversely, in applications where confidentiality provides value — or at least does not impede the system operation — the addition of the 32-bit MAC actually demonstrates better performance than higher integrity modes for messages under 48 bytes.

5.4 Application

This chapter has offered approaches that enable the transformation of Problem Context stakeholder needs into actionable Solution Context system requirements. Utility relationships provide a useful basis for the analysis of alternative security solutions: they contribute to a richer and more precise understanding of the exact nature of 'adequate security' within the specific system instantiation, and provide a set of evaluative objectives and measures directly tied to the stated security objectives. Conveyance of security requirements via economically specified negative use cases promotes extensibility, constrains analysis, and reduces the potential for premature security lock-in — enabling the analysis of security requirements against other functional and non-functional concerns. Holistic analysis of these concerns using quantifiable, parameterised utility relationships enables the development of adequately secure, functional systems that are efficient and

resilient. Together, these techniques enable the security engineer to support business, managerial, and technical decision-makers with reasoned, anchored analyses. In addition to improved security, these approaches contribute to robust software engineering practice by providing clarification to what are often vague, generic descriptions of security (e.g. "The system shall be secure against all attacks") and inform the decision making in the face of scarcity and uncertainty.

Development of complex economic analyses on this foundation is full of possibility, inviting the examination of security concepts such as the balance of insurance and security action (e.g. [110]), the role of proactive and reactive security (e.g. [112]), or alternative conceptualisations of security (such as deterrence, as described in Section 5.2.3). Constructions such as game theory are "becoming just as important to the security engineer as the mathematics of cryptography" [9], providing a rich set of tools for more intricate and insightful analyses. As has been observed, "while some investment in information security is good, more security is not always worth the cost" [98]. There is no lack of security failures that result from bypassing or undermining security mechanisms, often due to their interference with the functionality or usability of the systems they were put in place to protect. Integrated methodologies supply an important, and to date lacking, linkage between secure software engineering and the burgeoning information security economics research community — leading to more insightful and informed security design.

Beyond requirements, the approach espoused in the preceding sections serves as the basis for economically rational reasoning throughout the system lifecycle. Consideration of the economic impact of functional requirements empowers the system and software designer to balance the security needs with other goals, such as maintainability, usability, and the rest. Trade-offs between security and other concerns are often not adequately accounted for in traditional software architecture decision methods (e.g. [209]), which rely on one-time cost, risk estimation and relative rankings to derive a 1-to-n list for investment. Bodin, Gordon and Loeb [210] provide an intriguing direction with their application of the Analytic Hierarchy Process (AHP) to the problem of selecting between competing designs. However, their approach is devoid of links to the supporting activities called for by guidance such as NIST SP 800–160 or ISO/IEC/IEEE 15288, leaving the practice disconnected from other system security engineering activities. This area is ripe for further empirical and theoretical investigation.

Chapter 6

Software Security Economics in Practice

Cybersecurity is not "build the best solution to win", it's "build the best solution to lose least".
— Michael Borohovski

The goal of an economic approach to secure software is to realise an efficient and balanced strategy to meeting the security goals of a system. To complement the research presented in Chapters 3–5, this chapter applies these concepts to a software project. This includes the development of an investment strategy for lifecycle security concerns (Chapter 4), which informs the constraints on the secure software development process (Chapter 5). These constraints must then be weighed against current norms for enterprise security (Chapter 3). An illustrative case study is employed to demonstrate these ideas in the context of a real-world problem.

Following a description of the context and environment for this case study, the application of this research is explored through the lens of 'security contexts' as outlined by NIST SP 800-160 [7] and presented in Chapter 5. An economic-based analysis is developed, supporting the decisions made within each context. This is accomplished through augmented engineering processes, activities and tasks identified in the incorporated International Standard ISO/IEC/IEEE 15288, "Systems and Software Engineering — System Life Cycle Processes" [183] and its relationship to the SwSec activities identified by the Building Security In Maturity Model (BSIMM) [4]. The result is an approach to explicitly define the SP 800-160 notion of "adequate security" in economic terms, providing a basis for systematic study and decision making. This is followed by reflection on the potential,

promise and challenges raised by the ideas presented in this dissertation. The chapter concludes with a description of the software developed in support of this research.

6.1 Example Application

Identifying investments that realise adequate security is a challenging task, made more difficult by pervasive uncertainty regarding attack realisation, resulting losses, and countermeasure effectiveness. Nowhere is this more palpable than in the space of research and development (R&D), where threats, assets, and environment may be ill-defined, variable or even theoretical — if considered at all. This uncertainty is compounded by questions relative to functional outcomes, deployment enterprise, and user expectations. Further complications arise from the antithetical nature of research, for which high opportunity costs may be paid if research progress is stymied by over-zealous security efforts.

However, as demonstrated throughout the software engineering literature (e.g. [21]), it is precisely this environment in which a lifecycle approach provides the most benefit. The nature of research software development often leaves the engineering of systems to other organisations (or entities under the same organisation), who may lack the domain expertise and/or reach-back to the technology originators required to understand security trade-offs. This can result in a high level of prototype code reuse due to the specialised knowledge required to realise complex algorithms, or an incorrect assumption regarding security measures undertaken in the prototype itself. Such information asymmetries are further compounded by issues of incentivisation during R&D, as resources are focused on technical challenges rather than other concerns. Finally, this lack of security consideration may be exacerbated during engineering, where focus is placed on productising complex technologies (which to non-technologists may already appear fully functional — rendering additional investment unnecessary). As a result, the environment is rife for market failures — in the case of secure software economics, realised as vulnerabilities. Within this setting, methods to guide software and system security processes are a prerequisite to the development of effective secure software practice. It is against this backdrop that this example is presented.

6.1.1 Background

The project selected for study, Project A[1], is one of a series of projects under a larger programme portfolio. The overall goal of Project A is to process and visualise temporal data related to the allocation and utilisation of resources, enabling the direction of human and material capital. The programme to which Project A belongs is focused on the development of novel technologies that contribute to a larger system-of-systems framework. Development is occurring within a government research laboratory specialising in basic through applied research; therefore, any operational use of the system is expected to involve the transfer of the technology for engineering and development.

Within the laboratory environment and for this case study, there are four key roles that impact security decisions:

1. *Leadership* is the collective term for the technical, managerial elements responsible for the oversight of the overall technology investment of the laboratory, embodying the 'Business' role identified in Section 5.1.1. Leadership plays a minor role in this case study, having already authorised the project and set budget and scope[2]. However, Leadership remain the party to whom the programme manager must justify programme progress and execution.

2. The *Programme Manager (or PM)* is the crux of the effort. Corresponding to the 'Management' decision-maker in Section 5.1.1, the PM is responsible for the conceptualisation, definition, initiation (at the approval of the Leadership) and execution of the technical programme. PMs exercise local authority over the execution of allocated budget against the authorised scope and programme goals, to include allocation of resources and establishment of technical requirements.

3. For various aspects of the venture, the PM will have a *Development Team* to execute the programme design, development, test, and fielding. Development Team may be other government employees, partner researchers, or contracted expertise (most likely, a combination all three). While expected to act independently and apply expertise within established parameters, the Development Team relies on the PM to authorise any changes affecting system realisation. This often results in

[1]The name and some details of the case study have been changed to protect current and future operational concerns; however, the resulting conclusions are representative of those reached during the analysis. Monetary values are presented in the project's currency, US dollars (USD).

[2]While not present in this case study, one application of analyses such as those presented in Chapter 4 and Chapter 5 would be *ex ante* decision-support to inform and influence leadership programme decisions.

linear development cycles with PM sign-off on requirements, design, and testing documentation; however, the PM may delegate some decisions to a *Technical Lead* for the purpose of enabling more agile development methods or fostering creativity. In these cases the PM remains the responsible party for the overall programme, and establishes shared responsibility with the Technical Lead for the 'Technical' decision-making role. In this case study, the PM and Technical Lead have come to agreement on a general design, with authority granted to the Technical Lead for minor design changes and implementation decisions.

4. Finally, the PM may employ a *Lead Engineer*. In this context, the Lead Engineer has the responsibility to guide the technical aspects of the programme through recommendations to the PM. Like the PM, the Lead Engineer comes from the laboratory staff and is accountable to Leadership for effort execution. Unlike the Technical Lead, the Lead Engineer is likely to be involved in a number of efforts, and act in the capacity of an evaluator or advisor to the PM rather than a member of the Development Team. Among other responsibilities, the Lead Engineer is expected to act as the System Security Engineer described in Chapter 5, often pooling expertise through associations with other Lead Engineers that approximate the Software Security Groups (SSGs) espoused by BSIMM [4].

At the onset of this case study, Project A is a standalone development effort comprised of a PM, a small contracted Development Team (with one acting as the Technical Lead), and no formal Lead Engineer; this role was undertaken as part of the study, focusing on the system security duties. The project was nearing the end of a 10-month (43-week) first phase of funding at the level of $700K, resulting in roughly 18 KLOC (thousand lines of source code). At the conclusion of phase one, a phase two effort with a funding level of $800K is expected to expand the codebase moderately (as refactoring and development continue) and integrate this technology with an expansive, $20M framework being developed externally.

Given this project plan, it has been established that this system has high opportunity for operational deployment. Absent this assumption, the introduced vulnerability from this development effort tends to 0 (as all operational code would be the result of a separate effort), and subsequently Gordon and Loeb's assumption *A1* dictates that z also trends to 0. With no explicit investment into security, the vulnerability of developed software is assumed to be high ($v \approx 1$), as discussed in Chapter 4.

6.1.2 Problem Context

As discussed in Chapter 5, the Problem Context forms the basis upon which security decisions are made throughout the system lifecycle. Critically, this includes the definition of *adequate security* to be employed in developing the solution and later assessing trustworthiness. This, in turn, involves analysis to define constraints governing the subsequent processes [7]. Focus is placed on the technical management and technical process measures that define these constraints: security planning and security objectives. Under Project A, these decisions fall to the PM who is seeking help in establishing security priorities.

6.1.2.1 Security Planning

To start, an evaluation of Project A was conducted by the Lead Engineer in order to establish a range of (theoretically) effective information security investment. This corresponds to the 'Project Planning' and 'Project Assessment and Control' Technical Management Processes in ISO/IEC/IEEE 15288 [183], with no apparent correlation in the BSIMM.

With the phase one development effort currently underway, an *ex-post* analysis was conducted in order to gauge the current information security investment. Through discussion with the Development Lead, it was estimated that roughly two person-weeks were expended on security-related tasks in phase one; this level of effort was regarded to be appropriate by the Development Lead and PM. Undertaken security tasks included a mix of process-related measures (code reviews) and design/implementation measures (input format and boundary checking, and the implementation of a stand-in cookie authentication mechanism for replacement by PKI at a later date). These investments complemented the security goals of the programme, expending roughly \$32.56K (4.65%) of the phase one budget. For simplicity, this expenditure is counted entirely against the security budget — despite the dual security-quality nature of code review and input checking.

Standard Model: Gordon-Loeb. In the absence of a defined threat model for the final system, the standard Gordon-Loeb model was employed to model the security investment using the S^I security breach probability function with the parameters $\alpha = 0.5142$ and $\beta = 1$, as identified in Section 4.2.2.1 for 20 KLOC.

The initial analysis was limited to the scope of the phase one R&D effort. This could be rationalised that, as a prototype and prior to deployment (e.g. exposure to the public network), the development environment reduces the threat of attack and limits any loss to the project investment. In this case, a small threat potential is assumed ($t \approx 0$), as no threat would result in no investment; and loss limited to $\lambda = 700$ (the project phase value

in thousands of dollars). The latter is consistent with a standalone system, and ignores possible externalities resulting from a compromise of the software under development (e.g. a vulnerability that leads to the compromise of the underlying operating system that houses development tools, other projects, and personal information). Optimal investment in this instance is found by the first order condition for S^I [98]:

$$\begin{aligned} z^{I*}(v) &= \frac{(v\beta\alpha(t\lambda))^{\frac{1}{(\beta+1)}}-1}{\alpha} \\ &= \frac{(1\cdot 1\cdot 0.5142(t\cdot 700))^{\frac{1}{(1+1)}}-1}{0.5142} \\ t=0.01 &= \frac{0.5142(0.01\cdot 700)-1}{0.5142} && = 1.744 \\ t=0.10 &= \frac{0.5142(0.10\cdot 700)-1}{0.5142} && = 9.722 \end{aligned} \tag{6.1}$$

These results reflect a 0.25–1.39% investment into security over this phase, driven by the low threat estimate. The identified efficient investment falls well under the security investment made by the team, seemingly indicating gross over-investment in security. However, a more consistent picture is painted if the assumption that this code is likely to go into operation is taken to its logical conclusion. The high likelihood of transition, and subsequent code reuse, implies that the code is likely to encounter attack ($t \approx 1$).

$$\begin{aligned} t=0.90 &= \frac{0.5142(0.90\cdot 700)-1}{0.5142} && = 33.06 \\ t=0.95 &= \frac{0.5142(0.95\cdot 700)-1}{0.5142} && = 34.02 \\ t=1.0 &= \frac{0.5142(1.0\cdot 700)-1}{0.5142} && = 34.95 \end{aligned} \tag{6.2}$$

The resulting investment is in-line with the effort undertaken by the development team, and reflects forward-looking security thinking; for instance, investment in cookie-based authentication ensures the proper architecture is in place, but does not attempt to re-create costly infrastructure.

At phase two the combined R&D budget of Project A rises to $1.5M. Following the approach employed in phase one yields an estimate of efficient security investment of $52.07K at $t = 1$, or 3.47% of the R&D funding. By this estimate, the projection of expenditure is effectively reduced from phase one levels ($52.07K-$32.56K=$19.51K). Placing Project A within the context of the larger framework forms the basis for estimating the loss much higher, perhaps at the level of the overall system investment: $\lambda = 21500$ (the investment made into the R&D, as well as the framework itself; ignoring externalities and secondary costs). Efficient investment into security in this case is found to be $202.54K, or 0.94% of the total investment. As this represents a loss of the entire system, it would imply

that Project A — just under 7% of the system investment — would have consumed over 16% of the security budget for the entire system in phase one alone. With widely cited numbers for the cost of accreditation for EAL4 at \$175K–\$300K, this estimate appears inadequate to support pre- and post-deployment security investment.

These results can be interpreted in one of two ways. From the perspective that too little is spent on security, these results reinforce the discussion of Section 4.5: efficient security levels do not necessarily yield desired security results. Correcting for a more risk-averse attitude for security, as undertaken on Project A, would require revisiting a fundamental aspect of the Gordon-Loeb model's risk-neutral formulation. Alternatively, the perspective that security expenditures are inefficient and lead to over-investment calls into question the cybersecurity assessment processes and engineering judgement exercised. Regardless of perspective, this analysis fails to inform on a key aspect of the security investment: how much to invest toward pre-deployment, versus post-deployment.

Proposed Model: GL-SSE. Employing the GL-SSE model developed in Section 4.2.3 provides an alternative view that attempts to inform pre- and post-deployment considerations. Examining the pre-deployment security investment for Project A, Figure 6.1 depicts the ROSI for post-deployment investments of \$175K, \$300K, and \$645K at $v = 0.95$ with nominal efficiency ($\alpha_{S^{II}} = 1$). This corresponds to previously-cited bounds for EAL4 assessment, up to 3% of the effort's budget — the average US Government enterprise security investment [211]. These results indicate a wide potential range of productive investment (i.e. positive returns) for software security.

Focusing on the appropriate level of security investment during R&D for Project A, Figure 6.2 shows the ROSSP as calculated for various (high) starting vulnerabilities ($v = 1$, $v = 0.95$, and $v = 0.90$) when $S_{\text{PRE}} = S^I$ and $S_{\text{PST}} = S^{II}$. The results range from the full loss (at $v = 1$) to a small fraction of Project A's budget (at $v = 0.9$). At $v = 0.95$, a positive return is found up to $z_1 \approx \$1.7M$ for the overall system; by relative cost, this places $z_1 = \$119K$ (7% of the overall system investment) for Project A. With a \$32.56K phase one investment, this results an additional pre-deployment security investment of \$86.44K, or roughly 11% of the phase two budget.

GL-SSE analysis paints a different picture for security investment in Project A. The increased scope of security investment reflects common intuition regarding early investment into security, but with key caveats. First, the pre- and post-deployment investment are linked, with increases in one reflected in the other. While the most efficient SwSec investment will always be the first dollar spent, the range of positive return can be large

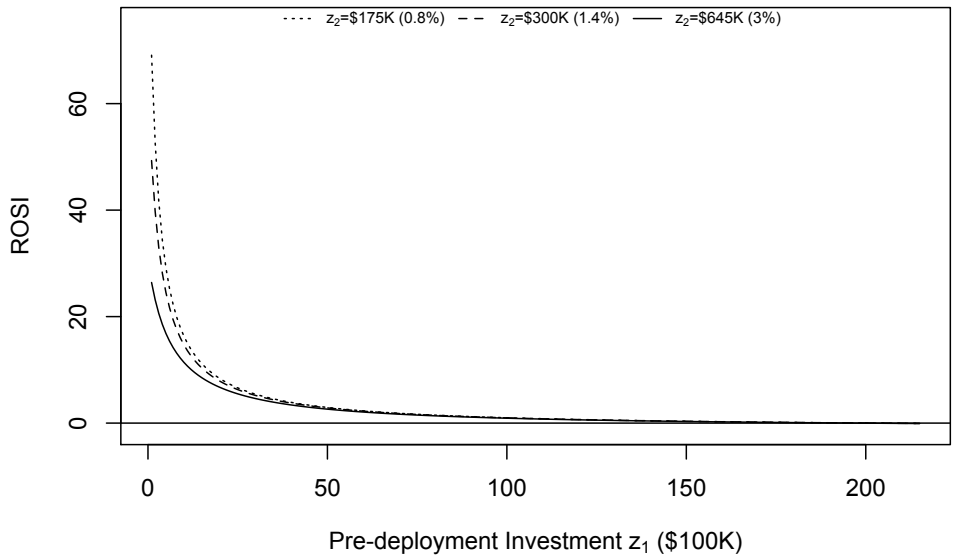


Figure 6.1: Return on Security Investment under GL-SSE ($\alpha_{SI} = 0.5142$, $\beta_{SI} = 1$, $\alpha_{SII} = 1$, $\lambda = 21500$, $t = 1$).

— much larger than the available budget in some cases. With the most efficient investment point far below the 'most secure' point (i.e. greatest reduction in v), reallocation of resources to the R&D effort for security processes could be argued.

As identified in [164], the less productive the investment, the more sensitive the calculation is to variation. This underscores the need for carefully selected data. An analysis that adjusts model parameters to this specific project team (e.g. team efficiency, languages and tools utilised, etc.) would produce a more calibrated result.

6.1.2.2 Security Objectives

As discussed in Section 5.2, security objectives can be specified through the construction of utility functions. These may be the result of rigorous analysis (for example, via game theory), resulting in security goals and constraint conditions. The need for such processes is identified in the Technical Process 'Stakeholder Needs and Requirements Definition' as well as the Technical Management Process of 'Project Assessment and Control' of ISO/IEC/IEEE 15288 [183]. BSIMM touches on this need in the 'Intelligence' domain practices (constituting Attack Models, Security Features & Design, and Standards &

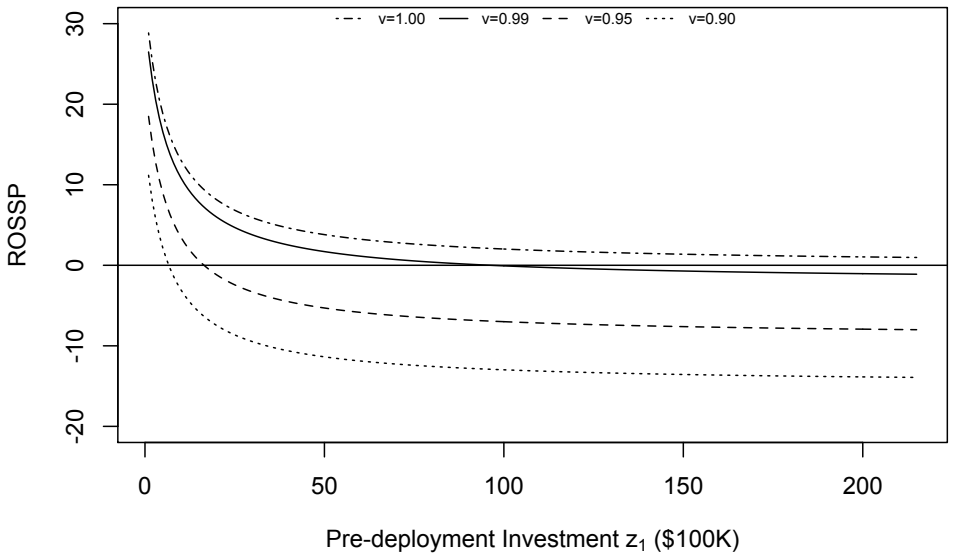


Figure 6.2: Return on Secure Software Process under GL-SSE ($z_2 = \$645K$, $\alpha_{S^I} = 0.5142$, $\beta_{S^I} = 1$, $\alpha_{S^{II}} = 1$, $\lambda = 21500$, $t = 1$).

Requirements). The desired outcome, in the case of software security economics, is for the security engineer to identify and convey system security threats and identify requirements leading to an efficient security design. This supports PM decisions regarding the nature, scope and construct of the security mechanism.

Given a defined security objective and threat model, such an analysis would ideally be incorporated into the previous Security Planning step via models such as IWL-SSE (Section 4.3.2). Alternative security goals, such as the deterrence construct in Section 5.2.3, could also be employed as a target for security investment planning. Unfortunately, Project A lacks such a well-defined threat model. In consultation with the PM, the anticipated threats and eventual deployment environment are more traditional, and highly reliant on the operational enterprise. As a result, a simplistic game based on standard economic motivations of attackers and defenders was developed for Project A. This serves as both a starting point for discussions between the PM and other stakeholders, as well as an illustration of the methodology.

The game in Table 6.1 is formed as one of partial information for the defender, with attacks (a) occurring with probability t. Decisions on investment into defensive measures d rest on system vulnerability (with defences v_d, and without v), investments z_d, and loss

Table 6.1: Simple game theory model for setting security goals.

	$a\ (t)$	$\neg a\ (1-t)$
d	$-v_d\lambda - z_d,\ v_d\epsilon\lambda - c_a$	$-z_d,\ 0$
$\neg d$	$-v\lambda,\ v\epsilon\lambda - c_a$	$0,\ 0$

λ. Attacker strategies rely on attack cost c_a and expected gain ($v\epsilon\lambda$ or $v_d\epsilon\lambda$, where ϵ is the fraction of the loss that can be monetised by the attacker). This formulation is essentially that of an economic attacker, whose strategy rests on benefit maximisation. In addition to the development of more complex defence strategies (such as deterrence, as discussed in Section 5.2.3), other security constructs may include differently-motivated attackers, or other costs and benefits such as reputation gain/loss, political equities, or legal liabilities.

Analysis of this game allows identification of economic considerations during the development process. As the defender, one approach is to consider if the utility in investing in a defensive approach (d) is greater than the utility of not investing ($\neg d$):

$$\begin{aligned} d &> \neg d \\ t(-v_d\lambda - z_d) + (1-t)(-z_d) &> t(-v\lambda) - (1-t)0 \\ -v_d t\lambda - z_d &> -vt\lambda \\ v_d t\lambda + z_d &< vt\lambda \\ z_d &< vt\lambda - v_d t\lambda \\ z_d &< (v - v_d)t\lambda \end{aligned} \tag{6.3}$$

Such representations make the 'levers' of information security investment explicit: under the assumption that the threat t is endogenous, the defender decision rests on the relationship between the security investment amount (z_d), vulnerability (v and v_d), amount at stake (λ), and attacker cost (c_a). Establishing these relationships within the Problem Context provides greater flexibility to the PM, who can choose security strategies that reduce v to v_d, architectures that attenuate the exposure to loss λ, or deployment targets that increase attacker cost c_a. Most relevant to this exposition, such an analysis supplies the mechanism by which the programme manager can identify desired investment points within the space of positive ROSSP.

Employing the security investment for Project A identified in Section 6.1.2.1, it can be shown that an expenditure of \$2.345M (\$1.7M pre-deployment, and \$645K post-

deployment) easily meets the desired objective with a very low expected residual vulnerability of $v = 0.0000038$ (under the assumptions of investment productivity captured by S^I and S^{II}):

$$\begin{aligned} z_d &< (v - v_d)t\lambda \\ 2,345 &< (0.95 - 0.0000038) \cdot 1 \cdot 21,500 \\ 2,345 &< 20,424.92 \end{aligned} \tag{6.4}$$

A more modest investment of \$100K into pre-deployment (with the same post-deployment security) also meets this objective, but with a slightly higher residual vulnerability ($v = 0.046$).

6.1.2.3 Discussion

While the analysis in this section employs specific parameters identified in previous chapters, it is by no means the only analysis that could (or should) be undertaken as part of a robust approach to security. Supported by the belief that 'triangulation' of estimates yields the best result [96], the assumptions underlying the models in Chapter 4 and Chapter 5 could be further evaluated relative to Project A. Ideally, this would result in multiple scenarios, allowing the Lead Engineer to produce for the PM an evaluation of how different estimates and outcomes affect the process. Following from Section 4.2.3, a re-examination of the assumptions driving the parameters could yield dramatically different results; for example:

- As discussed in Chapter 2, traditional accreditation schemes focus on the checklist-based configuration within an enterprise. Under an expectation that misconfiguration represents a smaller (or larger) portion of the vulnerability ($\delta \to 0$ or $\delta \to 1$), the positive benefit space for pre-deployment investment increases (or decreases) as discussed in Section 4.2.3.2.

- With the Project A Development Team employing modern, agile practices in a Continuous Integration Environment (CIT) it is reasonable to expect greater efficiency in the security practices (e.g. increase in α). As α approaches 1, the investment decreases while the peak ENBIS and resulting v stays constant. As a consequence, pre-deployment investments reduce v faster, resulting in a smaller positive ROSI (and therefore ROSSP) range with higher security efficiency.

As demonstrated, setting tighter goals for adequate security — e.g. a specific desired value for the vulnerability — further influences the security investment, the required

effectiveness of the investment, or potentially the assets placed at risk. This underscores a key point made in Section 4.5: this research shows that, given realistic values regarding the nature of the investment (i.e. the parameters of the security probability breach functions), it is very likely that the *efficient* outcome may differ from the *desirable* outcome. In this case, these tools can still be employed to identify net positive outcomes, with other combinations of z_{PRE} and z_{PST} leading to even lower values of v_{PST}. These must be considered in light of budget constraints and competing interests.

Finally, it should be noted that this simple example of a security goal has produced a variant of the Annual Loss Expectancy (ALE) calculation, as discussed in Chapter 2 and employed in the analysis of security policy in Chapter 3. This is not a coincidence; these security goals underlie the basic economic concerns of such measures. With the system set to be deployed within a traditional enterprise environment, such an analysis must be conveyed throughout the SDLC in order to support re-evaluation and further decision-making.

6.1.3 Solution Context

With the Problem Context defined, NIST SP 800-160 calls for the definition, realisation, and production of evidence for security aspects of the solution. Following the research presented in Chapter 5, this section focuses on the former: how to convey economic analyses within the security aspect definition, such that it is explicit and actionable throughout the Solution and Trustworthiness contexts. Such definitions and analyses of the security solution provides a means for the Lead Engineer to convey to the PM and the Development Team key aspects of potential threats and security decisions. This enables a dialogue between the PM, Development Team, and System Engineer as to the best approach to balance security and functional needs in the system

With Project A nearing the conclusion of its initial phase, development has moved beyond the application of these concepts organically within the lifecycle. As such, this section will present a representative example of the processes, inspired by the development under Project A and used by the Lead Engineer to foster discussion with the programme team.

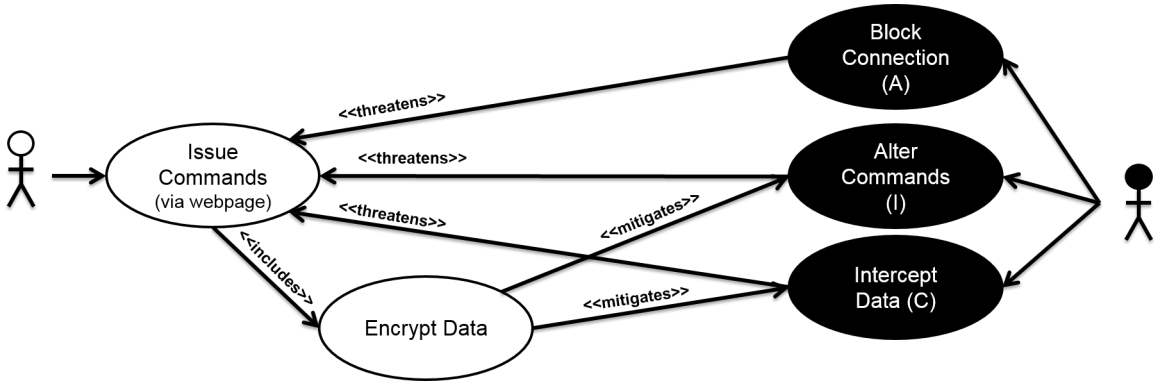


Figure 6.3: The *Issue Commands* use case, identified use cases, and potential mitigation.

6.1.3.1 Requirements

Requirements are an integral and heavily-researched aspect of the secure development lifecycle, as discussed in Chapter 2. These processes involve both the articulation of security-specific requirements, as well as the security aspects of functional requirements, as identified in the 'Security Features & Design' and 'Standard & Requirements' processes of the BSIMM and the 'System Requirements Definition' activity in NIST SP 800-160. Increasingly, attack models are seen as a critical part of this process, evidenced by their inclusion in many policies and guidance (e.g. inclusion of 'Attack Models' as the third process composing the Intelligence domain of the BSIMM [4]). A commonly employed approach that binds these three concepts is that of use/misuse cases, as discussed in Section 5.3.

A sample is presented in Figure 6.3, showing the use case "Issue Commands" along with potential misuse cases corresponding to security concerns Confidentiality ("Intercept Data"), Integrity ("Alter Commands"), and Availability ("Block Connection"). The mitigation "Encrypt Data" has been identified as addressing two of these misuse cases, with the third unaddressed. Following from the approach outlined in Section 5.3.1.1, this is captured in misuser (Table 6.2), abstract misuse case (Table 6.3), and concrete misuse case (Table 6.4) templates. As before, misuse case fields irrelevant to this discussion are omitted.

These parameterised misuser and abstract misuse case templates are populated directly from the information generated in the Problem Context (Section 6.1.2.2). Threat t is equated to probability of attack initiation p_a, while vulnerability v is equated to the probability of success p_s, for a given cost of attack c_a[3]. In the abstract misuse case 'Obtain

[3]As in Section 5.3.1.1 it is assumed that p_a and p_s are independent.

Misuser Target: Seek System Commands **Summary:** An attacker seeking to intercept commands between a user and a system. **...** **Related Use Case(s):** Issue commands (via Web Interface)
Utility Considerations Attacker cost (c_a) Probability of successful attack = Probability of attack (threat, t) · probability of success (vulnerability, v) = $(p_a \cdot p_s)$ **Constraints:** Attacks only if the expected pay-off is higher than cost; $a < p_{s\|a} \cdot \epsilon\lambda$, where $\epsilon\lambda$ is the monetisation value to the attacker.
Abstraction Level: Façade **...**

Table 6.2: Parameterised misuser template. Fields not relevant to this discussion have been omitted for this example.

Name: Obtain Commands **Summary:** Attacker seeks to obtain commands being issued to the system. **Known Specialisations:** Obtain Commands by Interception (client-server), Obtain Commands by Malware, ... **...**
Assumptions: ... **Mitigation Guarantee:** The attacker is dis-incentivised to attack ($c_a > p_{s\|a} \cdot \epsilon\lambda$): $[p_{s\|a}, \lambda, \epsilon] \rightarrow 0$ or $c_a \rightarrow \infty$ **Related Business Rule(s):** Only the user issuing a command should have knowledge of the issued commands.
Stakeholders and threats: Observation of user commands may undermine mission success, compromising the investment in the system ($\lambda = \$21.5M$). **...**

Table 6.3: Generalised (abstract) misuse case for 'Obtain Commands'. Fields not relevant to this discussion have been omitted for this example.

Name: Obtain Commands by Interception (client-server) **Summary:** A misuser seeks to obtain commands in the 'Issue commands (via Web Interface)' use case by eavesdropping on the communication between system components. ...
Misuser Profile: Seek System Commands — UNMITIGATED **Prob of attack (p_a):** 0.99 (all attackers) **Prob of success (p_s):** 0.95 (high-high), 0.5 (high-low) **Attacker cost (c_a):** 20 person-hours **Mitigation:** Assuming positive ϵ, $c_a < p_{s\|a} \cdot \epsilon\lambda$ for system value λ

Table 6.4: Misuse case specialisation 'Obtain Commands by Interception (client-server)'.

Misuser Profile: Seek System Commands — MITIGATED **Prob of attack (p_a):** 0.99 (all attackers) **Prob of success (p_s):** ≈ 0 (Data encrypted) **Attacker cost (c_a):** 20 person-hours **Mitigation:** Assuming positive ϵ, $c_a > p_{s\|a} \cdot \epsilon\lambda$ for system value λ

Table 6.5: Updated 'Obtain Commands by Interception (client-server)' with the 'Encrypt Data' use case implemented.

Commands' these utility concerns are employed to identify mitigation conditions, coming from a solution to the security objective game in Section 6.1.2.2. Isolation of specific parameters highlights the options available to the system architect.

Following from these abstractions, the concrete misuse case combines the various generalisations into applicable and actionable security requirements. The utility considerations from the misuser profile are populated — ideally, from threat intelligence and historical data. Notional data is shown in Table 6.4 without mitigations, employing the 'skill-focus' dichotomy for describing attackers[4]. With the introduction of the 'Encrypt Data' use case, this can be updated as shown in Table 6.5, where the encryption of data has reduced $p_s \approx 0$.

As discussed in Section 5.3.2, the generation of requirements that do not specify a design solution often pose a challenge for security, leading to premature design lock-in or solutions that fail to meet the desired system functionality. The 'Encrypt Communications' mitigation does not fall into this trap (although specific assumptions regarding the

[4] As described in, "Lessons from the Sony Hack", `https://www.schneier.com/blog/archives/2014/12/lessons_from_th_4.html`.

nature of the solution may impact the parameter affected and resulting value). Specification of a solution at the design stage requires further insight, as potential specifications include data or link level encryption under a variety of schemes — necessitating further analysis to balance security and functionality concerns. For this purpose, utility expressions within the 'Issue commands (via Web Interface)' will ideally specify functional requirements (for instance, a latency requirement or interface requirement with a legacy system), further constraining the choices of the system designer. Following the approach defined in Section 5.3.2, this secondary constraint can then be analysed in light of the attacker mitigation condition identified in Table 6.3. The resulting analysis permits the consideration of mechanisms with less than ideal security performance — for instance, the employment of specific cipher suites within the Transport Layer Security (TLS) protocol to meet hardware, latency, or legacy integration constraints.

6.1.3.2 Discussion

The requirements analysis presented in Section 6.1.3.1, although not derived from Project A, does apply the research within Chapter 5 to problem sets representative of Project A's security analysis. Deriving utility-based criteria from the analysis of Section 6.1.2 binds the problem and solution contexts in a way that provides formalism (short of formal methods) and repeatability. The described approach has multiple benefits over traditional methods: it makes assumptions on attackers and mitigations explicit, and ensures considerations on each are examined in light of system goals. When new attacks emerge that impact these parameters, the explicit nature of the security requirement criteria permits examination, recalculation, and further analysis.

An important aspect to this discussion is the variable nature of vulnerability that exists within this analysis. Within the problem context, vulnerability is treated as the probability of successful attack — a system-wide assessment of the intrinsic system state. Within the solution context, vulnerability is further equated to the success of a given attack — a subtle but important shift in definitions in order to permit the examination of a particular use case. These interpretations are subsequently supported by a third: that of specific vulnerabilities as they manifest within implemented systems. These related, but not equivalent, employments of the concept of vulnerability are further discussed in Section 7.2.2 as an important issue for further research.

6.1.4 Lessons Learned

This example brought many challenges to the fore, while highlighting the potential — and promise — of these concepts.

Starting with information security investment modelling, it is not clear that the efficient level of information security investment is ideal, or even desired; this is due to a number of issues raised throughout this research, made manifest by this example. Critically, this is a limitation of the stylised models employed in this analysis, and the assumptions upon which they are based. Examples include the failure to capture non-monetary concerns in loss calculations, the ever-changing and pervasive nature of the threat in the deployment environment, and the interconnectivity of such fielded systems to a substantial and valuable enterprise. This is further exacerbated by uncertainty in the parameter values, with the employment of generic parameters in place of (non-existent) historic measurements of software security and post-deployment effectiveness resulting in imprecise estimates. These issues relate to Gordon and Loeb's caveats regarding systems with potentially catastrophic losses and non-neutral risk postures [98]; both of which apply to Project A. At best, this experience reinforces the assertion in Chapter 4 that models are best employed as guidance, rather than direction.

Despite these limitations, the post-hoc investment level identified for phase one closely aligns with the effort undertaken by the Development Team and deemed appropriate by the PM (given stated project assumptions). In phase two, the planned integration influences the investment significantly, although the values for post-deployment investment and loss potential serve as grounded estimates rather than absolute insight. Provided this estimate and the methodology used in its generation, along with a plan of action and direction, the PM welcomed the case study outcomes (although it is unclear to what extent this can be attributed to the specific plan versus *any* external security consultation). The well-defined and traceable approach to deriving this estimate served as a catalyst for increased security emphasis within Project A, with additional security investment and a documented security development plan as tangible outcomes. Any future resource conflicts between the proposed security process and functional demands will provide a litmus test for commitment.

All parties involved found the NIST security contexts and BSIMM SwSec processes approachable (if not at times overly broad) constructs for incorporating these concerns. In the establishment of security objectives, the methodology of employing game theory in the development of security goals (Section 6.1.2.2) was found to be burdensome for

Project A. It is conjectured that this is an artefact of the system under study, which at its core is a standard web-based processing engine to be incorporated into a heavily-regulated enterprise framework and exposed to a wide array of potential attacks. The straightforward design of Project A coupled with the broad nature of the threat it faces renders it ill-suited for such an analysis. Absent unique security goals or a specific, directed threat model, the defender and attacker utilities derived reduced to generic economic concerns that are well-modelled and don't require game theoretic derivation. As a result, the contribution of this approach over the adoption of standard security goals — or the policy, accreditation, and checklist approaches already applied to the larger enterprise framework — is not immediately clear. The development of security goals for non-traditional systems or environments, such as deterrence goals (Section 5.2.3), may provide more fertile ground for such endeavours.

The derived example infusing these concepts to the secure software engineering process was insufficient to assess the utility of the approach over current methodologies. The return on the effort to develop generalised abstractions was unclear, although with the benefit of such abstractions lying in their reuse, insights will not be reached with example problems. Capturing such information was seen as beneficial to providing a basis for reasoning and traceability, with significant potential for later analysis. However, in the case of misusers (malactors) the values assigned were less rigorously derived than desired. While information regarding threat probability exists for some contexts, a lack of such information led to the assignment of largely qualitative, 'reasonable' values in this case. Additionally, rectification of a mitigation approach to a misuser largely resulted in binary results, rather than the nuanced gradation that probabilistic assessments would impress to the user. This condition speaks both to the lack of data (a recurring theme throughout this research) and the nature of vulnerability — both discussed in detail in Chapter 7. While the case study example is indicative of the approach's potential, here again the system under consideration has likely influenced the outcome: as an enterprise system component with few 'hard' requirements, there are few measurable constraints against which security can be considered. Emerging environments, such as the IoT scenario presented in Section 5.3.2, provide a richer environment for the future research required to evaluate such methods.

Despite the challenges, overall the application of this research to Project A and the development of workable examples has illustrated the potential benefits to be realised through the application of this research. These approaches have drawn interest from the

system integrator for its promise to reduce system integration and accreditation effort, as well as the system customer for potential application in future programmes. As tools for considering the tension between security and cost throughout a system's lifecycle, the impact of these measures cannot be known until system security costs are tallied against comparable systems. Even then, the true benefit may not be realised until the system is subjected to attack and found both sufficient and efficient.

6.2 Software Libraries

To support the development and application of the information security investment models, a software repository has been established on the software development platform GitHub (`https://github.com/`). GitHub supports cooperative development and sharing of codebases, permitting wide dissemination of the developed tools. This repository is identified by the project name `InfoSecEcon-InvestmentModels`[5], and is subdivided into four directories:

- `models`, the most significant portion of the codebase, is the repository for `R` source files that realise the various models developed and utilised under this research.
  - *Utilised models* include `R` variants of the IWL [112; 176] and Gordon-Loeb [98] investment models employed in Chapter 4. In addition, the Hausken [101] and Willemson [100; 102] alternative security breach probability functions, as well as the COSECMO [156] calculations are provided within this directory. Finally, implementations of many standard information security economic functions employed in the analysis conducted in Chapter 3 can be found in `InfoEconFunctions.R`, based on the code produced by Southern Methodist University (SMU) CSE 7338 Security Economics graduate course, fall 2014[6]. In addition to implementations of the model operation, many of these files contain supporting functions to reproduce published results and verify correct operation.
  - *Developed models* consist of the IWL-SSE and GL-SSE variants developed as part of this research.

[5] `https://github.com/CHeitzenrater/InfoSecEcon-InvestmentModels`
[6] `http://tylermoore.ens.utulsa.edu/courses/econsec/`

- `data`, which houses files containing the data derived from various datasets. This includes the data from the Information Security Breach Reports 2012–2015 employed in Chapter 3 and the COSECMO EAL data employed in the development of the GL-SSE in Chapter 4.

- `examples` contains examples of model usage, along with the code employed in the development of the papers published as part of this research.

- `tests` acts as a development area for experimental scripts under development. At different points this directory contained partially-developed versions of the code now resident in the `examples` and `models` directories. As of the time this dissertation was submitted, this directory is empty as all remaining code has been integrated into other source files.

In addition, the file `README` contains information for configuring the `R` environment to utilise this project.

This code was employed extensively in the research results contained in Chapters 3–5, in addition to its application to the case study described in Section 6.1. As a public repository, this codebase is open for external contribution and usage, and stands as a tangible outcome of this research[7].

[7]Distribution A. Approved for public release: distribution unlimited. Case number: 88ABW-2017-3847. The codebase was reviewed on 9 Aug 2017.

Chapter 7

Conclusion

There is no such thing as perfect security, only varying levels of insecurity.

— Salman Rushdie

This chapter summarises the findings of this research, identifies limitations to applicability, and captures future directions for investigation.

7.1 Contributions & Findings

The contributions and major findings of the research presented in Chapters 3–5 of this dissertation are summarised below:

- Chapter 3 set out the motivation for this research through an analysis of a current cybersecurity accreditation scheme. Examination of the Cyber Essentials scheme, in conjunction with the Information Security Breaches Survey, demonstrated that conditions exist under which even the most 'basic' cybersecurity measures can be found to be economically unviable. This investigation underscored the need to consider cybersecurity controls relative to available resources, and the impact of ongoing investment on effectiveness — specifically, costs relative to maintenance, such as scale and manpower. Under a worst-case approach to breach impact, this chapter motivates the need for a lifecycle approach to cybersecurity. This analysis was initially developed in [24], and subsequently updated and expanded in [25]. ***Contribution:*** *Demonstration of a method for, and an example of, economic analysis of a security policy.*

- Given the need for a lifecycle approach, Chapter 4 explores models for considering cybersecurity expenditures prior to the deployment of a system (with software security the paradigm for such investments). Two models were developed to demonstrate this approach: the first is a single-period, utility-focused model based upon the Gordon-Loeb model for information security investment — the most popular and widely-recognised information security economic model. A variant of this model, Gordon-Loeb for Secure Software Engineering (GL-SSE), captures pre- and post-deployment information security investment [26]. The second model, based on the Iterated Weakest Link model of information security investment, provides an analysis of pre-deployment security through a multi-period, numerical optimisation with game-theoretic elements. The IWL for Secure Software Engineering (IWL-SSE) is presented in [27]. These models are examined relative to the Return on Security Investment (ROSI), Earned Net Benefit of Information Security (ENBIS), and a newly developed metric capturing the benefit of investments in pre-deployment security: the Return on Secure Software Process (ROSSP).

  Analysis of these models provides insight into the impact and value of such investments, generating an understanding of the limits of different security investment approaches, the impact of factors such as threat, uncertainty and value-at-risk, and the effective scope of pre- and post-deployment security investment providing positive returns. These findings reinforce many common conceptions: the benefits of software security investment when considering the total cost of ownership; the insurmountability of securing truly flawed software; the value of formal methods in certain contexts; and the reduction in post-deployment security demands given investments in pre-deployment security measures. In addition, non-obvious insights emerged regarding the value of balanced pre- and post-deployment investment approaches and the effectiveness of SwSec. These initial findings provide a tantalising view of a largely unexplored topic, while limitations (discussed in Section 7.2) and unexplored aspects of this approach offer a vast research landscape. Challenges facing the development of information security economic models for secure software engineering are presented in [28], and a research agenda for further investigation is presented in [32]. ***Contribution:*** *Establishment of an approach and initial models to explicitly consider pre- and post-deployment security (e.g. SwSec and enterprise investments). Analysis of lifecycle models for insights into the nature of economically efficient software security practice, and the development of a metric (Return*

on Secure Software Process) to quantify these results.

- Armed with methods to identify and balance efficient pre- and post-deployment security investment, the outcome of such an analysis must be integrated into the secure software engineering process in order to derive the desired security outcomes. Chapter 5 examines how information security economic analyses can be represented and utilised within the Systems Development Life Cycle (SDLC). This is framed by NIST SP 800-160, in order to demonstrate how economic approaches contribute to the definition of "adequate security" and a robust basis for evaluation.

  In conjunction with an analysis of information security investment, development of security goals and an attacker model are essential to defining the Problem Context. Threat, loss, vulnerability and uncertainty estimates establish the parameters for investment, while specific threat models can drive the investment approach (e.g. the IWL's weakest-link attacker). Accompanying the threat model with utility-based objectives allows for the development and specification of alternative security goals (such as deterrence, as illustrated in [29]) and a means to express and evaluate the current security approach. This information can be applied further in the development process through the extension of accepted requirements techniques, such as use/misuse cases (as discussed in [31]), and employed to evaluate security goals against other system requirements and constraints (as demonstrated in [30]). Speculation as to the utilisation of this information within architecture and design processes — and beyond — concludes this chapter. ***Contribution:*** *Development and demonstration of methodologies to explicitly incorporate information security economic concerns within secure software development practice.*

Additionally, Chapter 6 demonstrated the employment of these concepts to an example programme inspired by a system currently under development. This analysis provides a brief summary of the application of these concepts within a traditional (secure) software development lifecycle. While not a formal and tested process, the aspects of this approach contribute to the development of a broader economics of software security, as discussed in [32].

7.2 Assumptions & Limitations

There remain numerous barriers to the realisation of a robust economics of secure software engineering practice. This section outlines the criticisms, unanswered questions, and outstanding assumptions that limit the employment of these research results.

There are some who challenge the fundamental premise of quantified cybersecurity approaches (e.g. [118]). Short of this view, valid questions have been raised regarding the interpretation of models [212] and ability to project [213] any human activity — let alone one as shrouded in mystique as computer security. Rather than dismiss the proposed approach, the message is one of the messiness of probability estimation, the existence of 'black swans', and the need to understand the role of uncertainty. Others question the employment of specific metrics, such as return on investment [214], as appropriate in the context of security. Ultimately, the message is one of understanding the scope and limitations of the analysis — echoing the provided guidance to validate, triangulate, and interpret results.

Research in software security investments is stymied by a lack of empirical data upon which to develop, evaluate, compare and employ security practices [69]. Software security, much like information security economics, is a recent area of focused investigation that emerged from years of related research. Software quality underpins many of the approaches to software security, with reviews and evaluations serving as a mechanism for imposing security into development processes such as requirements, design, and implementation. While some of the copious data on software process can be extrapolated to the security domain (for instance, the correspondence between code reviews for quality versus security-oriented code reviews), other areas have only begin to receive attention by the software engineering community (either due to recent inception, such as static code analysis, or differentiation in approach, such as risk-based testing). While the phenomena of information security breaches — and subsequent legal and societal attention that has resulted — has provided a windfall in the amount of data regarding information security costs, this data must be further refined to delineate cause, effect, and correlation between attacks, failures, and defences. There is reason to believe that the emerging cybersecurity insurance sector will correct this deficiency [215].

While the challenge of complete, concise, and relevant data underpins much of this research, additional assumptions limit the applicability of specific aspects:

- As an analysis of a specific cybersecurity policy, generalisation of the results from

Chapter 3 must be undertaken with care. Building on this approach through repetition on a broad array of policies — especially those formed on a different basis, such as the risk-based NIST Cybersecurity Framework [72] — would foster a deeper understanding and more substantiated conclusions. With respect to Cyber Essentials, consideration of a broader set of controls (specifically, incorporation of controls 2 and 3) would provide a more comprehensive analysis — although, as demonstrated this is not necessary to identify limitations to economic viability. While numerous assumptions regarding the nature of the software, hardware, security controls, control efficiency, costs, and applicable attacks were made to enable the analysis, these are easily overcome relative to a specific situation through the employment of historical data using the same methodology.

- Modelling real processes — especially those involving human action — is often fraught with peril. Even with ideal datasets, the best that can be achieved is a model of the statistically average case, with some probability. In addition, these models themselves incorporate subtle assumptions; some of which may not be apparent on first glance:
  - GL-SSE employs the well-accepted assumption that SwSec reduces vulnerability — far from controversial on its face, but this assertion rests on the employed definition of vulnerability; this is further discussed in Section 7.2.2. For the presented example, it is further assumed that vulnerability reduction manifests along EAL levels with an associated cost escalation. The Gordon-Loeb model assumes risk neutrality in the defender's approach, the ramifications of which are addressed in Section 7.2.1.
  - In addition to the assumptions inherent within the IWL model (specifically, the independence of vulnerabilities and their distribution), IWL-SSE relies on further assumptions regarding the nature of secure development. One such assumption is that SwSec increases certainty regarding the vulnerabilities present in a given system (further explored in Section 7.2.2). While seemingly intuitive, no study could be found to confirm this assertion. Furthermore, the related assumptions of independence, linearity of effectiveness and correctness of remediation in the employment of SwSec obscure complex relationships between related vulnerabilities, the skill of developers, and difficulty of fixing insecure implementations.

  - Both the GL-SSE and IWL-SSE models assume a linear software development process. In GL-SSE, this assumption is inherent in the derivation of the Security Breach Probability Function parameters from the COSECMO model, itself based on a dataset of programmes employing waterfall-style development. In IWL-SSE, the discrete time sequencing of the phases and growth model for remediation costs are coherent with sequential development. Relaxation of this constraint in IWL-SSE is explored in Section 7.3.
  - The majority of information security investment modelling approaches are predicated upon the Bernoulli Loss Assumption, which reduces an unknowable probability distribution (the probability of a given loss between zero and the maximum loss, itself a difficult quantity to estimate) to a singular probability of set loss. This discretisation renders the model tractable and understandable, but obscures the complexity of assigning a probability of loss — the fundamental goal of the Security Breach Probability Function in GL/GL-SSE, and the attacker-defender interplay within IWL/IWL-SSE. Relaxation of this assumption would create more realistic models at the expense of greater complexity and need for further data estimation.

- The application of software security economic approaches to 'everyday' projects remains limited by the amount of data and level of expertise required. Through the example presented in Chapter 6, the challenges of executing such a process — especially in the absence of tool support — are made obvious. The limited application of these techniques has yet to yield the gains expected by systematic employment; this is especially true in the case of generalised economic misuse cases. Due to this complexity, the environments under which functional constraints can be modelled — and subsequently analysed — may be limited to specific environments (e.g. IoT and other embedded systems).

In addition to these contribution-specific limitations, there are two important notions underlying many of these assumptions — themselves areas of intense research in their own right. These areas — risk and vulnerability — pervade the research, and are covered in the following subsections.

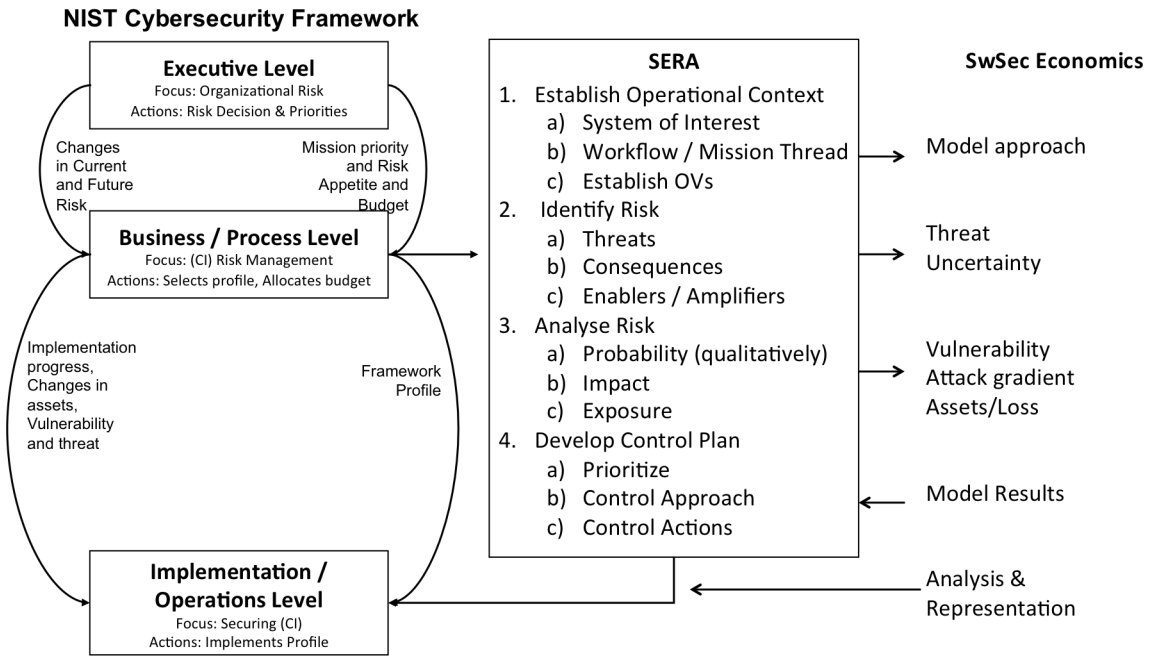


Figure 7.1: Representation of the interactions between the NIST Cybersecurity Framework, SERA, and relationship to software security economics.

7.2.1 Risk

Risk is at the core of modern information security approaches, including much of the widely-cited cybersecurity guidance (e.g. NIST Framework for Improving Critical Infrastructure Cybersecurity [72] and the Security Engineering Risk Analysis (SERA) Framework [216]). These approaches require the enumeration and ranking of risks, and the determination of which risks are acceptable. Risk-based approaches have been proposed throughout the SDLC, to varying degrees of success. Figure 7.1 depicts the relationship between the NIST and SERA framework, along with connections between the research presented in this dissertation[1]. This figure captures the varied levels of risk assessment necessary for a comprehensive approach to cybersecurity.

Risk-based security engineering both informs, and is informed by, an economic-based approach to secure software engineering. Data and measurements undertaken for risk analysis coincide with that needed for investment modelling, informing the model choice and many parameters. In addition to translating values such as 'threats' and 'consequences' into 'threat probability' and 'monetizable loss', a critical element is the specification of the

[1]The relationship between the SERA and NIST was validated with the SERA authors.

risk appetite. Risk appetites influence the model constructs, with risk-aversity resulting in concave utility representations while risk-seeking behaviours are represented by convex utility functions [217]. Such decisions alter the metric used in the optimisation of the outcome.

Likewise, the decision support provided by economic information security investment models can play a critical role in the identification and prioritisation of controls, a necessary element to meet the varied constraints of a programme. As with other security advice (e.g. the BSIMM), the focus of a risk-based approach is to specify the tasks necessary to achieving a secure system. The goal of this research is to supplement this advice with insight as to the investment required to achieve these goals.

7.2.2 Vulnerability

Vulnerability is a term for which numerous definitions exist, with subtle differences that impede a common understanding. Throughout this dissertation, the use of this term shifted slightly in adherence with the context of each contribution.

- In Chapter 3, vulnerability refers to the probability of a given class of breach agnostic to the attacker. This definition is specific to the type of threat, but non-specific to the system characteristics. Use of the Information Security Breach Survey to supply a probability for a specific type of attack against an unspecified system embodies this view.

- In Chapter 4, vulnerability is employed coherent with the Gordon-Loeb model (and subsequently GL-SSE). In this case, vulnerability refers to the probability of attack success — the success of an attack against any weakness. This is a complementary definition to that in Chapter 3, with vulnerability independent of attack, but specific to the system (or, information set) under examination.

- In Chapter 5, the use of the term vulnerability converges to the execution of a specific type of attack against a specific system weakness — the definition of vulnerability that is in common use by the Common Vulnerability and Exposures (CVE) database [121] and employed by vendor patching schemes.

The shifting definition can be illustrated using the probability of attack and probability of success distinction: Chapter 3 combines probability of attack with probability of success (the former being unknowable), while Chapter 4 focuses on probability of success (using

a separate term, t, to capture the probability of attack). Chapter 5 focuses on concrete terms, separating the existence of a flaw and opportunity for attack corresponding to that flaw. The over-loading of this term requires care when applying vulnerability metrics to information security investment modelling, and information security in general.

7.3 Future Directions

As an initial investigation into the problem of optimising information security expenditure throughout a system's lifecycle, work remains to realise a comprehensive economics of secure software development. Specific to the research presented in this dissertation, immediate paths for future research include:

- Given the lack of data, a primary direction for impactful research is the development of empirical studies that produce well-labeled data decoupling attacks, impacts, costs, and effectiveness. Such data could be used to further establish the relationship between IWL-SSE and GL-SSE parameters and reality (e.g. Δx and process maturity). Short of this, Butler *et al.* propose approaches to resolving the data–model gap [74].

- In the analysis of cybersecurity policy, consideration of a broader set of controls and comparison with data sources (such as made available by IBM and Ponemon [218]) would provide more robust analyses. Other benefits of achieving (or not achieving) a security accreditation should also be considered: on the negative end, to include lost business and reputation damage; and on the positive end, to include increased business opportunity, goodwill from consumers, or as a signal as part of a 'sheepskin effect' [219]. Such analyses would support a deeper understanding of the nature and role of cybersecurity policy.

- Numerous extensions and augmentations of the models presented in Chapter 4:

  - Generally, these models could be made more expressive and of higher fidelity (with a trade off in usability). In IWL-SSE, this could be supplied by richer compositions of error type, discovery stage, vulnerability source, cost variability and effectiveness both pre- and post-deployment. For GL-SSE, this could involve data-driven, purpose-developed security breach probability functions. This should be undertaken with care as increased SwSec representation risks

exacerbating issues with model complexity, rendering model selection and usage decisions more difficult [113].

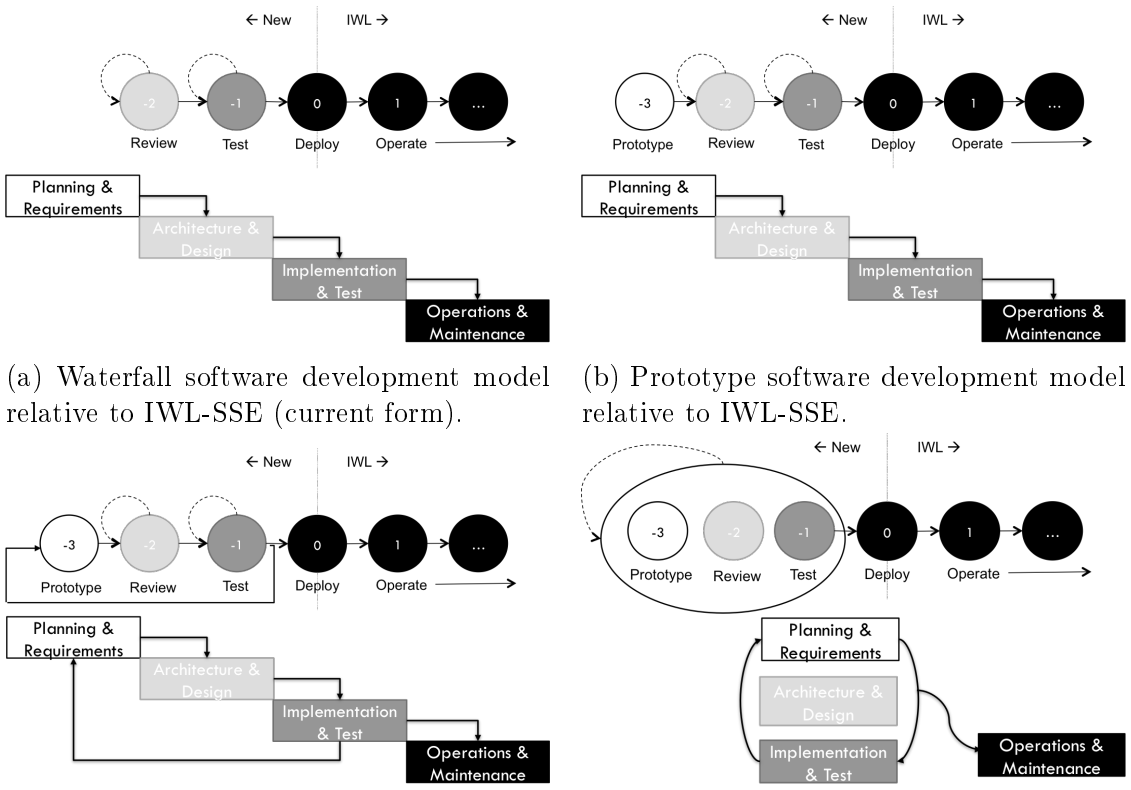
- Models assume stand-alone, greenfield development. Consideration of integrated defences and legacy systems — a major source of cybersecurity concern — would improve usability in real environments. Extension to coordinated strategies amongst sets of users while incorporating externalities and public goods offers the potential to reduce private over-spending in cybersecurity [220].
- A related point is the assumption that bugs and flaws correlate directly to singular vulnerabilities, when in fact this relationship is likely more complex [221]. Explicit incorporation of fix correctness, and by extension development aptitude, would yield more precise estimates.
- Development of additional information security investment models that incorporate attacker concerns (other than weakest-link) would provide a much more robust toolbox. Coupling of other concerns, such as quality considerations under a 'ship and fix' mentality [123], may also provide fundamental insight.

- Chapter 6 applied a subset of these practices to a relatively small project. This analysis was primarily *ex post*, leaving an *ex ante* analysis that drives decision making, in a substantial project and with measured outcomes, a laudable goal.

- Following from Section 7.2.2, the development of a formal definition and metric for vulnerability that transcends its common usage is a core problem in security engineering. Relative to the application of this research, such a definition is key to the development of a mechanism to reduce the system vulnerability probability in economic models requisite to the gains achieved through the remediation of concrete (e.g. CVE-defined) vulnerabilities.

Worthy of specific attention, a previously identified limitation of both models is the assumption of a linear, phased development process. Extending to alternative software processes is a logical step, to which thought has already been given. For GL-SSE, this would require a re-evaluation of the proposed security break probability functions against data from projects employing different process methodologies. Such an endeavour could result in different functional representations, different parameters, or both. In effect, this has the same outcome as application of the model to internal, historical data.

In the case of IWL-SSE, the approach to extension is necessarily more complex, due to the discrete software process representation. Figure 7.2 depicts four common soft-



(a) Waterfall software development model relative to IWL-SSE (current form).

(b) Prototype software development model relative to IWL-SSE.

(c) Iterative software development model relative to IWL-SSE.

(d) Agile software development model relative to IWL-SSE.

Figure 7.2: Depictions of different software development models (from [3]) relative to the IWL-SSE model.

ware process models, with Figure 7.2a representative of the current (baseline) IWL-SSE configuration. Considering each individually:

- *Prototype processes* (Figure 7.2b). Under the assumption of a prototype developed for the purpose of specifying system requirements, this can be conceptualised as targeting the system's vulnerability uncertainty, σ. In order to consider the effects this has on security investment, a few additional parameters are considered:
  - Prototype development incurs a one-time cost, I_{R}.
  - Differing prototypes may represent some or all of the overall system, with coverage denoted by the *prototype coverage* f: $0 < \mathrm{f} \leq 1$. The value of f would never equal 0, as that would imply a prototype with no correspondence to the system in development.

– Prototype code may be discarded, used in a limited fashion, or form the basis for the final development, resulting in a measure of *prototype reuse* g: $0 \leq \mathrm{g} \leq 1$. For example, a throwaway prototype (where no code is employed in the final system) has $\mathrm{g} = 0$.

Under an assumption that the prototype code is developed devoid of SwSec process, it is expected to have a negative impact on the overall codebase from a security perspective. This could be modelled as a decrease in the attack gradient Δx, proportional to the percentage of the covered, reused codebase in the final development: $\mathrm{p} = \mathrm{f} \cdot \mathrm{g}$. Development of the prototype itself can be modelled as a pre-design phase $t = -3$ with a cost c_{R}, which combines with c_{AD} and c_{IT} (the cost of conducting a review or test, respectively) into the overall cost of the software process, I_{P}:

$$I_{\mathrm{P}} = I_{\mathrm{R}} + I_{\mathrm{AD}} + I_{\mathrm{IT}} \tag{7.1}$$

with

$$I_{\mathrm{R}} = \quad (1 - \mathrm{p}) \cdot c_{\mathrm{R}} \tag{7.2}$$

Benefits of the prototype investment, relevant to the assumptions above, relate the unexamined portion of system requirements and the negative security impact of reused prototype code:

$$\sigma = (1 - \mathrm{f}) \cdot \sigma_{\mathrm{AD}} + (1 + \mathrm{p}) \cdot \sigma_{\mathrm{IT}} \tag{7.3}$$

This model captures a reduction of review uncertainty proportional to f, with a requisite increase in σ_{IT}. This speaks to a decrease in design uncertainty brought by the prototype, traded against negative effects of prototype reuse on the codebase (due to a lack of security process application). In a similar manner, the attack gradient is affected by the proportion of code reuse, with the result of raising the starting condition proportionally to reused code:

$$\Delta x = \sqrt{((1 + \mathrm{p}) + \alpha i_{\mathrm{AD}} + \beta i_{\mathrm{IT}})} \tag{7.4}$$

As a speculative model sketch, this formulation is based solely on the experience of developing IWL-SSE. It serves only as a notional approach; development of a robust model variant requires further study and analysis.

- An iterative process (Figure 7.2c) might on one hand be conceptualised as a multiple-run version of the SSE portion of the IWL-SSE, where the results from the previous iteration influence those of the next phase (thereby carrying along the presence, or absence, or good SwSec practice). Such an approach requires a re-evaluation of the cost escalation factors between phases. Alternatively, an iterative process might be conceived as a variant of the prototype process described above, where the prototype is of higher quality with full re-use (g = 1) and p is dictated by the requirements delta between individual phases. Presumably, this would require different calculations of σ and Δx that reflect a greater emphasis on security.

- Agile practices — notionally depicted in Figure 7.2d — might be modelled as the logical conclusion of this line of reasoning (in much the same way agile is seen as the logical conclusion of the need to reduce barriers between phases). An interesting question arises as to the cost escalations, if any, that result in the identification of security bugs and flaws in subsequent iterations of the process (e.g. sprints). Estimations of this kind are essential to modelling such processes.

Finally, IWL-SSE could be extended to account for the introduction of vulnerabilities during or after deployment (e.g. misconfigurations). Extension of the IWL-PT model [177] to include SSE constructs would be a natural step in this direction.

A lack of data currently hampers further analysis of such model variants. Empirical analysis of various development efforts, with a focus on the costs and benefits of security practices within such models, could add the necessary rigour to formalise and validate these model sketches.

In the broadest sense, realising economically-informed secure software practices requires further study into the three areas identified in [32]:

- *Models.* SwSec investment modelling requires the development of models that capture the essence of software process contribution. Due to the variability and dependency in 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 [134; 133; 35], in order to properly tune and validate model assumptions.

- *Representation.* Once 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, methods and models for juxtaposing security and functionality in order to inform investment are under-developed. Developing more robust methods and extending these concepts to further development practices (e.g. architecture — as discussed briefly in Section 5.4 — or risk-driven testing) will support a more comprehensive approach. Utility-based characterisations may integrate functional and security constructs in order to enable decision making within a variety of approaches; however, the issue of integration with risk-based processes (as discussed in Section 7.2.1) and the evaluation of these concepts as a coherent practice are the next step to realising this vision.

- *Analysis.* Critical to rationalising SwSec investment is the ability to identify the contribution. Starting with the execution of the effort, 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. SwSec analysis requires methods that can incorporate the varied indicators more effectively than current risk-based methods, yet retain compatibility with business practice. Just as a lack of security expertise amongst developers hinders software security [188], a lack of software engineering expertise will hinder the employment of these techniques.

Research in these 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.

As SwSec moves from purely empirical considerations toward the establishment of theoretical underpinnings, this research supports a deeper understanding of how software process impacts overall security goals. Applying these tenets to established approaches is essential to practical application, and will ultimately drive their adoption and impact. The combination of software engineering, statistical analysis, and security practice provides a

basis for a true science of software security — and increasingly the necessary conditions for practical security — with information economics poised to supply critical insights.

List of References

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

[2] Sindre, G., Opdahl, A.L. and Brevik, G.F.: Generalization/Specialization as a structuring mechanism for misuse cases. In: *Proceedings of the 2nd Symposium on Requirements Engineering for Information Security (SREIS)*, pp. 3:1–3:16. October 2002.

[3] Pressman, R.: *Software Engineering: A Practitioner's Approach.* 8th edn. McGraw-Hill, Inc., New York, NY, USA, 2014.

[4] McGraw, G., Migues, S. and West, J.: Building security in maturity model (BSIMM). Tech. Rep., Synopsis, Inc., September 2017.
Available at: `https://www.bsimm.com/`

[5] Sastry, N. and Wagner, D.: Security considerations for IEEE 802.15.4 networks. In: *Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe)*, pp. 32–42. October 2004.

[6] Armerding, T.: Is security really stuck in the dark ages? Website, May 2015.
Available at: `http://www.csoonline.com/article/2925351/data-protection/is-security-really-stuck-in-the-dark-ages.html`

[7] Ross, R., McEvilley, M. and Carrier Oren, J.: *SP 800-160 — Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*, November 2016.
Available at: `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160.pdf`

[8] McGraw, G.: Software security. *IEEE Security & Privacy*, vol. 2, no. 2, pp. 80–83, March 2004.

[9] Anderson, R. and Moore, T.: The economics of information security. *Science*, vol. 314, no. 5799, pp. 610–613, October 2006.

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

[11] HM Government: Information security breaches survey 2015. PDF, June 2015. Available at: `https://www.gov.uk/government/publications/information-security-breaches-survey-2015`

[12] Forni, A. and van der Meulen, R.: Press release: Gartner says detection and response is top security priority for organizations in 2017. Website, March 2017. Available at: `http://www.gartner.com/newsroom/id/3638017`

[13] Filkins, B.: IT security spending trends. PDF, February 2016. Available at: `https://www.sans.org/reading-room/whitepapers/analyst/security-spending-trends-36697`

[14] Kupsch, J.A. and Miller, B.P.: Manual vs. automated vulnerability assessment: A case study. In: *Proceedings of the 1st International Workshop on Managing Insider Security Threats (MIST) West*, pp. 83–97. June 2009.

[15] 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*, vol. 35, no. 4, pp. 534–550, July 2009.

[16] Al-Humaigani, M. and Dunn, D.B.: A model of return on investment for information systems security. In: *Proceedings of the IEEE International Symposium on Micro-Nano Mechatronics and Human Science (MHS)*, pp. 483–485. November 2003.

[17] Aslam, T., Krsul, I. and Spafford, E.: Use of a taxonomy of security faults. Tech. Rep. 96-051, Purdue University, September 1996. Available at: `http://docs.lib.purdue.edu/cstech/1305`

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

[19] Hein, D. and Saiedian, H.: Secure software engineering: Learning from the past to address future challenges. *Information Security Journal: A Global Perspective*, vol. 18, no. 1, pp. 8–25, February 2009.

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

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

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

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

[24] Heitzenrater, C. and Simpson, A.C.: Policy, statistics, and questions: Reflections on UK cyber security disclosures. In: *Proceedings of the 14th Workshop on the Economics of Information Security (WEIS)*, pp. 37–50. June 2015.

[25] Heitzenrater, C. and Simpson, A.C.: Policy, statistics, and questions: Reflections on UK cyber security disclosures. *Journal of Cybersecurity*, vol. 2, no. 1, pp. 43–56, December 2016.

[26] Heitzenrater, C. and Simpson, A.C.: Modelling software security investment. *Under preparation*.

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

[28] Heitzenrater, C. and Simpson, A.C.: Software security investment: The right amount of a good thing. In: *Proceedings of the 1st IEEE Cybersecurity Development Conference (SecDev)*. November 2016.

[29] Heitzenrater, C., Taylor, G. and Simpson, A.C.: When the winning move is not to play: Games of deterrence in cyber security. In: *Proceedings of the 6th International Conference on Decision and Game Theory for Security (GameSec)*. November 2015.

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

[31] Heitzenrater, C. and Simpson, A.C.: Misuse, abuse, and reuse: Economic utility functions for characterising security requirements. In: *Proceedings of The 2nd International Workshop on Agile Secure Software Development (ASSD)*. August 2016.

[32] Heitzenrater, C. and Simpson, A.C.: A case for the economics of secure software development. In: *Proceedings of the New Security Paradigms Workshop (NSPW)*. September 2016.

[33] Avizienis, A., Laprie, J., Randell, B. and Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, January 2004.

[34] *Military Handbook 1785: System security engineering program management requirements*, August 1995.
Available at: `http://quicksearch.dla.mil/qsDocDetails.aspx?ident_number=121174`

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

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

[37] Kissel, R., Stine, K.M., Scholl, M.A., Rossman, H., Fahlsing, J. and Gulick, J.: *SP 800-64 Rev. 2. — Security Considerations in the System Development Life Cycle*, October 2008.
Available at: `http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-64r2.pdf`

[38] van Wyk, K.R. and McGraw, G.: Bridging the gap between software development and information security. *IEEE Security & Privacy*, vol. 3, no. 5, pp. 75–79, September 2005.

[39] *Common Criteria for Information Technology Security Evaluation – Part 1: Introduction and general model*, April 2017.
Available at: `https://www.commoncriteriaportal.org/`

[40] *Payment Card Industry (PCI) Data Security Standard*, April 2016.
Available at: `https://www.pcisecuritystandards.org`

[41] Jaatun, M.G.: Hunting for aardvarks: Can software security be measured? In: *Proceedings of the IFIP International Cross-Domain Conference and Workshop (CD-ARES)*, vol. 7465 of *Lecture Notes in Computer Science*, pp. 85–92. Springer, August 2012.

[42] James, J.A.: Raising the bar for cybersecurity. Tech. Rep., Center for Strategic & International Studies, Technology & Public Policy, February 2013.
Available at: `https://www.csis.org/analysis/raising-bar-cybersecurity`

[43] Howard, M. and Lipner, S.: *The Security Development Lifecycle*. Microsoft Press, Redmond, WA, 2006.

[44] White paper: Building a culture of security. PDF, September 2016.
Available at: `http://www.adobe.com/content/dam/acom/en/security/pdfs/adb_security-culture-wp.pdf`

[45] Software development life cycle (SDLC). Website, 2017.
Available at: `https://www.synopsys.com/software-integrity/resources/knowledge-database/software-development-life-cycle.html`

[46] 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*, vol. 51, no. 7, pp. 1152–1171, July 2009.

[47] Butler, S.A.: Security attribute evaluation method: A cost-benefit approach. In: *Proceedings of the 24th International Conference on Software Engineering (ICSE)*, pp. 232–240. May 2002.

[48] Fléchais, I.: *Designing Secure and Usable Systems.* Ph.D. thesis, University of London, February 2005.

[49] Carin, L., Cybenko, G. and Hughes, J.: Cybersecurity strategies: The QuERIES methodology. *IEEE Computer*, vol. 41, no. 8, pp. 20–26, August 2008.

[50] Carin, L., Cybenko, G. and Hughes, J.: Quantitative evaluation of risk for investment efficient strategies in cybersecurity: The QuERIES methodology. Tech. Rep., Office of the Deputy Under Secretary Defense (Science & Technology) Software Protection Initiative (SPI), September 2007.
Available at: `http://www.securitymetrics.org/attachments/Metricon-3-Cybenko-Article.pdf`

[51] Verdon, D. and McGraw, G.: Risk analysis in software design. *IEEE Security & Privacy*, vol. 2, no. 4, pp. 79–84, July 2004.

[52] Mohammed, N.M., Niazi, M., Alshayeb, M. and Mahmood, S.: Exploring software security approaches in software development lifecycle: A systematic mapping study. *Computer Standards & Interfaces*, vol. 50, pp. 107–115, February 2017.

[53] Faily, S.: *A framework for usable and secure system design.* Ph.D. thesis, University of Oxford, Oxford, UK, 2011.

[54] Xie, N., Mead, N., Chen, P., Dean, M., Lopez, L., Ojoko-Adams, D. and Osman, H.: SQUARE project: Cost/Benefit analysis framework for information security improvement projects in small companies. Tech. Rep. CMU/SEI-2004-TN-045, Software Engineering Institute, Carnegie Mellon University, November 2004.
Available at: `http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7013`

[55] Tsiakis, T. and Stephanides, G.: The economic approach of information security. *Computers & Security*, vol. 24, no. 2, pp. 105–108, March 2005.

[56] 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*, vol. 2742 of *Lecture Notes in Computer Science (LNCS)*, pp. 122–137. Springer-Verlag, 2003.

[57] Roy, S., Ellis, C., Shiva, S., Dasgupta, D., Shandilya, V. and Wu, Q.: A survey of game theory as applied to network security. In: *Proceedings of the 43rd Hawaii International Conference on System Sciences (HICSS)*. January 2010.

[58] Hoo, K.J.S.: *How much is enough? A risk management approach to computer security*. Ph.D. thesis, Stanford University, January 2000.

[59] 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 Workshop on the Economics of Information Security (WEIS)*. June 2013.

[60] OWASP top 10 2013: The ten most critical web application security risks. PDF, June 2013.
Available at: `https://www.owasp.org/index.php/Top_10_2013`

[61] OWASP proactive controls for developers 2016. PDF, 2016.
Available at: `https://www.owasp.org/images/0/07/OWASP_Proactive_Controls_v1.pdf`

[62] Howard, M., LeBlanc, D. and Viega, J.: *24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them*. 1st edn. McGraw-Hill, Inc., New York, NY, USA, 2010.

[63] 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, August 2014.
Available at: `http://cybersecurity.ieee.org/images/files/images/pdf/CybersecurityInitiative-online.pdf`

[64] 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.
Available at: `http://www.safecode.org/publication/SAFECode_Principles_for_Software_Assurance_Assessment.pdf`

[65] McGraw, G.: A breakdown of the BSIMM–V with Caroline Wong. Podcast, October 2013.
Available at: `https://itunes.apple.com/us/podcast/cigital-silver-bullet-security/id154782182?mt=2`

[66] Caralli, R., Stevens, J., Young, L. and Wilson, W.: Introducing OCTAVE Allegro: Improving the information security risk assessment process. Tech. Rep. CMU/SEI-2007-TR-012, Software Engineering Institute, Carnegie Mellon University, May 2007.
Available at: `http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8419`

[67] Software assurance maturity model: A guide to building security into software development. PDF, March 2009.
Available at: `http://www.opensamm.org/downloads/SAMM-1.0.pdf`

[68] Building security in maturity model (BSIMM). Website. `https://www.bsimm.com/`.

[69] Morrison, P., Smith, B.H. and Williams, L.: Surveying security practice adherence in software development. In: *Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp (HoTSoS)*, pp. 85–94. April 2017.

[70] McGraw, G.: Seven myths of software security best practices. Website, October 2015.
Available at: `http://searchsecurity.techtarget.com/opinion/McGraw-Seven-myths-of-software-security-best-practices`

[71] Morgan, S.: Is poor software development the biggest cyber threat? Website, September 2015.
Available at: `https://www.csoonline.com/article/2978858/application-security/is-poor-software-development-the-biggest-cyber-threat.html`

[72] *Framework for Improving Critical Infrastructure Cybersecurity*, February 2014.
Available at: `http://www.nist.gov/cyberframework/`

[73] *ISO/IEC:27001:2013 — Information technology — Security techniques — Information security management systems — Requirements.*, October 2013.

[74] Butler, S., Jha, S. and Shaw, M.: When good models meet bad data: Applying quantitative economic models to qualitative engineering judgments. In: *Proceedings 2nd International Workshop on Economics-Driven Software Engineering Research (EDSER-2)*. June 2000.

[75] Schechter, S.E.: Toward econometric models of the security risk from remote attack. *IEEE Security & Privacy*, vol. 3, no. 1, pp. 40–44, May 2005.

[76] Anderson, R.J. and Schneier, B.: Guest editors' introduction: Economics of information security. *IEEE Security & Privacy*, vol. 3, no. 1, pp. 12–13, January–February 2005.

[77] Grimes, R.A.: Implementing a data-driven computer security defense. Tech. Rep., Microsoft IT Information Security and Risk Management, November 2015.
Available at: `https://gallery.technet.microsoft.com/Fixing-the-1-Problem-in-2e58ac4a`

[78] Anderson, R.: Why information security is hard: An economic perspective. In: *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC)*, pp. 358–365. December 2001.

[79] Moore, T. and Anderson, R.: Economics and internet security: A survey of recent analytical, empirical and behavioral research. Tech. Rep. TR-03-11, Computer Science Group, Harvard University, January 2011.
Available at: `https://dash.harvard.edu/handle/1/23574266`

[80] Anderson, R. and Moore, T.: The economics of information security. In: *Proceedings of the 4th Bi-annual Conference on the Economics of the Software and Internet Industries*. January 2007.

[81] Armstrong, M.: Competition in two-sided markets. *The RAND Journal of Economics*, vol. 37, no. 3, pp. 668–691, September 2006.

[82] Klemperer, P.: Network effects and switching costs: Two short essays for the New Palgrave. Tech. Rep. 2006-W06, Economics Group, Nuffield College, University of Oxford, March 2005.
Available at: `http://www.nuff.ox.ac.uk/users/klemperer/NewPalgrave.pdf`

[83] Varian, H.: Versioning of information goods. Tech. Rep., University of California, Berkeley, March 1997.
Available at: `http://people.ischool.berkeley.edu/~hal/Papers/version.pdf`

[84] Varian, H.R.: *Intermediate Microeconomics: A Modern Approach*. 8th edn. W. W. Norton & Company, New York, NY, 2010.

[85] Thaler, R.H.: *Misbehaving: The Making of Behavioral Economics*. W. W. Norton & Company, New York, NY, 2015.

[86] Hoo, K.S.: Information security: Why the future belongs to the quants. *IEEE Security & Privacy*, vol. 1, no. 4, pp. 24–32, July 2003.

[87] Odlyzko, A.: Providing security with insecure systems. In: *Proceedings of the 3rd ACM Conference on Wireless Network Security (WiSec)*. March 2010.

[88] Böhme, R. and Nowey, T.: *Dependability Metrics: Advanced Lectures*, chap. Economic Security Metrics, pp. 176–187. Springer Berlin Heidelberg, 2008.

[89] National Institute of Standards & Technology (NIST): *Federal Information Processing Standards: Guidelines for Automatic Data Processing Risk Analysis (NIST FIPS–65)*, August 1975.

[90] Cantor, M.: Calculating and improving ROI in software and system programs. *Communications of the ACM*, vol. 54, no. 9, pp. 121–130, September 2011.

[91] Mercuri, R.T.: Analyzing security costs. *Communications of the ACM*, vol. 46, no. 6, pp. 15–18, June 2003.

[92] Wang, J., Chaudhury, A. and Rao, H.R.: A value-at-risk approach to information security investment. *Information Systems Research*, vol. 19, no. 1, pp. 106–120, March 2008.

[93] Sonnenreich, W., Albanese, J. and Stout, B.: Return on security investment (ROSI): A practical quantitative model. *Journal of Research and Practice in Information Technology*, vol. 38, no. 1, pp. 239–252, January 2005.

[94] Mizzi, A.: Return on information security investment: The viability of an anti-spam solution in a wireless environment. *International Journal of Network Security*, vol. 10, no. 1, pp. 18–24, January 2010.

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

[96] Pfleeger, S.L. and Rue, R.: Cybersecurity economic issues: Clearing the path to good practice. *IEEE Software*, vol. 25, no. 1, pp. 35–42, January 2008.

[97] Schatz, D. and Bashroush, R.: Economic valuation for information security investment: A systematic literature review. *Information Systems Frontiers*, vol. 19, no. 5, pp. 1205–1228, October 2017.

[98] Gordon, L.A. and Loeb, M.P.: The economics of information security investment. *ACM Transactions on Information and Systems Security*, vol. 5, no. 4, pp. 438–457, November 2002.

[99] Baryshnikov, Y.: IT security investment and Gordon-Loeb's $1/e$ rule. In: *11th Annual Workshop on the Economics of Information Security (WEIS)*. June 2012.

[100] Willemson, J.: On the Gordon & Loeb model for information security investment. In: *Proceedings of the 5th Annual Workshop on the Economics of Information Security (WEIS)*. June 2006.

[101] Hausken, K.: Returns to information security investment: The effect of alternative information security breach functions on optimal investment and sensitivity to vulnerability. *Information Systems Frontiers*, vol. 8, no. 5, pp. 338–349, December 2006.

[102] Willemson, J.: Extending the Gordon and Loeb model for information security investment. In: *Proceedings of the 5th International Conference on Availability, Reliability and Security (ARES)*, pp. 258–261. February 2010.

[103] Huang, C.D., Hu, Q. and Behara, R.S.: An economic analysis of the optimal information security investment in the case of a risk-averse firm. *International Journal of Production Economics*, vol. 114, no. 2, pp. 793–804, August 2008.

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

[105] Ioannidis, C., Pym, D. and Williams, J.: Investments and trade-offs in the economics of information security. In: Dingledine, R. and Golle, P. (eds.), *Financial Cryptography and Data Security*, vol. 5628 of *Lecture Notes in Computer Science (LNCS)*, pp. 148–166. Springer Berlin Heidelberg, 2009.

[106] Cavusoglu, H., Mishra, B. and Raghunathan, S.: A model for evaluating IT security investments. *Communications of the ACM*, vol. 47, no. 7, pp. 87–92, July 2004.

[107] Robbins, E.H., Hustus, H. and Blackwell, J.A.: Mathematical foundations of strategic deterrence. In: Lowther, A. (ed.), *Thinking About Deterrence: Enduring Questions in a Time of Rising Powers, Rogue Regimes, and Terrorism*, pp. 137–165. Air University Press, 2013.

[108] Moore, T., Friedman, A. and Procaccia, A.D.: Would a 'cyber warrior' protect us: Exploring trade-offs between attack and defense of information systems. In: *Proceedings of the New Security Paradigms Workshop (NSPW)*, pp. 85–94. 2010.

[109] Varian, H.: System reliability and free riding. In: Camp, L.J. and Lewis, S. (eds.), *Economics of Information Security*, vol. 12 of *Advances in Information Security*, pp. 1–15. Springer, 2004.

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

[111] Fultz, N. and Grossklags, J.: Blue versus red: Towards a model of distributed security attacks. In: Dingledine, R.and Golle, P. (ed.), *Financial Cryptography and Data Security*, vol. 5628 of *Lecture Notes in Computer Science (LNCS)*, pp. 167–183. Springer Berlin Heidelberg, 2009.

[112] Böhme, R. and Moore, T.: The iterated weakest link: A model of adaptive security investment. In: *Proceedings of the 8th Workshop on the Economics of Information Security (WEIS)*. June 2009.

[113] Rue, R. and Pfleeger, S.L.: Making the best use of cybersecurity economic models. *IEEE Security & Privacy*, vol. 7, no. 4, pp. 52–60, July 2009.

[114] AFCEA International Cyber Committee: The economics of cybersecurity: A practical framework for cybersecurity investment. Tech. Rep., Armed Forces Communicaitons & Electronics Association, October 2013.

[115] Dor, D. and Elovici, Y.: A model of the information security investment decision-making process. *Computers & Security*, vol. 63, no. Supplement C, pp. 1 – 13, 2016.

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

[117] Cremonini, M. and Nizovtsev, D.: Understanding and influencing attackers' decisions: Implications for security investment strategies. In: *Proceedings of the 5th Workshop on the Economics of Information Security (WEIS)*. June 2006.

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

[119] Pandey, P.: "Context, content, process" approach to align information security investments with overall organizational strategy. *International Journal of Security, Privacy and Trust Management (IJSPTM)*, vol. 4, no. 3/4, pp. 25–38, November 2015.

[120] 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 Workshop on the Economics of Information Security (WEIS)*. June 2007.

[121] Common vulnerabilities and exposures (CVE). Website. `http://cve.mitre.org/`.

[122] National vulnerability database (NVD). Website. `https://nvd.nist.gov/`.

[123] Arora, A., Caulkins, J.P. and Telang, R.: Research note–Sell first, fix later: Impact of patching on software quality. *Management Science*, vol. 52, no. 3, pp. 465–471, March 2006.

[124] Arora, A., Telang, R. and Xu, H.: Optimal policy for software vulnerability disclosure. *Management Science*, vol. 54, no. 4, pp. 642–656, April 2008.

[125] Armin, J., Foti, P. and Cremonini, M.: 0-Day vulnerabilities and cybercrime. In: *Proceedings of the 10th International Conference on Availability, Reliability and Security (ARES)*, pp. 711–718. August 2015.

[126] Edmundson, A., Holtkamp, B., Rivera, E., Finifter, M., Mettler, A. and Wagner, D.: An empirical study on the effectiveness of security code review. In: Jürjens, J., Livshits, B. and Scandariato, R. (eds.), *International Symposium on Engineering Secure Software and Systems (ESSoS)*, vol. 7781 of *Lecture Notes in Computer Science (LNCS)*, pp. 197–212. Springer Berlin Heidelberg, 2013.

[127] Wurster, G. and van Oorschot, P.C.: The developer is the enemy. In: *Proceedings of the New Security Paradigms Workshop (NSPW)*, pp. 89–97. September 2008.

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

[129] Ozment, A. and Schechter, S.E.: Milk or wine: Does software security improve with age? In: *Proceedings of the 15th Conference on USENIX Security Symposium (USENIX-SS)*, vol. 15. USENIX Association, August 2006.

[130] Caulkins, J., Osman, H., Hough, E.D. and Mead, N.R.: Optimizing investments in security countermeasures: A practical tool for fixed budgets. *IEEE Security & Privacy*, vol. 5, no. 5, pp. 57–60, October 2007.

[131] Gordon, L., Loeb, M. and Zhou, L.: Investing in cybersecurity: Insights from the Gordon-Loeb model. *Journal of Information Security*, vol. 7, no. 2, pp. 49–59, March 2016.

[132] Tanaka, H., Matsuura, K. and Sudoh, O.: Vulnerability and information security investment: An empirical analysis of e-local government in Japan. *Journal of Accounting and Public Policy*, vol. 24, no. 1, pp. 37–59, January-February 2005.

[133] Baca, D., Carlsson, B. and Lundberg, L.: Evaluating the cost reduction of static code analysis for software security. In: *Proceedings of the 3rd ACM SIGPLAN Workshop on Programming Languages and Analysis for Security (PLAS)*, pp. 79–88. June 2008.

[134] Goseva-Popstojanova, K. and Perhinschi, A.: On the capability of static code analysis to detect security vulnerabilities. *Information and Software Technology*, vol. 68, no. C, pp. 18–33, December 2015.

[135] McGraw, G. and Routh, J.: Software [In]security: Measuring software security. Website, June 2009.
Available at: `http://www.informit.com/articles/article.aspx?p=1357183`

[136] Aminnezhad, A., Mahmod, R. and Abdullah, M.T.: Survey on economics of information security. *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 7, p. 99, 2016.

[137] Woody, C. and Alberts, C.: Considering operational security risk during system development. *IEEE Security & Privacy*, vol. 5, no. 1, pp. 30–35, January 2007.

[138] Panaousis, E., Fielder, A., Malacaria, P., Hankin, C. and Smeraldi, F.: Cybersecurity games and investments: A decision support approach. In: Poovendran, R. and Saad, W. (eds.), *Proceedings of the 5th International Conference on Decision and Game Theory for Security (GameSec)*, vol. 8840 of *Lecture Notes in Computer Science (LNCS)*, pp. 266–286. Springer International Publishing, 2014.

[139] McGraw, G.: Software [In]security: Paying for secure software. Website, April 2008.
Available at: `http://www.informit.com/articles/article.aspx?p=1189519`

[140] Department for Business, Innovation and Skills: Information security breaches survey 2012. PDF, April 2012.
Available at: `http://www.pwc.co.uk/audit-assurance/publications/uk-information-security-breaches-survey-results-2012.jhtml`

[141] Department for Business, Innovation and Skills: Information security breaches survey 2013. PDF, April 2013.
Available at: `https://www.gov.uk/government/publications/information-security-breaches-survey-2013-technical-report`

[142] Department for Business, Innovation and Skills: Information security breaches survey 2014. PDF, April 2014.
Available at: `https://www.gov.uk/government/publications/information-security-breaches-survey-2014`

[143] Brecht, M. and Nowey, T.: A closer look at information security costs. In: Böhme, R. (ed.), *The Economics of Information Security and Privacy*, pp. 3–24. Springer Berlin Heidelberg, 2013.

[144] Imperva Application Defense Center: Hacker intelligence initiative, monthly trend report #14. PDF, December 2012.
Available at: `http://www.imperva.com/docs/HII_Assessing_the_Effectiveness_of_Antivirus_Solutions.pdf`

[145] Kührer, M., Rossow, C. and Holz, T.: Paint it black: Evaluating the effectiveness of malware blacklists. In: Stavrou, A., Bos, H. and Portokalidis, G. (eds.), *Proceedings of the 17th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, vol. 8688 of *Lecture Notes in Computer Science (LNCS)*, pp. 1–14. Springer, September 2014.

[146] Smith, M.: Patching windows is a major time sink for IT departments. Website, May 2011.
Available at: `http://www.csoonline.com/article/2925351/data-protection/is-security-really-stuck-in-the-dark-ages.html`

[147] Florêncio, D. and Herley, C.: Sex, lies and cyber-crime surveys. In: Schneier, B. (ed.), *Economics of Information Security and Privacy III*, pp. 35–53. Springer New York, 2013.

[148] Hulthén, R.: Communicating the economic value of security investments: Value at security risk. In: Johnson, M.E. (ed.), *Managing Information Risk and the Economics of Security*, pp. 121–140. Springer US, 2009.

[149] Romanosky, S.: Examining the costs and causes of cyber incidents. *Journal of Cybersecurity*, vol. 2, no. 2, pp. 121–135, December 2016.

[150] Schneier, B.: The cost of cyberattacks is less than you might think. Website, September 2016.
Available at: `https://www.schneier.com/blog/archives/2016/09/the_cost_of_cyb.html`

[151] Jacobs, J.: Analyzing ponemon cost of data breach. Website, December 2014.
Available at: `http://datadrivensecurity.info/blog/posts/2014/Dec/ponemon/`

[152] Wilson, T.: Poor priorities, lack of resources put enterprises at risk, security pros say. Website, July 2015.
Available at: `https://www.darkreading.com/vulnerabilities---threats/poor-priorities-lack-of-resources-put-enterprises-at-risk-security-pros-say/d/d-id/1321308`

[153] Moore, T.: The economics of cybersecurity: Principles and policy options. *International Journal of Critical Infrastructure Protection (IJCIP)*, vol. 3, no. 3-4, pp. 103–117, December 2010.

[154] Herley, C.: So long, and no thanks for the externalities: The rational rejection of security advice by users. In: *Proceedings of the New Security Paradigms Workshop (NSPW)*, pp. 133–144. September 2009.

[155] Boehm, B., Clark, B., Horowitz, E., Madachy, R., Shelby, R. and Westland, C.: Cost models for future software life cycle processes: COCOMOÂ® 2.0. *Annals of Software Engineering*, vol. 1, no. 1, pp. 57–94, December 1995.

[156] Colbert, E. and Boehm, B.: Cost estimation for secure software & systems. In: *Proceedings of the ISPA/SCEA Joint International Conference*. May 2008.

[157] Mead, N.R., Allen, J.H., Conklin, W.A., Drommi, A., Harrison, J., Ingalsbe, J., Rainey, J. and Shoemaker, D.: Making the business case for software assurance. Tech. Rep. CMU/SEI-2009-SR-001, Software Engineering Institute, April 2009.

[158] Laird, L.M.: The limitations of estimation. *IT Professional*, vol. 8, no. 6, pp. 40–45, November 2006.

[159] Jones, C.: A short history of the lines of code (LOC) metric, version 6.0. PDF, December 2012.
Available at: `http://www.ifpug.org/Documents/Jones-LinesofCodeMetricV6.pdf`

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

[161] Neill, C.J. and Laplante, P.A.: Requirements engineering: The state of the practice. *IEEE Software*, vol. 20, no. 6, pp. 40–45, November 2003.

[162] Neill, C.J. and Laplante, P.A.: Requirements engineering: The state of the practice revisited. PDF, 2008.
Available at: `https://www.projectsmart.co.uk/white-papers/requirements-engineering-the-state-of-the-practice-revisited.pdf`

[163] Farrow, S. and Szanton, J.: Cybersecurity investment guidance: Extensions of the Gordon and Loeb model. *Journal of Information Security*, vol. 7, no. 2, pp. 15–28, March 2016.

[164] Naldi, M. and Flamini, M.: Calibration of the Gordon–Loeb models for the probability of security breaches. In: *Proceedings of the 19th IEEE International Conference on Modelling and Simulation (UKSim-AMSS)*, pp. 135–141. April 2017.

[165] Yang, Y., Du, J. and Wang, Q.: Shaping the effort of developing secure software. *Procedia Computer Science*, vol. 44, pp. 609–618, 2015.

[166] Klein, G., Andronick, J., Elphinstone, K., Murray, T., Sewell, T., Kolanski, R. and Heiser, G.: Comprehensive formal verification of an OS microkernel. *ACM Transactions Computer Systems*, vol. 32, no. 1, pp. 2:1–2:70, February 2014.

[167] Witherspoon, L.: The Total Economic Impact of Synopsys software testing tools: Coverity and Defensics. Tech. Rep., Forrester Research Inc., September 2016.

[168] Novak, J., Krajnc, A. and Žontar, R.: Taxonomy of static code analysis tools. In: *Proceedings of the 33rd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 418–422. May 2010.

[169] Baca, D., Carlsson, B., Petersen, K. and Lundberg, L.: Improving software security with static automated code analysis in an industry setting. *Software: Practice and Experience*, vol. 43, no. 3, pp. 259–279, 2013.

[170] Okun, V., Delaitre, A. and Black, P.E.: *Special Publication (SP) 500-297: Report on the Static Analysis Tool Exposition (SATE) IV*, January 2013.
Available at: `https://samate.nist.gov/docs/NIST_Special_Publication_500-297.pdf`

[171] Nord, R.L. and Ozkaya, I.: Software vulnerabilities, defects, and design flaws: A technical debt perspective. Tutorial at the 1st IEEE Cybersecurity Development (SecDev), 2016.

[172] Baca, D., Petersen, K., Carlsson, B. and Lundberg, L.: Static code analysis to detect software security vulnerabilities – Does experience matter? In: *Proceedings of the 4th International Conference on Availability, Reliability, and Security (ARES)*, pp. 804–810. March 2009.

[173] Velicheti, L.M.R., Feiock, D.C., Peiris, M., Raje, R.R. and Hill, J.H.: Towards modeling the behavior of static code analysis tools. In: *Proceedings of the Cyber and Information Security Research Conference (CISR)*, pp. 17–20. April 2014.

[174] Sullivan, K.J., Notkin, D., Fuggetta, A. and Favaro, J.: 1st workshop on economics-driven software engineering research (EDSER-1). In: *Proceedings of the 21st International Conference on Software Engineering (ICSE)*, pp. 699–700. May 1999.

[175] Schneier, B.: Information security and externalities. *European Network and Information Security Agency (ENISA) Quarterly*, vol. 2, no. 4, pp. 3–4, January 2007.

[176] Böhme, R. and Moore, T.: The 'Iterated Weakest Link' model of adaptive security investment. *Journal of Information Security*, vol. 7, no. 2, pp. 81–102, March 2016.

[177] Böhme, R. and Félegyházi, M.: Optimal information security investment with penetration testing. In: *Decision and Game Theory for Security*, vol. 6442 of *Lecture Notes in Computer Science (LNCS)*, pp. 21–37. Springer Berlin Heidelberg, 2010.

[178] Bird, J.: Bugs and numbers: How many bugs do you have in your code? Website, August 2011.
Available at: `http://swreflections.blogspot.co.uk/2011/08/bugs-and-numbers-how-many-bugs-do-you.html`

[179] Boehm, B. and Basili, V.R.: Software defect reduction top 10 list. *IEEE Computer*, vol. 34, no. 1, pp. 135–137, January 2001.

[180] Peeters, J. and Dyson, P.: Cost-effective security. *IEEE Security & Privacy Magazine*, vol. 5, no. 3, pp. 85–87, May 2007.

[181] Department of Homeland Security: Software assurance. PDF, 2016.
Available at: `https://www.us-cert.gov/sites/default/files/publications/infosheet_SoftwareAssurance.pdf`

[182] Mailloux, L.O., McEvilley, M.A., Khou, S. and Pecarina, J.M.: Putting the "systems" in security engineering: An examination of NIST Special Publication 800-160. *IEEE Security & Privacy*, vol. 14, no. 4, pp. 76–80, July 2016.

[183] International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE): *Systems and Software Engineering — System Life Cycle Processes*, May 2015. Available at: `https://www.iso.org/standard/63711.html`

[184] McEvilley, M.: Towards a notional framework for systems security engineering. In: *Proceedings of the National Defense Industrial Association (NDIA) 18th Annual Systems Engineering Conference*. April 2015.

[185] Jabbour, K.T. and Ratazzi, E.P.: Deterrence in cyberspace. In: Lowther, A. (ed.), *Thinking About Deterrence: Enduring Questions in a Time of Rising Powers, Rogue Regimes, and Terrorism*, pp. 37–47. Air University Press, 2013.

[186] Morral, A.R. and Jackson, B.A.: Understanding the role of deterrence in counterterrorism security. Tech. Rep. OP-281-RC, RAND Corporation, November 2009.

[187] Herley, C.: Why do Nigerian scammers say they are from Nigeria? In: *Proceedings of the 11th Annual Workshop on the Economics of Information Security (WEIS)*. June 2012.

[188] Tøndel, I.A., Jaatun, M.G. and Meland, P.H.: Security requirements for the rest of us: A survey. *IEEE Software*, vol. 25, no. 1, pp. 20–27, January 2008.

[189] Salini, P. and Kanmani, S.: Survey and analysis on security requirements engineering. *Computers & Electrical Engineering*, vol. 38, no. 6, pp. 1785–1797, November 2012.

[190] Firesmith, D.G.: Engineering security requirements. *Journal of Object Technology*, vol. 2, no. 1, pp. 53–68, January-February 2003.

[191] Robertson, S. and Robertson, J.: *Mastering the Requirements Process*. 1st edn. Addison-Wesley Professional, Harlow, UK, 1999.

[192] Sindre, G. and Opdahl, A.L.: Eliciting security requirements with misuse cases. *Requirements Engineering*, vol. 10, no. 1, pp. 34–44, January 2005.

[193] 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)*, pp. 4–5. June 2001.

[194] Hope, P., McGraw, G. and Antòn, A.I.: Misuse and abuse cases: Getting past the positive. *IEEE Security & Privacy*, vol. 2, no. 3, pp. 90–92, June 2004.

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

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

[197] McDermott, J.: Abuse-case-based assurance arguments. In: *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC)*, pp. 366–374. December 2001.

[198] Alberts, C., Behrens, S., Pethia, R. and Wilson, W.: Operationally critical threat, asset, and vulnerability evaluation (OCTAVE) framework, version 1.0. Tech. Rep. CMU/SEI-99-TR-017, Software Engineering Institute, Carnegie Mellon University, September 1999.
Available at: `http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13473`

[199] Alberts, C. and Dorofee, A.: *Managing Information Security Risks: The OCTAVE Approach*. Addison-Wesley, Castleton, NY, USA, 2003.

[200] Nuseibeh, B.: Weaving together requirements and architectures. *IEEE Computer*, vol. 34, no. 3, pp. 115–117, March 2001.

[201] Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S., Kumar, S.S. and Wehrle, K.: Security challenges in the IP-based Internet of Things. *Wireless Personal Communications*, vol. 61, no. 3, pp. 527–542, December 2011.

[202] Atzori, L., Iera, A. and Morabito, G.: The Internet of Things: A survey. *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, October 2010.

[203] *IEEE Standard for Local and metropolitan area networks — Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, September 2011.

[204] *IEEE Standard for Local and metropolitan area networks — Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer*, April 2012.

[205] Apvrille, A. and Pourzandi, M.: Secure software development by example. *IEEE Security and Privacy*, vol. 3, no. 4, pp. 10–17, July 2005.

[206] Haley, C., Laney, R., Moffett, J. and Nuseibeh, B.: Security requirements engineering: A framework for representation and analysis. *IEEE Transactions on Software Engineering*, vol. 34, no. 1, pp. 133–153, January 2008.

[207] Botta, M., Simek, M. and Mitton, N.: Comparison of hardware and software based encryption for secure communication in wireless sensor networks. In: *Proceedings of the 36th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 6–10. July 2013.

[208] Camp, L.J. and Wolfram, C.: Pricing security: A market in vulnerabilities. In: Camp, L.J. and Lewis, S. (eds.), *Economics of Information Security*, vol. 12 of *Advances in Information Security*, pp. 17–34. Springer, 2004.

[209] Kazman, R., Asundi, J. and Klein, M.: Making architecture design decisions: An economic approach. Tech. Rep. CMU/SEI-2002-TR-035, Software Engineering Institute, Carnegie Mellon University, September 2002.
Available at: `http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6265`

[210] Bodin, L.D., Gordon, L.A. and Loeb, M.P.: Evaluating information security investments using the analytic hierarchy process. *Communications of the ACM*, vol. 48, no. 2, pp. 78–83, February 2005.

[211] Ryan, B.: The US government is not spending enough on cybersecurity. Website, September 2015.
Available at: `http://www.businessinsider.com/us-government-cybersecurity-spending-2015-9`

[212] Tanz, J.: The Super Bowl and the black swanning of America. Website, February 2017.
Available at: `https://www.wired.com/2017/02/super-bowl-black-swanning-america`

[213] Dubner, S.J.: How to be less terrible at predicting the future. Podcast, January 2016.
Available at: `http://freakonomics.com/podcast/how-to-be-less-terrible-at-predicting-the-future-a-new-freakonomics-radio-podcast/`

[214] Howard, R.: Cybersecurity spend: ROI is the wrong metric. Website, June 2017.
Available at: `https://www.csoonline.com/article/3200270/network-security/cybersecurity-spend-roi-is-the-wrong-metric.html`

[215] Woods, D. and Simpson, A.C.: Policy measures and cyber insurance: A framework. *Journal of Cyber Policy*, vol. 2, no. 2, pp. 209–226, August 2017.

[216] Alberts, C., Woody, C. and Dorofee, A.: Introduction to the security engineering risk analysis (SERA) framework. Tech. Rep. CMU/SEI-2014-TN-025, Software Engineering Institute, Carnegie Mellon University, December 2014.
Available at: `http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=427321`

[217] Böhme, R. and Moore, T.: Security metrics and security investment. PDF, September 2012.
Available at: `https://tylermoore.ens.utulsa.edu/courses/econsec/reading/lnse-secinv2.pdf`

[218] Ponemon Institute LLC: 2017 cost of data breach study: Global analysis. Tech. Rep. SEL03027-USEN-01, Ponemon Institute LLC and IBM, June 2017.
Available at: `https://www.ibm.com/security/data-breach/`

[219] Spence, M.: Signaling in retrospect and the informational structure of markets. *American Economic Review*, vol. 92, no. 3, pp. 434–459, June 2002.

[220] Cordes, J.J.: An overview of the economics of cybersecurity and cybersecurity policy. Tech. Rep. GW-CSPRI-2011-6, The George Washington University Cyber Security Policy and Research Institute, June 2011.

Available at: https://pdfs.semanticscholar.org/0691/25e47997e8f494f5922d315c81c9d4afe0ca.pdf

[221] Massacci, F. and Nguyen, V.H.: An empirical methodology to evaluate vulnerability discovery models. *IEEE Transactions on Software Engineering*, vol. 40, no. 12, pp. 1147–1162, December 2014.

Appendix A

Gordon-Loeb Model Function Definitions

Since the original Gordon and Loeb paper, various assumptions have laid the foundation for the form of various security breach probability functions. They are presented below; the reader is referred to the referenced works for more details.

A.1 Assumptions

The three original assumptions for security breach probability from [98] consist of the following:

- $A1$: $S(z,0) = 0 \forall z$ (If the system completely invulnerable it will remain perfectly protected for any amount of information security investment, including a zero investment)
- $A2$: $\forall\, v$, $S(0,v) = v$ (If there is no investment in information security, the probability of a security breach — conditioned on the realisation of a threat — remains the system's inherent vulnerability)
- $A3$: $\forall\ v \in (0,1)$ and all z, $S_z(z,v) < 0$ and $S_{zz}(z,v) > 0$ where S_z and S_{zz} and denotes the first and second partial derivative with respect to z. Furthermore, $\forall\ v \in (0,1), \lim\ S(z,v) \to 0$ as $z \to \infty$ (As the investment in security increases, the system is made more secure, but at a decreasing rate. Through investment the probability of a security breach $t \cdot S(z,v)$ can be made to be arbitrarily close to zero).

In addition to these three assumptions, the following have been proposed by Hausken in [101] to describe alternative conceptions of security investment (which we denote with 'H' subscript):

- $A4_H$: $\forall v \in (0,1)$, and all $z, S_z(z,v) < 0$, $S_{zz}(z,v) < 0$ for $0 \leq z < z_i$ and $S_{zz}(z,v) > 0$ for $z > z_i$, where z_i is some intermediate investment such that $S_{zz}(z_i,v) = 0$, and $\lim S(z,v) \to 0$ as $z \to \infty$ (For all investment under an intermediate amount the security increases at an increasing rate, beyond which the security continues to increase but at a decreasing rate)

- $A5_H$: $\forall v \in (0,1)$, and all $z \leq z_u > 0$, $S_z(z,v) < 0$, $S_{zz}(z,v) > 0$, and $S(z,v) = 0 \forall z \leq z_u$ (for all investments greater than zero but less than a certain point the security increases at a decreasing rate; however, once this point is reached the adversary is incapable of breaching the system)

- $A6_H$: $\forall v \in (0,1)$, and all $z \leq z_u > 0$, $S_z(z,v) < 0$, $S_{zz}(z,v) < 0$, and $S(z,v) = 0 \forall z \geq z_u$ (for all investments greater than zero but less than a certain point the security increases at an increasing rate; however, once this point is reached the adversary is incapable of breaching the system)

- $A7_H$: $\forall v \in (0,1)$, and all $z \leq z_u > 0$, $S_z(z,v) < 0$, $S_{zz}(z,v) = 0$, and $S(z,v) = 0 \forall z \geq z_u$ (for all investments greater than zero but less than a certain point the security increases at a constant rate; however, once this point is reached the adversary is incapable of breaching the system)

Others have proposed additional assumptions to capture further concerns. In [102], Willemson proposes a class of assumption to capture what he sees as a lax condition regarding the monotonicity of v that allows the security breach probability at an arbitrary vulnerability level (denoted as β in [102]) to be greater than at 0 vulnerability but lower than at $v = 1$. Additionally, Willemson proposes the following augmentation to $A3$ in order to address the concern that, unless $v = 0$ initially, it is impossible to increase v to 0. We use Willemson's designation denoted with 'W' subscript:

- $A3'_W$: $\forall v \in (0,1)$, and all z, S is twice differentiable; furthermore, $S_z(z,v) \leq 0$ and $z, S_{zz}(z,v) \geq 0$ while $\forall v, \lim S(z,v) = 0$ as $z \to \infty$ (As the investment in security increases, the system is made more secure, but at a decreasing rate. Through investment the probability of a security breach $t \cdot S(z,v)$ can be made to be equal to zero)

- $A3''_W$: $\forall v \in (0,1)$, and all z, $S_v(z,v) > 0$ and is twice differentiable (for all investments greater than zero the amount of vulnerability addressed will increase with increasing investment)

A.2 Security Breach Probability Functions

From these assumptions, various classes of security breach probability functions have appeared in the literature. The original paper proposed two such functions, each conforming to $A1$—$A3$ [98]:

- S^I(Conforming to $A1$, $A2$ and $A3$):

$$S^I(z,v) = \frac{v}{(\alpha z + 1)^\beta} \tag{A.1}$$

- S^{II}(Conforming to $A1$, $A2$ and $A3$):

$$S^{II}(z,v) = v^{\alpha z + 1} \tag{A.2}$$

Where $\alpha > 0$, $\beta \geq 1$ are measures of the productivity of information security; that is, the probability of a security breach is decreasing in both α and β (or just α in the case of S^{II}).

Hausken subsequently proposed an additional set of functions in [101], relative to the assumptions $A4_H$—$A7_H$ defined above:

- $S^{III\text{-}H}$(Conforming to $A1$, $A2$ and $A4_H$):

$$S^{III\text{-}H}(z,v) = \frac{v}{(1 + \gamma(e^{\theta z} - 1))} \tag{A.3}$$

- $S^{IV\text{-}H}$(Conforming to $A1$, $A2$ and $A5_H$):

$$S^{IV\text{-}H}(z,v) = \begin{cases} v(1 - \mu z^k) & \text{when } z \leq \mu^{-1/k} = z_u \\ 0 & \text{when } z > \mu^{-1/k}, 0 < k < 1 \end{cases} \tag{A.4}$$

where $\mu > 0$.

- $S^{V\text{-}H}$(Conforming to $A1$, $A2$ and $A6_H$):

$$S^{V\text{-}H}(z,v) = \begin{cases} v(1-\omega z^k) & \text{when } z \leq \omega^{-1/k} = z_u \\ 0 & \text{when } z > \omega^{-1/k}, k > 1 \end{cases} \tag{A.5}$$

  where $\omega > 0$.

- $S^{VI\text{-}H}$(Conforming to $A1$, $A2$ and $A7_H$):

$$S^{VI\text{-}H}(z,v) = \begin{cases} v(1-\lambda z^k) & \text{when } z \leq 1\backslash\lambda = z_u \\ 0 & \text{when } z > 1\backslash\lambda \end{cases} \tag{A.6}$$

In [100], Willemson proposes a family of functions that conform to his relaxed version of assumption $A3$:

- $S^{III\text{-}W}$(Conforming to $A1$, $A2$ and $A3'_W$):

$$S^{III\text{-}W}(z,v) = \begin{cases} v(1-\frac{z}{b})^k & \text{when } 0 \leq z < b \\ 0 & \text{when } z \geq b \ (b > 0, k > 2) \end{cases} \tag{A.7}$$

- $S^{IV\text{-}W}$(Conforming to $A1$, $A2$ and $A3'_W$):

$$S^{IV\text{-}W}(z,v) = \begin{array}{ll} S^{III\text{-}W}(z,v) & \text{when} z \leq b' \\ S^{III\text{-}W}(b',v) & \text{when} z > b' \text{for} b' \in (0,b) \end{array} \tag{A.8}$$

- $S^{V\text{-}W}$(Conforming to $A1$, $A2$ and $A3''_W$):

$$S^{V\text{-}W}(z,v) = \begin{cases} v(1-\frac{z}{b})^k & \text{when } 0 \leq z < b \\ 0 & \text{when } z \geq b \ (b > 0, k > 1) \end{cases} \tag{A.9}$$

Note that Willemson chooses not to provide an explicit example of a function in the class $S^{IV\text{-}W}$, instead choosing to define this in relation to $S^{III\text{-}W}$; this is reflected in the definition presented.