Imperial College London
Faculty of Engineering
Department of Computing

# Improving Resilience to Cyber-Attacks by Analysing System Output Impacts and Costs

Jukka-Pekka Eemeli Soikkeli

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Imperial College London
July 2022

## Statement of Originality

I hereby declare that this thesis and the research presented in it are my own work, and that all information derived from the work of others is properly acknowledged.

Jukka-Pekka Eemeli Soikkeli

## Copyright Declaration

# Abstract

Cyber-attacks cost businesses millions of dollars every year, a key component of which is the cost of business disruption from system downtime. As cyber-attacks cannot all be prevented, there is a need to consider the cyber resilience of systems, i.e. the ability to withstand cyber-attacks and recover from them.

Previous works discussing system cyber resilience typically either offer generic high-level guidance on best practices, provide limited attack modelling, or apply to systems with special characteristics. There is a lack of an approach to system cyber resilience evaluation that is generally applicable yet provides a detailed consideration for the system-level impacts of cyber-attacks and defences.

We propose a methodology for evaluating the effectiveness of actions intended to improve resilience to cyber-attacks, considering their impacts on system output performance, and monetary costs. It is intended for analysing attacks that can disrupt the system function, and involves modelling attack progression, system output production, response to attacks, and costs from cyber-attacks and defensive actions.

Studies of three use cases demonstrate the implementation and usefulness of our methodology. First, in our *redundancy planning* study, we considered the effect of redundancy additions on mitigating the impacts of cyber-attacks on system output performance. We found that *redundancy with diversity* can be effective in increasing resilience, although the reduction in attack-related costs must be balanced against added maintenance costs. Second, our work on *attack countermeasure selection* shows that by considering system output impacts across the duration of an attack, one can find more cost-effective attack responses than without such considerations. Third, we propose an approach to *mission viability analysis* for multi-UAV deployments facing cyber-attacks, which can aid resource planning and determining if the mission can conclude successfully despite an attack. We provide different implementations of our model components, based on use case requirements.

# Acknowledgements

I would like to thank my supervisor Professor Emil Lupu, for his support and guidance throughout my research, for constructive feedback that pushed me toward better writing and presentation of material, and most importantly, for guiding me toward useful research directions within the context of the research group, while allowing me significant freedom to pursue topics that suit my skills and interests.

Thanks to my collaborators Luis Muñoz-González, Giuliano Casale, and Cora Perner, for their contributions to our works, for providing feedback that helped improve my writing, and for being enjoyable to work with.

I am grateful for the funding I received from the EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, grant reference EP/L016796/1), without which this research would not have been possible.

I am thankful to my HiPEDS CDT cohort, for all the shared experiences, both good times and hard work, during the first year of the programme. Special thanks to Na Lee, Kenny Malpartida Cardenas and Miguel Cacho Soblechero for being an excellent team for the HiPEDS group projects; Dan Iorga, George Rizos, George Theodorakis, Johannes Wiebe, Alex Tasos and Nadeen Gebara for being great company at HiPEDS events and at free time. A further thanks to Na for being a good friend at Imperial and outside of it when work was hard and a break was needed.

I wish to thank everyone in my research group, RISS, for having been great company to share an office, work, and have breaks with. Special thanks to Luis and Erisa for offering help and providing good discussions when needed, and to Luca, Aaron and Javi for great chats about research and life when peer support and perspective were needed.

I am grateful to my family for their endless support throughout my studies, and to Katherine, without whose companionship and support the past few years leading up to this thesis would have been immeasurably harder.

# Contents

**Abstract** **ii**

**Acknowledgements** **iii**

**1 Introduction** **1**

1.1 Motivation and Objectives . . . . . 4

1.2 Contributions . . . . . 7

1.3 Publications . . . . . 9

1.4 Thesis outline . . . . . 10

**2 Background and related literature** **11**

2.0.1 Resilience in general . . . . . 12

2.0.2 Outline of our methodology . . . . . 15

2.0.3 Structure . . . . . 17

2.1 Resilience approaches . . . . . 17

2.1.1 Cyber resilience works . . . . . 17

2.1.2 Other resilience works with relevance . . . . . 26

2.2 Attack impact assessment . . . . . 36

2.2.1 Attack impact assessment works . . . . . 36

2.2.2 Attack modelling approaches usable for impact assessment . . . . . 39

2.2.3 System production and performance modelling . . . . . 44

2.2.4 Business cyber security investment . . . . . 47

2.3 Discussion and chapter summary . . . . . 49

**3 Methodology for cyber resilience impact analysis 50**

3.1 Summary of the methodology . . . . . 51

3.1.1 Sample instantiations of the methodology to applications . . . . . 55

3.2 Chapter summary and discussion . . . . . 61

**4 Attack progression modelling 62**

4.1 Attack paths . . . . . 63

4.1.1 Exploits of vulnerabilities in identical system components . . . . . 65

4.2 Attacker behaviour . . . . . 68

4.2.1 Attacker actions . . . . . 68

4.3 Defensive actions . . . . . 72

4.3.1 Attack detection . . . . . 73

4.3.2 Countermeasures . . . . . 74

4.3.3 Recovery . . . . . 77

4.4 Our attack progression modelling approaches . . . . . 78

4.4.1 Simulating attack steps with probabilistic choices . . . . . 80

4.4.2 Pre-defined attack scenarios . . . . . 80

4.4.3 Petri net modelling . . . . . 82

4.5 Chapter summary and discussion . . . . . 83

**5 Production modelling** **85**

5.1 Dependency modelling with DGs . . . . . 85

5.2 The propagation of performance impacts . . . . . 86

5.2.1 Network status function . . . . . 87

5.2.2 Production-function based modelling . . . . . 92

5.2.3 Performance modelling with QN models . . . . . 93

5.3 Chapter summary and discussion . . . . . 97

**6 Attack impact assessment** **99**

6.1 The impact of a single attack outcome . . . . . 100

6.2 Expected impact . . . . . 102

6.3 Impact assessment evaluation outputs . . . . . 105

**7 Cost modelling** **106**

7.1 Valuing production disruptions . . . . . 107

7.1.1 Smoothly-varying costs . . . . . 108

7.1.2 Trigger-level costs . . . . . 110

7.2 Costs of defensive actions and resilience improvements . . . . . 113

7.3 Valuing future impacts . . . 115

7.3.1 Countermeasure effect based on short and long-run "trajectories" . . . 116

7.3.2 "Time until loss" metric . . . 119

7.4 Other attack costs . . . 121

7.5 Chapter summary and discussion . . . 123

**8 Applications: Resilience planning 124**

8.1 Introduction . . . 125

8.2 Related work . . . 128

8.3 Overview of the approach . . . 129

8.3.1 Threat model . . . 132

8.4 Running case . . . 133

8.4.1 Summary of key assumptions . . . 135

8.5 Model . . . 137

8.5.1 Attack and dependency graphs . . . 137

8.5.2 Attack progression . . . 137

8.5.3 Attack detection . . . 140

8.5.4 Attack scenarios and outcomes in the case study . . . 141

8.5.5 Recovery modelling . . . 142

8.5.6 Performance modelling . . . 143

8.5.7 Costs . . . 143

8.5.8 Cost minimisation over attack scenarios . . . 144

8.6 Evaluation . . . 147
8.6.1 Baseline results . . . 147
8.6.2 Sensitivity to parameter changes . . . 149
8.6.3 Long-term maintenance costs . . . 154
8.6.4 Scalability . . . 160
8.7 Conclusion . . . 163

**9 Applications: Countermeasure selection 164**

9.1 Introduction . . . 165
9.2 Related work . . . 167
9.3 Impact analysis modelling . . . 168
9.3.1 Attack and dependency graphs . . . 168
9.3.2 Attack impact analysis . . . 168
9.3.3 Performance measurement and resilience . . . 171
9.3.4 Attacks, countermeasures and recovery . . . 171
9.3.5 Costs of actions . . . 173
9.3.6 Recovery process . . . 174
9.3.7 Sample impact analysis for CM selection . . . 175
9.4 Countermeasure selection . . . 177
9.5 Evaluation . . . 180
9.5.1 Results for the sample graph . . . 182
9.5.2 Results for randomly generated graphs . . . 184

9.6 Conclusion . . . . . 188

**10 Applications: Mission viability analysis** **189**

10.1 Introduction . . . . . 191

10.2 Related work . . . . . 193

10.3 Scenario and Threat Model . . . . . 194

10.4 Modelling approach . . . . . 196

10.5 Model implementation . . . . . 198

10.5.1 Connectivity and attack progression . . . . . 198

10.5.2 Actions to mitigate attacks . . . . . 200

10.5.3 Petri net (SWN) model . . . . . 201

10.5.4 Mission performance modelling . . . . . 205

10.5.5 Pre-planning and during mission use . . . . . 206

10.6 Analysis results . . . . . 208

10.6.1 Analysis using the SWN model . . . . . 208

10.6.2 Mission success analysis . . . . . 210

10.7 Conclusion . . . . . 211

**11 Conclusion** **213**

11.1 Summary . . . . . 213

11.2 Applications for our methodology . . . . . 216

11.3 Future Work . . . . . 217

**A Redundancy planning appendices** **220**

A.1 Complexity: Attack outcomes to evaluate, and variant cases . . . . . 220

A.1.1 Variant attack cases for attack scenarios . . . . . 220

A.1.2 Attack outcomes to evaluate for a given server allocation . . . . . 221

A.1.3 Worst case complexity: Optimisation bounds and attack outcomes . . . . . 222

A.2 Attacker capabilities . . . . . 226

A.3 Sensitivity to diversification cost $c_{as}$ . . . . . 228

A.4 Sensitivity to attack scenario weights . . . . . 230

A.5 TUL optimisation result tables . . . . . 231

**Bibliography** **232**

# List of Tables


2.1 Cyber resilience practices and their objectives, as in [BG11] . . . . . 18

2.2 Comparison of papers proposing practical solutions to improve cyber resilience . . . 23

2.3 Comparison of papers on resilience in networked systems . . . . . 31

4.1 Privileges and vulnerabilities in the case study . . . . . 67

8.1 Workload service demands and parameters . . . . . 134

8.2 Privileges and vulnerabilities in the case study . . . . . 138

8.3 Parameter values and sensitivity ranges . . . . . 147

8.4 Sensitivity to recovery time $t_r$ . . . . . 151

8.5 Sensitivity to detection probability $p_d$ . . . . . 152

8.6 Sensitivity to loss of customers . . . . . 158

9.1 Comparing mean values of performance metrics, our approach (CICM) vs AIA . . . 184

9.2 Comparing mean values of performance metrics, our approach (CICM) vs PLE . . . 185

10.1 Parameters in the SWN model . . . . . 202

10.2 Area coverage over time for an individual vehicle, ha/min . . . . . 205

10.3 Parameter values used . . . . . 209

10.4 Total expected mission coverage, ha . . . . . 211

A.1 Variants to attack Scenario 1 due to diversification . . . . . 221

A.2 Effective allocations from attack outcomes, Scenario 1 attack variant 1a . . . . . 223

A.3 Scenario variants for attack Scenario 1 . . . . . 227

A.4 Sensitivity to conditional probability of exploits $P(A|R)$ . . . . . 227

A.5 Sensitivity to diversification cost levels . . . . . 229

A.6 Sensitivity to attack scenario weights, attack cost minimisation . . . . . 230

A.7 Sensitivity to detection probability $p_d$, TUL optimisation, Case 1 defence . . . . . 231

# List of Figures


2.1 Resilience calculation from performance . . . . . 13

2.2 Methodology block diagram . . . . . 15

2.3 Sample attack graph . . . . . 40

2.4 Different AG approaches . . . . . 41

3.1 Methodology entities . . . . . 53

3.2 Summary of our approach to assessing the effectiveness of redundancy allocations . . 56

3.3 Activity diagram showing how our countermeasure selection would be deployed . . . 58

3.4 Countermeasure selection approach using impact analysis over time . . . . . 59

3.5 Mission viability analysis of a UAV mission . . . . . 60

4.1 Connectivity for a UAV mission, and potential high-level attack paths that ensue . . . 66

4.2 Network topology for the J2EE case study . . . . . 67

4.3 Prototype AG with multiplicity notation, and an expanded AG instance . . . . . 68

4.4 The SWN model, with mitigation parts highlighted . . . . . 82

5.1 DG, redrawn from [AJ17, Fig. 9] . . . . . 88

5.2 Production dependency graph, as used in [SPL21] . . . . . 93

5.3 Queueing network for the J2EE case study . . . . . 94

6.1 High-level representation of our attack impact assessment . . . . . 100

6.2 Impact mapping in implementation case studies . . . . . 101

6.3 Hierarchical probability weight assignment with multiple attack scenarios and variants 103

7.1 Cost impact calculation based on normalised output . . . . . 109

7.2 Sample cases of performance relative to a trigger-level . . . . . 111

8.1 Summary of our attack impact analysis approach . . . . . 131

8.2 Queueing network and network topology for the J2EE case study . . . . . 134

8.3 AG and DG in the J2EE case study, with attack scenarios . . . . . 138

8.4 Optimal allocations in the case study, baseline, $\beta = 0.001$ . . . . . 148

8.5 Impact of varying recovery time $t_r$ in the case study . . . . . 150

8.6 Months until cumulative maintenance cost exceeds attack-time benefit (TUL) . . . . 154

8.7 TUL and cost savings when minimising attack costs or maximising TUL . . . . . 156

8.8 TUL ranges, varying customer loss assumptions and $t_r$ . . . . . 159

9.1 Sample system network topology, and the IAG . . . . . 170

9.2 Patching and recovery actions and their effects on the system . . . . . 172

9.3 Sample attacker moves and their effects on the system . . . . . 175

9.4 The stages of the baseline trajectory calculation . . . . . 180

9.5 Simulation results for the sample graph . . . . . 183

9.6 Sensitivity to detection delay and cost assumptions . . . . . 187

10.1 Modelling components in the mission viability analysis application . . . . . 190

10.2 Sample analysis, attack that does not spread . . . . . 197

10.3 From connectivity to attack propagation stages . . . . . 199

10.4 Stochastic well-formed net for attack stages . . . . . 202

10.5 Expected number of sensor vehicles online . . . . . 210

10.6 Coverage rates during attack . . . . . 211

# Chapter 1

# Introduction

Cyber-attacks cost businesses millions of dollars every year, and one of the key components of costs from cyber-attacks is the cost of business disruption from system downtime. It accounted for a loss of $4 million per year on average for large international companies, or 31% of the total cost related to cybercrime, according to [BLDC19]. Not all cyber-attacks can be prevented, and therefore there is a need to consider the *cyber resilience* of systems, i.e. the ability of a system to withstand cyber-attacks and recover from them. Resilience relates to how the system responds to an event, from its start through to when recovery is achieved. Thus, approaches to improve cyber resilience can target different aspects, such as increasing robustness to attack impacts or improving recovery speed, with various system design aspects, capability improvements or reactive responses.

Previous works discussing the resilience of systems to cyber attacks typically either offer generic high-level guidance on best practices for organisations [CAC+10, BG11, MMPP19, CBL+20, CAH21, CBL+21], consider attack impacts in a limited way [SSL17, CYS+18, SLS19, HSKG20], or apply to very specific types of systems with special characteristics (e.g. power networks [YWZ17, YWR18], smart grids [LYGH21, BJY+21] and microgrids [WRS20], where attacks can be mitigated with control theoretic solutions that do not generalise to other settings). There is a lack of an approach to evaluate the cyber resilience of a system that is more generally applicable yet provides a detailed consideration for the system-level impacts of cyber attacks and defences, including effects due to component inter-dependencies and cascading effects.

A related problem is determining the effectiveness of investments in capabilities intended to improve cyber resilience, highlighted by the UK Government's finding that UK businesses believe that "commercial incentives to invest in cyber security are not clear" [Off21, p.23]. An indication of the importance of this is provided by Accenture 2020 State of Cyber Resilience [BLDC20], which finds that businesses among the best cyber resilience performers were four times better at both finding breaches and stopping attacks than average-performing firms, three times better at fixing breaches quickly, and two times better at reducing breach impact. These leaders were found to focus their cyber security spending differently, and target different metrics of cyber resilience success than other firms: they are focusing more on speed of detection, response and recovery, while other firms were more concerned on whether they were nominally achieving various resilience-related indicators and outcomes. Such differences in approach combined with findings that some investments in security are failing while costs of security technologies are increasing [BLDC20] lead us to conclude that modelling that can find the drivers of success would benefit companies, enabling them to replace investment decision making based on rules of thumb and tick-box exercises.

Cyber resilience approaches need to help determine the effectiveness of investments in order to increase cyber resilience and minimise long-term costs. For example, Linkov et al. [LK19] criticised risk-assessment approaches for being too targeted to known threats, whereas resilience investments could help mitigate the impact of all attacks including unknown ones. The modelling must strike a balance between generality and system-specific detail – abstract enough to be reasonably generally applicable, but capturing enough system detail to provide meaningful impact analysis that can evaluate the effects of resilience improvements. For example, abstracting the model to the level where we only consider the extent of investment rather than how it is targeted [GL02, RMEB19] can leave business leaders in a situation where money is spent in an untargeted way without the expected results in terms of security or resilience. A similar weakness exists in works proposing high-level methods (frameworks, standards, key performance indicators (KPIs) and maturity models [CAC+10, MMPP19, CBL+20, CBL+21, CAH21]) to guide companies to improve their cyber resilience. While such approaches are useful in providing suggestions on generally usable policies to increase resilience, the works have not provided a way to evaluate the impacts of the resilience policies proposed, or the cost of the investments involved. *This is key if the problem of cyber resilience*

*is to be taken seriously: if the financial implications due to lack of preparation to attacks are not laid bare, decisions might be based on vague notions of effectiveness, or only made to meet minimum standards or regulations.* Consequently, a resilience evaluation approach requires an appropriate level of abstraction, to aid with evaluating the impact of resilience improving actions and techniques, and thus help companies make better decisions on investing into cyber resilience and cyber security.

Further, to evaluate the effectiveness of improvements into resilience, one has to consider the overall picture of the impact of attacks onto the business, as system interdependencies cause effects to spill over from the directly compromised components to other parts of the system. For this, estimating the cost of impacts in monetary units is beneficial, as monetary valuation provides a generic measure that can be applied across different contexts. It *enables expressing the impact and cost of any attack to any part of a system*, provided that direct and indirect effects are taken into account. Also, monetary values can be *compared across time* in a straightforward way, even across longer periods of time using time discounting. This is important for cyber resilience analysis, as comparisons must be made across time to evaluate the benefits of resilience improvements, especially if attacks are expected to be infrequent. Additionally, a methodology that expresses the impact of an attack in monetary terms can help better integrate cyber security into business decision making, as the effect of security investment can be quantified with relation to the output and revenue of the organisation. This could benefit regulators as well as companies, and is a timely issue, as the National Cyber Strategy 2022 of the UK Government identified a need for using regulation and incentives to encourage "embedding cyber security as a core part of good business" [Off21, p.65].

In this thesis we propose to address these gaps in the literature by providing a methodology for evaluating the cyber resilience of a system based on the performance of the system in producing its output, which uses a cost evaluation that allows relating the impact of an attack to the costs of actions that are intended to improve system cyber resilience. Our approach has two key aims: **a)** to provide cyber resilience evaluations at a sufficiently detailed level to consider system-level impacts (and costs) from attacks, while being applicable to various systems where attacks can affect output performance; **b)** to enable relating the costs from cyber attacks directly to the financial performance of the company (via the output performance), thus offering a way to assess, in monetary terms, the benefits from defence actions and investments into cyber security and resilience capabilities.

## 1.1 Motivation and Objectives

The motivation for the specific approach we take is twofold: 1. cyber resilience is an increasingly important area of study, with research gaps to be found in the provision of modelling that directly focuses on evaluating the system-level attack impacts but is also reasonably generally applicable; 2. expressing the impacts of attacks in terms of monetary costs could provide a useful way for businesses to evaluate the cost-effectiveness of investments intended to improve cyber security and cyber resilience of the business. We now relate these to the objective of the thesis, and address them in turn.

The objective of this thesis is to propose an approach for evaluating and reasoning about the cyber-resilience of networked systems, in which attack impacts are reported in monetary terms over the duration of the attack event. Specifically, we propose a methodology for evaluating attack impacts across the whole duration of an event from the initial impact until recovery is achieved, in comparison to a reference level that is expected if attacks do not occur, and provide related approaches to reasoning about resilience decisions. We focus on systems where availability is key, i.e. where an attack can cause expensive business disruption by rendering parts of the system unavailable. The intent is to provide modelling that highlights the key generic aspects of attack response, at a level of abstraction that is general enough to be applicable to many contexts but specific enough to enable businesses to make useful decisions on capability investments and responses to attacks.

We propose evaluating the business impacts from component unavailability via modelling its effect on the production of system output. We use 'production' as a generic term to refer to the process by which a system output is produced, whether the specific instance was, for example, the provision of services, fulfilling a mission, or a physical production line. Accordingly, the components of this production could be the various services that together constitute a web application, different tasks required to be fulfilled to complete a mission such as a fire survey, or different sub-processes required for a production line.

We approach the quantification of cyber attack impacts from the perspective of the production effects incurred during and after the attack, and relate these to financial costs. The idea is to quantify the direct and production-impact costs related to the attack, which in general include the costs of prepara-

tion (investment into security capabilities, and redundant capacity), the losses due to disruption (loss of systems availability or performance), and the costs of defensive actions such as attack containment, patching and recovery. While the production-impact assessment component of our methodology could alone aid the design of cyber resilient systems, as it enables investigating the impacts of actions (and lack of them) onto the production performance of the system, the most benefit is gained when the impact assessment is combined with cost modelling, to obtain cost impacts.

We believe that by emphasising the financial impact of security decisions, our framework could lead to improvements in the cyber-resilience of systems, and to more effective security investment decisions. The benefits of our proposed approach are threefold: a) An organisation can bring decisions over cyber security into the heart of business decisions, as the costs and impacts on system performance are considered. b) Cyber security teams can better establish the costs and benefits of measures, allocate their budgets more effectively and better convince management of appropriate levels of security spending. c) Modelling of this kind can guide regulation on requirements for cyber security investments in businesses, as it could help undercover systemic impacts arising from insufficient incentives to invest in cyber security.

In order to form our methodology to address these objectives, we have had to address several research issues, the key ones of which are:

- Providing an attack impact assessment model that is generic and combines several components (attack progression modelling, incl. defence modelling; system ”production” modelling; performance modelling; cost modelling).
- Evaluating which types of modelling components suit the proposed methodology and its aims, and providing implementations of different approaches.
- Evaluating the costs of attacks and defensive actions in a way that takes into account cascading effects in interdependent systems.
- Providing a modelling approach to attacks which aim to disrupt multiple instances of components, thus requiring multiple privilege copies, instead of targeting one goal node, as commonly assumed by attack graph analysis approaches. Implementing an approach to evaluate the im-

pacts of the various outcomes of such attacks.

- Addressing the problem of relating the cost estimates during an attack to costs during future time periods, where attacks may or may not occur.

**Scope:** We focus on the performance of producing system output, and attacks that can impact this performance. This focus helps narrow our scope both in terms of the security attribute of interest, and the category of attacks we consider in terms of their consequences. From the perspective of the traditional security attributes, the focus on output production means we are interested in availability, and leave integrity and confidentiality outside of our scope.

On costs of attacks, Accenture [BLDC19] categorises cybercrime related costs based on consequences of attacks: business disruption (from downtime), information loss, revenue loss and equipment damages. Information losses account for the largest share (45%) of all cybercrime related losses among the firms in their survey, followed by business disruption (31%), revenue loss (20%), and equipment damage (4%). Our focus on output production comes under the category of *business disruption*, while the costs relating to the other categories would need to be captured with additional modelling. For *revenue loss*, we propose a simple model of losses from customer attrition due to reputational damage, discussed with relation to redundancy planning in Section 8. *Information losses* in the sense of stolen data, such as data breaches and espionage, do not typically have systemic effects on the production process, but their costs arise from the value of the information (or the penalty due to its loss). We do not model these for two reasons: 1. they do not relate to output performance, but occur as breaches to certain systems, and 2. the costs cannot be mitigated using resilience methods like redundancy and recovery, but only by traditional cyber security approaches focused on preventing attacks. Together, these two points mean that attacks aiming to steal information can be analysed using the existing approaches to cyber risk analysis and system hardening. However, attacks that delete or encrypt information, such as ransomware attacks, lead to business disruption, the impact of which can be captured with our modelling. While *equipment damage* could be straightforward to add to our model where the consequence of a certain attack step (exploit) was component damage, it may be hard to determine in advance if damage is an expected outcome of an exploit, and thus if it should be included in analyses made in preparation for defence against attacks. As equipment damage is a

minor cost category relative to the others, we chose to leave a detailed study of it outside of our scope. Note that this refers to the direct cost of the damage itself. In contrast, any business disruption caused by equipment damage, e.g. due to a longer recovery time, could be modelled using our approach.

In our choice of actions and resilience-improving techniques to model, we selected among ones that relate to the performance of production, and are fairly generally applicable to various systems. The MITRE Corporation's cyber resiliency engineering framework [BG11, BG13] lists various architectural design techniques to improve cyber resilience. The 13 techniques range from segmentation and isolation to randomness and moving target defence (MTD) techniques in the proactive side, and also include reactive techniques with different types of dynamic reconfiguration, reconstitution and composition, in addition to deception. Given the number and variety of these, in building our methodology we have focused on providing general ways to model resilience techniques that relate to the performance of production. In particular, among the architectural techniques discussed by [BG11, BG13], we demonstrate how *redundancy* and *diversity* can be modelled within our methodology, and how our methodology can then be applied to redundancy planning (Chapter 8) and mission viability analysis (Chapter 10). In addition to these techniques, we consider defence responses in terms of *countermeasure selection* in Chapter 9.

## 1.2 Contributions

The key contributions of this thesis are:

Proposing **a modelling methodology** for evaluating the cyber resilience of systems based on their output performance during attacks, which can be used to estimate the cost-benefits of actions intended for improving resilience to attacks. As part of this, we explain the modelling requirements for conducting production-impact assessment over time to enable resilience analysis, and for enabling analysis of cost-effectiveness of actions in a way that considers effects during attack events and when attacks do not occur, and in future periods. For the latter point, we propose ways in which future periods can be accounted for: a comparison approach evaluating benefits of actions in possible future states of an attack; and a metric (time until loss, TUL) that relates during-attack benefits of an action

to its costs accruing over time whether attacks occur or not.

Providing **implementations of the methodology** for use in three different types of analysis: redundancy planning, reactive countermeasure selection, and mission viability analysis. Further, we show different approaches to implementing some of the key modelling components, and provide discussion on when the alternatives are applicable. In addition, we provide process descriptions and algorithms for conducting the analysis involved in these use cases, for example, how a countermeasure is chosen when an alert is received, using an algorithm employing our methodology for impact assessment.

Introducing **case studies of specific use-case applications** of the methodology: application to determining a cost-effective allocation of redundancy (with and without diversity) to the components of a system; application to the problem of attack countermeasure selection, where we choose defensive actions based on their expected impacts on the costs arising from an attack event; application of parts of the methodology on evaluating the viability of a multi-UAV mission if an attack occurs.

The analysis conducted in the use cases further leads to a number of **important results**; we now summarise the studies provided, and the main results from them: A key use-case for the methodology is planning for capabilities that can improve resilience to cyber-attacks, and in Chapter 8 we provide an instance of such analysis from the perspective of redundancy planning. Here, the attack impacts are evaluated using a queueing network model, and the attack progression modelling is done with a focus on specific scenarios, and how redundancy affects the attack progression and possible outcomes of the attacks. We evaluate our approach using a case study, and find that *the optimal diversified redundancy allocation can yield a substantial reduction in costs (including attack impacts and maintenance costs) relative to cases without diversity and without redundancy*. Importantly, our method enables *quantifying* the benefits, and comparing different alternatives. This also shows the importance of attack modelling when planning component redundancy, as while random events or failures can be mitigated with limited redundancy without diverse implementations, cyber attacks can replicate exploits easily on the redundant components if no diversity is provided.

Our countermeasure selection application in Chapter 9 demonstrates the usefulness of our methodology in reactive defence in terms of countermeasure selection. We show, via simulated attacks in a case study and in randomly generated synthetic graphs, that *a countermeasure selection strategy*

*that prioritises balancing costs over a longer term leads, on average, to more cost-effective choices of actions than alternative strategies.*

Chapter 10 looks into using the methodology to determine the viability of a mission when attacks occur, with a flavour of redundancy planning. We focus on multi-UAV missions, and show how our methodology can be used to determine the viability of a mission in the face of an attack affecting the availability of components, for example, determining the number of vehicles required for a mission with a given coverage requirement, how many redundant vehicles should be available to act as replacements in case of attack, and how different parameters affect the coverage reachable within a mission if an attack occurs. *Results from the case study highlight the importance of the speed of containment and appropriate recovery capacity, and provides a way to quantify their impacts and relative importance.*

## 1.3 Publications

Parts of the work presented here have been included in the following papers:

- Jukka Soikkeli, Luis Muñoz-González, and Emil Lupu. Efficient Attack Countermeasure Selection Accounting for Recovery and Action Costs. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES '19)*, ACM, 2019 [SML19]

- Jukka Soikkeli, Cora Perner, and Emil Lupu. Analyzing the Viability of UAV Missions Facing Cyber Attacks. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 103–112. IEEE, 2021. [SPL21]

- Jukka Soikkeli, Giuliano Casale, Luis Muñoz-González, and Emil Lupu. Redundancy Planning for Cost Efficient Resilience to Cyber Attacks. In press, to appear in IEEE Transactions on Dependable and Secure Computing. [SCML22]

## 1.4 Thesis outline

The thesis is structured as follows: Background information and related literature are presented in Chapter 2. After this, the middle part of the thesis introduces our methodology, and how it is constructed. Chapter 3 provides an overview of the methodology, and how we have applied it in our works to different use cases, while the chapters that follow it go into the details of the different key modelling components: Attack progression modelling is discussed in Chapter 4, production modelling in Chapter 5, how these two combine in assessment of production impacts is the topic of Chapter 6, and cost modelling is introduced in Chapter 7. After explaining the methodology components, we present the studies in which we applied the methodology to different use cases. Chapter 8 contains our redundancy planning work, Chapter 9 introduces how the methodology can be applied to reactive countermeasure selection, and Chapter 10 discusses mission viability analysis. Finally, Chapter 11 concludes with a discussion of the thesis achievements and directions for future work.

# Chapter 2

# Background and related literature

In a general sense, resilience is the ability of a system (e.g. an organism, a network, a country) to withstand and recover from adverse events such as natural disasters, epidemics, system faults or cyber-attacks. The concept of cyber resilience focuses on the cyber dimension of systems, and is defined by [LK19] as "the ability of the system to prepare, absorb, recover and adapt to adverse effects, especially those associated with cyber-attacks."

From the point of view of cyber security, concerns over the resilience of a system reflect the realisation that not all attacks and events can be fully avoided. For example, there exist previously unknown vulnerabilities (zero-day vulnerabilities) and exploits, but also unexpected component failures, accidents and natural disasters which can affect the functionality and security of a system. These make a system's ability to maintain and/or recover functionality the key feature when a breach (or a more generic failure) happens. Furthermore, sometimes recovery/dealing with the consequences can be cheaper or otherwise more sensible than attack prevention, for example, if prevention would involve taking measures that significantly reduce the usability of a system.

There is an extensive literature on methods to counter cyber security events (cyber-attacks), reviewed in [NPMK17]. Typically, the methods in this area focus on selecting between known approaches to known events, in effect assuming that the full set of possible attacks is known in advance. As noted by [NPMK17], the countermeasure provision solutions either neglect unknown (zero-day) exploits, or do not provide concrete solutions for dealing with them. As stated, these methods focus on the choice

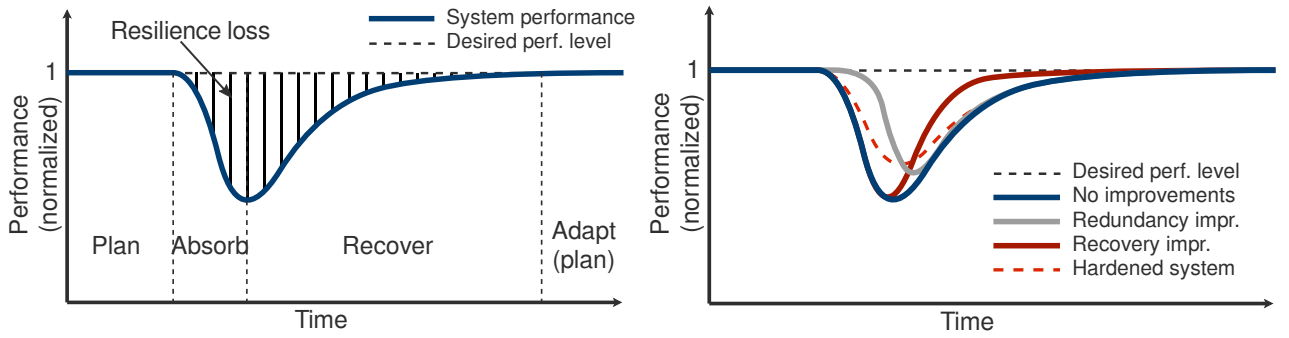
of countermeasures for cyber attack defence, and as such mainly involve the stages before and during an attack: modelling the system, and the potential attacks and countermeasures; identifying the attack and making a countermeasure choice. What is left out is the consideration for the long term – how do the actions by attackers and defenders impact the system during and after the event has passed, and how to improve the system to reduce the overall impact that occurs over time. *This longer term view is a key aspect separating the aims of cyber resilience from those of traditional cyber security.*

The research literature specific to cyber resilience, i.e. resilience to cyber attacks, is still relatively new: while there are plenty of descriptions of research programs and challenges [LK19, LKL21, KGTL21], and various frameworks and maturity models listing generic managerial actions to take to prepare organisations for resilience [CAC+10, BG11, CBL+20, CAH21], few works have proposed practical approaches that involve quantifying the impact of cyber attacks on a system's functionality. The examples of frameworks with quantification tend to apply to very specific systems with non-generalisable features (e.g. power grids [YWZ17, YWR18, LYGH21, BJY+21, WRS20]), or leave room for improvement in attack impact assessment [HSK19, HSKG20]. Hence our focus on building a methodology with attack impact assessment at its core.

Before moving into a fuller review of the most relevant areas of research for this thesis, we first set the scene by briefly describing research into resilience in general, and then provide an outline of our approach to impact analysis for cyber resilience.

### 2.0.1 Resilience in general

In the previous decade, two surveys covering the literature on resilience in the wider context were published, by [BDB11] and more recently by [HBRM16]. According to [HBRM16], the key fields where the topic of resilience is researched include environmental sciences and ecology, different subfields of engineering, psychology and psychiatry, and social sciences. Given this variety, the definitions of resilience applied differ somewhat, with some focusing only on the ability to endure a disruptive event at a good level of functionality, an aspect which we shall refer to as robustness. Even among the ones considering recovery a key feature, there are differences: some insist on recovery to




(a) Performance curve and resilience

(b) Impact of various system improvements

Figure 2.1: Resilience calculation from performance

the original level of performance, while others accept a more adaptive recovery. We term "adaptation" to mean ways in which a system can adjust during an event and after it, for example, continuing system functions despite a lower level of performance, or preparing for new events by improving capabilities (e.g. adding redundancy and diversity to components). We take the view that both recovery and adaptation are key aspects of resilience, and of evaluating the impact of adverse events.

Hosseini et al. [HBRM16] provides a classification of the literature up to 2015, with the key splits being between qualitative and quantitative approaches, and between generic and domain-specific approaches. The **general approaches** provide resilience metrics that are flexible enough to be applied to multiple fields, typically based on a comparison of the system performance before and after an event. Many such metrics adopt an approach in which a performance measure is observed over a window of time, including the event and a recovery period, and the value of resilience is given by comparing the cumulative value of the performance metric over the time window to what would have been expected during normal performance. Fig. 2.1 illustrates this type of an approach, initially popularised by [BCE+03] for measuring earthquake resilience, labelled with stages of resilience according to the classification in [GMG+16]. The curves represent system performance (normalised relative to the desired normal performance level), and the loss of resilience due to an adverse event is measured by the area between the curve and that of the desired performance. There are various approaches to the specific quantification of resilience from such a curve: some relating the loss area to that below the desired performance from the start of the event to the time of full recovery (e.g. [BCE+03]), others such as [GMG+16] specifying a time window with start and end times that do not have to coincide with the event start and recovery end, with other measures also proposed.

**Domain specific approaches** rely on modelling a particular system to examine the impact of the system structure on resilience. While these frameworks are not general, they are able to provide more direct insights into ways in which the resilience could be improved. The studies of this kind reviewed by [HBRM16] include approaches to resilience in transportation networks (airport runway, road, rail, waterway, and public transportation networks), fire and rescue service, water reservoir, supply chains, engineering resilience in a petrochemical plant, and organisational resilience, in addition to large-scale complex networks.

With respect to the wider resilience literature, we focus on resilience in man-made computer-enabled systems that can be affected by cyber-attacks. Within this context, our aim is an approach that applies to the systems and processes for the production of an organisation's output (whether it is services or products). Our focus excludes approaches that apply to particular types of specialised systems, such as power systems, but do not generalise elsewhere, and works focusing on specific types of events that do not relate to cyber attacks.

Among projects aiming to define what resilience means in terms of actions or stages relative to an event, there is a rough consensus on the different aspects of resilience. Sterbenz et al. [SHÇ+10] split the strategy of ResiliNets into four actions relating to an event (defend, detect, remediate, recover), and two for improving the system and these activities in longer term (diagnose, refine). In the MITRE Cyber Resiliency Engineering Framework [BG11], the four goals of cyber resiliency practices are: anticipate, withstand, recover, evolve. The US National Academy of Sciences [U.S12] definition of resilience splits it similarly into four parts: 1) plan and prepare; 2) absorb; 3) recover; 4) adapt. As apparent, the MITRE goals in [BG11] and NAS terms [U.S12] map readily into each other.

In Fig. 2.1 we labelled the stages of resilience relative to the event impact according to the terms used by NAS [U.S12], following [GMG+16]. To set our work in this thesis within the context of these stages, the methodology we propose is intended for impact assessment via modelling the performance of a system during the different stages of a cyber attack event, i.e. those mainly coinciding with the *absorb* and *recover* aspects of resilience. The most likely applications for the analysis enabled by the methodology are for planning purposes, in the *plan* and *adapt* aspects of resilience, as used in our works on redundancy planning in Chapter 8 and mission viability analysis in Chapter 10. However, it

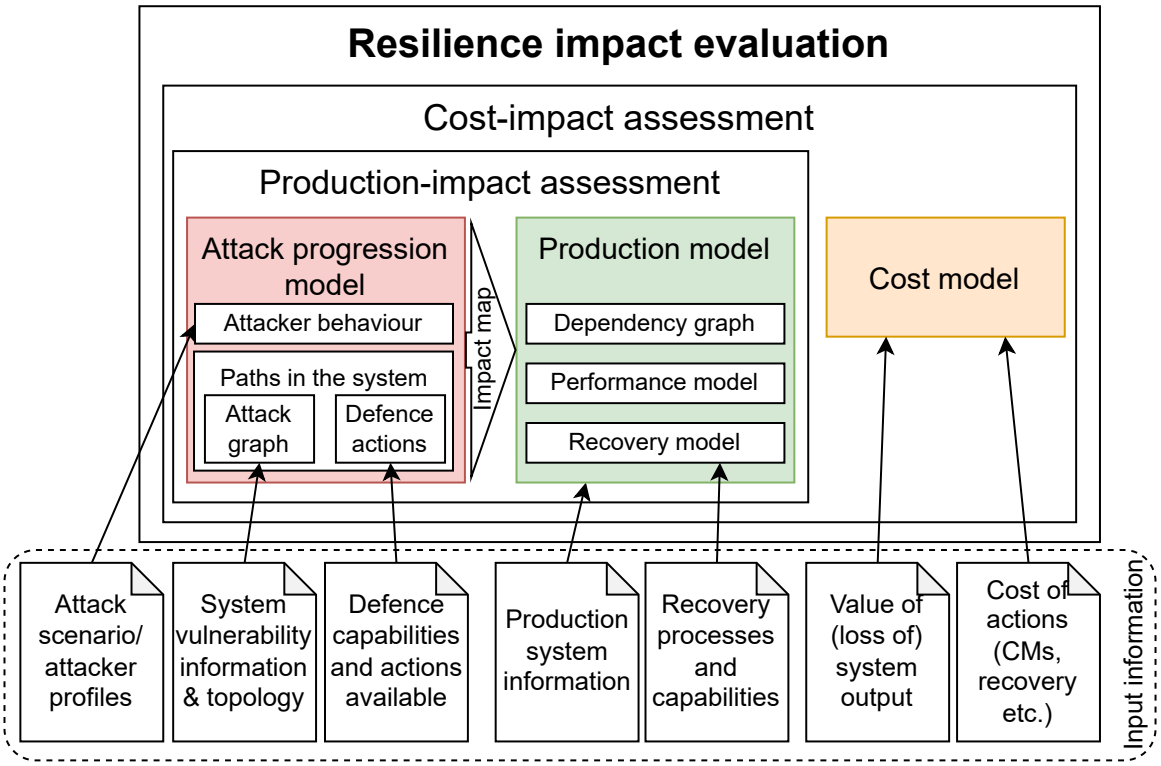



Figure 2.2: Methodology block diagram

could also help guide the response to an event (thus touching on the absorb and recover aspects), as proposed in our work on countermeasure selection in Chapter 9.

### 2.0.2 Outline of our methodology

To enable comparing existing works to our methodology for cyber resilience impact analysis, we now briefly outline the key aspects of our approach. Our focus is the resilience of system output production when facing cyber-attacks that aim to disrupt the system. For this, we follow the performance-curve approach to resilience, as in Fig. 2.1a. Consequently, the core of our methodology is about assessing attack impacts on production, and on expressing this in terms of financial costs. With these impacts, we can then quantify the resilience of the system against attacks, and evaluate the improvement in resilience from, and the cost-effectiveness of, actions and capability changes that are intended to improve cyber resilience. For example, Fig. 2.1b illustrates possible effects from improvements to redundancy, recovery and system hardening. Fig. 2.2 provides a high-level overview of the components of the methodology, and the input information required. The key components are the following:

1. **Attack progression model**, describing how cyber attacks progress in the system, containing:
    - A description of the *possible paths an attack can take* within the system;
    - *Defensive actions* that affect attack progression, such as detection and mitigation actions.
    - *Attacker behaviour* modelling, including attacker choices and capabilities;
2. A **production model** of the system components (services, processes) required to produce the output of the system, and their functional interdependencies. This includes:
    - A *dependency graph* describing the functional interdependencies between the components necessary for the production process;
    - A definition of *system performance* based on the production model, and how the compromise of production components affects the output (e.g. performance degradation of x%, or reduction in redundant capacity);
    - A *recovery model*, describing how production components are recovered, and how quickly.
3. An **impact map** between the attack and production models, defining which components in the production model can be impacted by attacks, and from which parts of the attack model;
4. A **cost model**, describing the costs (to the system owner) of attacks and defense actions, and the cost of investments into capabilities.

The impact evaluations using our methodology are conducted over time, from the start of an attack event until a specific time horizon, by which time recovery is expected to be completed.

We use the methodology for choosing actions intended to improve the resilience of a system, such as improvements to capabilities, and selection of reactive responses to attacks. We have focused our implementation efforts on a few key techniques that are applicable to a variety of systems and can be modelled in a general way: patching vulnerabilities, redundancy provision, and diversity (with respect to redundancy). Choosing them is a function of many things, including their prominence in existing literature, so comparisons can be made, but we also believe that implementing these enables us to capture key aspects of resilience and attack impact assessment with regard to architectural techniques for resiliency planning (redundancy, diversity), and for reactive response using hardening-type countermeasures (patching). In addition, to cover the whole duration of an attack event, we model

attack detection and recovery from attacks in a straightforward manner.

### 2.0.3 Structure

This chapter is split into two major parts: *existing resilience approaches* in Section 2.1 covers literature specifically discussing the concept of resilience, and includes works focusing on cyber resilience in particular, and related works on measures and frameworks for resilience in networked and cyber systems. Impact assessment is fundamental to resilience evaluation, so in the second part of this chapter we discuss related works that seek to address this challenge. *Attack impact assessment* in Section 2.2 includes approaches for evaluating the impacts of cyber attacks, and separate subsections for attack modelling and system performance modelling that could potentially be used for attack impact estimation. A further subsection discusses how investment into cyber security has been approached in the research literature. Finally, Section 2.3 provides a discussion of our work in relation to the literature, and concludes.

## 2.1 Resilience approaches

### 2.1.1 Cyber resilience works

The existing literature on cyber resilience can be broadly split into the following categories: 1) descriptions of frameworks; 2) resilience preparedness approaches, using maturity models or key performance indicators (KPIs); 3) impact assessment works; 4) practical solutions to improving resilience to attacks; 5) resilience metrics specific to cyber-attacks; 6) works focusing on specific components or specialised systems (e.g. smart grid). We now provide an overview of studies within these categories, save for the last one. We do not discuss works within the last category here, as works that provide solutions tailored to specific components or systems are intimately linked to the specifics of the target system, e.g. specialised targets such as maintaining the balance in a power grid. Examples of such include works on cyber resilience of CPS control processes [JF18, CCD+21], power systems (smart grids [LYGH21, BJY+21], microgrids [WRS20]).

| Technique | Understand | Prepare | Prevent | Constrain | Continue | Reconstitute | Transform | Re-architect |
|---|---|---|---|---|---|---|---|---|
| Adaptive response | | | | X | X | X | | |
| Analytic monitoring | X | X | | X | | X | | |
| Coordinated defence | | X | X | X | X | X | | |
| Deception | X | | X | | X | | | |
| Diversity | | | X | | X | | | X |
| Dynamic positioning | X | | X | | X | | | X |
| Dynamic representation | X | X | | | | | X | |
| Non-persistence | | | X | X | X | | | X |
| Privilege restriction | | | X | X | | | | |
| Realignment | | | | X | | | X | |
| Redundancy | | | | | X | X | | |
| Segmentation | | | X | X | | | | |
| Substantiated integrity | X | | | X | X | X | | |
| Unpredictability | X | | X | | X | | | |

Table 2.1: Cyber resilience practices and their objectives, from Table 3 in [BG11]

Cyber resilience techniques and practices were discussed in works by the MITRE Corporation [Gol10, BG11, GMP11, BG13] on cyber resiliency engineering, focusing on architectural design techniques that improve resilience to cyber attacks. Of these, [Gol10, BG11] provide extensive discussion on cyber resilience for mission assurance, and propose a set of generic techniques that aid cyber resilience. These techniques, as included in the description of the MITRE Cyber Resiliency Engineering Framework [BG11], are listed in Table 2.1. Most of the techniques relate to the design/architecture of a system or a mission (diversity, redundancy, segmentation, privilege restriction), or to the design of response strategies (adaptive response, coordinated defence).

Goldman et al. [GMP11] classify architectural design techniques for resilience into *proactive* techniques, for structuring a system architecture in a way that increases resilience to attacks, and *reactive* ones that are used in response to an attack. These techniques are all architectural in nature, from segmentation and isolation to randomness and moving target defence (MTD) techniques in the proactive side, while reactive techniques include different types of dynamic reconfiguration, reconstitution and composition, in addition to deception. The techniques are, essentially, about specific defence capabilities (and designs) which can increase the robustness of a system, but largely miss the other aspects of resilience, such as recovery. In this thesis we focus on a more general impact evaluation, and how to assess the effect of resilience improvements. Among the techniques in [GMP11], we provide a sample implementation and analysis that considers component *diversity*, modelling its impacts on the system

and comparing results to cases where it is not applied. We chose to focus on diversity as it is relevant for a wide array of systems, and is an important security consideration, for example, when redundant capacity is needed. Our evaluation approach could be applied to other architectural changes as well, and our methodology would provide a good foundation for their analysis in future work.

Whilst describing a framework for cyber resilience engineering, including the goals, objectives and actions, these "frameworks" do not provide an implementation of the approach in terms of a model or decision rules for choosing what actions to take, and what the impact would be. In this sense, as a method for impact assessment is not proposed, the frameworks discussed in [Gol10, BG11, GMP11, BG13] are similar to a maturity model approach, such as those we shall discuss next.

#### 2.1.1.1 Business cyber resilience planning: frameworks for assessing preparedness

A section of literature considers resilience planning for businesses from the perspective of generic actions that can improve the cyber resilience of a business, e.g. key performance indicators and maturity model goals such as "have a resilience plan", "have security technology". Examples of these are maturity models such as the CERT-RMM [CAC+10], and more recent works by [Nat18, MMPP19, CBL+20, CAH21, CALH21]. We shall now briefly summarise what they do, and then argue that despite their benefits in general applicability, their lack of impact evaluation leaves a need for resilience evaluation based on impact assessment, such as ours.

CERT Resilience Management Model (CERT-RMM) [CAC+10] is a process improvement model, providing a definition of the process for operational resilience management, including 26 process areas across 4 operational resilience management areas (enterprise management, resilience engineering, operations management, process management). The practices contained in these processes have a management perspective, i.e. they represent actions to "direct, control and manage" operational resilience. Their process definition is intended to be used as a benchmark for organisations to: 1. identify their current operational resilience capability level; 2. to set targets; 3. measure the gap between current performance and the targets; 4. develop action plans to meet the targets. The model provides descriptions of actions that an organisation might take to implement a resilience process, but does not specify how the actions would be deployed in practice. CERT-RMM does not cover

processes used for producing products or services, but focuses on the specific processes required to manage operational resilience.

NCSC Cyber Assessment Framework (CAF) [Nat18] is an approach to assessing how an organisation manages cyber risks to its essential functions. The guidance is outcome-focused, providing 14 principles specifying what must be achieved, broken down to further sub-outcomes. Assessment of how well the outcomes are met is based on expert judgement of associated 'indicators of good practice' (IGPs), which are statements describing organisational characteristics such as "Logs are reviewed almost continuously, in real time" (part of outcome C1.c Generating Alerts in CAF v3.1).

Carias et al. [CBL+20, CALH21, CAH21] work on forming a cyber resilience framework for SMEs. In [CBL+20] they form a list of policies to implement and domains within which these fit (e.g. the policy "Develop and communicate a cyber resilience strategy" within the domain "Governance"), and an implementation order for these, developed based on analysis of existing frameworks and feedback from industry experts. In [CALH21, CAH21] the authors develop a progression model (a type of maturity model) to accompany the framework, describing stages of achieving different policies. They provide a spreadsheet tool to aid SMEs in assessing how well they meet the various policies under the various domains, and gain a view of the overall maturity of their cyber resilience preparedness based on the progression model. This, like other maturity models, is a model for management decisions, without a way of estimating the cost of the actions (stages) or their impact and value.

Marrella et al. [MMPP19] propose a maturity model for business process resilience, for processes where multiple businesses cooperate. This work considers the impact of data inputs to a process with multiple business parties, where each party provides some data tasks. The threat they consider is a failure of one of the parties to provide their input to the process – the specific reasons for such failure are not considered, i.e. cyber attacks are not modelled. They build a maturity model for *resilience awareness* in multi-party business processes, with five levels (no awareness, failure awareness, data resilience, milestone resilience, process resilience). The aim of the model is to support process designers to be aware of the resilience of the processes during design time.

These approaches provide lists of generic policies and processes to improve resilience management, and thus have the benefit of being generally applicable to many types of businesses. However, they

provide little guidance for how to evaluate the benefit of each of these policies, as their impact on the cyber resilience of the company, and to the company's finances if an attack does occur, is not addressed by these works. As a consequence, they a) leave a lot of freedom for the company to inadvertently allocate investment inefficiently, and b) potentially reduce the incentive to invest in resilience in the absence of regulatory control.

Indeed, [CAH21] points out that there is gap in resilience capability levels between companies, especially for SMEs, which they explain by a lack of knowledge and suitable tools to evaluate the benefits from resilience improvements. This does not seem only restricted to smaller companies, as Accenture [BLDC20] also finds a large difference among large international corporations between the very best performing firms and others in terms of cyber resilience, and that the best are systematically better and target different things in order to ensure resilience, suggesting there are differences in their capabilities. For these reasons, there is a need for approaches that aim to evaluate the impacts of potential attacks and the cost involved with key enablers of cyber resilience (such as attack detection, redundancy, recovery), which can then provide an indication of what level of investment is cost-effective.

#### 2.1.1.2 Attack impact assessment for resilience

In addition to our work presented in this thesis, attack impact assessment for resilience has been considered by Haque et al. [HSK19, HSKG20], who have studied the cyber resilience of cyber-physical systems. They use graphical models for attack impact analysis, as is common in works on attack impact assessment, discussed in Section 2.2.1 below.

In [HSK19], they build a model for evaluating the resilience of an energy delivery system. They use a *network criticality metric* of a "vulnerability graph" to approximate system functionality after an attack, and then apply a sigmoid function to approximate recovery from this lower functionality level to normal functionality within a given time frame. This gives their estimate of the performance impact until recovery is completed. This approach provides an approximate, but not a detailed model of performance over time. The main weakness of this work is the lack of a model of attack propagation within the system – they only consider that attacks lead to random removals of edges in their graph, e.g. random communication links becoming unavailable. This could be an appropriate model to

evaluate the impacts of component failures, but cyber attacks typically do not impact components randomly but follow a strategy based on the vulnerabilities they can exploit within the system.

Their later work on mission impact assessment in CPSs [HSKG20] has a more detailed attack model, which they combine with estimates of attack impacts in a joint graph. However, the details of how this graph evaluates resilience and mission impact remain unclear. The work estimates the level of system functionality after certain components have been compromised, but does not consider the impacts of attacks over time. That is, there is no consideration for attacker behaviour and speed, nor the recovery process after attack. Thus, their model appears to only focus on the robustness aspect of resilience, minimising the peak impact, and not the over-time aspects (recovery and adaptation) and the losses that accumulate. The application they discuss is measuring the impact of a specific attack on a mission (maintaining SCADA functionality), and finding a security hardening investment that yields the minimum mission impact, using a simple model of security investment. Their investment model makes strong assumptions, such as all assets can be hardened, and that this can be achieved in a smooth linear fashion (e.g. investing 30% of a maximum level yields a 30% reduction in impact). Further, as the model does not evaluate attack impact and its cost over time, their investment model cannot be used to estimate how cost-effective the investment is.

#### 2.1.1.3 Practical solutions for resilience to cyber attacks

There is a strand of literature focusing on proposing what we shall call "practical solutions" for improving resilience to cyber attacks. By this, we mean that instead of frameworks, theoretical contributions or resilience measurement approaches, the works suggest ways that could be used in practice to improve the resilience of a system to attacks. Table 2.2 shows a few general patterns in this part of the literature. First, many of the works focus on specific types of attacks, such as zero-day attacks ([TCNFW16, ZWJ+16]) or lateral movement [CCR+19], instead of providing an approach more generally applicable to various attacks. Second, while many of the papers have defined resilience in a manner matching our "full" definition that includes recovery and adaptation (although with differences in some details), others have a limited definition including robustness only, or discussed resilience as a desirable concept without clearly specifying it. Third, the papers considered

here do not define a direct resilience metric, but consider deviations in some system performance level ([CRC+15], [TCNFW16], [CRDP17]), or the hosts reachable by lateral movement attacks [CCR+19].

**Comparison of studies proposing practical solutions to cyber resilience**

| Study | Notes | Events considered | Resilience definition | Resilience metric | Modelling and tools | Summary |
|---|---|---|---|---|---|---|
| [CRC+15] | proof-of-concept | failures, targeted attacks (only shown for DoS) | robustness only | based on deviations in general metrics of system health | graph-based | algorithm to recommend resilience-improving actions |
| [TCNFW16] | early-stage work on a modelling tool | zero-day attacks | full | descriptive rules to infer resilience | epidemiological | incident response choice framework |
| [NY17] | position paper | cyber threats | full | - | - (proposed approach only) | integrating security engineering to systems engineering process |
| [CRDP17] | active defence approach for ICS | cyber attack on industrial controller | - | based on performance level deviations | - | an active defence method for industrial control systems |
| [CCR+19] | designing resilience to lateral movement | lateral movement | robustness only | based on the reachability of lateral movements | tripartite graph of the enterprise (users, hosts, applications) | network hardening for robustness to lateral movement |

In resilience definition column, "full" refers to a definition that covers all parts included in our definition, such as robustness and recovery.

Table 2.2: Comparison of papers proposing practical solutions to improve cyber resilience

Choudhury et al. [CRC+15] propose a graph-based modelling framework for enterprise cyber-resilience, with an automated recommendation engine which monitors system health metrics and provides recommendations of actions to be taken if degradation in a metric is observed. The model consists of a set of graphs, representing the physical network, application level behaviour, permissions (users to resources), and host graph connecting applications to machines. The authors state that these can all be generated from network traffic data. In addition, a mission definition for the business is specified. The paper shows a proof of concept case-study with one health metric, quality of service in applications, and where recommendations involve blocking network flow from a subset of users, to fight DoS attacks. They run simulations to show that the system works in this simplified case.

The approach is interesting in providing automated resilience tools for a network, and combining different aspects of the enterprise. However, as presented, the framework is a tool for reacting to changes which are of a known type, in a deterministic way. It is unclear how this would be made to behave in unknown situations, which are of key importance for resilience. In addition, while things work in the proof of concept with only one metric, using multiple metrics could cause issues, as taking actions to improve one may have effects on another.

Tran et al. [TCNFW16] proposes a modelling tool for incident response and recovery to be applied to zero-day malware attacks. The tool is a system dynamics model for assessing the impact of countermeasures (investments, incident handling), using simulations that investigate incidence rate and recovery rate, and chooses the most effective countermeasure option. An epidemiological model of zero-day malware propagation is used. The model is demonstrated in action using a simulated phishing attack which plants a zero-day malware in a closed network.

While the proposed framework provides a quantitative tool for countermeasure selection for recovery after zero-day attacks, it is effectively an incident-response tool with responses that have some implications for resilience. The treatment of resilience is superficial, defining it as "the ability of the network to withstand a zero-day malware attack, provide discreet system and capability decommissioning, and enable recovery within an acceptable time frame." [TCNFW16, pp. 29] However, they do not use a metric, but descriptive rules – in the model, resilience is shown if the recovery rate is larger than the incident rate at any time $t$. Alternatively, resilience exists if, when comparing the incident rates under the best and worst case scenarios, it is shown that the incident rate decreases under the best case scenario.

Nejib et al. [NY17] is a position paper discussing security engineering approaches that should be integrated into systems and software engineering processes to enable NATO systems to plan/prepare, absorb, recover and adapt to threats, across the lifecycle of the system. However, as an early stage position paper, it does not propose any detailed plans for a framework or metrics, but just discusses the importance of integrating security engineering into the lifecycle of systems.

Chaves et al. [CRDP17] propose a resilience strategy for industrial control systems, using an active defence method to react to cyber attacks. This differs from a traditional industrial control redundancy strategy, aimed at recovery from failures, where the redundant controllers are identical to the ones they replace. This means that traditional strategies are susceptible to a common-cause failure by being vulnerable to the same attack. The paper shows an implementation of a method to achieve resilience via moving target defence and redundancy, using a resilience metric based on measuring a system performance level (dissolved oxygen in wastewater) over the time window when the event takes place.

The strategy is a good example of practical approaches to improve cyber-resilience, but it is only one specific solution, not a general framework that can apply in a wider range of systems. Further, the paper does not provide a way to do cost analysis. While the paper touches on the issue of cost in the conclusion, the approach does not contain the analysis required to determine if it is cost efficient to build the capacity to run the system in different configurations. These come with design and setting up costs that are likely to be much higher than in traditional redundancy strategy using the same vendor's controllers. A related aspect is the efficiency of attack detection, which is not considered in the paper. If it is not clear that the system was affected by a cyber attack, the strategy will be less effective, thus affecting the cost-benefit analysis.

An approach to improve network resilience to lateral movement was proposed by Chen et al. [CCR+19]. Their approach involves modelling systems using a tripartite graph of users, hosts and applications, and lateral movement attacks are modelled as walks over this graph. The model proposes network improvements to reduce the reachability of a lateral movement attack, i.e. how far an attack can spread in the network. The improvements considered are network segmentation and hardening of edges and nodes, where the former refers to making access from an application to a host secure, and the latter means securing a host, modelled as reductions to compromise probability.

For our purposes, the approach as described in [CCR+19] has the following key weaknesses: there is no modelling of production or performance, as improvements are quantified in relation to reachability only; the model does not provide an analysis of resilience over time, but currently only captures robustness (as impacts are not modelled); and the model is specific to lateral movement attacks.

There are also studies that propose actions that apply specifically to controllers in a control system. However, works that use the specifics of controllers to improve resilience, such as building security functionality into controllers [WDCO15], or by using control signals to respond to an attack (such as in [YWR18]) are not generalisable to systems without controllers, and are thus outside of the scope of this thesis.

To summarise, while several approaches for practical resilience in cyber systems have been proposed, they do not provide a comprehensive resilience framework as this stage. This is because many of them are at early stages (planning or proof-of-concept level), or do not consider all the aspects of

resilience that we find important, in particular recovery and adaptation. In addition, some approaches are focusing e.g. on controller design and are not generally applicable.

#### 2.1.1.4 Resilience metrics specific to cyber attacks

In addition to general resilience measurement, such as the performance-curve approaches mentioned in Section 2.0.1, some works have proposed other types of metrics for measuring resilience specifically in relation to cyber threats. Examples of such approaches are the use of model-checking to prove system properties that deny attacks of specific types [KAsR15], or metrics that should correlate with resilience, e.g. resource diversity based metrics proposed by [ZWJ+16]. These approaches could be useful in specific settings, but are not appropriate for use cases where system output performance is of key interest, such as our resilience impact evaluation methodology. For example, neither model-checking or indirect metrics enable the estimation of impacts on the system during attack events, which is key for production impact evaluation. As such other metrics are not of use for our methodology, we shall not discuss them further, but acknowledge that such works exist.

### 2.1.2 Other resilience works with relevance

#### 2.1.2.1 Business Continuity Management, Operations Research, Business Resilience

Business continuity management (BCM) is a framework for organisations to prepare for adverse events in such a way that the survival of the organisation is ensured after the events. It takes into account the organisation's overall goals, and focuses on what the critical functions are for delivering the organisation's product/service, and how to ensure their continued production after an event.

There is an ISO standard regarding the requirements for business continuity management systems, ISO 22301, originally introduced in 2012 [ISO12] and updated in 2019 [ISO19]. However, the existence of the standard does not mean the approaches to BCM taken by companies are effective, as the standard is not compulsory and leaves a lot of room for variation in its implementations. Indeed [SSB20] lists several limitations of BCM, which mostly boil down to lack of appreciation by man-

agement toward the BCM process, in addition to the ambiguity of the tasks that are included in BCM. Combining insufficient managerial enthusiasm with the vagueness of the approach leaves extensive room for managers to undermine the process where it might interfere with other business goals. These weaknesses support our view that to ensure that system resilience is taken seriously by management, an approach is required to quantify the impact of the lack of resilience onto the core business. This is a key reason for our focus on performance quantification and cost evaluation.

Some works discuss business continuity planning (BCP) in relation to resilience as we consider it here, including recovery. Sahebjamnia et al. [STM15] provides a resilience approach that integrates business continuity and disaster recovery planning (DRP). They propose considering both BCP and DRP jointly, with the key problem being the choice of how many resources to allocate on available plans for each, i.e. trying to optimise the allocation of resources between response and recovery. They do not model how the events occur, but focus on the recovery stage assuming random disruptive events take place. The model in [STM15] is a neat theoretical model, setting out the problem of choosing the allocation of resources to the response and recovery, for various key processes. However, the model assumes a great deal of information is available when it is applied – the impact of each disruptive event is assumed to be known and reflected by the parameters, as are their likelihoods, and the effectiveness of different resource levels at maintaining/recovering production. This means that the model does not provide the users a way to evaluate the different quantities (impacts, probabilities, etc.) relating to a particular type of event (e.g. a cyber attack), but only shows how one would solve the allocation problem if this was known. Additionally, the impact of the events is reflected at the level of different services/components, but how this impact occurs, including how the disruption of components affects others, is not modelled. In contrast, we think it is necessary to model how the attacks occur, and what their likelihoods are, to quantify the impact within a system. Considering the interdependencies of various components/services to model their effect on each other and the overall output of the system is also necessary.

Furfaro et al. [FGS16] consider cyber attacks in the context of business continuity planning, focusing on systemic risks in the banking sector. They provide a graphical modelling approach to requirements specification for BCP, using a goal-oriented modelling methodology GOReM that is based on UML. The modelling is used to identify stakeholders and their goals, and process steps for handling incidents

in the banking context. The approach does not provide mathematical modelling that would be required for impact and resilience analysis, but instead is intended to aid with BCP and DRP in terms of identifying activities that are needed in response to incidents.

Suresh et al. [SSB20] propose a BCM methodology focusing on supply chains, with an approach that tries to combine BCM modelling with supply chain resilience/risk management approaches. The proposed methodology consists of high-level phases that supply chain risk management should include in order to be prepared for catastrophic events. However, as the paper does not provide an implementation or evaluation of the methodology, this study is more a description of a research direction rather than a fully developed approach.

A few papers in the resilience literature discuss approaches to business resilience, which could be used as an implementation of business continuity planning. Zahoransky et al. [ZKA14, ZBK15, ZHLB16] propose a resilience approach to business process management (BPM). The works propose using process logs for automated modelling of business processes, and use these models to analysis of resilience characteristics of the processes. The analysis examples provided in the papers focus on compliance, i.e. that the workflows in the process comply with rules, regulations and standards. The events that cause disruptions are not modelled, but the historic logs used for building the model are assumed to include cases of disruption to the process. In [ZKA14] the analysis aim is to build a probability density function for the timeliness of the overall process, based on historical data on the time profiles of the different tasks. In [ZHLB16] the historical log data is tested, using model checking techniques, for violations of various conditions that the process should comply with.

While the approach in [ZKA14, ZBK15, ZHLB16] models and analyses business processes, which is relevant to us, the modelling is at the level of process tasks and does not contain detail on the services and components needed to provide these tasks. Thus this approach does not provide the link from the system components (and their vulnerabilities) to the process. Hence, we cannot make a cyber-attack analysis approach based on this, as there is insufficient information to model the propagation of attacks and the components impacted by them.

Losada et al. [LSO12] propose a model for optimising investment into facility protection against worst-case attacks, where protections reduce the time to recovery. This is a high level model of supply

from several facilities. The proposed facility problem assumes that there are multiple facilities that can provide the same service the customers, and the aim is to provide the service at the fastest rate (i.e. from the closest facility). The protection model assumes that an investment to a 'protection resource' provides a given level of reduction in recovery time to a facility. That is, there is no detailed modelling of the system in the facility (the production process, or attack progression), just the assumption that an attack will cause a disruption, and an investment will reduce the recovery time.

Due to the assumption of identical facilities, this approach does not generalise to modelling processes with different types of components, and the attack model only applies in specific cases. The approach could work for a system with identical components, such as a group of UAVs, as such a setting could support an attack model of this type if the attacks could target any node in the graph equally. However, if the progression of an attack across the components is dependent on connectivity or vulnerabilities, a more detailed attack progression model would be needed, and this particular model would not apply.

Giahi et al. [GMH20] propose a model for optimising engineering designs for resilience according to the risk-appetite of risk-averse firms, as opposed to strictly profit-maximising risk neutral firms. Their application area is on the design of engineered systems, in their case a wind turbine, and thus does not match our aim of providing a generally applicable modelling methodology. Further, they do not model cyber attacks, but test their designs against random adverse conditions. However, the approach to resilience they adopt is similar to ours in that they model the expected performance of a design during adverse events until recovery. They relate the performance curve to expected profits, and incorporate risk-aversion in one of two ways: using an exponential utility function, or with a Value-at-Risk (VAR) constraint. While the approaches to modelling risk-averse decision makers are known in economics, the paper shows an example of how risk-aversion can be incorporated into the decision making process with regard to designing a system, trading off profit for added resilience.

As discussed in Section 2.1.1, there is a strand of literature that addresses resilience planning for businesses by listing generic actions that can improve the cyber resilience of a business, e.g. "have a resilience plan", "have security technology" etc. Several works, [CAC+10, MMPP19, CBL+20, CAH21] approach this with the use of maturity models. As these works were already discussed in Section 2.1.1, we shall not discuss them further here, but simply acknowledge their relation to BCM.

Overall, while BCM and business resilience works have a similar focus as this thesis does, on resilience to adverse events, none of the works reviewed above approach impact assessment in a way that meets the aims of our proposed resilience impact assessment methodology. That is, while some provide process models and estimates of costs due to events, none of the works provide detailed attack modelling, or a generally applicable quantitative model of how the compromise of one component affects the performance of the overall process (e.g. in [LSO12] the facilities are considered identical, and thus their performance modelling does not generalise to processes with various components).

#### 2.1.2.2 Resilience in networked systems

Given that a wide spectrum of systems can be described as networked systems, there are a considerable number of publications discussing resilience in such systems. As our focus is on cyber resilience and thus cyber networks, this section considers papers in the literature with approaches and insights that are applicable reasonably widely to networked systems, leaving out literature appearing too specific to a particular setting. We have, for example, excluded papers relating to community resilience and transport system resilience, and those focusing on only one narrow type of event.

For our purposes, the main limitation of network resilience approaches, as discussed in this section, is that approaches that are based on *network performance* typically cannot be used as-is on a more general system. For example, network robustness, structural resilience, and graph robustness metrics are approaches that focus on evaluating the performance of the network structure, where each node is essentially functionally identical, apart from their location in the network. However, in other systems different components can have differing roles and performance impacts on the overall system, and then the network structure is not enough to describe system performance or resilience.

**Resilience approaches to networked systems**

A comparison of the key features of the papers on resilience in general networked systems is provided in Table 2.3. A few research themes can be seen: resilience analysis frameworks, structural resilience estimation, component importance, and network resilience prediction using graph topological metrics.

| Comparison of studies in networked system resilience | | | | | | |
|---|---|---|---|---|---|---|
| Study | Study type | Events considered | Resilience definition | Resilience metric | Modelling and tools | Summary |
| [SHÇ+10] | resilience framework for communication networks | link, node and protocol failures | incomplete (no recovery or adaptation) | the range of operating states at which the system maintains acceptable service | networks; simulated events | framework for resilience and survivability in communication networks |
| [GMG+16] | resilience analysis framework | disruptions inactivating nodes, causing cascading failure | full | based on performance over time | dependency networks; simulated events | metric for resilience analysis |
| [YWZ17] | resilience analysis framework | disruptions to power systems | full | based on performance over time | dynamic Bayesian network | predictive resilience analysis for dynamic engineered systems |
| [GBB16] | structural resilience analysis | perturbations (node removal, link removal, global link weight reduction) | full | binary (resilient or not), given by the stability of a desired fixed point of a resilience function | graphs with weighted interactions (ecological and gene regulatory networks) | analytical tools for deriving simpler dynamics that predict the resilience of a complex system |
| [WL16] | structural resilience metric | virus with specific rules | - | predictive metric | networks, converted to a type with specific rules | providing a proxy metric for structural resilience |
| [ZMSG18] | network design optimisation for resilience | any disruptive event | full | based on performance over time; network flow | design optimisation problem | finding optimal network structure designs based on resilience constraints |
| [FPZ16] | component importance for resilience | disruptive events in critical infrastructure networks | full | based on performance over time, but with emphasis on recovery | network; stochastic ranking to rank components | two metrics for measuring components' importance on system resilience |
| [ZKL+17] | component importance for resilience | disruptive events in complex networks | full | based on performance over time | network, Monte Carlo simulation algorithm for ranking | three metrics of component importance for resilience |
| [BB18] | component importance for resilience | disruptive events in critical infrastructure networks | full | based on performance over time (ratio of recovery over loss) | network (no model) | one component importance metric updated with Bayesian kernel approach |
| [AS15a], [AS15b], [AS15c], [Ale16] | graph robustness metrics as resilience predictors | targeted attacks on nodes with highest centrality | incomplete (no recovery) | sum of the flow robustness values while attacking nodes (3 measures) | graphs with various traditional and random topologies; simulated attacks | evaluating different graph robustness metrics from literature as resilience predictors |
| [SC17] | graph energy as resilience predictor measure | targeted attacks on nodes and links of highest centrality | - | - | graphs with real-world network topologies; simulated attacks | considering graph energy as a resilience predictor measure |
| [WSD15] | resilience metric based on network topology | random disruptions and targeted attacks | - | aggregate multiple-path reachability of all demand nodes | supply graphs (random and scale-free); simulated attacks | topology-based resilience (robustness) metric |

In the "resilience definition" column, dashes mean that the paper did not provide a clear definition, and "full" refers to a definition that covers all parts included in our definition, such as robustness and recovery.

Table 2.3: Comparison of papers on resilience in networked systems

**Resilience frameworks:** Sterbenz et al. [SHÇ+10] discuss resilience and survivability in communication networks, from the perspective of service provision in networks such as the Internet, and

discuss implementation of such strategies with relation to a resilient networking framework called ResiliNets. Although the framework focuses on a specific type of system and thus includes details that do not generalise wider, they quantify resilience in an interesting way. They define resilience as "*the ability of the network to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operation*" [SHÇ+10, pp.1246]. While this is closer to what we would call robustness, the paper also discusses the adaptation and evolution part which we are interested in. In particular, they discuss a framework for resilience evaluation in terms of operational states and service parameters. In their definition, resilience is defined by the range of operational conditions that yield an acceptable service level, with an ideal resilient network being one where the service level remains in the acceptable region even if the operational conditions are severely degraded.

Despite mentioning adaptation, the authors' focus is on the design and implementation of survivable (robust) networks, not in adaptation, which they only count as an enabler of resilience. Further, for our aim of making a generally applicable approach, a key weakness of the paper's treatment is that it considers networks where service provision of data transmission (relating to connectivity, reliability, and speed) is the main consideration, while systems with more involved services have to consider a wider selection of effects from events, such as the cost of the loss of data, reputation effects etc.

Ganin et al. [GMG+16] proposes another metric for network resilience which defines resilience relative to the level of network functionality over time. They build on similar approaches applied to seismic disaster resilience by [BCE+03] and [CRB10], but make the approach more general to enable its application to more varied settings. The explicit aim of the paper is to provide a quantitative measure of resilience which can capture four key aspects of resilience as promoted by the National Academy of Sciences (plan and prepare; absorb; respond and recover; adapt), which are not considered by e.g. the method in [SHÇ+10]. The approach is for networked systems, and is applicable across physical, information and social domains.

The authors provide an analytical definition of their measure of resilience, which applies to networks modelled as graphs. They use a performance measure they term 'critical functionality', which measures the overall functionality in the network at each point in time. This functionality will drop in the face of adverse events that affect the functionality of the nodes or edges of their graph, and can rise

back due to recovery or adapting. With this, they define their measure of resilience as the time integral of critical functionality from an initial time to a control time, $T_C$. In effect, resilience is the cumulative level of functionality in the network during the time window. This approach is largely equivalent to a [BCE+03] type resilience triangle metric, such as in Fig. 2.1a, but with critical functionality as the performance metric and a fixed end time $T_C$. Given that $T_C$ can be chosen freely, and as the measure is mainly intended to be used for analysis via simulations, the time window can be made to include both the time before and after an event has taken place. Thus the resilience measure can cover all of the aspects of resilience in the NAS definition: planning, absorption, recovery and adaptation, and it will also react to all efforts to improve critical functionality and resilience during them.

The paper proposes using the measure as part of a resilience analysis framework, by comparing the metric's values for a system configuration (or different configurations) in adverse event simulations. This is shown by example in two kinds of networks, Directed Acyclic Graphs (DAGs) representing dependencies between network nodes, and coupled networks where some nodes have dependencies across the two networks.

The approach for measuring resilience by [GMG+16] is useful for our purposes as it is a quantifiable measure that can be calculated automatically if the functionality levels of nodes can be observed. This is useful for analysing the impacts of different resilience improvement techniques and attack responses, as it enables running simulations to test the resilience impacts of the techniques and response approaches under different attack scenarios.

The limitation of the measure as proposed in the work is that it is only applicable to networks representable by a graph where the functionality of nodes and edges is the measure of interest. While this holds for some applications in multiple domains, as the paper states, other measures are required where the assumptions do not hold. For example, situations where the nodes are heterogeneous and services provided by some nodes are more important cannot be captured in a very satisfactory way with this metric by just weighting the nodes, but some other measurement approach would be required to reflect this. Also, the measure is silent on levels of functionality that are required for a service to be viable – the discussion only assumes that a network can be at varying levels of functionality (or completely down), but in many cases it is more useful to know what level of specific service provision

is available, not how many of the nodes would technically be able to function.

Yodo et al. [YWZ17] performs predictive resilience analysis using a Dynamic Bayesian Network (DBN). In the paper, a DBN is used for modelling the relationship between the system, reliability and restoration. Disruption is modelled as random events that can cause components of the system to fail, while a restoration node represents a recovery process. The system's dynamics over time are modelled as dependencies between the system states at different time slices, and resilience is quantified by the binary state of a node (resilient or not) based on the state of restoration and reliability nodes. The approach is demonstrated in a case study of comparing the resilience of two power grids serving customers at different locations under three disruption scenarios, with disruptions applied on the transmission subsystem, the distribution subsystem, and both. Being a high-level model, this approach could be used for resilience modelling and quantification in many settings, although the models would have to be adapted to each setting. On the other hand, the paper concedes that the DBN approach may not be tractable in realistic-scale complex systems.

**Structural resilience of networks:** Some papers, such as [GBB16, WL16], have studied how the structures of networks help them be resilient in face of disruptive events. In a similar vein, [ZMSG18] consider the resilience of the maximum traffic flow in a network, and search for designs (choice of links between nodes to build) that enable satisfying a performance recovery requirement. These works consider whether a network can exhibit resilience to some desired level, but do not suit our purposes for performance-based attack impact evaluation, as they only consider the performance and resilience of the network itself, and thus do not apply to settings where the production performance of the underlying system is of interest.

**Component importance:** Some papers have considered the importance of different network components for resilience. Fang et al. [FPZ16] proposes two metrics for measuring the importance of network components based on their impact on system resilience: 1. optimal repair time, reflecting the repair priority of the component, and 2. resilience reduction worth, measuring the potential loss in system resilience due to delay in the component's recovery. A Monte-Carlo method is used for generating probability distributions of the metrics for all components, and a stochastic ranking is used to rank components' importance.

Zhang et al. [ZKL+17] proposes three metrics of component importance: 1. structure importance, which measures the extent to which a component's failure affects the system resilience; 2. redundancy importance, capturing the influence of component's redundancy (having a backup component) to the overall system resilience; and 3. reinforcement importance, which is the extent to which improving the component's individual robustness affects the system resilience. A Monte-Carlo simulation algorithm is used to rank components by importance.

Another paper on component importance, [BB18], revisits the "Resilience Worth" component importance measure, combining the previously used probabilistic assumptions with a Bayesian kernel approach allowing prior information on component characteristics and historical disruption data.

While component importance metrics, such as those discussed in the papers mentioned here, could be used to provide an indication as to what components need to be defended the most, or need most redundancy, overall these approaches are not relevant for attack impact evaluation, which is the core of our methodology. Thus, we do not consider such works further.

**Graph topology metrics:** Some works have considered the use of various graph metrics as predictors of network resilience. Metrics considered include standard graph metrics (various centrality, connectivity, criticality, diversity metrics) in [AS15a, AS15b], spectral metrics [AS15c, Ale16], and multi-path reachability [WSD15]. While none of these metrics measure resilience directly, they are proposed as predictors of the resilience of the network measured in terms of network flows. Most of these works ([AS15a, AS15b, AS15c, Ale16, SC17]) are on network resilience, while [WSD15] relates to supply-demand networks, where the aim is to ensure that each demand node has a functioning path to a supply node. As such, they are concerned with the availability of network connectivity and link availability.

Graph metrics such as those discussed in the works mentioned could be used for predicting the resilience of networks with functionally identical components that have equal importance for the overall system performance apart from what is caused by differences in location. As a consequence, they are valid when the performance of interest is the availability of network flows, e.g., in communication networks. However, this is not the case for production systems (such as production of services or products) or mission processes, as in such cases most components cannot be considered identical in

their impact on the production performance. Some subsets of components will be identical to each other, e.g. identical server copies for the same task, but in general a server used for task A is not identical to another used for task B in terms of their process impact. As a consequence, graph metrics could work as resilience predictors for only a very specific set of systems.

## 2.2 Attack impact assessment

### 2.2.1 Attack impact assessment works

Several approaches to evaluate the impacts of cyber-attacks on a system and its components have been proposed, most of them representing system components in some type of a dependency graph, and providing a model of how attacks affect these components.

Kheir et al. [KCBCD10] proposed an approach to evaluating the cost of responses to attacks, including a way to evaluate response impacts on the system. They model *privilege dependencies* between services in a system, and calculate the extent of disruption that different attack responses cause onto normal users of the system. The attacks considered are ones where an attacker grants inappropriate privileges to unintended users, or revokes privileges from legitimate users. Their model is evaluated using a coloured Petri net (CPN). However, the approach does not fit the purposes of our resilience impact estimation, as it does not consider impacts over time, nor does it consider service performance beyond determining if groups of users are still available to use the system or not.

Jakobson [Jak11] proposes a mission impact assessment approach, based on an "impact dependency graph". This graph shows how a mission depends on its various sub-tasks (mission steps), the services required to fulfil these tasks, and assets (hardware, software) that the services in turn depend on. The mission impact assessment uses these dependencies to evaluate the overall impact arising from attacks on assets, the direct impacts of which are approximated using CVSS [MSR06, FIR16] vulnerability scores. While the approach to using a dependency graph to assess overall impact is similar to other approaches proposed since, the attack model in the paper only considers whether an attack to a given asset has occurred or not, without providing an attack propagation model for considering multi-

step attacks. Consequently, the approach as presented cannot be used for assessing the likelihood of different attack outcomes, the time it takes to proceed or how a response to an attack could occur.

The combined use of attack graphs (AGs) and dependency graphs (DGs) for analysing the impact of cyber attacks has been explored by Albanese et al. [AJPS11, AJ17]. Kotenko and Doynikova [KD14, KD15, KD16, DK15] employ these graph tools in a countermeasure selection system, and Shameli-Sendi et al. [SSLHC16] use similar graphs for selecting countermeasures in an intrusion response system.

Both the Albanese et al. and the Kotenko & Doynikova approaches rely on connecting nodes in the attack graph with relevant parts of the DG for the use of dependency information to quantify the impact of a given attack step. However, the approaches differ significantly on how the DGs themselves are used. The method by Albanese et al. [AJPS11, AJ17] holds the DG alongside the AG as part of dynamic analysis, with the effects of attack steps on the DG held as key for impact assessment. By contrast, Kotenko & Doynikova [KD14] focus on enriching the system's AG with topological and service dependency information, and using this augmented AG for the analysis. The DG is employed as a source of component importance information at the pre-processing stage. The [KD14] approach, therefore, implicitly assumes that any changes occurring during an attack do not significantly affect the information obtainable from the dependency graph.

In our application to attack countermeasure selection in Chapter 9, our proposed approach has similarities to the countermeasure selection techniques in [KD15] and [SSLHC16] in the use of AGs and DGs, using costs to quantify attack impact, and basing countermeasure decisions on costs. The latter similarity also applies to [KCBCD10] discussed above. While these works focus on containment in terms of stopping attacks or reducing the risk of them reaching pre-determined goals, we aim to find a strategy that is efficient over longer time, considering all costs until achieving recovery to previous functionality.

More recently, a few works [SSL17, CYS+18, HSKG20] have proposed the use of impact graphs built using the MulVAL tool [OGA05] for graph based network security analysis, which has often been used in the literature on attack graphs. These approaches are similar to ours in combining attacker information and component dependencies ("entity dependencies" [CYS+18]) for estimating

impact. The key difference is the evaluation of impacts, which is based on pre-evaluated estimates of vulnerability severity, and not directly evaluated based on process/production impact as we do it.

In [SSL17], the focus is on mission impact assessment for cloud computing. They build a "mission impact graph" that links attack graph information to their impact on services and mission tasks. Their assessment can show which missions are under threat by attacks, and if a given asset or service has been compromised, which missions are affected by it. However, the approach does not actually quantify the impact, e.g. what is the extent of performance loss, or what the cost of a breach is. For their case study of a data breach (stealing confidential information) there is no need for assessing system performance impact as the attack steps do not cause disruption, but estimating the cost of the breach would be beneficial for determining the benefits of improving defences.

Cao et al. [CYS+18] quantify impacts on business processes using CVSS [MSR06, FIR16] scores (specifically the base scores) of vulnerabilities. They calculate the overall impact by passing the impact scores to the business processes via their dependencies. The [CYS+18] approach uses three layers of information: the business process, describing tasks required for a given process and their dependencies; the services required by these processes; the assets where these services are located. The connections between these layers are used to map attacks relating to the assets, which are then mapped to the services and tasks to evaluate the impact to the process.

The approaches in [SSL17, CYS+18] (and [HSKG20], discussed in Section 2.1.1.2) provide a useful straightforward way of estimating impacts, which should be intuitive for users of AGs for risk analysis. However, for the purposes of resilience evaluation, the approaches are lacking in not providing an estimate of business impact over time. For this, one needs a more detailed model of production impacts from disruption (i.e. not just based on CVSS scores), including the duration of disruption.

As this section shows, there have been various works discussing attack impact assessment by combining system structure information in some version of a component dependency graph with a representation of how attacks can impact the system components. However, none of the works meet all the requirements we have for analysis of resilience, and for being able to study the cost-effectiveness of approaches to improve resilience.

## 2.2.2 Attack modelling approaches usable for impact assessment

### 2.2.2.1 Attack graphs

An attack graph (AG) is a network risk-assessment tool that provides a graphical representation of actions that an attacker can take to reach an attack goal, for example root access on a given server, by exploiting vulnerabilities that exist in the system. While a given vulnerability may not pose much threat on its own, they may be usable as part of a multi-step attack path to the attack goal, by chaining together exploits of multiple vulnerabilities. Attack graphs show all the vulnerabilities, and the privileges that can be gained by their exploitation, which can be used as part of such a multi-stage attack toward the goal. Given this, they can be used for assessing the risk that a specific part of a system becomes compromised due to an attack, and for analysing how system risk is affected by changes to the system configuration (e.g. changing the network topology, or patching vulnerabilities).

Various definitions and ways of creating attack graphs have been proposed. The earliest approaches to AGs suffered from poor scalability, with the graphs exploding in size before realistic network sizes were reached, making them impractical for application on medium-to-large networks, as discussed in a review of the early literature by [LI05]. Later approaches have improved the usability of AGs; surveys of the more recent research have been provided by e.g. [ZWC+19] and [LDB20].

The introduction of **Logical Attack Graphs** by [OBM06] made AGs more viable. The nodes in a logical attack graph are logical statements (fact and derivation nodes), and edges show dependencies between the nodes. A derivation node directed from a fact node represents a reason [e.g. an attack step] for the fact to take place. A fact node directed from a derivation node represents a requirement for the derivation node, i.e. precondition for the attack step in the derivation node. The logical attack graph concept reduces the graph size required to represent an AG, and makes graphs simpler and faster to generate.

The usefulness of attack graphs is best appreciated via an example. Fig. 2.3 shows a sample attack graph commonly used for illustration purposes in the literature, versions of which have been used for example by [FW08], [AJPS11], [MGSBL17]. The left side shows the network with two hosts (Host 1 and Host 2), a router and a firewall, while Host 0 represents a machine in the internet, a potential

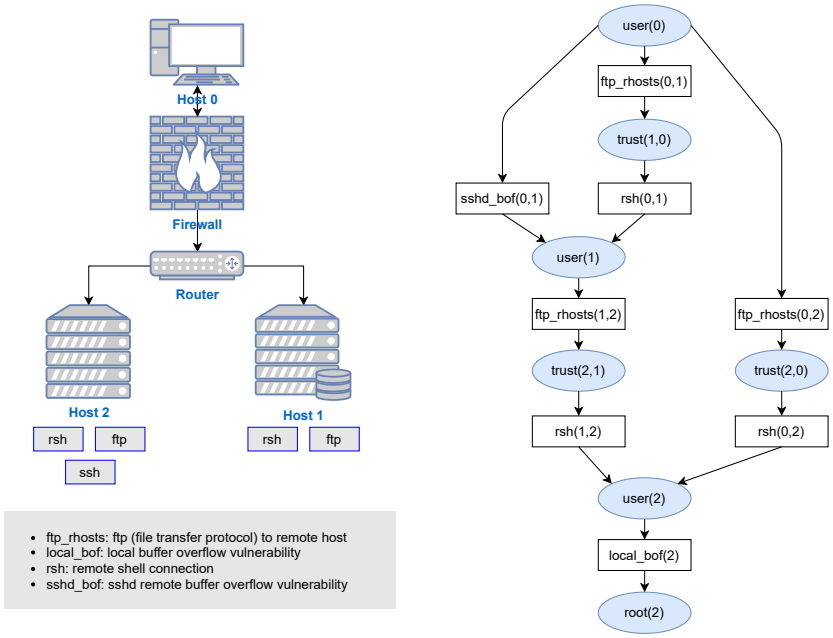



Figure 2.3: Sample attack graph

attacker. The graph on the right hand side is a logical attack graph related to the network, where the rounded nodes represent privileges that a user can have on some machine in the network, and the boxes are vulnerabilities that can enable an attacker to obtain more privileges via exploits. For example, the attacker can reach the goal node "root(2)" representing root access to Host 2 by first exploiting the ftp_rhosts vulnerability to gain the trust of Host 2, then the rsh vulnerability to get user privileges at Host 2, and finally a local buffer overflow vulnerability to gain root access at Host 2, i.e. via the path [user(0); ftp_rhosts(0,2); trust(2,0); rsh(0,2); user(2); local_bof(2); root(2)].

The usability of AGs as tools for risk assessment was improved by **probabilistic AGs**, which use estimates of vulnerability exploit probabilities to provide network risk measures. A typical approach to providing risk probability estimates treats the Common Vulnerability Scoring System (CVSS, [MSR06, FIR16]) scores of the severity of vulnerabilities as a reflection of the likelihood of a successful exploit, and derives estimates of the probabilities of compromise at the nodes of an AG based on these. For example, [WIL+08] and [OS12] introduce methods for risk-evaluations using simple probability estimation methods with logical AGs.

Additional rigour to the risk assessment using probabilistic AGs was introduced by combining Bayesian inference with attack graphs. Liu et al. [LM05] introduced the concept of **Bayesian attack graphs**, using Bayesian network to model potential attack paths, as an alternative attack graph type providing

improved probabilistic analysis. Poolsappasit et al. [PDR12] proposed the use of Bayesian AGs to improve the risk assessments of networks, by using them to dynamically determine the likelihood of different nodes becoming under attack during the deployed phase of the network.

Albanese et al. [AJPS11] introduced another approach to probabilistic analysis in attack graphs, the concept of Probabilistic Temporal Attack Graph. The innovation here is the use of time windows during which vulnerability exploits can take place at a certain probability. The time to exploit a vulnerability $v_i$ can start $x$ time units after the exploit of a prerequisite vulnerability $v_j$ (a vulnerability that has to have been exploited to enable exploit of $v_i$), and can be exploited until time $y$ (at which time it's assumed that the vulnerability is patched or the attack is discovered). Within this time frame, the probability of a successful exploit is $\rho(x, y) \in [0, 1]$. The graph proposed also differs from many other approaches by having the nodes denote vulnerabilities, with security states/conditions included implicitly in the edges connecting the vulnerabilities. Figure 2.4 shows this approach and how it relates to a traditional logical AG, in addition to a Bayesian AG approach from [MGSBL17].

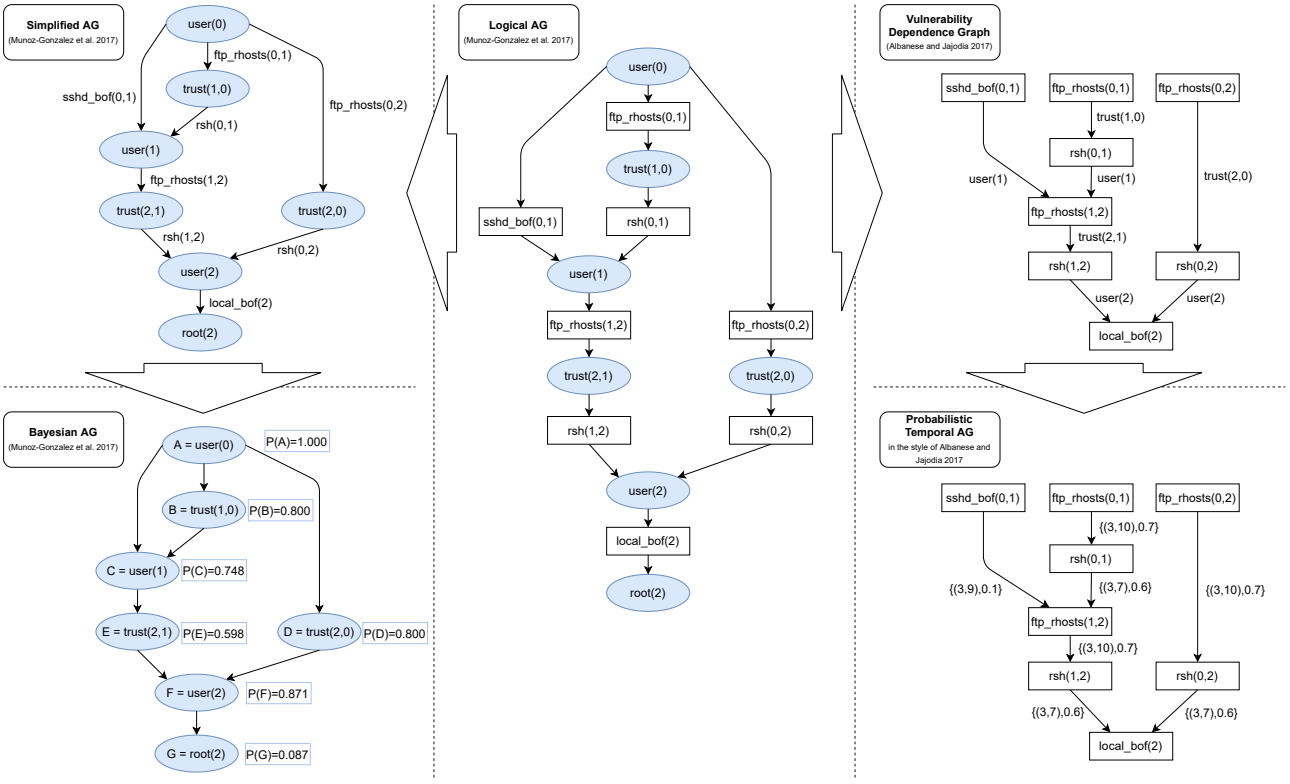



Figure 2.4: Different AG approaches: Logical AG and two approaches to simplify the representation, and two related approaches for probabilistic analysis. The graphs are adapted versions of ones from [MGSBL17] and [AJ17], using the same logical AG sample (in the middle) to illustrate their methods.

There have been works that use attack graphs to analyse risks in terms of mean-time-to-compromise

(MTTC) such as [MBFB06, LB08, LFK+11, NWJS13]. These studies have focused on evaluating the time required to compromise a system, and do not model impacts and cascading effects in inter-dependent systems as we intend to do.

The usefulness of attack graphs for our proposed resilience methodology relates to the modelling of attack progression. An AG can be used for risk analysis regarding possible event scenarios, and this information can be combined with impact analysis methods to aid planning and strategy creation. For our purposes, we do not benefit as much from the standard risk evaluation use of AGs, but use the AG structure to model the possible paths of attacks in a system, to be used in analysis of possible outcomes of specific attack instances. As an example of this difference, a BAG analysis will provide probabilities that an attack will compromise (or will have compromised) the various nodes in the AG, given a specific state of attack. If we combine these probabilities with the cost obtained from dependency models, we would get a set of risks of "loss", without a time aspect specified (as time is not represented by the BAG). This can be good for prioritising risk scenarios, but not for analysis where time is an important element, or for comparing various outcomes of specific attacks. This latter part is needed for evaluating the effectiveness of actions that affect the attack behaviour or improve system resilience, i.e. redundancy, recovery of nodes, allocation of additional resources.

#### 2.2.2.2 Other attack modelling formalisms

**Petri net approaches:** Stochastic Petri nets (and variants such as stochastic activity networks) allow for the modelling of transition time (with distributions), enabling discussing the duration of exploit steps. They are typically used for analysing time to reach certain nodes, providing a time distribution.

Petri net formalisms have been used before for attack modelling in works such as [MT04, LLC+18, DW06, CH19]. Madan and Trivedi [MT04] proposed using generalized stochastic Petri nets (GSPNs) as a tool to solve a security quantification model that incorporates attacks and responses, an attack graph variant called attack response graph (ARG). Their approach is to closely replicate the ARG in GSPN form. The method proposes a straightforward conversion of standard AGs to GSPNs; however, it yields a complex representation using a high number of places and transitions as each AG node is given its own place and related transitions.

Li et al. [LLC+18] use a GSPN to model the impact of coordinated topology attacks to smart grids. Their use of a GSPN for attack impact modelling in the presence of countermeasures is superficially similar to our evaluation of expected attack impacts in our mission viability assessment example in Chapter 10. However, a key difference in the attack modelling is how the attack progresses: the propagation of the compromise within the system, from component to another, is a key feature of our work for assessing the impact across the different steps of an attack, while their paper models attacks to different components as individual attacks without a direct link between compromises.

Some works have extended features that add to the expressiveness of the Petri net formalism, at the expense of the closed-form solution method that is one of the key features of formalisms such as GSPN. For example, Dahl and Wolthusen [DW06] introduced a Petri net formalism for attack scenarios where the transition firing times are distributed within an interval, while Collins and Huzurbazar [CH19] propose a formalism that allows for arbitrary firing time distributions. Such models require simulated solution, as they do not satisfy the memoryless property that would allow them to be solved as a CTMC. Simulation approaches have benefits such as supporting added model features, but drawbacks on solution time due to the numerous iterations required to obtain approximate average behaviour. However, as the simulation tools are not shared by the authors and the analyses and models are used in different contexts, direct comparisons are not possible.

A weakness of Petri nets as attack models is that they typically assume random transitions based on what transitions are enabled. This is not a good description of intelligent attackers in settings where choices are available, and therefore needs to be controlled with the model structure and inhibitor arcs etc., or by providing an extended modelling formalism with custom features, which means losing some of the benefit of using a well-known formalism. By comparison, the ADVISE model [LFK+11] discussed below added some limits to this, making attacker choices based on utility over a few parameters (costs, detection probability, goal).

**ADVISE:** Another related graphical attack model is the attack execution graph (AEG) proposed in ADVISE [LFK+11]. Several aspects separate the AEG from typical AGs, but the key conceptual difference is that, in addition to specifying attack steps towards a goal, the AEG specifies skills and system knowledge that attackers require for specific steps. This structure, used in conjunction with

attacker profiles, enables explicit modelling of the behaviour of different attacker types. By comparison, in standard AGs attacker capabilities are typically not modelled explicitly, but approximated by assumptions on success probabilities for attack steps.

Overall, as a formalism ADVISE is in many ways similar to a stochastic Petri net, but adds priorities over the choices of moves of the attacker, so the *attacker steps are not random, but based on a more realistic model of attacker decisions*. It also adds attacker capabilities, although the papers describing ADVISE do not provide an approach to estimating and setting the parameter values for these, making them seem hard to calibrate.

**Attack emulation:** Automated adversary emulation has been studied by e.g. [AMS+16, AMS+17, MAA+18]. A prominent outcome of work in this field is the CALDERA tool [MIT19]. CALDERA is an open source attack emulation platform, originally intended for automated red-team exercises (creating an adversary profile and launching it on the system to be defended). Its use cases have been extended to also include automated incident response and manual red-team assessments, both taking advantage of the emulation agents provided by the platform. These evaluations are run on the system that is to be tested/defended, like in normal red team exercises, as opposed to models of systems.

From the perspective of our work, attack emulation platforms such as CALDERA pose the difficulty of requiring an environment to operate in, i.e. setting up an evaluation would require building the case study system in detail. Thus, the use of such a tool would require an extensive and detailed modelling exercise, emulating the system and its environment in detail. By contrast, in its current form, our methodology is intended to operate at a higher level of abstraction, which is more agile to use when planning and evaluating changes to system architecture. However, for implementations of our methodology in practice, an emulation tool could perhaps provide features of the attack progression model component of our methodology, as part of our impact assessment.

### 2.2.3 System production and performance modelling

To estimate the impact of cyber attacks on the performance of systems, one needs to model how the system produces its output, and how to express the performance of this process. Some systems

can be modelled at a high level as a network with nodes that are functionally equivalent as part of the network. For such systems, the performance of the overall network can be modelled with various graph and network metrics; specific resilience approaches for them were discussed in 2.1.2.2. However, the production performance of systems where components of interest have different roles as part of the overall process require different modelling. In this section we discuss some approaches to such modelling, with a particular focus on how they could work as part of our overall approach, and be combined with attack progression modelling.

Practically any production process can be represented in terms of dependencies between components or stages of production. For example, to produce output (i.e. sales), an e-commerce requires both a web server and an order processing server, which themselves likely depend on many other services. Similarly, the manufacture of physical goods or the provision of services such as healthcare involve steps that depend on each other to produce the final output. Furthermore, almost any model of a production process must describe how different components or tasks connect together to form the overall process, and thus dependencies are inherent in modelling formalisms: e.g. Petri nets, queueing networks, stochastic activity networks have dependencies built in, and even purely graphical business models such as BPMN (business process model and notation) describe how tasks connect to form a process. It is unsurprising, therefore, that a dependency model of some type is used in all the attack impact assessment works discussed in Section 2.2.1.

In addition to describing dependencies between system components or tasks, attack impact evaluation needs to address the way in which the overall system performance is affected by cyber attacks affecting some of the components. Ways in which performance impacts have been modelled in the literature include: a) impact rules based on dependency types, e.g. dependency functions or interaction rules [AJPS11, AJ17, SSL17, CYS+18] that describe the nature of dependency between a given component (e.g. service) and other components that it depends on for fulfilling its purpose, combined with metrics representing the impacts of particular attack stages based on CVSS scores [CYS+18, HSKG20] or service utility values [AJPS11, AJ17]; b) a performance model, such as a queueing network or a Petri net, used to describe the performance of a system/process. Note that, while many BCM and operations research approaches to business resilience discussed in Section 2.1.2.1 above consider process models, they have not provided generally applicable quantitative models for the kind of performance

estimation we are concerned with here.

It is worth noting that the performance-based view of resilience has similar modelling requirements for representing system performance, and the impact of changes on it, as the literature on *performability* [Mey80, Mey92], which focused on joining performance and reliability evaluations in settings where faults cause degradation in system performance. Indeed, Meyer [Mey13] proposed conducting performability-based system resilience evaluations. However, when focusing on cyber attacks, standard performability modelling as used for faults is not enough: While faults are random and independent from others (save for cascading impacts), adversarial attacks can follow a strategy to target multiple components to yield more devastating impacts. Thus, the modelling of attack progression is key, even if one wanted to model impacts by evaluating performability rather than performance. Future research could investigate merging aspects of performability modelling into our methodology.

The rest of this section explains some approaches on performance modelling relevant to impact assessment, and to what extent these have been used in conjunction with cyber attack models.

**Performance modelling:**

*Queueing networks:* Queueing networks (QNs) are a modelling formalism for evaluating the performance of systems that contain elements involving queues. That is, situations where a station processes tasks, and where unprocessed tasks form a queue before being processed in turn. Examples of such queueing include a server that processes requests, a call handler answering calls, etc. A QN contains several such stations, for example a factory production line can be represented as a QN with different stages of production represented as queueing stations, or a multi-tier software application where the output of one process is the input to another. A thorough explanation of QNs and their uses can be obtained from e.g. [BGDMT06].

QNs can used to model the performance of processing systems, and have been used for example for reliability and system performance [BGDMT06], resource provisioning [HWIL09], capacity planning [KL12], and resource management [GGQ+14], [HGG+14]. As a rigorous and commonly used formalism, it provides a useful performance modelling formalism for use in a sample implementation of our methodology. In Chapter 8 we use QN modelling as part of an approach for planning redundancy

(with diversity) to mitigate the effects of cyber attacks.

*Petri nets:* Petri nets provide expressive formalisms for modelling distributed systems. Several variants enable quantitative performance modelling, since the advent of stochastic Petri nets (SPNs) and generalised stochastic Petri nets (GSPNs) [MBC+95, BK02]. They can be used to model varied situations thanks to the expressiveness of the formalism. A drawback in their use is that detailed models are complex to build, and the models are often used for high-level descriptions of processes. Petri nets have been used for attack impact evaluation in e.g. [KCBCD10, LLC+18].

**Combining attack modelling with performance evaluation:** For the purposes of our methodology, we must be able to both express the progression of cyber attacks, and evaluate system performance. In previous literature, attack modelling has rarely been combined with detailed performance evaluation. For example, while AGs have been used for attack impact evaluation by [AJPS11, AJ17, SML19, SSL17, CYS+18], their impact evaluations have not used detailed performance models, but simpler models of component dependencies. Similarly, while the Möbius tool [DCC+02] supports both performance evaluation with Stochastic Activity Networks (SANs, a variant of stochastic Petri nets) and attack modelling with ADVISE [LFK+11], no published works that use Möbius combine these capabilities of the tool to investigate attack impacts on system performance.

Some studies have examined the impact of DDoS attacks using QN models. For example, Shan *et al.* [SWP17] investigate the impact of DDoS attacks on web applications using a QN model, and [YTGW14, LJZY20] examined how cloud platforms can maintain availability in the face of DDoS attacks. However, DDoS attacks represent a simpler special case of an attack where attacker actions do not have to occur within the system. By comparison, as part of our methodology, we want to express attacks where multiple stages take place inside the system, exploiting its vulnerabilities, and hence require a model of how the attack can propagate within the system.

### 2.2.4 Business cyber security investment

In considering the impact and cost-effectiveness of improvements intended to increase cyber resilience, our work shares some similarities with the literature on cyber security investment. The topic

of investment into cyber security has been studied from many angles since the work by [GL02], often using economic models. The key difference between our approach, assessing the cost-effectiveness of investment choices, and economic ones such as [GL02, GLL03, Lam16, CPG18] is that we aim to provide insights into investment via a model that attempts to capture more detailed responses to detailed attacks and recovery dynamics, instead of abstracting to the basic economic decisions over timing and overall amounts to invest. In this way, we hope to gain additional insights into how the specifics of the system design can affect different aspects of cyber security investment decisions.

Kheir et al. [KCBCD10] proposed an approach to evaluating the cost of responses to attacks, called return-on-response-investment (RORI), based on the return-on-investment (ROI) approach to evaluating the effectiveness of investments. Their approach is to estimate the costs (and benefits) involved with a given response using a coloured Petri net (CPN) model representing privilege dependencies between services in a system, and calculating the extent of disruption that different attack responses cause onto normal users of the system. The attacks considered are ones where an attacker infects privileges, or revokes privileges from legitimate users. Their model does not consider impacts over time, nor does it consider service performance beyond determining if groups of users are still available to use the system or not. Thus the approach does not fit the purposes of our impact estimation, and response cost estimation in the manner proposed is not valid in our case.

There have also been studies about how best to split investment across several competing uses. For example, [FPM+16] and [KMH+16] propose decision support models for optimising investment across various 'security control measures', classes of actions taken across a whole organisation. This problem differs from ours, as investments in their models aim at reducing the organisation's general risk from cyber attacks via generic enterprise-wide measures, such as deploying firewalls or enforcing a passwords policy, while our aim is a detailed look into defence measures to lower attack impacts.

Closer to our particular focus, [NKK+17] investigated how cyber security investment should split between security and recovery. The key difference between our approach and theirs is that we specify the status of each service in the system, and their interdependencies, and make choices over them, while their model has a two-state description of the whole system, abstracting away system details.

## 2.3 Discussion and chapter summary

This chapter provided background on cyber resilience, and on our impact assessment methodology, and discussed the literature related to the work in this thesis, in the ares of: cyber resilience; resilience; business continuity and operations research; attack impact assessment; and cyber security investment.

In this thesis we propose an approach to evaluating cyber resilience of systems within organisations, based on evaluating the performance and cost impacts from cyber attacks. This can aid with evaluating the costs and benefits from actions to improve resilience at the design stage, help with reactive choice of response actions, and evaluate the cyber resilience of a mission.

As discussed in the previous sections, while there are works and strands of literature that relate to our approach, they each have limitations that stop them fulfilling the aims of our proposed approach. The literature areas closest to our focus on cyber resilience in businesses are business cyber resilience and business continuity management. Existing works on business cyber resilience lack impact assessment to make them truly effective, as they leave uncertainty over the effectiveness of proposed policies. Works on business continuity management have similar issues, as discussed by Suresh et al. [SSB20] who point to problems of getting executives to provide sufficient resources and funding for BCM purposes, which again seems likely to stem from the lack of impact and cost evaluation.

Resilience works which are not specific to resilience to cyber attacks do not provide attack propagation modelling, and therefore their assessment may not apply when the events are cyber attacks rather than random events, as cyber attacks are not random and can impact multiple components simultaneously.

Existing works on attack impact assessment have not considered impacts and costs over a longer run [AJ17, SSL17, CYS+18, HSKG20], or use simplistic attack models or impact measures [HSK19].

Finally, existing works focusing on cyber security investment have considered the overall level of spending on security, with stylised models. Our work, by contrast, aims to assess the cost-effectiveness of specific actions to improve cyber resilience or respond to attacks, by evaluating their impact on the outcomes of attacks and the costs related to them.

With this background laid out, the next chapter will discuss the structure of our methodology in detail.

# Chapter 3

# Methodology for cyber resilience impact analysis

Our approach to analysing cyber resilience of systems, and the methodology introduced in this thesis, is focused on the impact of cyber attacks on the production of system output, and thus the resilience of this production. As a consequence, the core of our methodology is about quantifying the impacts of cyber attacks (and defensive actions) in terms of how well the system performs in producing its output, and enabling expressing the impacts in monetary units for evaluating the cost-efficiency of actions. Hence, impacts are measured in terms of the final output of a production process, and effects on intermediate processes are valued based on their impact on this final output. This allows expressing the impacts in monetary terms, based on the value of the output/business lost due to disruption (plus any costs of replacing or fixing faulty/compromised components).

As we mentioned in Chapter 1, there are several benefits to estimating attack impacts in terms of monetary costs, as they allow forming an estimate of the value of specific security investments. Most importantly, it plays a useful role in the impact evaluation itself, for two reasons: **1.** a monetary value provides a *common unit of measurement* for attack impacts, security investments, various maintenance costs etc., and **2.** monetary values can be *compared across time* in a straightforward way, even across longer periods of time using time discounting (using Net Present Value or other more general formulations of intertemporal utility [MWG95, Ch.20]). These are both of practical value for

cyber resilience analyses, as they enable meaningful quantitative comparisons between, and aggregation of, different quantities of interest. For example, a common unit of measure enables comparing system output impacts with costs of mitigation actions. Further, where a production process yields multiple outputs which may be asymmetrically affected, it allows for aggregating the various output impacts into one overall impact measure across all outputs. In addition, as attacks may be infrequent but cause extensive damage when they succeed, cyber resilience analysis requires comparisons to be made across time to evaluate the benefits of resilience improvements, especially if attack events are expected to be infrequent.

With relation to existing works, our aims for the proposed approach differ in the following key ways: **a)** Unlike many works on resilience, we aim to quantify impacts and costs of the resilience techniques to aid decision making, and thus go beyond simply listing techniques to consider (MITRE works [Gol10, BG11, BG13]) or objectives and indicators to match (maturity models like [CAC+10, MMPP19, CBL+20]); **b)** Our impact assessment is in terms of system performance over time, considering impacts from attack and defence actions, unlike attack impact assessment works focusing on the immediate impact [KC13, AJPS11, AJ17]; **c)** Our aim to model the cost to the system owner, based on the performance impacts over time, separates our approach from works on attack impact that use static estimates of impact and thus cannot reflect the full costs due to attacks, such as [CYS+18, SSL17, HSKG20].

## 3.1 Summary of the methodology

The central aim of our methodology is building a way to evaluate the impact of cyber attacks from the perspective of the resilience of system output performance, and doing so, enable analysing the effectiveness (and cost-effectiveness) of actions intended to improve cyber resilience and respond to attacks. These objectives form the reasoning behind the key components of our methodology, which is as follows. To be able to discuss system output performance, we need a *production model* for the system, which includes a *performance model*. To consider the impacts of cyber attacks, we must model *attack progression* in the system, which also includes the way in which attacker behaviour

and defensive actions, such as attack detection or countermeasures, affect how attacks can proceed. In addition, an *impact map* is used to define how different components of the production model are affected by the stages of this progression. For expressing resilience in terms of production performance over time, we must model *time aspects*, such as the speed of attacker and defence actions, and *recovery* of production after disruption. Finally, to enable evaluating the cost-effectiveness of resilience-improving actions, we require a *cost model*, setting the impacts in monetary units (e.g. the *value* of lost production, not just the amount) so they can be compared to the costs of taking the action, and the value of other actions.

Other approaches have not provided these desired features together and in full: while impact assessment works such as [AJ17, SSL17, CYS+18] link attack steps to system impact, they do not provide an analysis of the over-time impact (resilience aspects), or the cost effectiveness of the actions. Resilience works have not modelled the progression of general attacks, but focus on special cases such as DoS [YTGW14, CRC+15, SWP17, LJZY20], zero-days [TCNFW16, ZWJ+16], lateral movement [CCR+19], or attacks to specialised systems [WDCO15, CRDP17, JF18, YWR18]. Works on business resilience and business continuity have not provided detailed attack modelling, nor a generally applicable quantitative model of how the compromise of individual components affects the performance of the whole process, as discussed in Section 2.1.2.1.

Our approach is mainly intended for analysing systems where attacks can disrupt the output production process of the system. This is because the aim is to expose trade-offs between cyber resilience and costs, and use them to find cost-efficient choices for resilience improvement. Hence, the methodology is best suited for analyses of systems where the security attributes of availability and/or integrity are of concern. For confidentiality or safety impacts, cost trade-offs or performance focus are not appropriate, making our methodology less relevant for settings where these are of primary concern.

The way in which we structure the different modelling entities associated with our resilience-impact assessment methodology, and how our implementation examples relate to this, is shown in Figure 3.1. A colour coding is used to signify which classes (or implementation details) are used in each of our three studies where we have used the methodology. At the top level, the full *resilience impact* evaluation consists of a *production impact model*, and a *cost model* to assign a monetary value to the

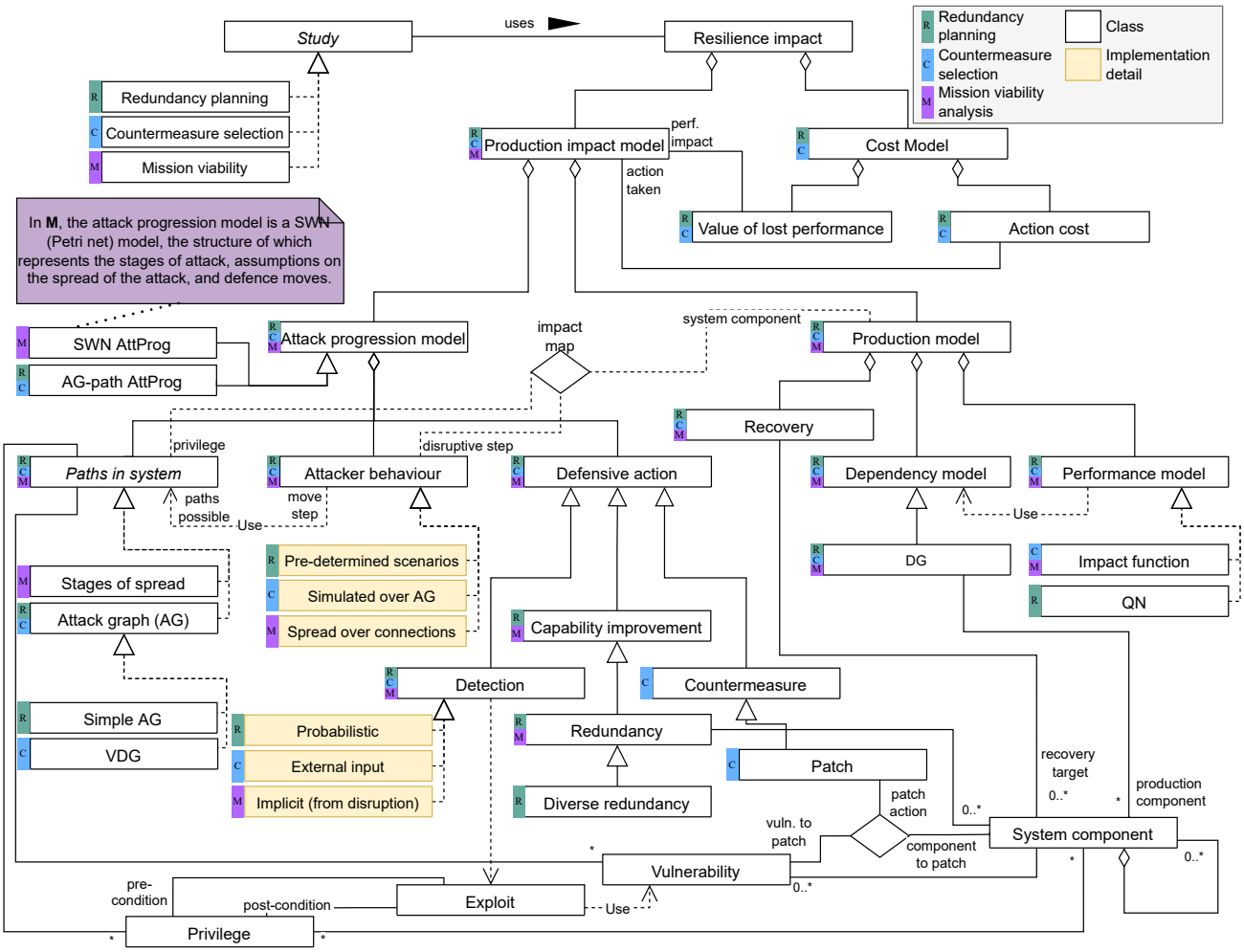



Figure 3.1: Methodology entities

impact and action costs. The performance impact evaluation requires an *attack progression model*, and a *production model*, with an *impact map* that relates disruptive attack steps to components of the production model that are impacted by the steps. The attack progression model contains a description of possible *attack paths in the system*, *attacker behaviour* and *defensive actions*, and the production model requires a *dependency model*, a *performance model* and a *model of recovery*. The figure also shows sub-classes and implementations of the main classes, e.g. 'Defensive action' has the sub-classes 'Detection', 'Capability improvement' and 'Countermeasure', and 'Performance model' has the implementations 'Impact function' and 'QN'. In addition, the figure also shows interactions and associations among the modelling components, and between them and system characteristics in terms of system components, privileges and vulnerabilities (and their exploits).

The methodology is intended to be extensible so that it can be adjusted when applied to specific settings. Indeed, as can be seen from the colour coding in Fig. 3.1, we have provided multiple im-

plementations for some of the sub-components of the methodology, which enable the methodology to be instantiatiated according to the differing requirements of our studies. The simplest differences occur when the use-case considered does not require a particular feature, e.g. the *defence actions* (apart from detection) in the studies differ, as one of our works focuses on reactive countermeasures while the others consider resilience improving capabilities (redundancy, diversity). There is more difference in the ways the *attacker behaviour* and *paths in the system* have been modelled, arising from a combination of differences in the system considered, the attack modelled, and the analysis conducted in the use case. While attack paths, behaviour and attack detection are consistent features, their implementations differ. In the resilience analysis and CM selection works these model components are largely similar, but have some differences arising from the use cases, e.g. the modelling of reactive countermeasures or capability improvements are due to the focus in the use case. However, the system and attack type considered in the mission viability analysis work introduce larger differences, as the analysis required for the use case is more convenient to do using an SWN-based attack progression model. The later sections, and chapters that follow, go into more detail on the reasons for the implementation differences.

Due to its ability to evaluate the impacts of attacks, defensive actions, and changes to the system, both in terms of system output impact (provided by the production impact model) and its monetary value (using the cost model), our methodology can be used to aid decision-making on questions such as: 1. Investment into resilience-improving capabilities, such as redundancy, diversity and other 'architectural techniques' relating to the system design; 2. Formulating a response strategy to cyber-attacks, or choosing between reactive responses to attacks, based on either expected performance impacts or costs; 3. Determining if improvements would be required to related capabilities, such as detection capability or recovery and its speed, or checking if planned improvements are justified, by performing analysis on their effect on impact on output performance and costs; 4. Deciding an appropriate service-level agreement (SLA) to offer to customers, by assessing the impact of different SLAs on the expected amount of penalties to be paid due to attack scenarios; 5. Assessing what level of cyber-resilience is provided by a given system setup, for a set of attack scenarios.

To apply the methodology, the key modelling components (attack progression, production model, impact map, and cost model) must be elaborated, and set-up to solve for the measures of interest. As part

of this, some of the component implementations, and the approach to solving for the different attack outcomes, may have to be adjusted based on the specific system and the question to be answered. We have provided a set of sample instantiations that are varied in terms of the system, attack type, and the use case, and we chose to change aspects of the modelling based on these differences. In the next sections we provide an overview of the works, while the changes considered, and the reasons for them, are discussed in the chapters that follow.

### 3.1.1 Sample instantiations of the methodology to applications

Here we summarise the different implementations of the methodology that we used in our studies of redundancy planning (Chapter 8, published in [SCML22]), countermeasure selection (Chapter 9, [SML19]), and mission viability analysis (Chapter 10, [SPL21]).

These studies and the related implementations were chosen based on what we identified as the key problem areas where our methodology could contribute to as an analysis approach, and also as these studies provide different challenges and pose opportunities for trying different modelling approaches to some of the components of the methodology.

#### 3.1.1.1 Redundancy planning in [SCML22]

In Chapter 8 we show how this methodology can be applied to estimating the optimal level of redundancy for production components in a system whose output performance can be affected by cyber attacks. We chose redundancy planning as a fruitful setting to demonstrate how the methodology can be applied to evaluate changes in system design in terms of their impact on cyber resilience and attack impacts. Redundancy and diversity are two of the cyber resilience techniques identified by [Gol10, BG11], thus the focus fits well within the cyber resilience literature. Further, this study enabled us to link the problem of cyber resilience into performance analysis using a modelling technique, queueing network, that is well established in reliability and performability research.

Our approach to evaluating the effectiveness of redundancy designs, using our impact assessment methodology, is summarised in Fig. 3.2. The effect of a *redundancy allocation*, which describes the

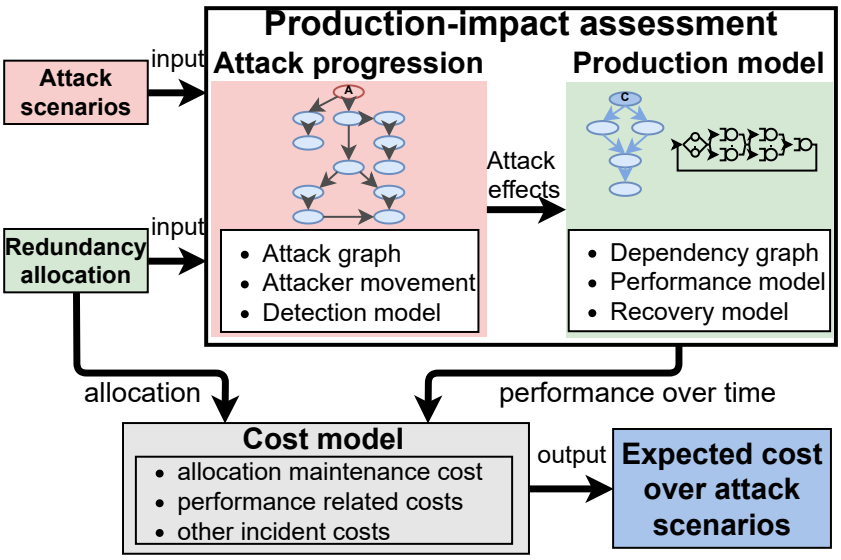



Figure 3.2: Summary of our approach to assessing the effectiveness of redundancy allocations

level of redundancy (and whether it is done with diversity) used at different components, is evaluated under a set of attack scenarios. This evaluation involves first our production-impact assessment, yielding estimates of production performance over time in the various possible attack outcomes, after which our cost model expresses these in terms of monetary units and calculates the expected cost over the set of attack scenarios for the particular redundancy allocation.

We now briefly outline how the key modelling components are represented in the case study used in Chapter 8. We consider a system that is a multi-tier application where the output is a web-service, and the *production model* components are the servers processing requests to the application. We use a DG to represent the component dependencies of the production, and a queueing network (QN) model to evaluate *system performance*. The QN modelling enables us to estimate the detailed performance impacts from changes in the system component allocation and from damage due to attacks.

The *attack progression* is modelled using an AG, behaviour assumptions and attack detection. To analyse the expected cost of cyber attacks to the system, an estimate of the relative likelihood of different attack outcomes is required. We thus model the detection of individual attack steps, which split an attack into different outcomes according to the number of steps that succeed without being detected. We assume that once detected, the attack can be contained (stopped from spreading further), but can still disrupt the production system using the privileges obtained by that point. The *links between the attack and production models* (which describe the *impact map*) occur when an obtainable

privilege can be used to disrupt a production component. In Fig. 3.2 these are represented by the arrow labelled “attack effects”.

*Recovery* in the case study represents both recovering the services affected by an attack, and removing the privileges held by the attacker. The latter is important, as failing to remove the privileges would allow the attacker to repeat the disruption. Recovery determines the duration of the cumulative loss of production, which is a major part of the costs arising from cyber attacks. Finally, we define a *cost model* to quantify the losses due to the production disruption, as well as the direct costs (acquisition and maintenance) of the servers and redundancy. In the particular situation represented in the case study, the cost of disruption is based on penalties for failing to meet an SLA.

The work presented in Chapter 8 implements the redundancy analysis use case, with an algorithm to estimate the costs that occur during attacks when different redundancy allocations are applied, and relating these to maintenance costs incurred to consider the cost-effectiveness of the allocation. The approach is evaluated using a case study of a system, and sensitivity analyses applied to estimate the effect of various parameters and assumptions made.

#### 3.1.1.2 Countermeasure selection

The countermeasure selection approach in Chapter 9, published in [SML19], is based on evaluating the costs and benefits of different countermeasures (CMs) considered for application during an ongoing attack. The approach considers the cost of system disruption over the duration of the attack event until recovery, with disruption caused by both the attack and the defensive actions themselves. This application uses the impact assessment and cost model parts of the methodology, to evaluate impacts as part of the CM selection algorithm, with the attack progression implemented with simulated attacker and defender actions to compare the effect of different countermeasure choices.

The application is of interest for cyber resilience, as when an attack escapes preventative cyber security measures, reactive responses to attacks are an important consideration for ensuring that system production can continue and normal performance recovered in a timely manner. Our specific study was conducted to demonstrate how our resilience impact assessment methodology could be used to

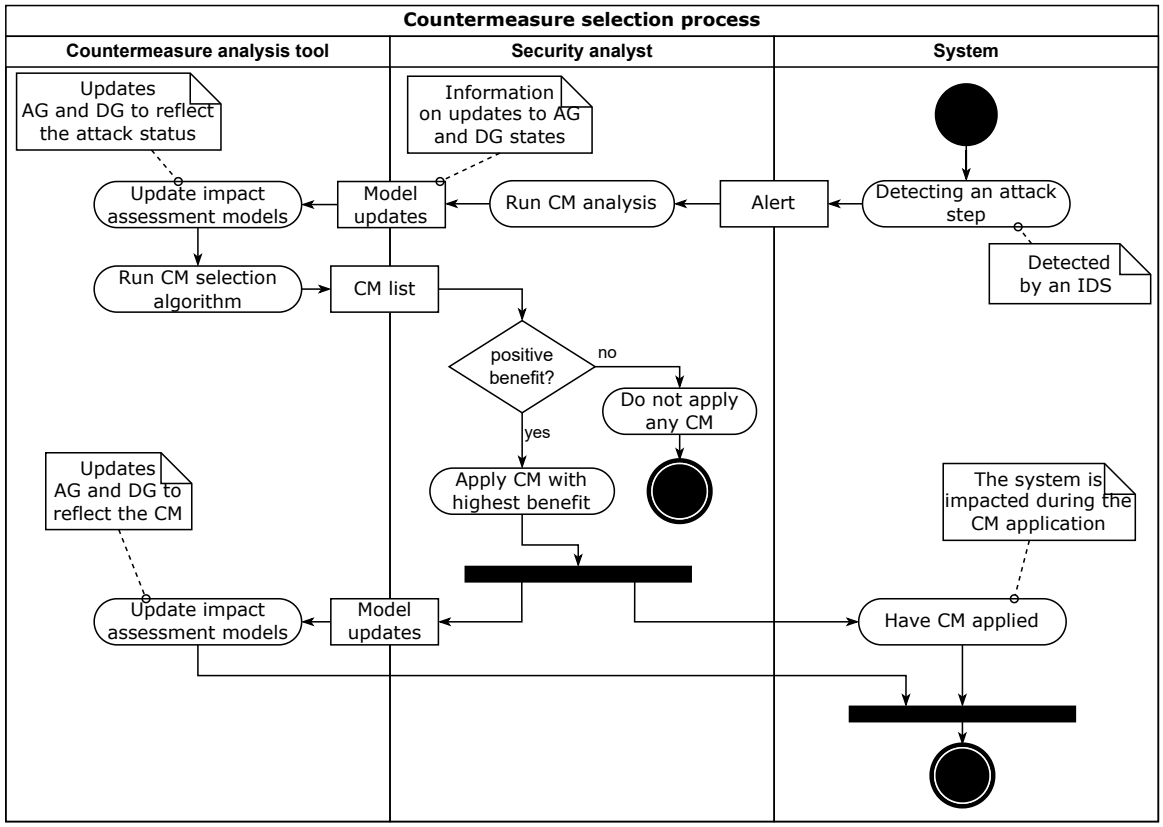



Figure 3.3: Activity diagram showing how our countermeasure selection would be deployed

aid reactive attack responses, and the example case of patching choice was chosen as it provides a clear example for how the countermeasure selection approach could be implemented. Further, the patching choice use-case enabled comparison to existing works, and thus allowed for indicating the benefit, for the cost-effectiveness of reactive responses, from considering performance over time to determine impacts from attack and defensive actions.

Fig. 3.3 shows how our countermeasure selection approach would be used to choose responses to attacks to a system. First, an attack detection by an IDS would be passed to a security analyst, who would initiate the countermeasure selection analysis. They would send model updates, based on the attack state, to the countermeasure analysis tool, where our CM selection algorithm would be run. A list of proposed CMs would be returned, and the highest benefit CM would be applied, conditional on the benefit being positive. This process is run each time an attack step is detected, and is repeated for as long as further attack steps occur, that is, until the attack is made to stop.

Figure 3.4 gives a visual summary of how the CM selection algorithm itself operates to provide the list of CMs, using our impact assessment methodology. When an attack step has been detected, we

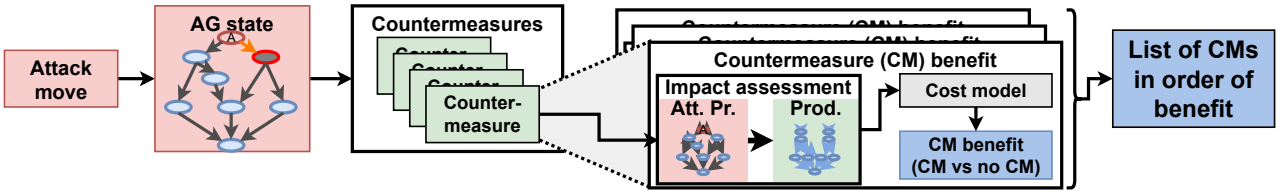



Figure 3.4: Countermeasure selection approach using impact analysis over time

use the system attack graph (AG) to determine the attack state in terms of the system, and create a list of possible countermeasures (CMs). For each of these, we use our attack impact evaluation model to estimate the CM's effect on the system production, calculating the benefit from applying the CM by comparing its effect (and cost) against what would be expected to happen in the absence of a countermeasure. The list of these are then returned, for the highest benefit one to be applied.

Chapter 9 provides an implementation of the CM selection algorithm, and evaluates its performance in choosing patching actions against alternative ways in which such actions could be chosen, in a case study sample system and with randomly generated synthetic attack and dependency graphs. The evaluation is done with the aid of a simulation of the attack process, including detection and the attack steps taken. The results suggest that including considerations for action costs and the long term can improve the cost-effectiveness of the choice of actions during an attack.

#### 3.1.1.3 Mission viability analysis in [SPL21]

Chapter 10 lays out our work applying our methodology to mission viability analysis. In it, we build a model for evaluating the viability of a fire survey mission, conducted using a group of unoccupied aerial vehicles (UAVs) with various roles, when facing a cyber-attack that can spread among the UAVs. We chose this as one of our applications, as our approach to evaluating performance impact from attacks is a good fit for analysing attack impacts on missions. Further, this particular type of mission, with a dynamic network between UAVs and a short mission length, provides a useful contrast to the other case-studies discussed above, as it offers different challenges in terms of implementing attack progression modelling, production performance and the impact map. Doing so, it helps highlight some of the general considerations required to model these aspects of the methodology.

The approach we take to evaluating the viability of a mission in this work is summarised in Fig. 3.5,

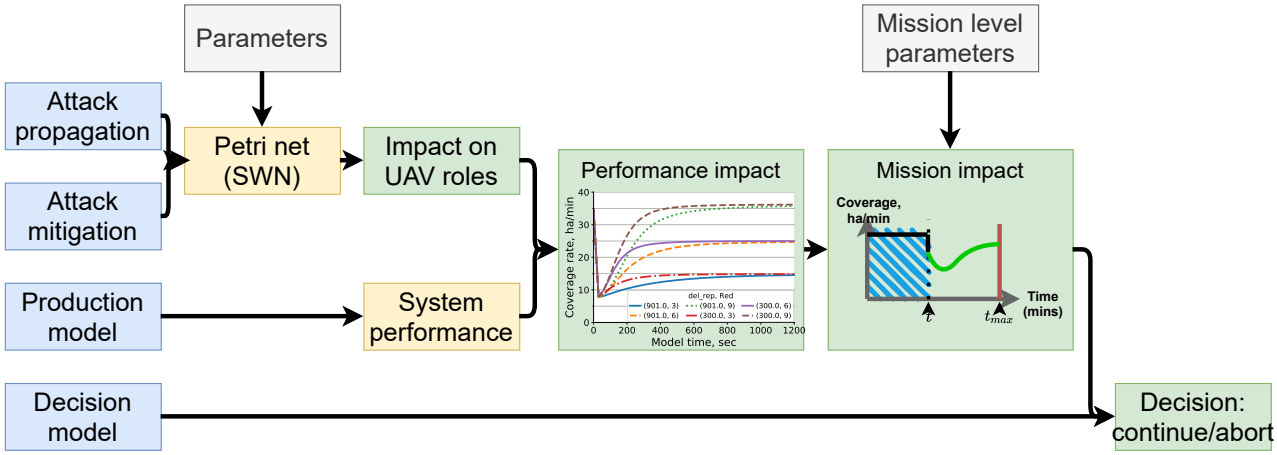



Figure 3.5: Mission viability analysis of a UAV mission

which represents the analysis process and its main components. In this work, the attack progression model and the impact map are implemented as a stochastic well-formed net (SWN), a type of Petri net. The SWN is built to reflect the structure of the inter-UAV communication network, rules about attack propagation and mitigation, and recovery mechanisms. To find the impact of an attack on individual UAV roles during the mission, the SWN is initialised with parameters that describe the speed of propagation, mitigation and recovery, and the capacity for recovering the mission. The SWN model allows for analysis of attack impacts over time using transient stochastic analysis techniques of continuous time Markov chains (CTMCs). This enables evaluating the impacts of the different possible interleavings of timed actions using a well known closed form solution approach, instead of forming a simulation. This aspect is important to capture in this case study due to the attack type (able to spread quickly across the UAVs), as this makes the impact on the mission depend critically on the relative speeds of the attack and defensive actions.

The production model leads to a description of system performance as a function of UAV roles. When combined with the output of the SWN model, this lets us estimate the impact of an attack on mission performance. Combined with parameters describing the timing of the attack during the mission and maximum mission duration, we can form an expected impact on the progress toward the mission goal, and if this goal is still achievable or if the mission should be aborted.

In this particular application we did not model monetary-valued costs, but merely considered impacts on the performance of the mission. A model of financial costs was not desired as the setting concerns

firefighting, where all available resources were considered to be provided, and safety concerns override any desire for cost-based decision making. However, in a different setting where the mission is unrelated to safety concerns, a cost model could be applied to consider the cost-effective allocation of resources etc. While we have not investigated such an application, it is a feasible subject for future research.

## 3.2 Chapter summary and discussion

This chapter set out the structure of our methodology for cyber resilience impact analysis, which is based on evaluating how disruptive cyber-attacks can impact the output production performance of a system. The various modelling components were described briefly, and various use cases for the model were listed. We then provided outlines of our works that provide specific implementations of the methodology in three different use cases.

The chapters that follow will go into more detail on the modelling of the different components of the methodology, including the different implementations we provided for some of them due to specific requirements of the case studies we have considered.

# Chapter 4

# Attack progression modelling

In the methodology described in this thesis, attack progression refers to the way in which an attack advances within a system, including steps to move within the system by exploiting vulnerabilities that exist in it, and actions taken to launch the impact stage of the attack (e.g. disrupt parts of production processes).

As we aim to use our methodology to aid with evaluating and choosing resilience improving actions, we are primarily concerned with attack actions that cause system impacts, or ones that could be detected and/or affected by defence actions. Thus, as the defence has no control over attack activity that occurs outside of the system, and as they cause no system impacts by themselves, from our modelling perspective all such activity can be abstracted into one action, the entry to the system. In the context of the Lockheed-Martin cyber kill chain (CKC) [HCA11], our methodology is concerned with phases three to seven (Delivery, Exploitation, Installation, Command and Control, Actions on Objectives), which relate to activities occurring within the system. By contrast, activities relating to the first two phases (Reconnaissance and Weaponization) can occur fully outside of the system and would thus not be part of our attack progression modelling. Similarly, the equivalent MITRE ATT&CK® [SAM+18] tactics of Reconnaissance and Resource Development are out of our scope.

Our modelling of attack progression consists of three parts: 1. The *paths an attacker can take* within the system, given the system setup and vulnerabilities, expressed using *attack graphs*; 2. The *behaviour of an attacker* during an attack, in terms of choice of actions given capabilities, the paths

available, and in response to any defensive actions; 3. *Defensive actions* that can mitigate or respond to an attack, which thus affect the actions that are available to an attacker.

## 4.1 Attack paths

We model the paths attackers can take within the system using **attack graphs** (AGs). In essence, an AG for a given system represents the steps that an attacker could take within the system, stringing together vulnerability exploits to proceed further in the network. Various types of attack graphs have been proposed, as discussed in Section 2.2.2.1. In this section we focus on the requirements our methodology places on AGs, and give the AG definitions used in the applications in Chapters 8-10.

At its simplest, the attack graph used in our modelling methodology can be a simple directed graph with system privileges as nodes, and edges as vulnerabilities whose exploits enable an attacker holding one privilege to obtain others. Such a graph can be defined as:

**Definition 4.1** ***(Attack Graph)*** *Given a set of system privileges $P$ and a set of vulnerabilities $V$, an attack graph G is the directed graph $G = (P, V)$, where $P$ is the set of nodes and $V \subseteq P \times P$ is the set of edges.*

A more complex AG could be used as part of the methodology, as long as it can work together with the other components of the methodology, but the simple version is sufficient. This simpler graph conveys the requirement of our methodology, which is to represent the attack paths available by obtaining privileges using vulnerability exploits, and emphasises the point that the methodology is not dependent on a specific AG definition. We have used this AG definition in our work on redundancy planning, in Chapter 8.

In our work on countermeasure selection (Chapter 9), we applied a slightly different AG definition, a compact version of a logical attack graph proposed by [AJ17] called a vulnerability dependency graph. This was used in order to enable direct comparisons of our methodology to the impact assessment approach by [AJ17], to show the benefits of our methodology over a more static impact assessment for use in countermeasure selection. This definition is as follows:

**Definition 4.2** ***(Vulnerability Dependency Graph, [AJ17])*** *Given a set of vulnerability exploits $V$, a set of security conditions $C$, a require relation $R_r \subseteq C \times V$, and an imply relation $R_i \subseteq V \times C$, a vulnerability dependency graph G is the directed graph $G = (V, R)$, where $R = \{(v_i, v_j) \in V \times V | \exists c \in C, (v_i, c) \in R_i \wedge (c, v_j) \in R_r\}$ is the edge set.*

The vertices consist of nodes representing vulnerability exploits such as "remote exploit of vulnerability V on host A", while the edges implicitly contain security conditions such as "user access to host A". A require relation $R_r$ between a security condition $c$ and a vulnerability exploit $v_i$ means that $c$ must be satisfied for $v_i$ to be exploited, and an imply relation $R_i$ between $v_j$ and $c$ means that exploit of $v_j$ leads to condition $c$ being satisfied [WNJ06]. The edges in the set $R$ link pairs of vulnerabilities that are connected via a security condition by what [AJPS11] call a "prepare-for" relation. A prepare-for relation exists between $v_i$ and $v_j$ if $v_i$ has an edge to security condition $c$ representing an imply relation (exploit of $v_i$ implies the attainment of security condition $c$), and $c$ has an edge to $v_j$ representing a require relation (the exploit of $v_j$ requires the condition $c$ to have been obtained by the attacker). This AG representation leaves the security conditions implicit, as the edges are defined as going through a condition but these are not shown, with the end result resembling a dependency graph of the vulnerabilities.

The graphs in definitions 4.1 and 4.2 have different concepts represented by the nodes: in 4.1 nodes represent privileges, with vulnerability exploits as the edges connecting them, while in 4.2 the vulnerabilities are the nodes, and edges show how these vulnerabilities can be stringed together with the security conditions the vulnerabilities enable. These are both approaches to reduce the amount of information carried around in the AG, and either type could be generated from a traditional Logical Attack Graph, as shown in Fig. 2.4 of the background section. Either approach can be used with our methodology, as long as they can represent where the disruptive impacts of attacks occur in the system that is modelled. That is, using 4.1 is preferable when disruptive effects occur due to actions that are enabled by specific privileges, while 4.2 is suited for when a disruptive effect occurs as a direct result of a vulnerability being exploited (e.g. if the exploit itself causes a DoS to a component). However, as part of our impact assessment, definition 4.1 can represent either case if, in the impact modelling, the information is added on whether the privilege is used for disruption in a discretionary

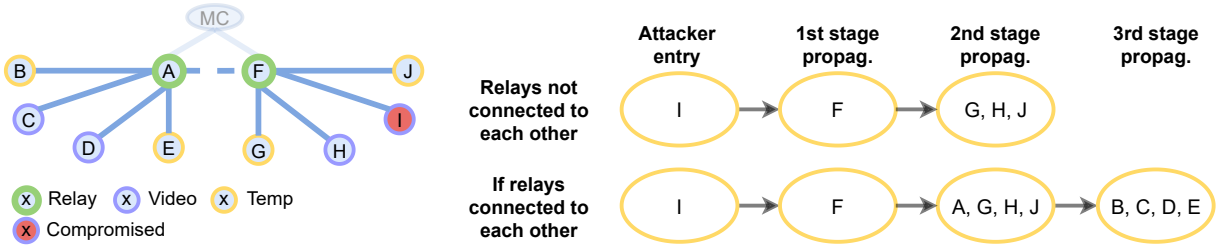
way, or whether a disruption occurs immediately when the privilege is obtained.

### 4.1.1 Exploits of vulnerabilities in identical system components

In our modelling, the steps on the attack paths do not have to represent the use of multiple different exploits, but can be exploits of the same vulnerabilities in identical systems or components. The key is to distinguish between steps in terms of the impact they have on the system, and on whether there is a need to distinguish between them from the defense perspective. For example, if an attacker requires action A to enable action B, but A cannot be observed or detected by the defense and does not have a separate system impact, then the model does not have to represent A and B separately.

**Simple attack paths:** An example of a situation where attack paths are simple is provided in our work on mission viability analysis, in Chapter 10. In that work, we provide an implementation of our methodology for missions involving multiple unoccupied aerial vehicles (UAVs), where the attacks of interest are targeting the UAVs directly (not via attacks on e.g. the mission control). In that setting, assumptions over the mission connectivity and exploit actions available for the attacker lead to attack paths in which the attack propagates across the vehicles in the network in a straightforward sequence of steps, with differences arising based on which node is attacked first. Fig. 4.1 shows a sample of the potential paths in this setting. The graph in Fig. 4.1a depicts the connectivity between the UAVs in the mission, in a situation where the video node $I$ has been compromised by an attack. The nodes represent UAVs in three different roles (relay, temperature measurement and video). Fig. 4.1b shows two alternatives for attack progression given the connectivity: 1. Relays ($A$ and $F$) are not connected to each other, meaning that only half of the vehicles can become compromised by an attack on $I$; 2. The relays are connected to each other, which makes it possible for the compromise of $F$ to also compromise $A$, and the vehicles connected to $A$.

**Attack graphs with multiplicity notation:** In our work on redundancy planning, discussed in detail in Chapter 8, the attack graph contains both stepping-stone exploits of various vulnerabilities to proceed within the system, and repeated exploits in identical system components. As the work is about determining appropriate levels of redundancy, it requires considering cases with different numbers of




(a) Mission connectivity (b) Attack paths

Figure 4.1: Connectivity for a UAV mission, and potential high-level attack paths that ensue

some of the components. Therefore, we took the approach of representing the attack paths in terms of a "*prototype AG*" that shows the general paths and possible multiplicities, while the analysis uses *expanded instances of the AG* with the component multiplicities instantiated to specific numbers. While an AG could be set up to represent these identical components as a group, in one node instead of separate nodes, the appropriate granularity to use depends on how the attack impacts the components, individually or as a group.[1] In our case we want to be able to represent various outcomes of an attack where different numbers of servers in a cluster are compromised. To better explain this further using an example, shown in Fig. 4.3 below, we shall give some background detail on the case study system.

Our case study in Chapter 8 is based on an e-commerce setting using J2EE services. We chose it as its architecture is representative of widespread service-based systems and its performance model is well understood. The system and workload characteristics we use are based on the queueing model in [KB03] on a multi-tier application benchmark by SPEC [Sta02]. While the performance modelling details of the system largely remain as in [KB03], our case study adds assumptions about the specific network topology and vulnerabilities present in the system, in order to model attacks. The assumed topology is shown in Fig. 4.2. The application servers provide clients access to the services over the internet, and the database resides in a separate subnetwork. The rest of the network is split into two subnetworks containing administrators and users, respectively.

In addition to this topology, we made assumptions about vulnerabilities that exist in the system, enabling the attacker to obtain privileges. These vulnerabilities and privileges form the attack graph shown in Fig. 4.3, below. The specific privileges (represented by the AG nodes), and the vulnerabilities (the edges), are listed in Table 4.1. These represent possible vulnerabilities that could have

[1] This also depends on whether the defence can stop the attack from spreading to all of them, and effectively impact them in a group even if they require individual exploits.

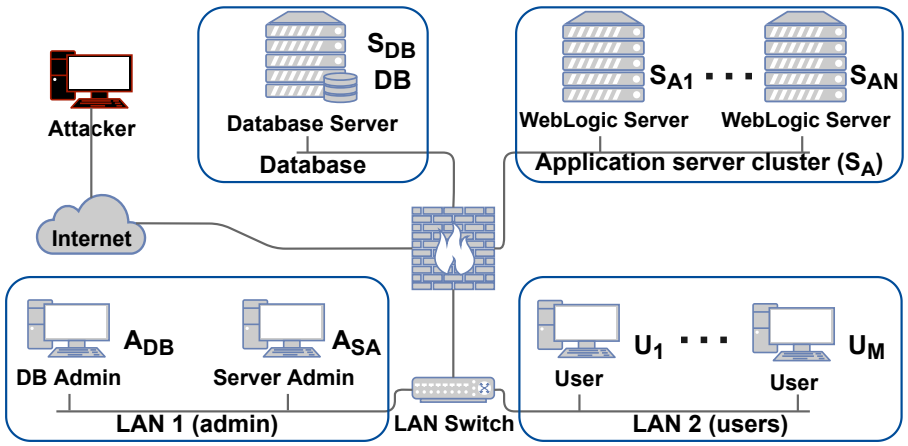



Figure 4.2: Network topology for the J2EE case study

Table 4.1: Privileges and vulnerabilities in the case study

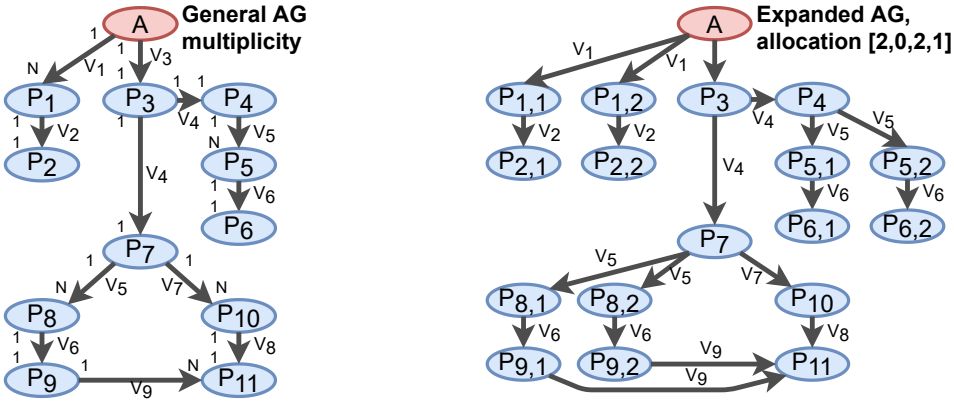
| Privileges | | | | Vulnerabilities | | | |
|---|---|---|---|---|---|---|---|
| $P_1$:U, WebLogic 7 | $P_5$:U, SuSE 8 (AS) | $P_9$: A, SuSE 8 (DS) | $P_{1A}$: U, alt. server | $V_1$:CVE-2003-0151 | $V_5$:CVE-2004-1175 | $V_9$:CVE-2004-0638 | $V_{1A}$: obtain user priv. |
| $P_2$:A, WebLogic 7 | $P_6$:A, SuSE 8 (AS) | $P_{10}$: U, Oracle 9i DB | $P_{2A}$: A, alt. server | $V_2$:CVE-2003-0640 | $V_6$:CVE-2004-0495 | | $V_{2A}$: priv. escal. (U-A) |
| $P_3$:U, LAN 2 | $P_7$:A, DB Admin | $P_{11}$: A, Oracle 9i DB | $P_{5A}$: U, alt. OS (AS) | $V_3$:Phishing attack | $V_7$:CVE-2002-0965 | | $V_{5A}$: obtain user priv. |
| $P_4$:A, Server Admin | $P_8$:U, SuSE 8 (DS) | | $P_{6A}$: A, alt. OS (AS) | $V_4$:CVE-2006-5051 | $V_8$:CVE-2004-1707 | | $V_{6A}$: priv. escal. (U-A) |

Abbreviations: U – User; A – Admin; AS – application server; DS – Database Server; DB – Database; alt. server – alternative server; priv. – privilege; escal. – escalation

occurred in the case study system, relating to the appropriate component versions and time period for the hypothetical system. In practice there could be various vulnerabilities that can achieve the same attack outcomes, including attack techniques that do not require application vulnerabilities.

The multiplicity of servers (and adding diversity) affects the structure of the AG for the system, as the compromise of different server copies can require obtaining separate instances of some privileges. Fig. 4.3 shows how we represent the general pattern (the 'prototype AG') and how this relates to the underlying AG which depends on the server multiplicities. *Each server allocation will generate a different underlying AG*, due to the server multiplicities. For example, the right-hand side panel of Fig. 4.3b shows the underlying AG corresponding to the server allocation [2,0,2,1], representing the case with two application servers, two database processing servers and one database.

We use *multiplicity notation* to simplify an AG based on its general structure, without showing the full complexity arising from duplicates of privileges in different server copies. As shown in Fig. 4.3a, the multiplicities are given at the ends of the edges, in the style of UML class diagrams, and are used to show whether a given source privilege enables a vulnerability exploit that can be used to obtain one instance of a destination privilege (1-to-1 relation), or multiple (1-to-N). For example, with the privilege $P_5$, multiple copies of privilege $P_8$ can be obtained (in different instances of the system




(a) *"Prototype AG"*: general structure and multiplicities

(b) Expanded AG for [2,0,2,1]

Figure 4.3: Prototype AG with multiplicity notation, and an expanded AG instance

component in question, in this case database processing servers) by exploiting the vulnerability $V_5$: in Fig. 4.3b there are two database processing servers, and thus two copies of $P_8$ ($P_{8,1}$ and $P_{8,2}$), and two exploits of $V_5$ are required to obtain both of these privileges.

## 4.2 Attacker behaviour

To evaluate the production performance over time, and how different system improvements or defensive actions affect this, the modelling of attack progression must consider the attacker behaviour, not only the possible paths available. The description of the attack behaviour needs to consider the attack steps on the possible paths, including the speed at which these steps are taken, but also how the attack would proceed if a defensive action stops the attack following the intended path. This section describes our modelling of attacker behaviour, starting with a description of the actions available to attackers, and then describing how we model the choices of actions in our application case studies.

### 4.2.1 Attacker actions

We model **attacks** as sequences of attack steps. We consider two types of attack steps: "move steps" and "disruptive steps". In a *move step* the attacker moves in the AG by exploiting vulnerabilities to obtain privileges. That is, it corresponds to a single move between nodes in the AG, with potential

to lead to further steps. Commonly, one such step would correspond to a single vulnerability exploit, but, depending on the granularity of information in the AG, it could also represent a sequence of several exploits that in combination lead to the next AG node[2].

Some privileges can enable an attacker to disrupt system components. After obtaining such a privilege, the attacker gains the ability to take a *disruptive step* to cause disruption to a component. In our impact modelling, explained in more detail in Sec. 6.1, the possible disruptive steps form links from AG elements to components of the production model, which we call an 'impact map'. Some vulnerability exploits can lead to disruption directly, e.g. causing a denial of service, and in such cases the disruptive step occurs coincidentally with the move step. This is the assumption used in the model in our countermeasure selection work discussed in Chapter 9, and in the UAV mission scenario in Chapter 10. In other cases, the exploits used for move steps do not themselves cause disruption. Then, the timing of launching the disruption is at the discretion of the attacker, as long as it continues to hold the required privileges. This is the assumption used in our redundancy planning model in Chapter 8, where the attack is assumed to first try to collect privileges, and then launch a disruption when it has reached the set of privileges it wants, or when it is detected. Modelling situations involving both disruptive and non-disruptive move exploits could be done; this would require tracking whether a step is disruptive or not, and adding priorities to the attacker behaviour in terms of the order in which it completes move steps, prioritising those that will not cause disruption and possible detection.

Attacks enter the network via particular "entry nodes" which are directly exploitable by the attacker – either externally from the Internet, or from a part of the network the attacker has direct access to. The potential next steps are then determined by what vulnerability exploits or disruptive actions the entry node enables, which is shown in the AG structure as edges from the entry node.

**Time-to-exploit:** The time needed to exploit a vulnerability, i.e. the time taken by the move steps of an attacker, is determined based on estimates of the time it takes to exploit the vulnerabilities required. In general, these estimates would be based on the capabilities, skills and system knowledge of the attacker. In our work, we assign parameter values to these based on approximate values derived

[2]Such an aggregation could occur, for example, if the exploits cannot be detected or stopped separately by the defence, and therefore do not need to be modelled separately. For example, in our mission viability analysis work in Chapter 10 (in [SPL21]), we consider the various exploits and techniques used when compromising a UAV to form a single attacker step, as the defence is not able to detect or stop these stages separately.

from the available literature.

Further, as we are concerned with attacks on system performance rather than objectives that require attackers to persist in the system and where slow movement might be an advantage, we assume that attackers take move steps as fast as their abilities allow, and there is no delay between attempting steps. For the speed of attack progression, this is the most aggressive (worst case) scenario. For cases where faster attack progression helps the attacker, our assumption captures the worst case impact. In practice, attacks might delay steps to minimise detection. However, the cases in which such delays can benefit the attacker are ones where the launch-time of the disruption is discretionary, and in these cases, the impact steps of the attack would be launched in one go (to avoid detection). In such cases, the speed of the move steps does not matter for the system impact observed for a given attack outcome, only the timing of when the disruption begins. Thus, our fast-movement assumption does not impact the results – although the speed could impact the likelihood of detection, these should be set separately based on realistic assumptions of detecting the various steps in practice.

In contrast to move steps, disruptive steps are assumed instant, so take place immediately after the attacker has obtained the privileges required for its goal, or if further move steps are not possible due to the actions of the defence. Further, it is assumed that when the disruption is launched, disruptive steps are taken simultaneously from all the privileges the attacker has obtained by that point.

The way we assign time-to-exploit to move steps in our work has varied based on the specific model used. In our redundancy planning work (Chapter 8), we use exploit time estimates based on findings used in [MBFB06, LB08, NWJS13], and distinguish between three different cases as follows: the first exploit of a given vulnerability takes 24h ($t_e = 24$); using the same exploit elsewhere in the network (reuse exploit) takes 6h ($t_{er} = 6$); once the attacker has obtained a privilege on a server, the time to replicate the exploit on other servers of the same type (repeat step) is 4 hours, $t_{ers}$.

In the countermeasure selection model (Chapter 9), the sample evaluation uses a generic value for exploit time (not directly set to practical estimates), and the time taken by response actions is set relative to this base unit used for exploit time. In a practical implementation in a specific setting, what time this base unit represents would be set based on estimates of exploit times.

In the mission viability work (Chapter 10), we set mean exploit time values based on estimates of plausible exploit times for UAV missions, and varied them among a set of values in the evaluation. The attack progression model is done using a Petri net (specifically a stochastic well-formed net, SWN). To enable closed-form solution of the model using continuous-time Markov chains (CTMC), the time required to exploit a given vulnerability is drawn from an exponential distribution with a specific mean, which we set based on the exploit time estimates.[3]

#### 4.2.1.1 Choice of actions

To determine how an attack would proceed in a network, and how it will be affected by defensive actions, the choice of actions by the attacker has to be modelled. In the model implementations we use in Chapters 8-10, our modelling of the attacker choices has the following general features: **1.** The attacks are assumed to aim at disrupting the production performance of the system, via impacting the availability of components. **2.** A given instance of an attack has a specific target, in terms of a privilege or a *set* of privileges, that the attack proceeds toward. Thus, for the purpose of the evaluations, the attacker is assumed to know enough of the system to be aware of what they are working toward. This can be seen as a worst case assumption on which the defence is based, i.e. while it may not reflect the knowledge of each attacker, it is safer to build the defence against it than other knowledge assumptions. The other reason for this assumption is that it limits the number of path instances to consider, as attacks with inefficient progress or impact are excluded. **3.** The attacks continue their progress with *move steps* until they reach their specific target, or if they are stopped from progressing by the defence. Defence actions can lead to attack outcomes with partial impact, that is, if an attacker cannot progress further with move step, it may still be able to take *impact steps* with the privileges they have obtained, and these can cause system disruption.

These features limit the attacker modelling in our sample implementations to essentially worst case attack types on the targets given, excluding paths where attackers are slowed down by lack of knowledge or act inefficiently. Various more complex approaches to modelling attacker choices have been proposed, some of which assume planning with limited information as discussed in [SBH12, MAA+18],

[3] Technically, what is drawn from the exponential distribution is a transition's firing *rate*, the inverse of which gives the time until firing. The latter, in the case of transitions that represent exploits, is the time-to-exploit.

and choices based on estimates of attacker utility [LFK+11]. We have yet to extend our behaviour modelling to such approaches, as the available tools related to them do not currently fit with our modelling for the overall methodology: attack emulation with e.g. CALDERA [MIT19, MAA+18] functions in a system instance, not a model, and thus could not be used for evaluating planned changes or a new system architecture, unless a full system emulation was conducted; the Möbius tool where ADVISE [LFK+11] is implemented is closed source and out of maintenance.

Beyond these general points, the specifics of how choices are modelled in our sample implementations (Chapters 8,9,10) differ somewhat between the applications due to the analysis requirements and modelling used. As the actions available to the attacker also depend on what the defence does, we shall discuss these different approaches in some more detail in Section 4.4 below, after introducing the defensive actions considered.

## 4.3 Defensive actions

Actions by the defence impact the actions that are available for the attacker, in terms of both move steps within the system, and disruptive actions. Given this, the modelling of attack progression should account for defensive actions that could be used to mitigate and respond to attacks.

However, it is not only the interplay between attack and defence that causes the need for defence modelling as part of our approach. This need is heightened by our aim to evaluate the output performance due to an attack event, as defensive actions can both mitigate the overall cumulative performance impact but also potentially reduce performance temporarily. There is also a need to consider future periods where the response includes actions whose short-term impacts differ from their lasting effects, such as with redundancy planning and patching. These aspects relate to the overall impact assessment, considering both attack progression and production, which is discussed in later sections. However, they must be addressed when modelling the actions.

We divide the actions by the defence into three categories, each of which must be addressed in the modelling: **1.** Attack *detection*; **2.** Attack *countermeasures* (including capability changes such as redundancy); and **3.** *Recovery*.

### 4.3.1 Attack detection

Given our use of AGs as part of our attack modelling, we model the detection of attacks as the detection of an attack step along the AG. We do not go into detail on how the detection works (e.g. what activity is tracked, what IDS tool is used, etc.), but assume that there is a way to detect attacker activity relating to the move steps along the AG, and that we can assign a probability to detecting such activity.

The specific use-case to be analysed affects the requirements for detection modelling, reflected in the differences in its implementation in our works, in Chapters 8-10. The most straightforward case is when detection is external to the model, as in our countermeasure selection work in Ch. 9, where it is assumed that the countermeasure selection process starts once an IDS has made a detection and sent an alert. As a consequence, in that work we do not model detection explicitly, although we do consider how far the attack has progressed before the detection is made. Another simple case occurs when attacks are assumed to cause noticeable disruption at each move step within the system, which is the assumption made with regard to the attacks to the group of UAVs considered in Ch. 10. In this case, detection happens automatically at each move step when the disruption occurs. However, where neither of these two cases apply, as in our redundancy planning work in Ch. 8, then detection must be modelled explicitly, in order to consider the different attack outcomes and their probabilities.

There are four key assumptions we use in the detection modelling in the redundancy planning application in Chapter 8: First, the probability that a given move step is detected is $p_d$, and this probability is equal for every move step. This is a simplifying assumption which could be relaxed, with context specific probability values used, at the cost of added computation required for determining the overall probability of the various detection outcomes. Second, we assume that detection can only occur in an AG node where the attacker is currently acting, i.e. the detection does not occur with a delay in a location that was previously compromised. This represents intrusion detection that acts on the latest activity and does not have the capacity to investigate historical activity, which would only get investigated as part of a more detailed investigation after an initial detection. This assumption simplifies the modelling in terms of how the overall probability of detecting a multi-step attack is determined. While it also nominally limits the number of possible situations to consider in terms of where the

attack has reached with relation to a detection, it does not affect the number of different outcomes in terms of the attack impact (an attack that has successfully completed a given set of steps has the same impact regardless of what part of the activity caused the detection). Third, when the attacker takes a disruptive action, this is assumed to lead to a detection automatically. Fourth, once an attack is detected, we assume that the defence can contain it, i.e. will stop it from moving further along the AG. However, the attack can still cause disruption using the privileges already acquired.

### 4.3.2 Countermeasures

Attack countermeasures (CMs) are actions aimed at reducing the impact of an attack. For the purposes of our work, we classify these actions into two categories: *capability changes* affecting the network's security capabilities which can be made before an attack but not during one (redundancy additions, back-ups), and *dynamic countermeasures* which can be taken at any time.

As discussed in Section 2.1.1, there is a great range of techniques proposed for reducing attack impacts, such as the resilience practices listed in [BG11]. For our sample implementations, we have chosen to implement techniques that are generally applicable and that can be modelled in a generic way. Note, however, that for architectural approaches in [BG11] such as segmentation, evaluating the impacts of such designs could be done similarly to the approach we use for redundancy and diversity comparisons in Chapter 8: comparing the attack impacts and resilience characteristics of different designs against each other.

Our sample implementations, the countermeasures used are as follows. In our redundancy planning work in Chapter 8 we considered *capability changes* in terms of **redundancy** additions with and without **diversity**, while in the countermeasure selection implementation in Chapter 9 we focused on the selection of *dynamic* countermeasures among different choices, focusing on the case of **patching** of vulnerabilities. Our mission viability modelling implementation in Chapter 10 contains a flavour of both capability changes and dynamic CMs, as the model considers the impact of redundancy on mission viability, and also includes a generic **containment** action that can stop the spread of an attack, to enable analysing how the speed of containment affects redundancy needs.

Note that we do not consider **recovery** in itself a countermeasure. When discussing recovery, we distinguish between recovering components/services to functioning order, which is in itself not considered an attack countermeasure, and actions that stop an attack from progressing. The latter actions are considered CMs, but do not necessarily occur as part of the process of recovery.

#### 4.3.2.1 Redundancy and diversity

Redundancy and diversity are techniques to improve the resilience of a system to events that can disrupt the functioning of system components. They are *capability changes* that cannot be made during an attack event, but must be planned for and applied before an attack occurs, or after as part of adapting. For clarity, in this thesis we use the word "component" to mean not only physical components but also software and services that a system may be composed of. Simply put, *redundancy* refers to having additional capacity (additional component instances) beyond what is required during normal operation conditions, so that the system impact of an unexpected event impacting some instances of the component is reduced. For the purposes of cyber resilience to attacks that can disable a component, redundancy should come in the form of separate instances of a component, i.e. simply having excess capacity will not do if it is susceptible to be taken out in one attack. In this context, *diversity* refers to having different implementations of a component so that the implementations do not share the same vulnerabilities for attackers to exploit. Redundancy can be ineffective against cyber-attacks if it is not provided with diversity [Gol10], as an exploit that succeeds on one instance of a component can be repeated on the redundant but non-diverse instances.

In our redundancy planning work, Chapter 8, we propose an implementation of our methodology to investigate the cyber resilience impacts of different redundancy allocations (assigning different numbers of component instances to fulfil a given task in a process), with and without applying diversity to the redundancy. Diversity is modelled as providing *alternative* servers with different vulnerabilities from the "regular" server instances used as standard when diversity is not applied. In our modelling, we assumed that no vulnerabilities are shared between the regular and alternative servers. Extending the model to consider servers that share some vulnerabilities but not others could be done by e.g. using the similarity between vulnerabilities as in Li et al. [LFH20].

In the mission viability analysis (Chapter 10), redundancy comes in two forms: the main way is via additional UAVs held in reserve in case of an attack, which can be flown in to replace compromised UAVs; in addition, each UAV taking part in the mission can fulfil different tasks, and can swap to a different task if required.

In our modelling, we assume that redundant capacity gets swapped into use automatically when the need arises due to a disruption. In the redundancy planning work, this occurs instantly, whereas in the UAV mission example redundant vehicles must be flown into the appropriate position, which takes some time, reflected in the recovery time for a given UAV task.

#### 4.3.2.2 Patching

Patching a vulnerability stops an attacker being able to exploit that particular vulnerability. In our modelling, it is assumed to form a part of bringing the attack to an end and facilitate recovery of services. Patching is important for recovery, as simply resuming services without patching makes it possible for an attack to simply repeat using the same exploits.

In our applications on redundancy planning and mission viability analysis (Chapters 8, 10), which are planning-focused, we assume simply that patching of compromised systems occurs as part of the recovery of services. However, in our application to countermeasure selection in Chapter 9, we model a reactive response to an attack, and patching is explicitly modelled as a countermeasure that the defence can take in response to an attack. The effect of patching is to remove a vulnerability from the AG, restricting potential attack paths. In the model, described in Sec. 9.3.5, a patching action requires $t_P$ units of time to implement, and comes at a cost consisting of a direct cost and a production impact.

#### 4.3.2.3 Containment

Containment refers to actions to stop an attack from propagating further. To keep our modelling general, we have modelled the effect of containment actions on the spread of an attack, without going to detail of the specific way in which this is achieved. It should be noted that, where the actions taken

to achieve containment affect the connectivity within the system, or otherwise reduce production performance, the impact on the performance of the production process must also be modelled.

In our works on redundancy planning and countermeasure selection in Chapters 8 and 9, containment is not modelled separately but is implicit in the assumptions regarding to when an attack is stopped. In the redundancy planning work (Chapter 8) we assume that if an attack is detected, the defence can contain it, i.e. will stop it from moving further along the AG, but not from disrupting the system using privileges already acquired. In the model in Chapter 9, containment occurs when patching actions stop the attack's ability to spread. By contrast, in our mission viability analysis work (Chapter 10) we address it directly in the SWN model, where the containment action removes the ability of the attack to propagate further, as the speed of containment is one of the key parameters of the model. As stated above, the modelling is done at the level of the effect of containment actions, instead of detailed modelling of the specifics of how it is achieved.

### 4.3.3 Recovery

**Recovery** refers to actions that the network owner uses to recover the functionality of components (DG nodes) compromised by vulnerability exploits. In practice this could occur via e.g. the full replacement of a component, repairing any faults or errors that could be limiting functionality, or simply resuming services that were disabled. However, in the case of cyber attacks, it is important to ensure that the attack will not simply repeat itself when the functionality resumes, so the attack must be stopped first (for example via containment, patching, or using replacement components that do not have the vulnerabilities originally exploited).

In our application examples, we use a simplified approach where we do not differentiate between different ways in which recovery occurs. Instead we have an abstract recovery action which effectively replaces a compromised component with a working uncompromised instance, with the same functionality (and vulnerabilities) as before. We assume that a recovery action takes $t_R$ time. The UAV case study in Chapter 10 makes somewhat of an exception to this, as there we consider the possibility of recovering services by swapping tasks between UAVs, in addition to full replacement by redundant

vehicles. These cases are considered separately as they have different impacts on the overall number of vehicles taking part in the mission, and different recovery speeds.

Full detail on the implementation details in the different application examples are included in the applications chapters, Chapters 8-10.

## 4.4 Our attack progression modelling approaches

Evaluating the probabilities of different attack outcomes can be done using various modelling formalisms, either via simulation or analytical solution methods. In our applications we have employed three approaches: 1. simulating attacks into a system, based on an AG and assumptions over attacker behaviour and defensive actions; 2. evaluating the likelihood of the various outcomes of particular attack scenarios; 3. Petri net evaluation of a net that represents the attack propagation and defence actions, which affords an analytical solution via conversion to a CTMC model. We will discuss these approaches in some more detail in the subsections below, after first discussing why other possible formalisms were not used instead.

Other modelling formalisms that were considered but eventually not used for the work in this thesis include employing a Bayesian network to solve the AG probabilities [PDR12, MGSBL17, MGSPL17], using stochastic activity networks (SANs) to model the AG [SM01, SA14], and using the ADVISE attack modelling approach by [LFK+11].

Bayesian AG (BAG) approaches can be used to evaluate the likelihood of an attack reaching particular nodes in the attack graph. BAGs are tools for risk evaluation where the output shows, for each node, the probability that the node is reached (e.g. an asset is compromised) during an attack, via any path, and without a specific consideration for time. However, for the purposes of our work, this kind of measure of risk is not very useful, as we want to consider the probabilities of specific *outcomes* of attack scenarios. These outcome probabilities must account for things such as the detection of the attack occurring (in Ch. 8), the relative likelihood of a choice between alternative steps (in Ch. 9), and the probability that an attack step occurs before a defensive one (Ch. 9,10). These are not the same as the BAG probabilities. As a consequence, we chose not to use BAGs in our model implementations.

While there are various benefits to Petri net models (including SANs), as discussed in Section 2.2.2.2, they are not ideal for modelling attacker behaviour in multi-stage attacks over AGs. This is because of the way transitions between places occur randomly based on which transitions are enabled (and their firing rates), without a notion of strategy or a specific target to aim at unless imposed by the net structure. We do not believe this can generally reflect the choices of actions by intelligent attackers, although it can be applicable in some specific attack scenarios. In the UAV mission case study in Chapter 10, where we use a SWN model (a Petri net variant), this is not an issue due to the specifics of the scenario: the type of attack, inter-UAV connectivity and identical UAVs make the ensuing attack graph a special type where the steps occur based on connectivity/proximity rather than choice. Specifically, the attack goal is to spread across the UAVs to maximise mission disruption, and as all UAVs are compromisable in the same way across the inter-UAV network, no intelligent choices are needed, and the attack will simply spread to all UAVs that can be reached at that stage.

Newer formalisms, such as HCSAM by [SA14], are problematic to integrate to our modelling as they are not standardised and do not have publicly available tools and implementations. A similar issue exists with the ADVISE formalism [LFK+11] that is part of the Möbius tool [DCC+02, FKL+13]: while the ADVISE description suggests it should be a useful model of attacks and could be usable as part of our work, we were unable to test it as part of our methodology as the tool software was not functioning correctly, is closed source and out of maintenance. However, the use of such formalisms as part of our methodology could be investigated in the future if new tool implementations are made.

Further, as discussed in Section 2.2.2, the use of an attack emulator platform such as CALDERA [MIT19] for modelling attacker behaviour is not appropriate for the use cases we envision for our methodology, which involve analysis of potential actions and comparisons of various design versions. This is because such tools operate at a different level of abstraction, where the emulation applies to a detailed real system instance, and working at this level of abstraction in our analysis use cases would require ways to emulate whole system instances.

In the next subsections below, we briefly describe the three approaches used in the models for our application samples in Chapters 8-10.

### 4.4.1 Simulating attack steps with probabilistic choices

In the countermeasure selection application (in Chapter 9), our evaluation approach simulates attacker choices of exploits using edge probabilities based on CVSS [MSR06, FIR16] base scores. Such an approach has been common in works using AGs for risk assessment, e.g. in [FW08, PDR12, OS12, NWJS13]. In our simulations the attacks follow a path towards a goal node, which is picked from among the AG nodes with the highest availability impact on the final services (the single highest impact one, or drawn among the shared highest impact nodes). Each chosen edge is picked from those along the paths to the goal, based on a draw between viable candidates, where the distribution is based on the edge probability values (representing access complexity). This creates variety across simulations, approximating different attacker choices based on e.g. different skill levels. One attempted vulnerability exploit is allowed at a time, i.e. the attack cannot attempt multiple different exploits simultaneously. If a given step is not possible, due to a defensive action, the attacker attempts another exploit that is on a path to the goal. If the goal becomes unreachable, the attack will stop.

The choice of the countermeasures is on what AG nodes to patch given that an attack is ongoing, and when to recover a service compromised by an attack. We propose our own approach based on expected costs of actions, considering that patching actions cause temporary unavailability of system components (due to the application of the patch, including testing for and fixing any impacts to other services etc.). As a patching action causes short term disruption but has a long term benefit in terms of system hardening, our approach includes a way to balance the future benefit with the short term costs. We discuss the details of this in the cost modelling chapter, in Section 7.3, and in the chapter on the application itself, Chapter 9.

### 4.4.2 Pre-defined attack scenarios

Pre-defined attack scenarios, describing a set of AG nodes the attack tries to reach, are useful if the number of possible paths (considering different goals, routes to them, and different orderings of steps etc.) in an attack graph is high, but only a subset of the paths make sense for the attacks that the defence is most concerned about. For example, if the defence is concerned with production impacts,

then only those attack paths that disrupt components with production impact are of interest. In this sense, such pre-selection of scenarios is not very different from running risk analyses based on pre-determined goals, as done by many approaches, such as many Bayesian AGs.

What we term as attack scenario here is essentially a description of the target (a set of privileges in an AG, selected based on their impact on the system) the attack has. This does not determine what *specific path* (including the order of attack steps) a specific instance of the attack follows, or what the *outcome* of an attack instance is. So our use of pre-defined scenarios does not imply pre-determining the paths or the outcomes of attacks. The key point is, the *scenarios* describe a specific type of an attack, and target nodes for it, while the various different *outcomes* of those scenarios describe different outcomes that can occur, given path choices and defence actions. It is the latter which are ultimately used for assessing the impact of attacks.

The way we define attack outcomes from the scenarios in our redundancy planning work (in Chapter 8) is as follows. We consider a set of attack scenarios that are reasonably expected to cause significant disruption, and evaluate the resulting system performance under the different outcomes of those scenarios. The scenario input consists of the AG nodes targeted, and the weight given to the scenario, relative to others, in the overall analysis run. Based on the AG setup, the scenarios are then processed into *paths* (of move steps), and may split into *path variants* if there are different orderings of steps that can be taken to reach the target nodes. Finally, the attack *outcomes* are determined based on the paths (incl. variants) and defensive actions, which consist of detection and subsequent containment of the attack.

In accordance to what we explained about attacker behaviour in Section 4.2.1 above, in deciding the paths and variants for the scenarios, we focus on the worst case where the attack chooses the quickest path towards its goal. As part of this, we assume that the attacker knows what they are trying to achieve and how to proceed there. In addition, in determining the system disruption caused by the different outcomes, we assume that the attacker does not launch disruptive actions until obtaining all the privileges required to reach their goal, or if forced to act due to being detected. The latter is because disruption would alert the defence, so launching disruptive actions prematurely would limit the attack success. This means that the outcomes where the attacker does not reach the final

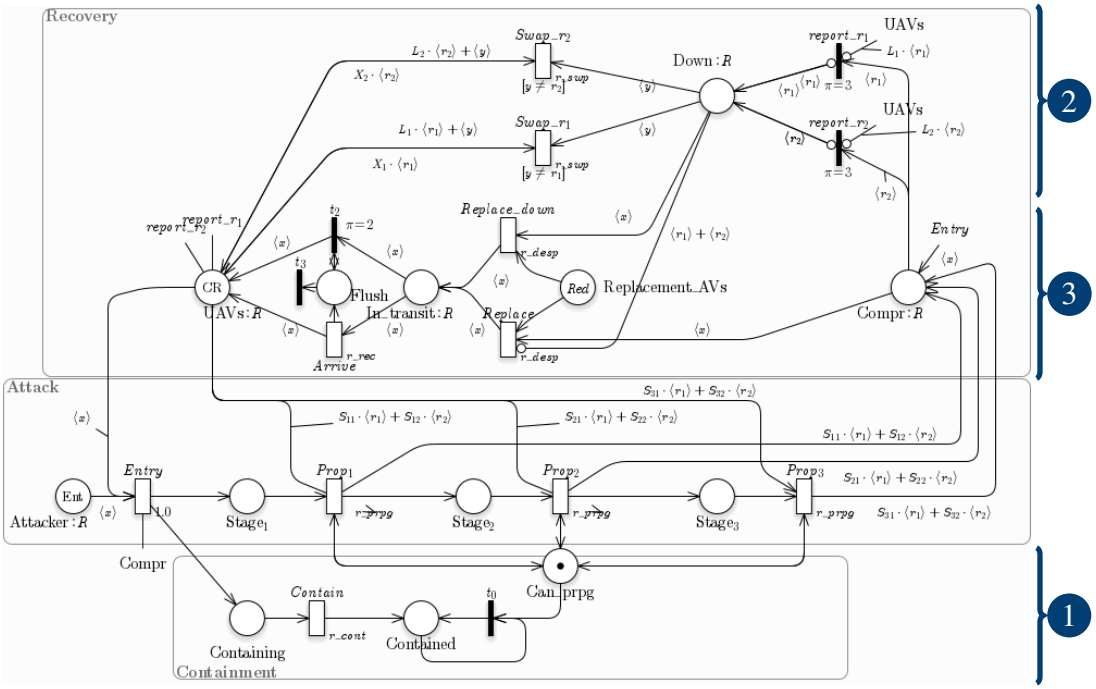



Figure 4.4: The SWN model, with mitigation parts highlighted

target privilege (or the final copy, when privileges in several server instances are targeted) still yield disruption, but just to a lesser extent than a fully successful attack.

We discuss how the outcomes are used in estimating the expected attack impact in Section 6.2, as determining the impacts also involves the production model, which is discussed in Chapter 5.

### 4.4.3 Petri net modelling

Our attack propagation modelling in the UAV mission viability analysis application in Chapter 10 uses a stochastic well-formed net (SWN), which is a Petri net variant with stochastic time and coloured tokens. The model represents the whole attack progression, including defensive actions to mitigate the attack (containment, and recovery via vehicle replacements).

The defensive actions to mitigate attacks are: **1.** *Contain the attack* by disabling a compromised part of the mission; **2.** *Swap roles/tasks between UAVs*; **3.** *Introduce a replacement UAV* to the mission. Role swaps and replacement UAVs act as a recovery mechanism for performance; the latter takes a longer time than the former. The graphical representation of the SWN is shown in Fig. 4.4, with the parts that refer to the different mitigation actions labelled with their corresponding numbers in the

above list. The attack is modelled in stages (shown in the mid section of Fig. 4.4 inside the box labelled "Attack"), with three stages shown in the SWN (extension to further stages is straightforward).

As the SWN of Fig. 4.4 is built for the specific assumptions of the attack and the mission in the case study setting, we leave more detailed description of how the model is built and what the different parts mean into Chapter 10, where the case study is described in detail. Thus, more detail on the specifics of this SWN, and how it represents attacks, is given in Section 10.5.3.

The SWN modelling allows us to assess how many UAVs are functional in their roles over time, given the progression of the attack and the containment and recovery actions. In conjunction with a production model of the mission, this allows us to estimate the viability of the mission, i.e. whether the mission target can be reached despite an attack, given different assumptions on parameter values on the speed of attack propagation, containment, and recovery actions, and the number of vehicles taking part in the mission and waiting in the replacement reserve.

There are some limitations to consider. A key aspect of the use of a Petri net for attack modelling is that the behaviour of the attacker and defence have to be represented by the structure – where these behaviours can be represented by the structure and firing rate assumptions, an SWN can be a good way to solve for impacts, enabling efficient stochastic analysis. However, this does not hold always. For example, for considering reactive choices of defence actions such as countermeasure selection, a different kind of analysis approach would be required, without the action choices being built into the SWN structure. Additionally, as already touched upon at the beginning of Section 4.4 above, for attacks where the goal is a specific resource rather than wide spread, a simple spreading process does not represent the behaviour of an intelligent attacker, and it would be hard to represent complex decision-making using only the structure of an SWN model.

## 4.5 Chapter summary and discussion

This chapter described the attack progression model part of our methodology, explaining the reasoning for the modelling and the different parts that it needs to address. It also introduced the approaches that we have used to implement the modelling in our applications in Chapters 8-10.

To fulfil its purpose as part of a resilience impact assessment methodology, the model of attack progression must have a representation of the *attack paths* that are possible within a system, describe *attacker behaviour* with respect to the possible paths and how it responds to defensive actions, and describe such *defensive actions* that can influence what actions are available for the attacker. In addition, there are important interactions between the attack progression and the production modelling (which is discussed in the next chapter), which must be considered to enable attack impact assessment. The model of attack paths, e.g. an AG, must be such that it can be linked to the impacts on system components in the production model, using an *impact map* (discussed in more detail in Chapter 6). The defensive actions must be modelled in such a way that reflects their impact on the system production in addition to their impact on the attack progression, e.g. any unavailability of components caused and the duration of it. Further, the costs of the actions must be considered by the cost modelling, discussed in Chapter 7.

The specific requirements for implementing the modelling depend on the use-case, and the specifics of the system and the type of attacks considered. In addition to discussing the aspects that need to be addressed by the model, we also gave examples of how the implementations can vary, with reference to the modelling we use in our use cases. For example, to estimate the impact of possible future attacks to a system, one must consider how the attacker actions might be detected, as this affects the possible outcomes of the attacks in terms of the extent of their success. While our implementations vary on this point between the use cases, each implementation must address attack detection in some way to enable impact analysis.

# Chapter 5

# Production modelling

To evaluate the impact of cyber attacks on the output performance of the system, we require a model of the process by which the system produces its output, including how different system components and different levels of production inputs relate to the level of output produced. We provide three instances of models that we use for this purpose, and use the term "production model" to refer to this class of models. This term was chosen to make it clear that the focus of the model is to describe the production process, and as the purpose of the model is similar to production functions used in economics. The common feature of all of our production model instances is a dependency graph (DG), representing dependencies between system components. The approaches differ in the way the production performance is modelled in relation to how attacks impact the functionality of the components, and how they affect others due to the dependencies in the system.

## 5.1 Dependency modelling with DGs

Practically any production process can be represented in terms of dependencies between components or stages of production. For example, to produce output (sales), an e-commerce requires both a web server and an order processing server, which themselves likely depend on many other services. Similarly, the manufacture of physical goods or the provision of services such as healthcare involve steps that depend on each other to produce the final output. Given this, **dependency graphs** provide

a straightforward way to model how a system depends on various components and sub-processes to produce its output.

A dependency graph (DG) describes the dependencies between the components of the production model. The simplest definition is a directed graph of components where edges represent dependency between components, such as the one used in our redundancy planning work in Chapter 8:

**Definition 5.1** ***(Dependency Graph)*** *Given a set of system components $N$, and a relation $E \subseteq N \times N$ with $(s,t) \in E$ meaning that component $s$ depends on component $t$, a dependency graph D is the directed graph $D = (N, E)$, where $N$ is the set of nodes and $E$ is the set of edges.*

This definition only describes which nodes a given node $n$ depends on, but leaves out the exact type of the dependency in terms of the impact on $n$. That is, the simple definition is silent on whether the functioning of node $n$ requires all of its dependencies to be provided (and functional) or a subset of them, and how the specific level of functionality (i.e. performance) of the dependencies affects the performance of $n$. For the purposes of impact assessment, this information is also necessary. Some works in the literature have built custom definitions of dependency graphs that include this performance impact directly as part of the definition, such as the "generalised dependency graph" in [AJPS11, AJ17], and various definitions [KCBCD10, Jak11, CYS+18] specify the type of dependency (logical OR, logical AND) but no further performance aspects. We separate these aspects from the basic definition of a dependency graph above, to better explain different approaches to evaluating the impacts occurring via dependencies, which we do in Section 5.2 below.

## 5.2 The propagation of performance impacts

While a basic DG provides the structure of how components depend on each other, we also need a way to quantify how the performance of a component is affected by an attack, how it affects components that depend on it, and how the performance of a component is affected by its dependencies. Quantifying these enables evaluating the system-level performance impacts.

Attack impact assessment works have quantified impacts by using dependency information and a

function to propagate impacts across dependencies, combined with some impact metrics assigned to individual components. The metrics have been based on utility values of system components [AJPS11, AJ17] or CVSS impact scores for attack steps [Jak11, CYS+18, HSKG20]. However, although not used in attack impact assessment works previously, there are other approaches to quantifying system performance, such as the performance modelling formalisms we discussed in Section 2.2.3, which could be more appropriate in settings to which they apply.

In the works where we have applied our methodology, included in Chapters 8-10, we have implemented three ways of modelling performance impacts in combination with a DG: network status function, production-function model, and queueing network performance modelling. The next subsections will discuss each of these in turn.

### 5.2.1 Network status function

In our work on cost-efficient countermeasure selection in Chapter 9, published in [SML19], we followed the approach by [AJPS11, AJ17] in using a set of "status functions" to represent interdependency impacts, i.e. how the availability status of an entity is affected by the statuses of the entities it depends on. This is also similar to the approaches used by [Jak11, CYS+18, HSKG20].

As the status functions in [AJ17] build on a part of their DG definition, termed Generalised Dependency Graph, we reproduce this definition here:

**Definition 5.2** ***(Generalised Dependency Graph, [AJPS11, AJ17])*** *A generalised dependency graph is a labeled directed acyclic graph $D = (H, Q, \phi)$, where: H is a set of nodes, corresponding to network components; $Q = \{(h_1, h_2) \in H \times H | h_1$ depends on $h_2\}$ is a set of edges; $\phi : H \rightarrow \mathcal{F}$ is a mapping that associates with each node $h \in H$ a function $f \in \mathcal{F}$ s.t. the arity of $f$ is equal to the outdegree of $h$*[*]*. For each node $h \in H$, $\dot{h}$ denotes the set of components that depend on $h$ and $\dot{h}$ denotes the set of components $h$ depends on.*

The first two points in definition 5.2 describe the basic structure of the dependency graph (which nodes depend on which others), while the mapping $\phi$ describes the type of dependency that each node

[*]"If $h$ is a terminal node in the dependency graph (i.e. it does not depend on any other node), we assume $\phi(h)$ is the constant (0-ary) function 1" [AJ17]

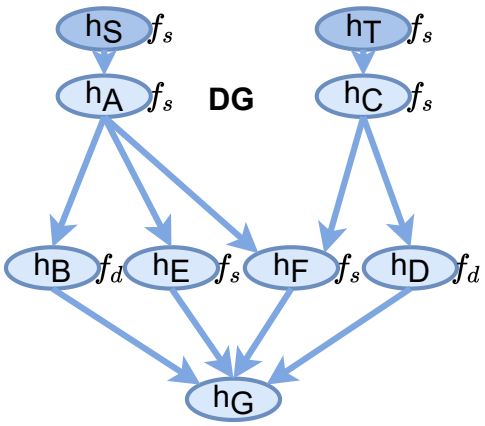



Figure 5.1: DG, redrawn from [AJ17, Fig. 9]

$h \in H$ has on its supplier nodes $\dot{h}$. We consider the same dependency function types as used by [AJPS11, AJ17]:

$$f_r(a_1, ..., a_n) = \begin{cases} 1 & \text{if } \exists i \in [1, n] \ s.t. \ a_i = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

$$f_d(a_1, ..., a_n) = \frac{1}{n} \sum_{i=1}^{n} a_i \tag{5.2}$$

$$f_s(a_1, ..., a_n) = \begin{cases} 1 & \text{if } a_i = 1 \ \forall i \in [1, n] \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

where $a_i$ represents the availability value of a network component which the current component directly depends on, and $n$ is the total number of components on which the current component depends (i.e. for $h_i$ we have $n = |\dot{h_i}|$). Here, $f_r$ is a *redundancy-type* dependency (logical OR), $f_s$ is *strict dependency* on all supplier nodes (logical AND), and $f_d$ means availability of $h$ is the mean of the availabilities of its suppliers (*degradation*). A sample DG of this type is provided in Fig. 5.1. The DG nodes shown are network components: $h_A$ to $h_G$ provide intermediate services (internal services), while $h_S$ and $h_T$ are customer-facing final services (the product of the organisation). The dependency types are given as labels next to the nodes. For example, for $h_T$ we have $f(s(\dot{h_T}, t)) = f_s(s(h_C, t))$.

With the dependency functions $f$, we can now describe the network status function, which maps network components to their availability statuses at time $t \in \mathcal{T}$, where $\mathcal{T}$ is the set of time points.

**Definition 5.3** ***(Network Status Function, [AJPS11, AJ17])*** *Given a generalised dependency graph* $D =$

*$(H, Q, \phi)$, a network status function for $D$ is a function $s : H \times \mathcal{T} \rightarrow [0, 1]$ such that $\forall h \in H$ and $\forall t \in \mathcal{T}$, $s(h, t) \leq f(s(h_{i_1}, t), \ldots, s(h_{i_m}, t))$, where $f = \phi(h)$, and $\dot{h} = \{h_{i_1}, \ldots, h_{i_m}\}$ is the set of components $h$ depends on.*

The definition states that a network status function $s$ assigns each DG node $h$ a value in the range $[0, 1]$, capped by $h$'s dependency function $f$ over the statuses of its dependencies. That is, the indirect status impacts from the components that $h$ depends on define a maximum availability level for $h$, but the status of $h$ can be below this if directly affected by an attack.

We give the functional form for the status of a node $h$ at time $t$ as:

$$s(h, t) = f(s(\dot{h}, t)) \prod_{v \in V_{e,t}} (1 - \eta(v, h)) \tag{5.4}$$

where $V_{e,t}$ is the set of AG nodes that are in an exploited state at time $t$; $\eta(v, h) \in [0, 1]$ is the availability effect that the exploit of vulnerability $v$ has on component $h$ (0 for no effect, 1 for fully unavailable); $\dot{h}$ is the set of components that $h$ is dependent on; and $f(s(\dot{h}, t))$ is the availability effect on $h$ from its dependencies.[2] The status of network component $h$ is composed of two effects on availability: 1. Compromise effect (direct): $(1 - \eta(v, h))$, the effect of compromise (vulnerability exploit) in the AG node $v$ which corresponds to component $h$; 2. Dependency availability effect (indirect): $f(s(\dot{h}, t))$, the effect of unavailabilities in the components that $h$ is dependent on, of type $f_r$ (redundancy), $f_d$ (degradation) or $f_s$ (strict dependence).

Finally, using the node status functions, we measure the performance impact onto the system with an overall service provision status, *service performance* (SP). It is the weighted sum of the statuses of the output services (final services, the "product" of the organisation), weighted by their relative utilities for the organisation. Mathematically, SP at time $t$ is given as:

$$SP_t = \sum_{h \in H_S} \frac{u(h)}{\sum_{h_k \in H_S} u(h_k)} s(h, t) \tag{5.5}$$

where $s(h, t)$ is the status (availability level) of component $h$ at time $t$, $u(h)$ is the utility the organi-

[2] A component *recovery* is represented as the removal of a vulnerability $v$ from the exploited set $V_{e,t}$, while *patching* vulnerability $v$ removes it the from the full set $V$.

sation derives from the service component $h$ (during its full availability), and $H_S$ is the set of components which are final services. With regard to the sample DG in Fig. 5.1 we have $H_S = \{h_S, h_T\}$, as the services are $h_S$ and $h_T$. For example, assuming utility values $u(h_S) = 10, u(h_T) = 7$, if node $h_C$ becomes unavailable at time $t$, service performance at time $t$ is:

$$SP_t = \frac{u(h_S)}{u(h_S) + u(h_T)} s(h_S, t) + \frac{u(h_T)}{u(h_S) + u(h_T)} s(h_T, t) = \frac{10}{17} \cdot 1 + \frac{7}{17} \cdot 0 = 0.588$$

While the approach to impact evaluation described above, which is applied as part of our methodology in the work shown in Chapter 9, is based on that by [AJPS11, AJ17], we made several changes to make it suit our methodology for resilience based impact analysis. We now briefly describe these changes, which relate to the network status function, and to how the overall impact is quantified.

Our version of the network status function (5.4) makes several changes in order to enable applying it to countermeasure selection and analysis with recovery included. Specifically, the functional form of the network node status function $s(h, t)$ in [AJ17] does not allow modelling recovery in a component's status due to the way the previous period's status $s(h, t-1)$ enters the function definition. A low availability status one period will punish the component's status in later periods, even if recovery was done. Therefore, we provide our own form for the node status function, (5.4), that can keep track of all vulnerability exploits that are in effect at time $t$ (either newly exploited, or previously exploited ones whose related security conditions remain compromised), each of which may have a direct impact on $h$. Our reformulation also introduces a multiplicative interaction between the direct and indirect availability effects, replacing the minimum function in the original. In this way, the direct and indirect effects interact to set the effective availability, in contrast to the dependency availability effect only acting as a cap on the effective availability level. These changes also enable multiple exploits affecting a node $h \in H$ directly, which the original formulation does not support.

As shown in equation (5.5), our evaluation of attack impact is based on the impact on the performance of the final services, i.e. the output of the system. This differs from the approach in [AJPS11, AJ17], who assign utility values to each component of the DG, or the approach of CVSS impact scores for attack steps in [Jak11, CYS+18, HSKG20]. We believe that the intermediate services provide value

only when at least a part of the final service to which they contribute is online (available). Therefore, we only set utility values for the final service nodes ($h_S$ and $h_T$ in the sample), with the value of the other nodes only reflecting the impact they have on the utility arising from the final services.

As mentioned in Section 5.2 above, several papers concerned with the impact of attacks have used CVSS [MSR06, FIR16] scores as impact metrics, as part of a dependency graph-based approach similar to that described above. Recent examples of such approaches are [CYS+18, SLS19, HSKG20]. The key benefit of CVSS scores is that the data is standardised and available for all vulnerabilities that have been given a CVE identifier, allowing for straightforward use in many settings with the support of tools for checking systems for such vulnerabilities such as Nessus [Ten], and others for building AGs and analysing them [OGA05, CYS+18]. However, the use of CVSS scores as impact metrics as part of system-level impact estimation has some important weaknesses. The base score, which was used as an impact metric by e.g [CYS+18, SLS19, HSKG20], is an aggregate across several aspects, and does not reflect impact only: a low base score could be assigned to a vulnerability which is complex to exploit even if it would have a high impact if successful. Therefore, using this as a measure of impact is not satisfactory as it mixes impact with aspects of attacker capabilities and system configuration. While CVSS also provides separate metrics for impact, these are non-numeric and limited in detail, consisting of three severity levels – high, low, or none. Additionally, as CVSS scores are only available for known vulnerabilities, a different approach is needed if estimating impacts of attack steps that do not correspond to vulnerability exploits but other attack techniques such as phishing.

We have chosen not to use a CVSS-score based approach, partly due to the drawbacks of their use as impact metrics explained above, but mainly as we focus on the production performance. We believe that modelling the impacts to the process is more intuitive when focusing on the production components instead of the vulnerabilities, as this allows the system experts to formulate what the impact of disruptions would be.

### 5.2.2 Production-function based modelling

A production function in economics is an abstract representation of a production process with a set of inputs, expressing the quantity of output as a function of quantities of inputs. They are used in economics for theoretical models of input allocation, see e.g. [MWG95]. This type of function provides a simple approach to approximating the performance of a production process where there are identical production inputs, and only the overall level of the inputs matters, not their order/location. Additionally, where a more detailed representation of performance is not available or feasible to model, such a function could be used to approximate the expected performance of a (sub-)process in terms of its dependencies in a DG. In fact, the functions for the functionality impacts in [AJ17], discussed in subsection 5.2.1 above, are essentially of this type, if we consider that this kind of model applies to the performance level of each node of the DG.

As an example, in the mission viability analysis application, in Chapter 10, we use this approach to estimate the performance, in terms of coverage reached, of a group of UAVs on a survey mission. We model how the different tasks, reflected by the UAV roles, combine to produce the mission output, which is the survey coverage of a certain area with the appropriate sensor inputs. For this, we use a function that resembles a Leontief production function[3], where a unit of coverage rate is achieved by a strict ratio of UAVs in specific roles. We choose this form as relay roles are typically not substitutes for sensor roles and vice versa, although in a different setting a more general type of function could be used. Fig. 5.2 gives a dependency graph representation of how the mission output (the node "Service" in the figure) depends on the different roles being filled. The dependency graph in Fig. 5.2 shows that an attack impacting a UAV in a Video role impacts the video service, taking the number of UAVs in the video role below the standard requirement of 4 (shown above the "Video" node of the graph). The production function for this system measures the coverage rate ($ha/min$), and the specific functional form for the sample shown in the figure is given as:

$$cov\_rate = r_n * min(\frac{x_{rel}}{x_{rel}^{req}}, \frac{x_{vid}}{x_{vid}^{req}}, \frac{x_{tmp}}{x_{tmp}^{req}}) \tag{5.6}$$

[3] An introduction to different production functions, including the Leontief input-output model, can be found in economics textbooks, e.g. [MWG95].

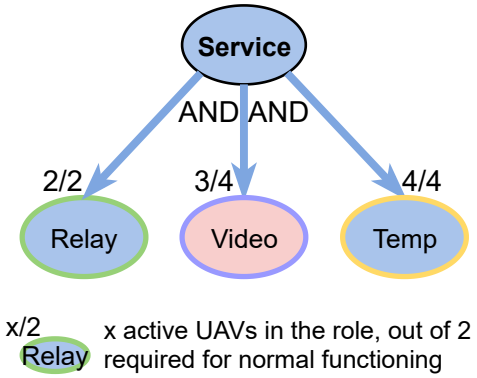



Figure 5.2: Production dependency graph, as used in [SPL21]

where $x_{rl}$ is the number of UAVs active in the role $rl \in \{rel, vid, tmp\}$ (relay, video, temperature) and the required number of UAVs in role $rl$ is denoted by $x_{rl}^{req}$. The multiplier $r_n$ represents the expected coverage rate during normal operation with the required number of UAVs in each role.

Full details on this approach is given in Chapter 10 where we discuss how we apply our methodology to the analysis of mission viability in a multi-UAV mission case study.

### 5.2.3 Performance modelling with QN models

While a dependency graph describes the *structure* of a production process, the *performance* of production can in some systems more naturally be evaluated using performance modelling approaches such as queueing models [LZGS84, BGDMT06]. Consequently, we provide a queueing network (QN) approach to attack impact evaluation, applicable to systems where performance is a key concern and the production process contains stages that can be modelled as queues.

Other approaches to modelling performance exist, such as Petri net variants like GSPNs [MBC+95] or stochastic activity networks (SANs) [SM01]. We have chosen to use a QN implementation, as queueing models are well understood and impactful, applicable to various commonly encountered systems, and have more readily available reference tools. Although other approaches could be suitable to be added to our methodology, our sample implementation uses QNs.

In the next subsections we will describe our approach to using QN models as part of attack impact evaluation, starting from a brief description of an example QN model for a production process, then

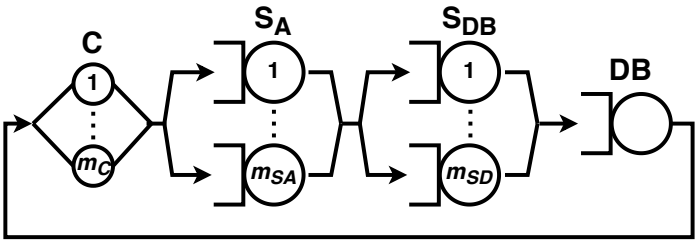



Figure 5.3: Queueing network for the J2EE case study

explaining how we use it in conjunction with attack modelling to estimate attack impacts.

#### 5.2.3.1 QN model of a production process

In our attack impact assessment for the redundancy planning work discussed in Chapter 8, a QN model is used to evaluate the performance of the system output when the attack impacts some of the components involved in the production process. In essence, we use the QN model as the performance evaluation mechanism relating to the system components represented in DG nodes. For example, Fig. 5.3 shows a QN that represents an e-commerce system that consists of application servers $S_A$, servers processing database requests $S_{DB}$, and a database $DB$. Each of these components are modelled as queueing *stations*, which process jobs at some service rates $\mu$, and jobs that are waiting to be processed form queues at the stations where they are waiting. The station $C$ in the graph represents customers sending jobs to the system, which gets processed in the server stations before a response is returned to the customer from the $DB$. The model shown is a closed model with $m_c$ customers that send new jobs each time their previous one gets fulfilled.

The figure shows that the application and DB processor stations have server multiplicities $m_{SA}$ and $m_{SD}$, respectively. In our redundancy planning work, we identify the appropriate number of each server type for the most cost-efficient performance during expected attacks. This also includes the number of databases in $DB$, although it is not shown in the figure as consisting of multiple components.

The purpose of a QN model is to evaluate the performance of the modelled system given a description of the processing capabilities of the stations, the multiplicity of processing components (servers, DBs) in the stations, and the workload that the system is under. The performance is measured by calculating

metrics related to the processing of jobs, such as throughput of jobs through a particular station, the number of jobs in a specific queue etc. In our redundancy planning case study, we are interested in the throughput of returning jobs to the station $C$, i.e. fulfilled customer requests, and whether this will continue to meet the conditions of a service level agreement (SLA) during periods when cyber attacks occur.

For systems where they are applicable, QN models can more accurately reflect the system performance impact of tasks competing for resources than higher level modelling based on dependencies only. This includes expressing issues such as bottlenecks due to imbalanced allocation of resources or their capabilities to process tasks, and cascading effects due to downtime of some of the resources. Thus, when such issues are of concern and the production system can be modelled using a QN, modelling performance using a QN can provide more accurate evaluation of attack impacts than a more generic approach based on dependencies only.

#### 5.2.3.2 Modelling attack outcomes

To evaluate the impact of an attack using QN models, we express the effect that the attack has on the availability of servers in the nodes of the production model. Impacts are evaluated for the different outcomes of the attack, including partial successes caused by detection. For each distinct impact stage of these attack outcomes, the QN is updated and its performance evaluated, to obtain an estimate of system performance throughout the attack.

The availability impact of a specific attack outcome is represented by a matrix $M_{attack}$, specifying the numbers of active (available) servers in different production nodes across the different *stages of impact* from that outcome. That is, $M_{attack}$ is a $I \times J$ matrix with elements $m_{i,j}$ specifying the number of active (available) servers at stage $i$ of the impacts of attack outcome, in node $j$ of the performance model. The total number of rows $I$ is the number of distinct impact stages that occur in response to a specific attack outcome, and the number of columns $J$ is the number of production nodes in the system for which redundancy is considered. The *impact stages* (rows) start from the normal system state, followed by the states reached after an attacker has taken disruptive steps, and those after defensive actions were taken. The last row is the state that prevails when the model time horizon

$t_h$ is reached. For example, in our diversified system with four server types, with the server allocation $[m_{SA}, m_{SAA}, m_{SD}, m_{DB}]$ the attack outcome where two servers in $S_{AA}$ are disrupted is expressed as:

$$M_{attack} = \begin{pmatrix} m_{SA} & m_{SAA} & m_{SD} & m_{DB} \\ m_{SA} & m_{SAA} - 2 & m_{SD} & m_{DB} \\ m_{SA} & m_{SAA} & m_{SD} & m_{DB} \end{pmatrix}$$

where: row 1 is the initial state; row 2 shows the available servers after two servers in $S_{AA}$ have been disrupted; and row 3 represents the situation after recovery.

The performance evaluation for an attack outcome uses the rows of $M_{attack}$ as server multiplicity inputs to a sequence of QN models used to estimate performance across the impact stages. The *duration of the impact stages* (rows of $M_{attack}$) is represented by the column vector $\tau_{attack}$, whose values are computed from the time-to-exploit and time-to-recover estimates. For the example above, the related duration vector is $\tau'_{attack} = [56, 3, 661]$. The effect from the attack materialises after 56 hours as it requires an exploit to obtain a stepping-stone privilege (24h), another to obtain the privilege enabling disruption of the first server (24h), and further two exploits to obtain these privileges on another server copy. These repeat exploits are assumed to take only 4h each as they are re-exploits of the same vulnerabilities as for the first server. After this, the disruptive action is launched. The second state is active for $t_r = 3$ hours before recovery of the services takes place, and the final state stays in effect for the time remaining until the time horizon, which is 720h (one month) here, so $t_h - 59 = 661$ hours.

The number of stages in $M_{attack}$ depends on the attack and recovery models. If the attacker causes disruption in several stages, the matrix would contain one stage for each change in the number of active servers. However, unless a disruption is unavoidable during an exploit, taking disruptive steps in several stages instead of in one go is not effective attacker behaviour, as it increases the likelihood of detection before the objective is reached. Similarly, recovery done in multiple steps would add stages. In our case study in Chapter 8, the $M_{attack}$ matrices consist of a total of three stages (including the starting and final situations), as all attacks yield one impact stage as all disruptive steps are taken in one go, and recovery occurs simultaneously in all servers requiring it.

We designed the approach above for representing attacks where a component instance is assumed to become fully unavailable when the attacker makes a disruptive step against it. Partial degradation of performance could be modelled with added modelling efforts, but even for such cases, we feel that the full unavailability impact is a useful "worst case" impact from an attack that has the kind of disruption intent that we consider. A more limited attack impact would better represent some stealthy forms of attacks, e.g. for gathering information, but it is unlikely for attacks where the aim is to disrupt production.

## 5.3 Chapter summary and discussion

This chapter focused on the details of *production modelling*, an important component of our methodology. It is used for modelling system output performance, required for measuring resilience based on performance over time, and is a key part of evaluating the impact from attacks as it enables calculating the overall output production impact of an attack on individual production components. The next chapter discusses how the attack impact evaluation is done, linking the production modelling with attack progression modelling.

In our methodology, the structure of the production process is represented with a dependency graph, which is a generally applicable way to describe how the different components or stages of production depend on each other to produce the system output.

To model the effect on production of a component being disrupted by an attack, we must model how performance impacts propagate from a component to the overall production. We proposed three approaches to do this: two related ones based on functions applied to the dependency structure, and QN modelling. When applying our methodology, the choice between the approaches comes down to which one best suits the situation to be modelled.

A *status function* based approach is good for modelling situations where dependency relations are important but no suitable performance modelling is available to describe the process in more detail, or where such added detail is not needed. An economics-type *production function* can be used to approximate production performance in settings where production levels depend on input quantities,

but where the order of the tasks within the process (represented as inputs to the production function) is not important. Both of these two approaches can be good for approximating impacts when the disruptions cause binary impacts on component availability, i.e. a copy of a component is either functional or fully unavailable, but may not be as good for situations where effects are more subtle and the accuracy of performance estimation is important.

*Performance modelling* approaches such as QNs are a good fit for attack impact evaluation in systems where the level of measurable performance is important, and where performance models apply. There are limitations, for example, QN models apply to systems and processes that have components representable as queueing models, but not all processes can or need to be modelled as such. As mentioned, there are other performance modelling frameworks that could be suitable for evaluating attack impacts as part of our methodology, such as the various variants of Petri nets.

# Chapter 6

# Attack impact assessment

Our approach to analysing attack impacts on a system is based on evaluating the effects that attacks have on the output performance of the system, via their impact on the components of the production process. The key parts of this evaluation are a model of the system production, a model of attack progression in the system, and an impact map relating steps along the possible attack paths to components of the production process. Additionally, within the attack progression modelling, the approach must consider any defensive capabilities and actions that can impact the outcomes of the attacks, such as attack detection and countermeasures.

The estimation requires both a method for evaluating the performance impact of a single outcome of an attack, and an approach to aggregate the results for all the attack outcomes to represent the expected impact of attacks to the system. An overview of the process of estimating the impact of an attack scenario is given in Figure 6.1. First, an attack scenario is entered into the attack progression model (labelled with number 1 in the figure). The attack progression model yields the various outcomes related to the scenario provided (number 2) in terms of attack paths over an AG, and the probabilities for these outcomes (number 3). Then, the attack outcomes are mapped into their effects in terms of disruption of system components (DG nodes) with an impact map (number 4 in the figure). The production model (number 5) will then evaluate the performance impact of the disruption from the attack outcomes, yielding performance impacts (number 6) for the different attack outcomes. The performance impacts can be used in one of two ways: they are either converted directly to an expected

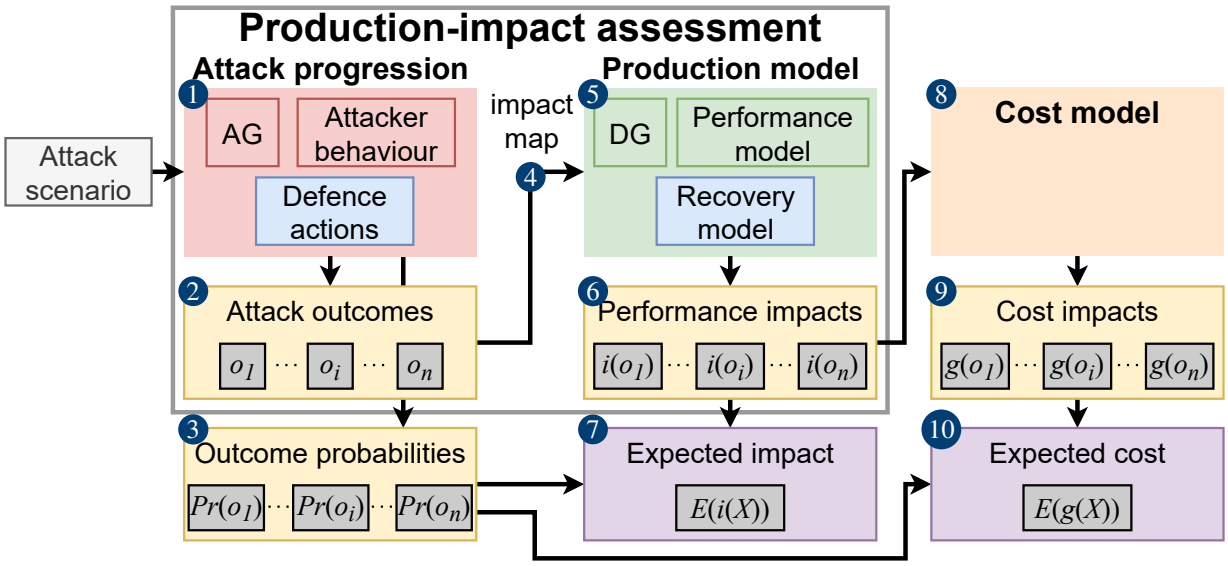


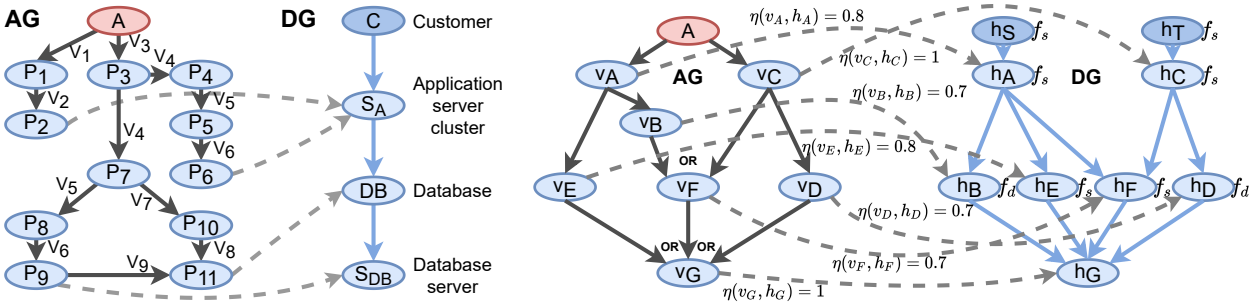

Figure 6.1: High-level representation of our attack impact assessment

performance impact for the attack scenario (number 7), using the outcome probabilities (number 3) as weights, or they are passed onto the cost model (number 8) to determine cost impacts of the different attack outcomes (number 9), and finally the expected cost for the scenario (number 10).

Attack progression, including the various outcomes, were discussed in Chapter 4, and approaches to evaluate the impact of a disrupted system component were introduced in Chapter 5. Here we first discuss how the attack progression model connects to the production model to determine which system components an attack outcome can affect, i.e. the impact map part (number 4) on Fig. 6.1, and thus what the impact of a single attack outcome is, in Section 6.1 below. Then, in Section 6.2 we explain how the impacts of attack outcomes are combined together to form the *expected impact* from attacks. Finally, Section 6.3 explains the types of output from the production impact assessment, whether this is intermediate impacts (number 6 on Fig. 6.1) to be passed into the cost model, or expected impacts (number 7). Thus, this chapter completes the explanation of the modelling behind our production-impact assessment, and leads into the cost modelling discussed in the next chapter.

## 6.1 The impact of a single attack outcome

The impact of a given attack outcome is determined based on connections specified between the attack progression and the production model. They map attack steps in the attack model into specific




(a) Redundancy planning case study, Chapter 8

(b) Countermeasure selection case study, Chapter 9

Figure 6.2: Impact mapping in implementation case studies

components of the production process that are affected by these attack steps.

Figure 6.1 shows the general idea of what is involved at an abstract level: attack outcomes, an impact map, and the production model. The *impact map* is a mapping from AG nodes to DG nodes, showing which system components can be affected by disruptive steps launched from a given attack graph node. Applying the map to the path in an attack outcome shows what DG nodes are directly affected, and then by applying the dependency structure of the DG and the performance model we find the system performance impact of the attack outcome.

This mapping and its implementation is intuitive to depict graphically as links between AG and DG nodes. To illustrate, Fig. 6.2 shows the sample cases used in the redundancy planning and countermeasure selection works in Chapters 8 and 9, where the mapping is visible as hyphenated arrows from AG nodes to DG nodes. In the mission viability analysis in Chapter 10, the mapping arises in the form of the roles fulfilled by UAV as part of the mission, which map to particular services in the DG. As the UAVs can switch roles if needed, in that study the impact mappings are dynamic; more detail on this is given in Section 10.4.

As already discussed with reference to Fig. 6.1, with the help of the impact map, the production model is able to process an attack outcome $o_i$ into a performance impact $i(o_i)$. The performance impact $i(o_i)$ is a time series of what the system production performance is given the attack outcome $o_i$, considering the system structure and recovery capabilities included in the production model.[1]

[1]If the same system is used to process multiple job types, such as in the case of the e-commerce case study in Chapter 8, then $i(o_i)$ returns a $t_{max} \times j$ vector, where $j$ is the number of job types and $t_{max}$ is the final time period.

## 6.2 Expected impact

In general, the expected impact over various outcomes is evaluated as follows. Let random variable $X$ represent attack outcomes, and denote by $i(o)$ the impact value of attack outcome $o$, and by $Pr(o)$ the probability of $o$. Then, the expected impact (when $X$ is a discrete random variable) is calculated as:

$$E(i(X)) = \sum_{i \in |\Omega|} Pr(o_i) * i(o_i) \tag{6.1}$$

where $\Omega$ is the sample space (i.e. the set of all outcomes) of $X$. Equation (6.1) looks deceptively simple as it hides the fact that the performance impacts $i(o_i)$ are not scalars but contain performance values over time, and thus suppresses the details on how the time dimension is dealt with. In what follows, we use (6.1) as the general pattern in which the calculation is done, but the exact way in which the time aspect is handled, and thus the exact calculation of the expected impact, can vary due to the specific evaluation purpose and modelling formalism used. We now explain how the expected impacts are calculated in the three sample implementations of our methodology in Chapters 8-10.

In our redundancy planning work (Chapter 8), we want to assess the impact of a set of attack scenarios $A$, and therefore the set of outcomes $\Omega$ considered in the expected impact calculation contains the outcomes of all scenarios in $A$. This means that the probabilities given to each outcome in $\Omega$ must also reflect the relative weights of the scenarios in $A$.

We calculate the probabilities of the different outcomes in $\Omega$ in a hierarchical manner, as shown in Fig. 6.3. The levels of this hierarchy are as follows: **1.** The set of **attack scenarios** $A$ considered in the analysis, which are assigned weights $w_a$ for $a \in A$ based on their relative likelihoods; **2.** A given attack scenario $a$ can split into **scenario variants** $V_a$ for $a \in A$, in two cases: a) if there are various orderings of attack steps that can be used for achieving scenario $a$, e.g. if $a$ contains multiple distinct targets to reach (in AG nodes), then the different orders in which these are reached form variants; b) when considering the effect of redundancy with diversity, obtaining the privileges to the diverse servers requires a different vulnerability exploit than the standard servers, and thus this adds variants reflecting whether the attacker has capabilities to exploit both standard and alternative server types or only one of them. When variants occur, we address these cases under the scenarios they relate to, so

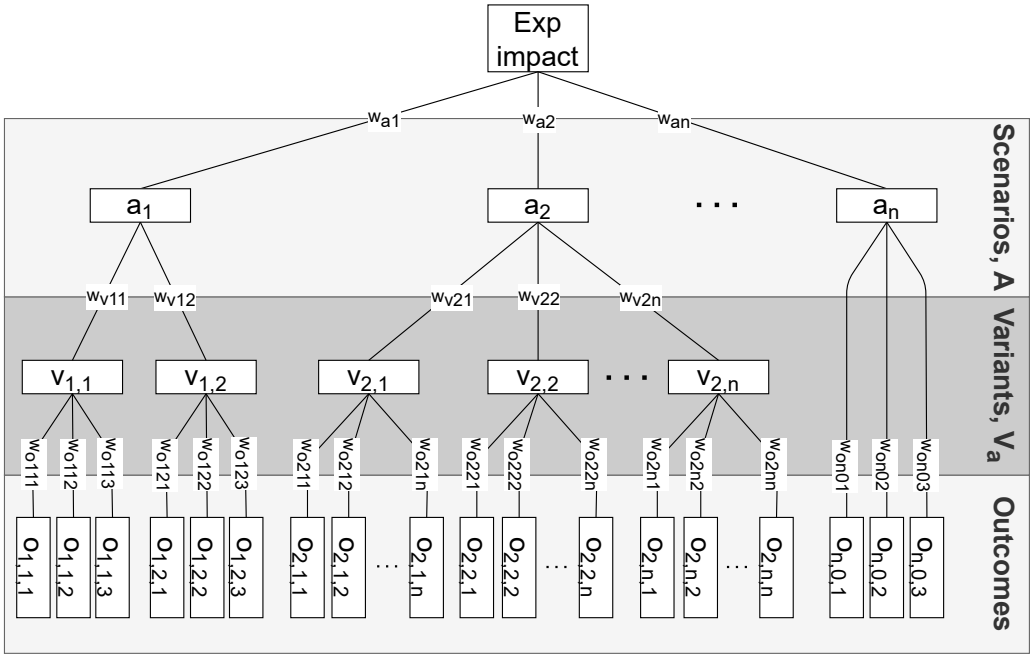



Figure 6.3: Hierarchical probability weight assignment with multiple attack scenarios and variants

$\sum_{v \in V_a} w_v = w_a$ for $a \in A$; **3.** Each attack scenario variant (or a scenario itself, if no variants exist) is split into different **attack outcomes** $O$, representing different levels of success. The outcomes are enumerated based on how far the attacker progresses along a given scenario variant before being stopped, either by the defence or a lack of capabilities required. For example, in the case study in Chapter 8 the different outcomes occur due to the defence detecting an attack step and stopping the attacker progressing further. Their weights in the probability calculation are under the relevant scenario (variant), e.g. $\sum_{o \in O_v} w_o = w_v$ for $v \in V_a, a \in A$. The weights $w_o$ for outcomes in $O_v$ relating to variant $v$ are determined based on the likelihoods of the different outcomes occurring for attack variant $v$. For example, in the resilience planning work, an attack variant splits to different outcomes due to detection, so the outcome weights are calculated based on the probability of detection; more details on this specific implementation are given in Section 8.5.3.

With the weights applied in this hierarchical manner, we can then find the probabilities $Pr(X)$ for the outcomes $o_i \in \Omega$ using the weights in the hierarchy levels, i.e.

$$Pr(o_i) = w_{o,i} * w_{v,i} * w_{a,i} \tag{6.2}$$

where $w_{o,i}$, $w_{v,i}$ and $w_{a,i}$ are the outcome, variant and attack weights that relate to outcome $o_i$. These

probability values could then be used to calculate the expected impact using equation (6.1).

If the intent in the use-case is to evaluate an *expected cost*, as is the case in our redundancy planning work, then we must evaluate the cost impacts of attack outcomes. For this purpose, the performance impacts for the different outcomes $i(o_i)$ are passed onto the cost model, as shown in Fig. 6.1, where the cost impacts $g(o_i)$ are evaluated using a model for valuing production disruptions (explained in Section 7.1). The expected cost $E(g(X))$ is then calculated similarly to the expected performance impact in equation (6.1), but with cost impacts $g(o_i)$ instead of performance impacts $i(o_i)$, and the result is a monetary-valued estimate of the cost that is expected to arise due to the attack scenario.

In the model in our countermeasure selection work in Chapter 9, the countermeasure selection analysis is intended to be run reactively during an attack. Thus, expected impact calculations are done each time a countermeasure choice is considered, and applies to the impact over the attacker's actions in the next $k$ time steps, relative to the current situation in terms of the extent of an attack. The outcomes $\Omega$ reflect the paths the attack could reach in that time window of $k$ time steps, from no progress at all up to the outcomes where the attacker attempts and succeeds in all its chosen moves that can be completed in the time given. The expected impact calculation is applied separately at each time step up to $k$, to find the expected impact over time. A more detailed description of this is in Section 9.4.

To estimate the effect of a countermeasure, a comparison of the expected impact is made between the cases where the countermeasure is applied, and where it is not. Further, as the countermeasure can have short term costs (from causing unavailability, or direct costs related to applying it) but permanent benefits, we also consider the impact the countermeasure would have in future attacks. Therefore, four sets of expected impact calculations are required to estimate the benefit from applying a countermeasure, two for considering the current attack, and two for future attacks. Further detail of how we do this is given in the discussion of cost modelling considering future periods in Section 7.3, and in the description of the countermeasure selection work in Chapter 9.

In the mission viability analysis (Chapter 10), the expectations of our measures of interest are evaluated by the CTMC solution of the SWN model, using the probability distributions of the transitions. Thus, we do not separately enumerate the possible outcomes and evaluate their individual probabilities. To be specific, we use the transient solution of the CTMC of the SWN model to solve over

time for the expectation of the number of tokens in places which represent UAVs fulfilling their roles. This shows how the number of functional UAVs in particular roles changes during an attack scenario, on average, given the probability distributions of the different transitions. The impact estimation is then done using the production function approach in Section 5.2.2 based on the time series of these measures, i.e. based on the expected numbers of functional UAVs in particular roles over time.

## 6.3 Impact assessment evaluation outputs

The output of the attack impact assessment, in terms of production performance, could be:

1. A *set of performance curves over time* for individual attack outcomes, i.e. impacts $i(o) \forall\, o \in \Omega$, to be passed on to cost analysis. This is relevant when the cost of an impact depends on system performance meeting or failing to meet requirements such as service level agreements (SLAs). In such cases, the expected impact calculation (6.1) cannot be used on performance impacts, as the average impact thus calculated can be above the required level and suggest no penalties, while individual outcomes cause penalties. Thus expected impact must be calculated on *cost impacts* $g(o)$, not performance impacts. This is the case in our redundancy planning work.

2. A *time-series of expected performance impacts* at each point in time. This is what is used in our countermeasure selection work to choose a countermeasure, and in the performance evaluation in our mission viability work.

3. The *expected performance impact, measured cumulatively over the duration of the attack*, i.e. translated into a single value representing the performance impact due to the attack (as opposed to a time-series in case 2 above). Examples of this could be the expected values of e.g. cumulative production until recovery, or the value of a resilience metric such as discussed in Section 2.1. This approach was not taken in our use-cases, but could be useful in some applications.

# Chapter 7

# Cost modelling

A cost model is required for quantifying the losses due to the disruption, as well as the direct costs (acquisition and maintenance) of investments into defensive actions such as countermeasures or redundancy. To enable analysing the impacts of attacks and defensive actions in monetary terms, as is intended as part of our methodology, the cost modelling needs to consider the following aspects:

- the value of lost output/low performance due to production disruptions (whether from attacker or defensive actions)
- direct costs of actions, e.g. costs of defensive actions such as the price of replacement components and installation
- costs and benefits occurring across future time periods (future benefits from patching, increased future maintenance costs from redundancy etc.)
- other costs arising due to an attack event, e.g. loss of customers due to reputation loss

The cost of disruption due to an attack is quantified based on the value of production lost (e.g. services not delivered) due to the disruption. We consider two ways in which this loss of production can lead to money-valued costs: costs varying smoothly with production loss (”smoothly-varying” costs), and costs due to failure to meet a limit value for required performance (”trigger-level” costs). Which form is applied depends on how the disruption costs incur for the system of interest. In the case studies in Chapters 8-10, we provide an example for both types: In our work on countermeasure selection, the

loss is evaluated by setting a value for the final service provision under normal conditions, and partial functionality reduces the value of output in proportion to the performance lost relative to the normal situation. This reflects lost business, such as transactions lost. In our work on redundancy planning, we consider a case study where the service provision is subject to a service level agreement (SLA), and the cost of disruption is based on penalties for failing to meet the SLA.

While our main focus is on attacks with production impact (business interruption), we acknowledge that even these kinds of attacks can cause other types of costs, e.g. customer loss due to reputation effects. Where such costs unrelated to output performance are of concern, an estimation of them could be done using appropriate system and organisation specific assumptions and added alongside the main cost estimation. An example of how this could be done is given in Section 7.4, where we propose two ways of modelling the costs due to loss of customers, one where any successful attack leads to a constant share of customers lost, and another where the number of customers lost varies with the size of the disruption from the attack according a sigmoid function.

The rest of this chapter discusses the four aspects of cost modelling listed above, and explains how we have addressed them in our model implementations.

## 7.1 Valuing production disruptions

Disruptions to system components reduce the output performance of the system, which can lead to costs in terms of production losses, or due to failure to meet a performance requirement. This section discusses how we estimate the cost of the performance impact from a given attack outcome, first in the case where the costs vary smoothly with the level of production performance ("smoothly-varying costs"), and then in the case where the costs arise due to failing to meet a specific performance requirement ("trigger-level costs").

The key piece of notation used in this section is the cost impact of an attack outcome $o$, which we denote by $g(o)$. While this section only discusses this in terms of attack outcomes, it is important to explain what the cost impact is considered to contain when there are defence actions associated with the outcome. That is, $g(o)$ is used to represent all production-impact costs relating to a given

outcome $o$, which consist of the performance impact of the attack steps included in the outcome, recovery related performance effects and costs[1], and the costs of performance impacts arising from any responses to the attack steps that form a part of the outcome. For example, where a defence action $a$ (such as a countermeasure) is applied, then the cost impact of an outcome which occurs (conditional on $a$) contains also the cost of any performance impacts that may arise as a direct consequence of $a$ being applied. Note also that where a recovery action is part of the outcome (which is typically the case, to return to normal performance levels), then not only is its impact on the performance considered, but also any other costs that might be associated with the recovery action.

### 7.1.1 Smoothly-varying costs

In cases where the production performance is directly linked to the economic performance of the organisation, e.g. if sales transactions not fulfilled due to a disruption are permanently lost (e.g. a competitor fulfils them instead), the cost of disruption varies smoothly with the level of performance. In this case, the cost due to a production disruption can be estimated based on the profit that would be made under normal production performance, and the deviation from this normal performance observed under the attack scenarios.

**The general approach:** To express the evaluation in general terms, the cost of production disruption due to attack outcome $o$ in a time window containing $t_h$ time units is:

$$g(o) = \sum_{\tau=1}^{t_h} (\pi_{ref}(\tau) - \pi(o, \tau)) \tag{7.1}$$

where $\pi_{ref}(\tau)$ is the reference value of profit from the production process expected at time $\tau$, and $\pi(o, \tau)$ is the profit at time $\tau$ if attack outcome $o$ occurs. This simply states that the cost is the deviation in profit from its expected level that occurred due to outcome $o$, cumulated over time up to the time horizon $t_h$. With the assumption that the profit from production varies smoothly with

[1] Recovery effects are considered as part of $g(o)$ as they are an essential part of determining the set of possible outcomes from an attack, e.g. a given attack path could create various different outcomes if recovery was applied differently.

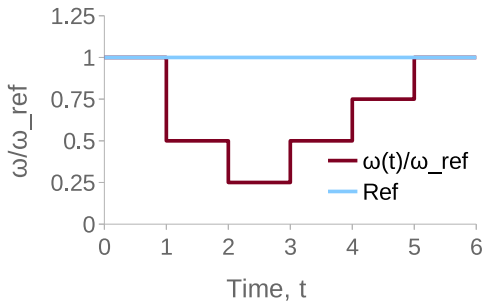



Figure 7.1: Cost impact calculation based on normalised output

production performance, we can express this in more detail as:

$$g(o) = \sum_{\tau=1}^{t_h} \pi_{ref}(\tau) \left(1 - \frac{\omega(o,\tau)}{\omega_{ref}(\tau)}\right) \tag{7.2}$$

where $\omega(o,\tau)$ stands for system output performance under attack outcome $o$ at time $\tau$, and $\omega_{ref}(\tau)$ is the reference level expected if no attack occurs. Further, if we assume that the reference performance and reference level of profit are constant over the time periods, we can express this as:

$$g(o) = \pi_{ref} \sum_{\tau=1}^{t_h} \left(1 - \frac{\omega(o,\tau)}{\omega_{ref}}\right) \tag{7.3}$$

where $\frac{\omega(o,\tau)}{\omega_{ref}}$ is the normalised performance, as often used for performance-curve based resilience analysis, e.g. by [GMG+16]. The performance impact is illustrated in Fig. 7.1, where we assume that attack $o$ occurs at time $t = 1$, system performance drops until $t = 2$ before recovery starts, and performance recovers to the reference level at $t = 5$. The area between the curves, the cumulative performance loss, is $\sum(1 - \omega(o,\tau)/\omega_{ref}) = 0.5 + 0.75 + 0.5 + 0.25 = 2$, so the loss of profits, as given by (7.3), is $g(o) = 2\pi_{ref}$.

This is essentially the approach we use in our work on countermeasure selection (Chapter 9), except that we consider several final outputs in that work. That is, we express the loss due to the unavailability of a system component in terms of the value of final services' production lost because of it. Expressing this using the general notation given above, this adds a summation over final service outputs:

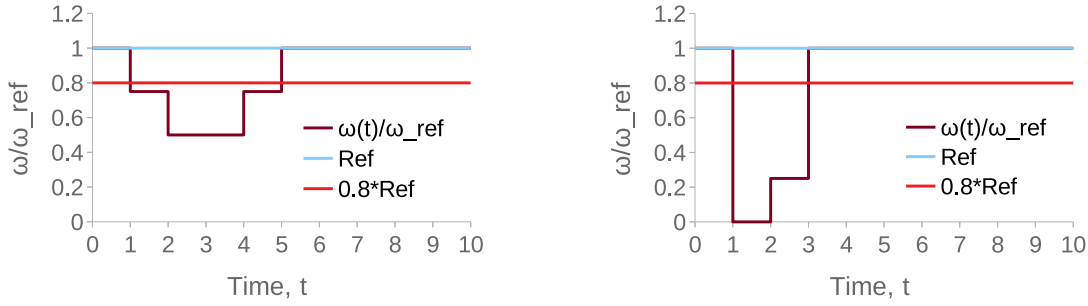
$$g(o) = \sum_{h_j \in H_S} \pi_{ref}(h_j) \sum_{\tau=1}^{t_h} \left(1 - \frac{\omega(o,h_j,\tau)}{\omega_{ref}(h_j,\tau)}\right) \tag{7.4}$$

where $h_j$ is a final service node in the set of final service nodes $H_S$, $\pi_{ref}(h_j)$ is the profit from the output of service $h_j$ (given by utility value $u(h_j)$ in the paper), and $\omega(o, h_j, \tau)$ and $\omega_{ref}(h_j, \tau)$ are the observed output of service $h_j$ under attack outcome $o$ and in the reference case, respectively. Note that the exact form of this calculation used in the paper (and in Chapter 9) differs somewhat from the above, as it is aligned with the specific modelling approach used, while the above expresses it in the context of the general approach.

### 7.1.2 Trigger-level costs

Costs that occur due to performance dipping below some trigger level can arise for contractual or regulatory reasons, e.g. service level agreements (SLAs), or due to process specific requirements where system functioning depends on maintaining some critical level of performance. There are two dimensions to this, the required level of performance, and the share of time the system is allowed to breach this level without a cost being triggered.

Figure 7.2 illustrates how this type of cost is formed by the way of two simplified example cases of disruption. In both cases, the red horizontal line shows a trigger level set at 80% of normal performance. In Fig. 7.2a, the performance drop is shallow but lasts for four time periods, while Fig. 7.2b shows a much more disruptive attack outcome, but which is recovered within two time periods. With a smoothly-varying cost, the first case would be preferable, as the overall production loss is smaller. However, with a cost determined by a trigger-level, the situation can flip around as the second outcome is below the trigger-level for a shorter time, or both outcomes can yield the same amount of penalty. The issue comes down to how long the system is allowed to be in breach of the required performance (trigger-level) until a penalty is applied (or another loss occurs). We shall call the share of time that a breach of the trigger level is tolerated the *disruption tolerance* of the SLA, and denote it by $\beta$. Assuming that the 10 time periods shown in the figure reflect the reference duration for the penalty (i.e. $\beta$ is measured against it), then if the disruption tolerance is below 20% of the overall time, i.e. $\beta < 0.2$, then the cases in both Fig. 7.2a and Fig. 7.2b lead to triggering the cost. If $0.2 < \beta < 0.4$, the first case leads to a penalty while the second does not, and if $\beta > 0.4$, neither case causes the penalty to be triggered.




(a) Limited disruption, slow recovery

(b) Extensive disruption, fast recovery

Figure 7.2: Sample cases of performance relative to a trigger-level

To express this using the same notation as above, with trigger-level costs, the cost impact $g(o)$ is:

$$g(o) = \mathbb{1}_{DT(o)>\beta} \cdot L \tag{7.5}$$

where $L$ is the monetary valued loss due to the breach (e.g. a penalty), and $DT(o)$ ("disruption time") is the share of time that performance is below the required level due to outcome $o$, and $\beta$ is the limit value, as a share of time, after which the disruption causes the cost to be triggered. The binary indicator function $\mathbb{1}_{DT(o)>\beta}$ has a value of 1 if $DT(o) > \beta$ holds, and is 0 otherwise. The disruption time is measured simply as:

$$DT(o) = \frac{\sum_{\tau=1}^{t_h} \mathbb{1}_{\omega(o,\tau)<\omega_{req}}}{t_h} \tag{7.6}$$

where $\omega(o, \tau)$ is the system output performance under attack outcome $o$ at time $\tau$, and $\omega_{req}$ is the required performance level ("trigger level").

We have modelled this type of a cost in our redundancy planning work in Chapter 8, in the form of penalties due to breaching the conditions of an SLA. As in the general description above, the costs of failing to meet the SLA depend on the extent to which the system deviates from a required performance: how much below required, and for how long. We use a performance based SLA, similar to Oracle's [Ora], with average throughput $X$ as the performance metric. Other types of SLAs exist, many of which are based on service availability, where the SLA condition only considers time spent with the service fully unavailable.

In the case study in Chapter 8, our SLA has only one credit level, for 100% service credit to be

provided to the customer if the condition is breached.[2] Due to the specifics of the case study, the notation and the exact form of the calculation shown in (7.5) differ somewhat from the general case described above. We shall now use the notation from the case study to show how SLAs relate to the above. The SLA breach condition is: *100% service credit if* $X(t, x, M) < 0.9 \cdot X_{ref}$ *for a disruption time greater than* $\beta$ *in a month.* Here, $x$ is a server allocation vector, $M$ is an attack outcome matrix, and $X(t, x, M)$ is the system throughput at time $t$ observed when server allocation is $x$ and the attack outcome $M$. $X_{ref}$ is the reference throughput that has been marketed to the client (the published performance), and $X_{req} = 0.9X_{ref}$ the required throughput. The 90% limit is as in the performance SLAs in [Ora]. $\beta$ is the limit value (as a share of time in a month) that performance can be below $X_{req}$ before the SLA is breached. The *SLA penalty cost* is calculated similarly to (7.5) above:

$$SLAp(x, M) = \mathbb{1}_{DT(x,M)>\beta} \cdot N \cdot cc \tag{7.7}$$

where $DT(x, M)$ is a disruption-time share (defined below), $\beta$ is as described above, $N$ is the number of clients, and $cc$ is the client charge for the overall service. The equation states that a penalty of $N \cdot cc$ is to be paid if $DT(x, M) > \beta$, otherwise the penalty is 0. $DT(x, M)$ is defined as:

$$DT(x, M) = \frac{\sum_{\tau=1}^{t_h \cdot S} \mathbb{1}_{X(\tau,x,M)<X_{req}}}{t_h \cdot S} \tag{7.8}$$

where $s \in S$ is a sub-unit of time used in the performance modelling, with each time unit $t$ split into $S$ fractions. $DT(x, M)$ is the share of time during which throughput is less than the required throughput, $X_{req} = 0.9X_{ref}$, out of all time periods up to the time horizon $t_h$. The time horizon is set to the SLA service commitment period, one month.

While the above example relates to SLAs, equations (7.5) and (7.6) could be used for other situations where a trigger level for performance is observed. For example, if a system has a critical level of performance that should be maintained, then the required level $\omega_{req}$ would be assigned this critical performance level, and the parameter $\beta$ set to 0 to show that any drop below $\omega_{req}$ is to be avoided. The form of the cost as a function of attack outcome $o$ would be the same as equation (7.5), but with

[2]The case of multiple credit levels would make the cost impact $g(o)$ a summation over $I$ terms with different penalty (loss) amounts $L_i$ and limit values $\beta_i$, where $i \in I$ and $I$ is the number of credit levels considered.

the cost level $L$ reflecting an incident cost/penalty that relates to the system in question.

## 7.2 Costs of defensive actions and resilience improvements

The costs of actions by the defence can consist of a combination of costs directly relatable to the action, and costs arising due to its impact on production. When production impacts occur, the cost is dependent on what the system state is when the action is taken. Therefore, we denote the cost of a generic action $a$ in state $s$ by:

$$c_a(s) = c_{a,d} + E(g(X_{a,s})) \tag{7.9}$$

where $s$ is the system state at the moment before the action $a$ is taken, $c_{a,d}$ is the direct cost of the action not related to production impacts, and $E(g(X_{a,s}))$ is the expected cost impact of an attack scenario conditional on the defence action $a$ applied in system state $s$. The notation $X_{a,s}$ is used to denote a random variable whose outcomes are attack outcomes $o_{a,s}$ among the set of possible outcomes to the ongoing attack, given system state $s$, if an action $a$ is applied. Note that, as part of the cost impact $g(o_{a,s})$, we also consider any impact the action $a$ may itself have on the system performance. The system dependence occurs because an action that might be disruptive in one state of the system may not be so in another, for example, if a service is already unavailable, then actions that would otherwise cause that service to be temporarily unavailable may yield a reduced production impact or none at all.

The expected cost impact $E(g(X_{a,s}))$ is calculated as:

$$E(g(X_{a,s})) = \sum_{o_{a,s} \in \Omega_{a,s}} Pr(o_{a,s}) * g(o_{a,s}) \tag{7.10}$$

where $g(o_{a,s})$ represents the cost impact of an attack outcome $o_{a,s}$ among the set of possible outcomes conditional on the action $a$ applied at system state $s$, and $Pr(o_{a,s})$ is the probability of outcome $o_{a,s}$.

Not all defensive actions involve both types of costs, e.g. the cost of disabling communications would likely only involve costs from production disruption, while capability improvements such as adding redundant components could come with (practically) no production impact when applied. In our application case studies, we modelled the costs of a defensive action with both direct and

production impact costs (patching action) in the countermeasure selection work (Chapter 9), and resilience capability changes with only direct cost components (redundancy and diversity) in Chapter 8. In the countermeasure selection work the evaluation of the cost of patching follows the approach given in equation (7.9) closely, although the notation used to describe it (in Chapter 9) differs from the above in giving more detail on the specifics of the model and the case study.

Regardless of whether the action itself causes disruption, it will still have an effect on the outcomes of an attack, either in terms of limiting the performance impact, or by affecting the paths an attack can take, or both. As a consequence, a performance impact term $E(g(X_{a,s}))$ is always a part of the effect associated with an action $a$, although this term could be negative (suggesting a benefit, not a cost). We have chosen to include it as part of the cost $c_a(s)$ in (7.9), to keep the approach consistent and avoid creating added notation for tracking overall monetary impacts separately from costs. Note that even in the cases where an action $a$ causes an impact, evaluating this cost impact must be done in the context of the attack scenario (or scenarios) involved, as the production impact from the action differs based on the attack outcomes considered (e.g. a patching action always causes disruption when an attack is not occurring, but may not cause added disruption during attack).

Improvements to the *resilience capabilities* of a system, such as component redundancy, are applied during planning, before an attack event takes place. As a consequence, there is no attack to consider when such improvement actions are applied, but the action will still have an impact on system performance during an attack, and this must be considered to determine the cost impact of the action. Thus, $E(g(X_{a,s}))$ must be evaluated, and the set of outcomes to include in the evaluation is based on what attack scenarios are plausible given the action.

The direct cost of an action may be expressed in more detail if the application requires it. For example, in our redundancy planning work in Chapter 8, the direct cost part of redundancy and diversity actions is the cost of server capacity (in terms of the number of servers). We express this capacity cost as[3]:

$$c_{c,d} = \sum_{i=1}^{N} (c_i \cdot x_i + c_{A,i} \cdot x_{A,i}) + C \tag{7.11}$$

[3]Note that here we denote the capacity cost by $c_{c,d}$ for consistency of notation with the general action costs discussed in this section, while in the original work (and in Chapter 8) we denote it by $\gamma$ instead of $c_{c,d}$.

where $x_i$ is the number of regular servers used for a given service (or cluster) $i$ and $c_i$ is the per-unit cost for a regular server (including server purchase/rental and maintenance costs). The alternative servers used for diversification are assumed to have a higher cost $c_{A,i}$ per server, and $x_{A,i}$ is the number of such servers. $N$ is the number of separate services (server clusters) in the system. The last term $C$ is a constant reflecting any fixed costs of capacity and/or diversification, i.e. costs that do not vary with the number of units.

As costs such as maintenance costs accrue over time, the per unit cost terms $c_i$ and $c_{A,i}$ involved in (7.11) must be expressed in terms of costs over an appropriate time period. For example, these could represent the costs that accumulate over the model time window up to the time horizon $t_h$, or they could represent the whole cost over the expected lifetime of the components. In our case study in the resilience planning work, we do not want to cumulate maintenance costs over the full server lifetime, but want to be able to separate the costs accruing during time periods when attacks occur from those relating to normal times. This enables us to form a metric (TUL, discussed in Section 7.3.2) relating the benefit of a server allocation during attack periods to the costs that accrue over time. For this purpose, we measure the costs $c_i$ and $c_{A,i}$ as monthly per-unit costs (equivalent monthly cost), as the window of time modelled is one month (corresponding to the SLA commitment period).

## 7.3 Valuing future impacts

When evaluating the cost effectiveness of defence actions that have an effect that lasts for longer than one attack event, the modelling must consider the impact in future periods to form a picture of the overall costs and benefits. For example, a patching action may cause disruption when applied and therefore a production loss, but has a positive impact that can help not only during the current attack but also against future attack attempts. Further, component redundancy and diversity have a lasting benefit but come with an added cost which must be covered whether attacks occur or not. As the benefits of these types of actions or improvements are felt with every attack attempt, the modelling must somehow address the *frequency of attacks* that can be expected, to determine if the benefit is worth the cost associated with an action or improvement.

We have approached the problem of valuing impacts occurring during future periods in two ways: **1.** using current and long-run "system trajectories" to evaluate the impact of a defence action on attack outcomes for both the current and future attacks, with the results combined for forming an estimate of the benefit of the action; **2.** forming a metric ("time until loss", TUL) that measures how frequent attacks would need to be to justify a resilience capability investment.

### 7.3.1 Countermeasure effect based on short and long-run "trajectories"

When a defensive action has both short and long-run impacts to system performance, the cost/benefit assigned to the action must consider both the current and future impacts. For example, patching a particular vulnerability in a system can lead to temporary system downtime, but makes the system more robust against future attacks.

In our countermeasure selection work (Chapter 9), the cost effectiveness of each countermeasure is evaluated by comparing its cost, including from overall expected performance impact, to the case where no countermeasure is applied. Thus, our approach makes a comparison between two potential evolutions ("trajectories") of the system: the "baseline trajectory" reflects how an attack would be expected to proceed within the network in the absence of the countermeasure, and its cost (denoted $E(g(X_{0,s}))$, as in equation (7.10) but with $a = 0$ reflecting the lack of action) includes what the impact would be in terms of costs due to availability loss, including the cost of recovery actions that would occur as part of the outcomes. The "deviating trajectory" given a countermeasure action measures the expected cost impacts when the countermeasure is applied ($E(g(X_{a,s}))$ in equation (7.10)). Further, as the impact of an action (countermeasure) can differ in the short and long term, e.g. patching requiring a component to be taken temporarily offline, we make a difference between the immediate and the longer-run impacts of the countermeasure.

The benefit of an action is the sum of two parts: its cost relative to taking no action during the current attack, and the cost reduction it would confer in future attacks, multiplied by the expected number of

future attacks within the planning horizon. This is given by:

$$B(a,s) = (c_0(s) - c_a(s)) + eaf(a) \cdot t_{ph} \cdot \left(c_0(s) - c_a^{LR}(s)\right) \tag{7.12}$$

where the term $c_a(s)$ is the cost of an action $a$ at state $s$, as in equation (7.9), and $c_0(s)$ represents the cost of no action at state $s$. The term $c_a^{LR}(s)$ is the cost of the action in the long run, after any temporary effects of applying the action have run their course. The number of future attacks is given by the multiplication of $eaf(a)$, the expected frequency of future attacks against which $a$ is effective, and the length of time until the end of the planning horizon, $t_{ph}$.

The benefit can be written in terms of the different components of costs, using equation (7.9), as:

$$\begin{aligned} B(a,s) = &(E(g(X_{0,s})) - c_{a,d} - E(g(X_{a,s}))) \\ &+ eaf(a) \cdot t_{ph} \cdot \left(E(g(X_{0,s})) - E(g(X_{a,s}^{LR}))\right) \end{aligned} \tag{7.13}$$

where $X_{a,s}$ is a random variable whose outcomes are system impact outcomes of an attack $o_{a,s}$ that are possible given that the action $a$ is applied at system state $s$; $X_{0,s}$ is similar notation for the case when no action is applied. $X_{a,s}^{LR}$ is used to denote the outcomes that can occur in the long run at state $s$ after action $a$ is applied, excluding any temporary system impacts from the action $a$.

Further, the form shown in equation (7.13) can be expressed equivalently using only the monetary values of performance, which we denote $V(o)$, instead of the cost $g(o)$. The value $V(o)$ is related to the cost $g(o)$ as follows: $g(o) = V_{ref} - V(o)$, where $V_{ref}$ represents the value of system output at reference performance. Due to this relation, when calculating differences between the baseline and deviating trajectories, we find the following equality:

$$\begin{aligned} (c_0(s) - c_a(s)) &= E(g(X_{0,s})) - c_{a,d} - E(g(X_{a,s})) \\ &= V_{ref} - E(V(X_{0,s})) - c_{a,d} - V_{ref} + E(V(X_{a,s})) \\ &= E(V(X_{a,s})) - E(V(X_{0,s})) - c_{a,d} \end{aligned}$$

In the implementation of the benefit calculation in our countermeasure selection work Chapter 9, the benefit calculation is done using the equivalence shown above. In that work, the benefit arising from

a countermeasure is expressed as:

$$\begin{aligned} B(v_i, s) = E(V(X_{v_i,s})) - E(V(X_{0,s})) - c_{cm} \\ + eaf(v_i) \cdot (t_h - t) \cdot (E(V(X^{LR}_{v_i,s})) - E(V(X_{0,s}))) \end{aligned} \tag{7.14}$$

where $v_i$ the node in the AG (representing a vulnerability) for which a countermeasure is considered, e.g. the current head of the attack path, or one determined to be risky. $E(V(X_{v_i,s}))$ is the expected monetary value of system output arising from the outcomes of $X_{v_i,s}$, i.e. attack outcomes given a countermeasure applied to AG node $v_i$ at system state $s$. The direct countermeasure cost is $c_{cm}$. The current time period is $t$, and $t_h$ is the last time period in the planning horizon considered.

In the implementation in our countermeasure selection work the expected attack frequency is approximated using the term $eaf(v_i)$, which is an estimate of the probability that an attacker will attempt to exploit node $v_i$ again. This is not based on the current compromise state, but on the probability to (re-)obtain the privileges for exploiting $v_i$ in the future via any path. We estimate $eaf(v_i)$ by approximating the probability of the shortest viable (not patched) path from the attacker node $A$ to $v_i$. The detail on how this is done in our countermeasure selection implementation is given in Section 9.4.

While the evaluation involves calculating various attack outcomes and their probabilities to formulate the trajectories, we do not solve for the whole network or the full time horizon, as such an approach would not scale to large network sizes. Instead, the method focuses on "neighbourhoods" of the current attack, by looking at potential states a few time-steps forward relative to the current "boundary" of the attack.

Our method described above accounts for the future in countermeasure selection by estimating the monetary benefit derived from the countermeasure during future attack events, and the expected number of times such events would be observed. This approach provides an estimate of longer term impact that is considerably more efficient to calculate than directly modelling all possible states in each future period over a long simulation time window. A weakness of the method is that it considers future impacts relative to the current situation, so excludes the impact of changes to the network or the environment that might happen in the future. However, such information is unlikely to be available at the time when the decision is to be made. In the case it was, it should be possible to establish the robust-

ness of the estimate to such changes, e.g. by calculating the impact of the countermeasure conditional on other changes being applied as well. However, we have followed a greedy strategy looking for the impact of the countermeasures independently, for the sake of tractability and scalability, especially as the evaluation is intended to be run when an attack is ongoing.

### 7.3.2 "Time until loss" metric

Resilience improvements that form a part of the system at all times, even if only actively beneficial during adverse events such as cyber attacks, can involve costs that accrue over time whether an event occurs or not. Thus, decisions over such improvements, and the extent to which they are applied, must consider how their impact during attack periods (in terms of cost saving achieved) relates to the costs they accumulate over time. A key part of this estimation is again the expected frequency of attacks, as the benefits occur during attack events, and the more attack events there are, the more viable the investment into the improvement becomes. Thus, at its core the problem is similar to that of comparing the costs and benefits from short term and long term impacts in the previous section, but here we propose another way of approaching it: instead of estimating a frequency of attacks based on parameters, which ultimately relies on assumptions about attacker choices and capabilities, we form a metric that shows what the frequency of attacks should be for a given resilience improvement to be cost effective relative to a reference point in which the improvement is not applied.

The metric, which we call "time until loss" (TUL), is as follows:

$$TUL(a) = \frac{c^A_{ref} - c^A_a}{c^m_a - c^m_{ref}} \tag{7.15}$$

where $c^A_a$ refers to the expected cost during a period when an attack occurs when improvement action $a$ is applied, and $c^A_{ref}$ is the cost with the reference setup; $c^m$ refers to maintenance costs (paid per each unit of time) whether an attack occurs or not, with $c^m_a$ denoting maintenance costs if action $a$ is applied, and $c^m_{ref}$ the maintenance costs associated with the reference. The numerator is the difference in expected costs during attack periods, i.e. the benefit of action $a$ over the reference during attacks, and the denominator measures the excess maintenance costs due to $a$ relative to the

reference. The numerator is measured in monetary value, and the denominator as monetary value per unit of time, and thus the metric is measured in terms of the time unit used for the maintenance costs in the denominator. The metric reflects the number of time units until the excess maintenance costs due to action $a$ exceed the benefit obtained from it during an attack (relative to the reference point used). Therefore, the metric is used to determine the cut-off attack frequency at which investing into $a$ switches from being financially justified to not: if the expected time between attacks is longer than the value of the metric, then investing in $a$ loses money relative to the reference.

The advantage of this metric is that it can be directly related to how frequently an organisation expects to be targeted by attacks, but avoids requiring a frequency of attacks as an input to analyses. Instead, analyses can be done and their results then compared to various values for expected attack frequency. Further, the metric enables forming optimisation problems to find an action that maximises the TUL metric, or ones that minimise overall cost conditional on a specific level of TUL.

To give an example of how the metric can be implemented, in our work on redundancy planning, discussed in Chapter 8, TUL is applied for the case of choosing the level of redundancy (including diversity) to apply to different services that form a part of a production process. In our case study used for that work, we measure TUL in months, as this coincides with the SLA commitment period considered in the case study. The actions $a$ considered in the work are different allocations of servers to services in the system, represented with a server allocation vector $x$, and thus the specific form of TUL used is:

$$TUL(x) = \frac{\phi(x_{ref}) - \phi(x)}{c^T x - c^T x_{ref}} \tag{7.16}$$

where $\phi(x) = c^T x + \sum_{M \in A} w_M \cdot SLAp(x, M)$ is the cost during an attack period for server capacity vector $x$, $c$ is a vector of cost coefficients applying to $x$, and $x_{ref}$ is the server capacity vector for the reference allocation. The numerator includes the overall cost during a month with an attack, while the denominator only concerns the maintenance costs for the allocation $x$.

## 7.4 Other attack costs

While our approach focuses on attacks that can impact system production, cyber attacks can also lead to various costs that may not be in proportion to the impact on production or the defensive actions taken. Examples of such effects are the value of business lost due to reputation impacts, and the value of data lost in a data breach, e.g. the business value of sensitive information lost, or a regulatory penalty due to lost customer data. Although we focus on losses due to business disruption, in cases where such additional costs are relevant to an attack scenario and could be mitigated by using the modelled defence actions, a way to estimate them is required when evaluating the benefits of defensive actions and resilience improvements, to avoid underestimating such benefits.

As there are various kinds of costs that are scenario and system dependent, we do not claim to have a single approach that can exhaustively capture all types of costs. Instead, in our work thus far we have proposed a way to model one important type of cost that does not have to occur in proportion to the performance impact of the attack, which is the loss of business due to reputational impact after a cyber attack event. As such reputation impact is not directly observable but is always an estimate, analyses that require the estimation of such business loss should test results for sensitivity to changes in the reputational impact assumptions over a range of values.

Our modelling of reputational cost (value of lost business due to reputation) relies on two key parameters: the *share of business lost* due to an attack, on average, which we denote by $\lambda$; the *duration of the the reputational impact* $t_L$, i.e. for how long the reduced level of business lasts before recovering. In addition, when the duration of the impact is long, the value of impacts after the first year are expressed in terms of their value in the initial year using net present value with discount rate $r$.

We consider two models for how the cost is applied: *constant loss share*, where any attack that breaches the system (even if not successful in its goals) causes a loss of $\lambda$ share of business regardless of the extent of the attack; *variable loss share*, where the share of lost business varies with the extent of the attack. In our implementation used in the resilience planning work, the variable loss share changes with the extent of production disruption caused, but it could also be based on e.g. the share of customers affected by a data breach. For example, in the redundancy planning work, we determine

the share of business lost using a logistic function with the following form:

$$f(y) = \frac{2\lambda}{1 + e^{-500(y-y_0)}} \tag{7.17}$$

where $f(y)$ is the share business lost (in the case study, the share of consumers lost) as a function of the size of the disruption $y$, measured by the "disruption-time share" as in equation (7.6). The logistic function shape parameters $L = 2\lambda$ (max value), $k = 500$ (logistic growth rate) and $y_0$ (mid-point of the curve) were chosen to yield a smooth sigmoid shape with values between 0 and $2\lambda$ (as $y \geq 0$), where the midpoint ($f(y) = \lambda$) is reached at a disruption-time share of $y_0 = 0.01$. This form provides smooth variation in the extent of business loss that limits the effect of small disruptions and caps the maximum share of business lost to twice the average loss share $\lambda$.

As the cost of business lost accrues over time, the length of time $t_L$ for which the number of customers will be dampened has a large effect on the overall cost effect. We assume that the impact of the attack on reputation will be temporary, so eventually customer numbers will return to the level that would have been expected in the absence of the attack. We estimate the value of the loss as the duration $t_L$ (in months) times the monthly loss of revenues due to the reduced level of business. To account for time discounting of the revenues (and losses) of future years, we discount the value of the losses during future years (from month 13 onward) at a rate of $r\%$ per year.

Consideration of "other attack costs" such as discussed above are important when they can impact the decisions that are modelled. For example, in the redundancy planning case, the overall cost of attack outcomes will affect the decision to invest in redundancy (with diversity), as if the maintenance cost of the added servers is a smaller share of the overall costs due to attacks, investment into the added capacity is more viable. While we have not modelled other instances of costs of this type, the approach described above can be modified to apply for other cases. Furthermore, when the extent of the cost is not dependent on the production performance, or other variables and parameters that are part of our modelling methodology, the costs can be modelled completely externally to the model. For example, the cost of data breaches could be estimated based on the number of records stolen, as done by [Dun18].

## 7.5 Chapter summary and discussion

In this chapter, we introduced the cost modelling part of our methodology for assessing cyber attack impacts. Cost modelling is needed for quantifying losses due to production disruption, making them comparable to cost of investments into defence capabilities and defensive actions. Further, expressing impacts in monetary units also enables meaningful comparisons of impacts across time.

Our cost modelling considers three classes of costs related to cyber attacks and defence: 1. The value of output loss due to attacks and defensive actions; 2. Direct costs of defensive actions; 3. Other costs arising due to attack events which do not directly impact production, but are significant in magnitude, such as reputational loss. We discussed how they relate to our attack impact assessment, and proposed ways in which they can be addressed in the cost model, with examples based on our case studies.

Future periods must be taken into account, both as benefits of investments to capabilities accrue over a long time, and because attack periods are exceptions, and the estimation must consider the impact (and costs) of measures when attacks do not occur. We proposed two approaches for this: comparison of short and long-term impacts of countermeasure actions, for evaluating the cost of actions that can disrupt production, and a metric, TUL, useful for assessing the cost-effectiveness of capability improvements that help during attacks but come with a continuing cost. While explained with reference to redundancy, the TUL metric can be applied to any improvement where the cost impacts can be calculated, and where a comparison to a reference makes sense.

This chapter completes the description of the parts of our methodology for resilience impact assessment. The chapters that follow contain the studies in which we applied this methodology to different applications, starting from resilience planning in Chapter 8, followed by reactive countermeasure selection in Chapter 9, and mission viability analysis in Chapter 10.

# Chapter 8

# Applications: Resilience planning

This chapter shows an application of our methodology to planning cyber resilience improvements, using our impact assessment methodology to evaluate the system production and cost impact of proposed improvements if cyber attacks were to occur. The work presented here is based on our work published in [SCML22], and focuses on the specific case of redundancy planning. The business case for considering cyber resilience improvements is convincing, as the business disruption arising from downtime or unplanned outages due to cyber-attacks costs companies millions of dollars each year [BLDC19]. Thus, improvements that can help minimise disruptions to system availability when attacks occur can be very beneficial, but the analysis of their effectiveness requires the ability to model the progression and impact of attacks in the system, to enable decision makers to balance investment to minimise business cost. Our methodology enables such analysis.

There are several classes of techniques for improving cyber resilience, including the 'architectural techniques' listed by [Gol10, BG11] for cyber resiliency engineering, discussed briefly in Sec. 2.1.1. Among these, we chose to focus on *redundancy* and *diversity* as they are generally applicable to many systems, and are a familiar consideration for system performance and resilience against faults, not just for when cyber attacks are considered, thus providing a useful bridge to the research communities and engineers who may not otherwise concern themselves with cyber security. While some aspects of the work here are specific to analysis of redundancy with diversity, the approach we use for comparing the expected production and cost impacts of cyber attacks could apply to other techniques

relating to architectural changes, such as privilege restriction and segmentation, if provided a model implementation of how these changes affect the attack progression and production models.

In the application here, we investigate the extent to which component redundancy, with and without diversity, can help mitigate the impact of cyber attacks that aim to reduce system performance. We estimate impacts, in terms of monetary costs, of penalties from breaching service level agreements (SLAs), and find optimal resource allocations to minimise the overall costs arising from attacks.

Our approach to finding optimal redundancy allocations relies on an implementation of our methodology for attack impact analysis, with an attack progression model based on attack graphs, system production performance modelled using queueing networks, and a cost model that enables relating the cost of SLA penalties to the excess maintenance cost from redundancy. We evaluate our approach using a case study of a website, and show how redundancy and diversity can improve the resilience of a system by reducing the likelihood of a fully disruptive attack. We find that the cost-effectiveness of redundancy depends on the SLA terms, the probability of attack detection, the time to recover, and the cost of maintenance. In our case study, redundancy with diversity achieved a saving of up to around 50 percent in expected attack costs relative to no redundancy. The overall benefit over time depends on how the saving during attacks compares to the added maintenance costs due to redundancy.

## 8.1 Introduction

In this chapter we apply an implementation of our attack impact assessment methodology to redundancy planning, specifically to optimise server capacity considering the cost of service provision given that attacks occur. For this, we use our methodology to evaluate the expected cost impacts of a set of attack scenarios under different server allocations. Then, we propose two optimisation problems for finding the most appropriate allocation for the scenarios considered: one that minimises the expected cost during the attack, and another that ensures redundancy is affordable during times without attacks.

The attack progression model implementation includes an AG representing the available attack paths, assumptions on behaviour based on attack scenarios that aim to disrupt system output, and modelling of detection to determine what outcomes can occur for each attack scenario. In the production

model, the attack impacts are estimated using a queueing network (QN) model which estimates the performance impact of downtime, taking into account interdependencies and cascading effects. Assumptions on recovery time are used to approximate the duration of the disruption due to the attacks. A cost model is used to assign a monetary value to the disruption caused by each attack outcome, and an expected cost is calculated over the outcomes of the scenarios considered.

We illustrate our model in an e-commerce case study. Our focus is on selecting server allocations such that the system performance, measured in terms of throughput of jobs, is maintained within the requirements of a service level agreement (SLA). The model is used to find the allocations that balance the cost of redundancy with the costs from attacks, including penalties for breaching the SLA. Beyond the case study considered, the analysis we propose is applicable to broader classes of systems where performance and availability are important, and/or where there are SLAs. This is the case, among others, in transactional systems, supply chains, and manufacturing systems.

The main contributions of the work presented in this chapter are as follows:

1. We investigate the effectiveness of redundancy to mitigate attack impacts, taking the service capacity analysis angle to redundancy and diversity. This differs from previous works on the use of diversity for security such as [BWJS18, BWJS19, LFH20], which focus on identifying which parts of a system to diversify, but without considering overall performance. On the capacity analysis side, where performance modelling is common, our work differs in our focus on malicious attacks. Attacks change the problem considerably, as malicious compromises are not random but follow a correlated approach to maximise damage, so simple redundancy is less effective than against random failures.

2. Attack modelling combined with detailed performance modelling is itself novel. Only a few works such as [CKN+13], [VTC+14], [GKK17] and [SWP17] consider both attacks and availability. Further, investigating the impact of multi-step attacks with detailed performance modelling using QNs has, to our knowledge, not been presented in the literature.

3. We introduce a metric for the cost-effectiveness of redundant allocations, Time Until Loss (TUL), reflecting the attack frequency required to make a redundant allocation financially viable compared to a reference allocation.

Our findings, summarised below, have wider implications for the resilience of systems, their design, operation and regulation. Whilst some may seem intuitive, the methodology we introduce provides the means to quantify them and to enable organisations to provision their investment in resilience to cyber attacks. The key findings are:

- Adding redundancy with diversity can improve system performance during attacks and help organisations avoid SLA penalties. By comparison, redundancy without diversity provides limited benefits.
- A key parameter for determining the benefits from a redundancy strategy is the probability of detection of individual attack steps, as this significantly impacts the overall attack success likelihood.
- Benefits from redundancy critically depend on whether attacks can trigger SLA compensation and thus losses.
- The frequency of attacks and the expected loss determine the long-term financial viability of a redundancy strategy. The benefit from the strategy must exceed the excess maintenance costs due to it.
- The SLA structure has a large impact on the appetite to invest in security, other things being equal: it can cap the compensation payable to customers, and thus the costs to the company.

We find that redundancy with diversity can reduce the costs arising from cyber attacks when diversification reduces the likelihood of an attack affecting services enough to trigger SLA penalties. However, the benefits that diversification affords during attacks must exceed the consequent excess maintenance costs incurred over time. Therefore, the final choice of whether to apply redundancy with diversity relies on a balance of various parameters, and the estimated frequency of attacks. Our methodology can aid such decisions.

The rest of the chapter is structured as follows: Section 8.2 discusses related literature; Section 8.3 provides an overview of our approach, after which a running case is introduced in Section 8.4; A specific instantiation of our methodology onto the running case is shown in Section 8.5; The evaluation of our method is discussed in Section 8.6, and Section 8.7 concludes.

## 8.2 Related work

In addition to works already discussed in Chapter 2 which relate to our methodology in general, on resilience, attack impact assessment and performance analysis, there are parts of literature that relate specifically to this particular application of our methodology. We discuss such works below.

**Resource planning and redundancy:** We investigate resource allocation planning for networked systems under attack scenarios. The topic has similarities to works using queueing networks for reliability and system performance [BGDMT06], resource provisioning [HWIL09], capacity planning [KL12], and resource management [GGQ+14], [HGG+14]. The key difference is that we focus on impacts from attacks (as opposed to failures or environmental effects), and how these could be mitigated with capacity planning using redundancy with diversity. Cyber attacks cause a different pattern of component unavailability that requires different modelling. Attackers do not act randomly but are goal-oriented, and likely to target whole services rather than individual components.

Existing works where redundancy is used for security often employ approaches similar to N-version programming [AC77] for fault tolerance, e.g. in [AAM+10], [ASP12] and [HWWC17]. Such works use redundancy solely to assess or improve the integrity of an output. By contrast, we consider the use of redundant capacity to maintain system availability during attacks.

Ge *et al.* [GKK17] use redundancy for availability considering the security impact of patching cycles and investigate the effect of different redundancy strategies. Although their work is related to ours, key differences exist in the specific problem addressed and the approach. For example, they model attack impacts relying on CVSS [MSR06] impact metrics, which are static, whereas we model impacts on system performance over time. Thus we can evaluate changes to impacts even when vulnerabilities do not change.

**Combining attack modelling and performance:** Attack modelling has rarely been combined with performance evaluation. For example, as we discussed in Section 2.2.3, while AGs have been used for attack impact evaluation by [AJPS11, AJ17, SML19, SSL17, CYS+18], their impact evaluations have not used detailed performance models, but simpler models of component dependencies.

One exception is Chen *et al.* [CKN+13], who combine workflow models with security information in an *Argument Graph*, to generate quantitative metrics of how the workflow performs with respect to a security goal. Their model is at a higher abstraction level than our QN-based approach in terms of performance evaluation, and less suited to model bottlenecks arising from downtime.

Some studies have examined the impact of DDoS attacks using QN models. For example, Shen *et al.* [SWP17] investigate the impact of DDoS attacks on web applications using a QN model, and [YTGW14, LJZY20] examined how cloud platforms can maintain availability in the face of DDoS attacks. However, DDoS attacks represent a simpler special case of an attack as they present themselves as increases in system load, without requiring attacker actions within the system. By comparison, we consider attacks where multiple stages take place inside the system, exploiting its vulnerabilities, and hence require a model of how the attack can propagate within the system.

**Diversity:** Existing studies on the use of diversity to increase robustness to attacks, e.g. [BWJS18, BWJS19, LFH20], focus on optimising network service diversity to resist attack propagation. By comparison, we determine the optimal number of (diverse) servers to be used for the services provided by a system in order to limit the impacts from attacks, with diversity arising in the different server types used for a given service. Therefore, our work is complementary to the literature on network diversity, not competing with it. Further, network diversity literature optimises metrics based on the properties of the network structure, various diversity metrics [BWJS19] and *'h safety metric'* in [BWJS18]. In contrast, we model the performance of a system while under attack, and estimate attack costs under different server allocations, including redundancy with diversity, and optimise costs.

## 8.3 Overview of the approach

We propose an approach to estimating the optimal level of redundancy for production components in a system whose output performance can be affected by cyber attacks, building on an implementation of our methodology to attack impact modelling described in Chapter 3. That is, we find a level of redundancy investment that provides sufficient robustness to mitigate losses due to attacks while being cost effective. At the core of this is the ability to quantify the impact of attacks, which we approach

from the perspective of the financial costs incurred before, during and after the attack. We consider both direct costs and production impact costs, including the costs of preparation (investment into redundant capacity) and the losses due to disruption (loss of system availability or performance).

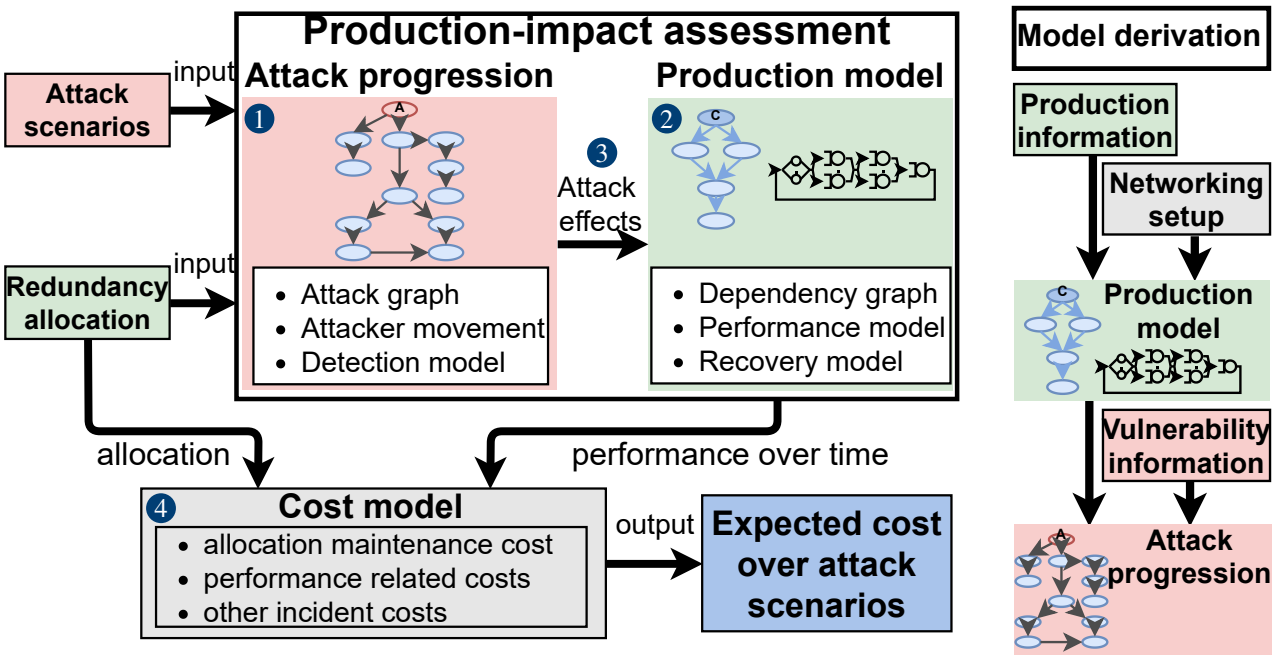
We focus here on the system design and resource allocation before attacks occur, using redundant capacity to increase the robustness of the system to cyber attacks. As we consider redundancy planning, the defensive actions of interest are *additions* of redundant servers, with and without *diversity*. Additions take place before the attacks, and redundant capacity is instantly functional when needed. We model diversity by adding "alternative" servers that do not have the same vulnerabilities as the "regular" servers (a regular server refers to the server type that would be the first choice if only one type was used).[1] Thus the redundancy state is described by a "component allocation", which specifies the number of components for each function and whether they are homogeneous or diverse.

Our analysis is conducted over an extensive set of the possible outcomes of a set of attack scenarios, based on which an expected cost of attack impacts is calculated. We use an optimisation algorithm to find a cost-efficient component allocation, which uses a genetic algorithm to find the optimal solution. This approach was chosen because we have an integer programming problem with a non-linear objective function.

Figure 8.1 summarises our process for evaluating attack costs, based on which the optimisation is done. Fig. 8.1a is a repeat of Fig. 3.2 which we used to summarise this work, although shown here with overlay numbers to better explain the details of the model components. As shown in Fig. 8.1a, attack scenarios and a component allocation are input to an impact assessment that determines system performance over time. A cost model is then applied to evaluate the expected attack costs. The optimal component allocation given the attack scenarios is found by optimising based on these costs. Fig. 8.1b shows how the production and attack progression models are derived from system information: the production model builds upon knowledge of the components required for production and their connectivity; attack progression is built to describe how the production can be attacked, using the production model and vulnerability information as inputs.

Following our methodology described in Chapters 3-7, the impact assessment approach used here

[1] To check whether servers share known vulnerabilities, databases such as NVD [NIS] can be used.




(a) Attack cost evaluation and key modelling components

(b) Deriving the models from system information

Figure 8.1: Summary of our attack impact analysis approach

integrates a model of attack progression ((1) in Fig. 8.1a) with a production model of the system ((2) in Fig. 8.1a), linking the privileges obtained during the attack with the damage they can be used to inflict upon the system using an impact map ((3) in Fig. 8.1a). A cost model is then applied to evaluate expected costs considering attack impacts and cost of redundancy provision ((4) in Fig. 8.1a).

We model *attack progression* ((1) in Fig. 8.1a) with an AG, showing the paths an attack can take, i.e., how vulnerabilities can be exploited in successive attack steps to acquire more privileges. To analyse the expected cost of attacks to a system, an estimate of the relative likelihood of different attack outcomes is required. We thus model the detection of individual attack steps, splitting an attack into different outcomes according to the number of steps that succeed without being detected. We assume that once detected, the attack can be contained, but can still disrupt the production system using the privileges obtained by that point. This models the damage that can be achieved even if the attacker does not reach the maximum level of privileges aimed for.

The *production* model ((2) in Fig. 8.1a) uses a combination of a DG to visualise the dependencies between components, and a QN model to evaluate *system performance*. Our running example is a multi-tier application where the output is a web-service, and the production model components are

the servers processing requests to the application. The QN modelling enables us to estimate the detailed performance impacts from changes in the system component allocation and from damage due to attacks, which is required for redundancy analysis.

To calculate the cumulative loss of production, a major part of the costs arising from cyber attacks, we must establish the *duration* of a disruption and thus need to model how the system can be recovered after an attack. *Recovery* in our model involves both recovering the services affected by an attack, and removing the privileges held by the attacker. Removing the privileges is important, as failing to do so would allow the attacker to disrupt the system again.

The *links between the attack and production models* ((3) in Fig. 8.1a), i.e. elements of the impact map, occur when an obtainable privilege can be used to disrupt a production component. In our running example below, these are shown as links between the AG and the DG.

Finally, we must quantify the losses due to the disruption as well as the direct costs (acquisition and maintenance) of the servers and redundancy, i.e. define a *cost model* ((4) in Fig. 8.1a). In the case study used here, the cost of disruption is based on penalties for failing to meet an SLA.

### 8.3.1 Threat model

We focus on attacks that aim to *reduce system performance* via *targeted multi-step attacks* that compromise system components and disrupt their *availability*, thus impacting the ability of the system to produce its output. An attack requires a sequence of steps within the system, exploiting vulnerabilities and gaining privileges. Some of the privileges enable the attacker to disrupt system components, and impact production in the system. Disruption to the availability of a component can be achieved in different ways: by damaging the integrity of the software/hardware, by deleting/encrypting data necessary for the component to operate, etc; our approach is agnostic to how this is achieved.

We do not consider attacks that only aim to steal data, as they do not directly affect system performance. While their impact on a business can be considerable, their costs can be modelled without a production model for the system, e.g. based on the number of records lost [Dun18]. If data breach losses need to be considered in addition to damages to production, our method can be adapted to add

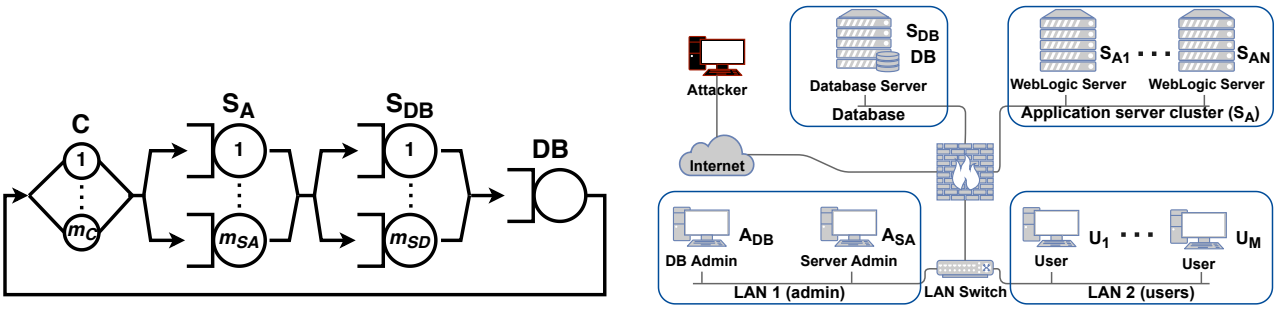
these costs, as explained in Section 8.6.3.2. We also do not consider attacks where the system is not breached, such as DDoS attacks on the external web-interface of a system. The impact of such attacks can be characterised solely in terms of the availability of the end-point, and the attack steps prior to the impact stage, e.g. recruitment of a botnet, do not take place within the system attacked. Such external steps cannot be detected or acted upon from within the system, and thus do not fit our attack progression and detection modelling.

We assume that the attacker progresses through the system by exploiting vulnerabilities of the system components and that each exploitation of a vulnerability has a time required for exploitation. As discussed in Sec. 4.2.1, we distinguish between attack steps performed to obtain privileges (*move steps*), and using these privileges to disrupt system production (*disruptive steps*). In the model used here, we assume that an attacker will seek to maximise the privileges obtained before causing disruption, but if detected, will cause what disruption it can with the privileges obtained before detection. We assume that detection leads to containment of the attack, i.e. being stopped from obtaining further privileges. The assumptions we make about attack scenarios are discussed in Section 8.4.1.

## 8.4 Running case

Our case study is based on an e-commerce setting using J2EE services. We chose it as its architecture is representative of widespread service-based systems and its performance model is well understood. The system and workload characteristics we use are based on the queueing model in [KB03] on a multi-tier application benchmark by SPEC [Sta02].

The basic structure of the QN, its workloads and parameter values are as in [KB03]. Fig. 8.2a shows the QN used (repeated here for convenience from Fig. 5.3). Customers (represented by node C) send jobs to the application servers in $S_A$, which process them and pass them on to the database. The jobs then go through the database processors in $S_{DB}$ before the queries reach the database $DB$ itself, from which a response is returned to the customer. The figure shows that the application and DB processor stations have server multiplicities $m_{SA}$ and $m_{SD}$, respectively. In our model we identify the appropriate number of each server type for the most cost-efficient performance during expected




(a) Queueing network in the case study

(b) Network topology in the case study

Figure 8.2: Queueing network and network topology for the J2EE case study

Table 8.1: Workload service demands and parameters

**Workload service demands at processing components**

| Job class | WLS CPU | DBS CPU | DBS I/O |
|---|---|---|---|
| NewOrder | 12.98ms | 10.64ms | 1.12ms |
| ChangeOrder | 13.64ms | 10.36ms | 1.27ms |
| OrderStatus | 2.64ms | 2.48ms | 0.58ms |
| CustStatus | 2.54ms | 2.08ms | 0.3ms |
| WorkOrder | 24.22ms | 34.14ms | 1.68ms |

**Parameters: client numbers and think times at different loads**

| Parameter | Low load | Moderate | Heavy |
|---|---|---|---|
| NewOrder clients | 30 | 50 | 100 |
| ChangeOrder clients | 10 | 40 | 50 |
| OrderStatus clients | 50 | 100 | 150 |
| CustStatus clients | 40 | 70 | 50 |
| WorkOrder lines | 50 | 100 | 200 |
| Cust. think time | 2s | 2s | 3s |
| Manuf. think time | 3s | 3s | 5s |

attacks. This also includes the number of databases in $DB$.

The service demands and parameter values for the Queueing Network (QN), from [KB03], are provided in Table 8.1. There are five classes of jobs processed in the system: NewOrder is a new order request; ChangeOrder is a request to change an existing order; OrderStatus requests the status for a given order, while CustStatus lists all orders by a given customer; WorkOrder is an order for widgets from a manufacturer.

The processing times for each job are given for the three processing components: WebLogic server CPU (WLS CPU), Database Server CPU (DBS CPU) and Database Server I/O (DBS I/O). Client numbers refer to the number of customers that are concurrently in the system creating jobs of the different classes, while the counts of manufacturing lines are the number of WorkOrder jobs in the system. Our analyses presented below were calculated using 260 concurrent clients, corresponding to the moderate workload case in [KB03]. Full details of the workload and queueing model used for the case study are provided in [KB03].

While the performance modelling details of the system largely remain as in [KB03], our case study adds assumptions about the specific network topology and vulnerabilities present in order to model at-

tacks. The application servers provide clients access to the services over the internet, and the database resides in a separate subnetwork, as shown in Fig. 8.2b (repeated here for convenience from Fig. 4.2). The rest of the network is split into two subnetworks containing administrators and users, respectively.

### 8.4.1 Summary of key assumptions

A set of assumptions are necessary to model the behaviour of the attacks, their detection, the recovery of the system, and the costs. This enables us to reason about the resilience of the system and analyse investments in capacity to withstand attacks. When applied in the context of a specific system, an organisation and an industry sector, prior experience offers a useful guide on the type of threats, damages and recovery to consider, and the model can be customised accordingly. In our case, we have chosen these assumptions with a view of ensuring the consistency and validity of the analysis in the case study. We have also explored the sensitivity to various parameters to ensure that our conclusions hold even if some assumptions are relaxed.

We consider that the attackers have well defined goals and sufficient knowledge of the system to progress towards those goals. This is appropriate, as the technology and typical deployment of the system it seeks to disrupt will be known to the attacker. We also assume that attacks seek to maximise the privileges acquired before launching disruptive steps, unless they are detected (or cannot progress), in which case they immediately cause the disruption they can. Thus, to evaluate the impact of the attacks, we consider a set of attack scenarios that are reasonably expected to cause significant damage, and evaluate the resulting system performance under the different outcomes of those scenarios. We believe this is more realistic than assuming the attack always chooses the shortest path (attackers have different skills, tooling, and preferences) or assuming the attack chooses its next steps randomly (which is both not realistic and computationally expensive). We have chosen a number of scenarios for our case study, shown in Fig. 8.3c, but do not place any limitation on the scenarios considered. Whilst this approach to attacker behaviour may omit some variation in the attack scenarios, this affects the calculation of probabilities for the different outcomes, but does not change the set of outcomes considered for impact. Therefore, the assumptions could be relaxed by providing a different model for the attack behaviour, without affecting the rest of our approach. Finally, we assume that the

disruptive actions are taken simultaneously on all servers to which the attacker has privileges. This is reasonable, as causing disruption in stages would likely be detected.

If an attack step was successfully performed without being detected, we assume that it will not be detected later in useful time, i.e., before the attack reaches the privileges it seeks to obtain. We believe this holds in most cases where attacks aim to cause disruption rather than simply to remain embedded in the system. If an attack only seeks to establish permanence and capability, it could be detected at a much later stage, but no disruption would have been caused. As a result, attack detection in our model is based on the latest attacker activity, i.e. the latest attack step attempted. In the sample analysis, detection probability is assumed equal at each attack step, regardless of where in the system it occurs. This is a simplifying assumption. Although, the methodology could be extended to consider a different likelihood of detection for each vulnerability, determining these values is difficult as they are context specific. Choosing the likelihood of detection randomly is also not appropriate. Once an attack is detected, we consider that the attack is *contained* i.e., that further attack steps in the attack graph are not possible. This may seem a strong assumption but it is reasonable as, typically, compromised systems are either disconnected or quarantined. If an attack is detected we assume the attack executes all the possible disruption steps with the privileges it has acquired so far. This is consistent with the assumption that the attack seeks to cause disruption to the production system.

We assume that the organisation can recover compromised servers and purge the attack from the system, e.g. via re-imaging and patching, and do not consider the recovery process to be over until the attack has been entirely removed. This is a practical assumption in most cases, as recovering a system without removing the attack would allow the attack to simply repeat its previous actions. We have considered that recovery occurs simultaneously at all disrupted servers. This disregards that recovery resources may be limited, but ensures our analysis is conservative as it minimises recovery time. A longer recovery would entail even larger losses and further favour the use of redundancy.

In our cost model, the costs considered in the optimisation include server costs (per month, including purchase/rental and maintenance) and losses due to attacks, in the form of penalties for SLA breaches. Other losses can also be considered, as explained in Section 8.6.3.2, but they do not affect the optimal choice found in the optimisation, and can thus be considered outside the optimisation.

In addition, further assumptions were made on the parameter values used in the case study, but these are example values and do not limit the modelling itself. These are explained under the relevant subsections of Section 8.5, and their values listed in Table 8.3 of Section 8.6. In general such parameter values should be set based on the system and context in which the methodology is applied.

## 8.5 Model

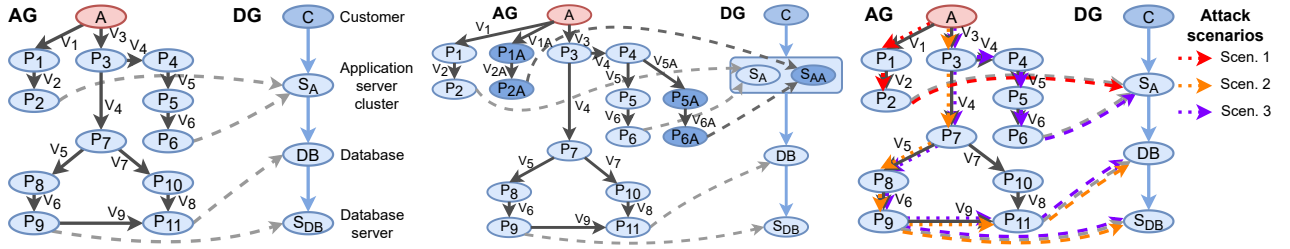
### 8.5.1 Attack and dependency graphs

**Attack Graphs:** We use Attack Graphs (AGs) [LI05], [LDB20], as the basis for our attack progression modelling, as discussed in Chapter 4. For the purposes of the application proposed here, we only require an AG that is a simple directed graph with system privileges as nodes, and edges as vulnerabilities whose exploits enable an attacker holding one privilege to obtain others. This was given as Definition 4.1. While a more detailed logical AG could be used, it is not needed for the analysis here.

**Dependency Graphs:** A Dependency Graph (DG) describes the dependencies between the components of the production model. For our implementation here, we use the simple DG definition from Chapter 5, Definition 5.2.

### 8.5.2 Attack progression

Fig. 8.3a shows the AG and the DG for the case study, visualised with connections between the graphs following the approach proposed in [AJPS11, AJ17]. The DG provides a view of the service dependencies for the customer-facing services, with the cluster of application servers treated as one node, as is the database server cluster. The edges from AG nodes to DG identify the system components whose operation can be disrupted if the attacker holds the given privileges. For example, an attacker obtaining the privilege $P_2$ can disable a server in the application server cluster $S_A$. The key targets for an attacker are the AG nodes enabling the most disruption, the three leaf nodes: $P_2$, $P_6$ and $P_{11}$.




(a) AG and DG for the J2EE case study (b) Diversified AG and DG (c) Attack scenarios in the case study

Figure 8.3: AG and DG in the J2EE case study, with attack scenarios

Table 8.2: Privileges and vulnerabilities in the case study

| Privileges | | | | Vulnerabilities | | | |
|---|---|---|---|---|---|---|---|
| $P_1$:U, WebLogic 7 | $P_5$:U, SuSE 8 (AS) | $P_9$: A, SuSE 8 (DS) | $P_{1A}$:U, alt. server | $V_1$:CVE-2003-0151 | $V_5$:CVE-2004-1175 | $V_9$:CVE-2004-0638 | $V_{1A}$:obtain user priv. |
| $P_2$:A, WebLogic 7 | $P_6$:A, SuSE 8 (AS) | $P_{10}$:U, Oracle 9i DB | $P_{2A}$:A, alt. server | $V_2$:CVE-2003-0640 | $V_6$:CVE-2004-0495 | | $V_{2A}$:priv. escal. (U-A) |
| $P_3$:U, LAN 2 | $P_7$:A, DB Admin | $P_{11}$:A, Oracle 9i DB | $P_{5A}$:U, alt. OS (AS) | $V_3$:Phishing attack | $V_7$:CVE-2002-0965 | | $V_{5A}$:obtain user priv. |
| $P_4$:A, Server Admin | $P_8$:U, SuSE 8 (DS) | | $P_{6A}$:A, alt. OS (AS) | $V_4$:CVE-2006-5051 | $V_8$:CVE-2004-1707 | | $V_{6A}$:priv. escal. (U-A) |

Abbreviations: U – User; A – Admin; AS – application server; DS – Database Server; DB – Database; alt. server – alternative server; priv. – privilege; escal. – escalation

The specific privileges represented by the Attack Graph (AG) nodes of Fig. 8.3a, and the vulnerabilities in the edges, are listed in Table 8.2 (repeated here for convenience from Table 4.1). The vulnerabilities that can be exploited to obtain the privileges are shown as labels on the edges of Fig. 8.3a, e.g. $V_1$. The vulnerabilities listed represent possible vulnerabilities that could have occurred in the case study system, relating to the appropriate component versions and time period for the hypothetical system. In practice there could be various vulnerabilities that can achieve the same attack outcomes, including attack techniques that do not require application vulnerabilities. For our model, the key is how the vulnerabilities relate to privileges, i.e. what privilege they require and what privilege can they be used to obtain, while the specific vulnerability identities are not as important.

The network configuration assumed in our case study allows two distinct attack methods. First, direct attacks over the Internet to the WebLogic servers in the application servers subnet are possible. These could be traffic to the appropriate application port, exploiting vulnerabilities in the application ($V_1$ and $V_2$ in the AG). Second, if an attacker obtains access to the network itself, other attacks become possible. For example, a phishing attack to a machine on LAN 2 can allow an attacker to obtain privileges in the admin machines (DB Admin or Server Admin) in LAN 1, enabling the exploitation of vulnerabilities in the Application server and Database subnets. The network is configured so that administrator machines can be remotely accessed via SSH (e.g. for remote access with a VPN), and thus have port 22 open to traffic from within the network. The computers have a vulnerability enabling an attacker to obtain privileges on the admin machines, which then allows compromising the

application servers or the database.

**The diversified case:** To investigate the effect of implementing *redundancy with diversity*, we evaluate the impact of adding *alternative* application servers with a sufficiently different configuration that they do not share vulnerabilities with the *regular* application servers otherwise used. While the term *diversity* can be used with regard to different system aspects, in our case it simply means that the components are sufficiently diverse not to share vulnerabilities. That is, the vulnerabilities in the alternative servers are not the same as those in the regular servers. Generalising to the case where the servers share some vulnerabilities but not others could be done by e.g. using the metric on vulnerability similarity between products by Li et al. [LFH20], and reflecting this in outcome probabilities.

This is shown in Fig. 8.3b, where a darker colour is used in nodes where servers provide redundancy with diversity to existing application servers. The DG has an additional node $S_{AA}$, representing these added servers which provide the same service as those in $S_A$ but are of the alternative type. Similarly, the AG has additional privilege nodes relating to servers in $S_{AA}$. These privileges are obtainable via vulnerabilities that differ from those in servers in $S_A$, but form similar patterns of attack.

With diversity, the attacker requires two different exploits to acquire privileges in all the servers providing a service, rather than a single one. To model the added attack difficulty due to requiring a further exploit, we set a probability that the attacker has the capability to exploit one of the vulnerabilities, the other, or both. An attack scenario to a diversified service is thus split into three cases, as the case of not having capabilities to exploit either is disregarded. We assume that the probability that an attacker can exploit a given vulnerability is independent from the probability of them being able to exploit a different one. In our case study we also assume, for simplicity, that these three cases of capabilities are equally likely to occur in attacks on a system that is diversified. However, we have also examined the impact of relaxing these assumptions; see Appendix A.2 for the results.

**Attack steps:** An attack consists of "move steps" and "disruptive steps". In a *move step* the attacker moves in the AG by exploiting vulnerabilities to obtain privileges. After obtaining a privilege, the attacker has the ability to take a *disruptive step* to make a server (or servers) unavailable. This is represented with the dashed lines from AG nodes to DG (i.e., the impact map).

In our case study the privileges relate to specific server instances, so a copy of a privilege applies to one server copy, not several. For example if there are two servers in $S_A$, the attacker requires two copies of the privilege $P_2$ (or of $P_6$) to disrupt both servers in $S_A$.

**Time-to-exploit:** The time taken by the move steps of an attacker is determined based on estimates of the time it takes to exploit the vulnerabilities required. In general, these would be based on the capabilities, skills and system knowledge of the attacker. As we are concerned with attacks on system performance rather than objectives that require attackers to persist in the system, we assume that attackers take steps as fast as their abilities allow, and there is no delay between steps.

Based on [MBFB06, LB08, NWJS13], the time to exploit a vulnerability is in the order of hours or days. In the case study, we use the following values for time to exploit: 24h for the initial exploit of each vulnerability; 6h for using the same exploit elsewhere in the network. Once the attacker has obtained a privilege on a server, replicating the exploit on other servers of the same type takes 4h.

In contrast to move steps, disruptive steps are assumed instant, so take place immediately after the attacker has obtained the privileges required for its goal. Disruptive steps are launched simultaneously from all privileges obtained.

### 8.5.3 Attack detection

We make three assumptions in our detection modelling. First, intrusion detection acts on the latest activity, so detection can only occur in an AG node where the attacker is currently acting; thus not taking into account delayed detection of earlier attack steps. Second, detection happens with equal probability at every node. Third, if an attack is detected during a move step (before disruption), the defence can and will stop it from moving further along the AG, but not from disrupting the system using the privileges already acquired. These assumptions simplify the model and allow us to focus the explanations and illustrations on the resilience aspects rather than the modelling.

Detection affects the probability of an attack reaching its intended goal, and leads to partial attacks when the attack is detected before reaching its final goal. In these cases, we assume that the attacker launches disruptive steps from the privileges it holds, simultaneously from all the nodes.

This detection model leads to a simple solution for the relative probabilities of the different outcomes of an attack. For an attack with a full length of $n$ steps, the probability of complete success is $(1 - p_d)^{n-1}$. Any sub-path with length $m > 0$ steps occurs at the probability $(1 - p_d)^{m-1} \cdot p_d$. In the outcomes yielding partial paths, the attack is contained at the point where it was detected.

**Defence after detection:** We consider two cases for analysis. In *Case 1*, attacks will impact all servers of the same type in a cluster if they compromise one of the servers, i.e. the defence cannot stop the attacker between repeating the exploits in identical servers. This reflects situations where stopping the attack requires countermeasures which themselves cause unavailability in other servers of the same type, e.g. blocking connectivity to a part of the network. In *Case 2*, it is possible to stop the attack between exploits of the same vulnerability on different server instances. Thus, when an attack step is detected, the attacker becomes unable to obtain further privileges.

### 8.5.4 Attack scenarios and outcomes in the case study

We consider three possible attack scenarios, and their respective partial success outcomes. The scenarios considered, shown in Fig. 8.3c, are:

- **Scenario 1**: attack to application servers over the internet (target: $P_2$); $[A \rightarrow P_1, P_1 \rightarrow P_2]$
- **Scenario 2**: attack to DB, as direct as possible (target: $P_{11}$); $[A \rightarrow P_3, P_3 \rightarrow P_7, P_7 \rightarrow P_8, P_8 \rightarrow P_9, P_9 \rightarrow P_{11}]$
- **Scenario 3**: attack to both the application servers (using privilege $P_6$) and DB, reusing vulnerabilities $V_4$ - $V_6$ that occur along both paths (targets: $P_6$ and $P_{11}$); $[A \rightarrow P_3, P_3 \rightarrow P_4, P_4 \rightarrow P_5, P_5 \rightarrow P_6, P_3 \rightarrow P_7, P_7 \rightarrow P_8, P_8 \rightarrow P_9, P_9 \rightarrow P_{11}]$

The relative weights given to these three scenarios should, in practice, be based on their likelihood of occurring. Such estimates could made based on past experience of the company or the industry sector. To simplify, in our baseline case we apply equal relative probabilities to the three scenarios, $1/3$ each. We investigated the sensitivity of our results to this equal-weight assumption in Appendix A.4. Although the expected cost values change, the results in terms of the optimal choice of server allocation remain stable, changing to a different choice only in cases with extremely uneven weighting

where no weight was given to one of the scenarios. Thus the equal-probability assumption is not affecting the overall characteristics of our results.

The weights of all the sub-paths (attack outcomes) within a given scenario must sum to the weight of that scenario, so that the total weight across all attack outcomes is $W = \sum_{M \in A} w_M = 1$.

### 8.5.5 Recovery modelling

When an attack has been detected, the defence will recover the servers affected, and purge the attacker from the system in $t_r$ time. This recovery time consists of the time to reboot servers, and the time to purge the attacker from the system. Failing to purge the attacker would likely lead to a re-occurrence of the impact and a longer overall downtime. While server reboot times are fast, in the order of seconds or minutes, purging the attacker requires detailed analysis and could take considerably longer, in the order of several hours. Thus our recovery times here are based on estimates on recovery from attacks, instead of server reboot times.

A report by Ponemon Institute on DoS attacks [Pon15] found an average downtime due to DoS of 9h among the companies they surveyed. Given the average number of DoS attacks among respondents was four, and 18% of attacks lead to no downtime, we estimate that the average downtime from an attack was $9\text{h}/(4 \times 0.82) \approx 2\text{h}45\text{min}$. As our attack modelling uses an hourly frequency,[2] we round this to 3h. Importantly, both 3h and the 2h 45min estimate lie between 0.1% and 1% of a month (43.2 minutes and 7h 12min, respectively), which are often used as boundaries for compensation in SLAs.

However, the attacks in [Pon15] likely consist largely of cases where the attacker is flooding the servers with requests, while we are mainly interested in the service availability impact of attacks penetrating the network of a company. Thus, we use another data point to represent recovery from more general attacks. Reporting the recovery time from the most disruptive breach faced by companies in the previous year, [FFM+19] found a mean time to recover of three days for attacks that had an economic impact. This 72h is the high value in our $t_r$ sensitivity range.

[2] Although the QN model uses a second frequency for estimating performance, due to the performance of the processing components, we use hours for the time unit of the attacks, as the SLA commitment period is one month.

These values are approximations, used to illustrate the approach. The recovery times could vary across different systems and attacks, based on factors such as vulnerability types, patch availability, and manpower capacity. Consequently, the values applied should reflect the application context. In practice, values could be set based on previous experience, or blue-team exercises.

### 8.5.6 Performance modelling

We assess the performance of the production model using QN models, solved using LINE [PC17, Cas20], an open source tool for specifying and analysing QN models that enables fast solving of QNs using fluid approximation. For this purpose, the impact of an attack is expressed in terms of the availability of servers in the nodes of the production model. Impacts are evaluated for the different outcomes of the attack, including partial successes caused by detection. For this, we use the approach we introduced in Sec. 5.2.3.2, with the $M_{attack}$ matrix showing the impact of attack outcomes in terms of servers available in different production nodes, and the vector $\tau_{attack}$ specifying the duration of these impact stages. See Sec. 5.2.3.2 for more details.

### 8.5.7 Costs

The costs relating to the provision of a service comprise the direct costs of server capacity (monthly per-unit costs), and penalty costs from failing to meet SLAs. The *capacity cost* is:

$$\gamma_i = c_i \cdot x_i + c_{a,i} \cdot x_{a,i} \tag{8.1}$$

where $x_i$ is the number of regular servers used for a given service (or cluster) $i$ and $c_i$ is the monthly per-unit cost for a regular server (equivalent monthly cost, including server purchase/rental and maintenance costs). The alternative servers used for diversification are assumed to have a higher cost $c_{a,i}$ per server, and $x_{a,i}$ is the number of such servers. Fixed costs of capacity and diversification do not feature in (8.1), as they do not impact the optimisation, but can be added afterwards if desired.

Our SLA has only one credit level, for 100% service credit. The SLA breach condition is: *100%*

*service credit if* $X(t, x, M) < 0.9 \cdot X_{ref}$ *for a disruption time greater than* $\beta$ *in a month.* Here, $x$ is a server allocation vector, $M$ is an attack outcome matrix, and $X(t, x, M)$ is the system throughput at time $t$ observed when server allocation is $x$ and the attack outcome $M$. $X_{ref}$ is the reference throughput that has been marketed to the client (the published performance), and $X_{req} = 0.9X_{ref}$ the required throughput. The 90% limit is as in the performance SLAs in [Ora]. $\beta$ is the limit value (as a share of time in a month) that performance can be below $X_{req}$ before the SLA is breached. The equations for the SLA penalty cost were given in Chapter 7, in equations (7.7)-(7.8).

**Costs in the case study:** We assume a server operating cost of $185 per month for regular servers. This is derived using a finding by [KBP+09] that IT capital costs were about 45% of the total annualised cost of server operating costs, and assuming that a new server costs $3000 and has an operating life of 3 years. For the SLA penalty costs, we assume that the baseline penalty for breaching the SLA in a given month is $50 per each concurrent client request during normal operation. With 260 concurrent client requests this penalty equates to $13 000.

### 8.5.8 Cost minimisation over attack scenarios

The optimisation problem for minimising costs over multiple attack scenarios is complex due to the non-linearities arising from the SLA penalties, as these are estimated using performance modelling. Our optimisation problem is:

$$\begin{aligned} &\min_{x}(c^T x + \sum_{M \in A} w_M \cdot SLAp(x, M)) \\ &\text{s.t.} \\ &\quad \text{variables in } x \text{ are non-negative integers} \end{aligned} \tag{8.2}$$

where $c$ is a vector of cost coefficients applying to the variables in the server allocation vector $x$; $M$ is an attack outcome matrix in the set $A$ of attack outcome matrices for all attack scenarios considered; $SLAp(x, M)$ is the SLA penalty in attack outcome $M$ for allocation $x$; and $w_M$ is the weight assigned to attack outcome $M$, with $\sum_{M \in A} w_M = 1$.

As apparent in (8.2), the SLA penalty cost is dependent on the average throughput level from the

attacks, which itself depends on the variables to be optimised. This cost term must be estimated during the optimisation. Thus, we cannot use solvers for standard MILP problems, and rely instead on a genetic algorithm solver (here, MATLAB's ga solver).

#### 8.5.8.1 Accounting for excess maintenance costs

The cost optimisation problem (8.2) does not consider potential added maintenance costs during times without attacks. We therefore account for the excess maintenance costs with the TUL metric that we introduced in Section 7.3.2. TUL is the longest time between attacks for which the benefits of the redundant allocation exceed the additional costs involved. Here it is measured in months, coinciding with the length of the SLA commitment period. If attacks are less frequent than the TUL estimate for an allocation, that allocation is not worth investing in (the reference is more cost effective). For a given server allocation vector $x$, TUL is calculated as:

$$TUL(x) = \frac{\phi(x_{ref}) - \phi(x)}{c^T x - c^T x_{ref}} \tag{8.3}$$

where $\phi(x) = c^T x + \sum_{M \in A} w_M \cdot SLAp(x, M)$, and $x_{ref}$ is the server capacity vector for the reference allocation. The numerator includes the overall cost during a month with an attack, while the denominator only concerns the maintenance costs for the allocation $x$.

Whilst we can estimate TUL for the optima to the problem in (8.2), that optimisation may not yield results that ensure the most favourable values for TUL as it focuses on the costs during attacks, ignoring the longer term. Thus, we form an alternative optimisation problem, max TUL:

$$\begin{aligned} &\max_x \left( \frac{\phi(x_{ref}) - \phi(x)}{c^T x - c^T x_{ref}} \right) \\ &\text{s.t.} \\ &\quad ||x||_1 > ||x_{ref}||_1 \\ &\quad \text{variables in } x \text{ are non-negative integers} \end{aligned} \tag{8.4}$$

The condition $||x||_1 > ||x_{ref}||_1$, where $||x||_1$ is the $l_1$-norm, states that the optimisation considers only allocations with more servers than in $x_{ref}$, i.e. with added redundancy.

#### 8.5.8.2 Speeding up the optimisation

As the evaluation of the queueing model is done as part of the objective function, we speed up the optimisation by doing it in stages and by using memoisation. The optimisation consists of two stages, the first of which involves a bound estimation to approximate for the average throughput in the queueing model. This evaluation is fast, and is used to run the optimisation across a large problem space to identify a region to focus on with the more expensive evaluation in the second stage. In the second stage we estimate the throughput of the model in the reduced search space using the fluid solver provided by LINE [PC17]. In this stage we also employ memoisation of the QN model results.

We use memoised performance values for the individual stages of an attack, as some QN configurations reoccur often during the optimisation. This means the performance (throughput) with a given configuration of the QN, in terms of numbers of active servers e.g. [2,0,2,1], is calculated once and added to a memo, and this memoised value is applied for every attack stage where that active server allocation occurs. In our case study this memoisation approach works well, as the transition from one steady state to another is very fast, due to the performance of the servers (multiple jobs per second) relative to the time window of interest for the attacks (one month), so the transition has a negligible effect to the overall performance over the time window. Given this, we can apply the steady-state QN solution method, and memoisation can be applied extensively.

This memoisation approach would be less appropriate if the transient performance during the transition from steady state to another was significant enough to be of interest, and thus transient QN analysis would be preferred. The use of memoised performance values for individual stages of an attack could lead to an evaluation error for the performance over time when transitions between two QN states are not fast, as then transitions from different QN states cause different performance profiles, so the value in the memo may not be appropriate. Thus, if the transition between states is slow, memoisation could lead to incorrect estimates of transitions being applied to different parts of attacks, and could therefore affect the estimates for time spent below a required performance. In summary, the memoisation can add an evaluation error if the transition time is significant relative to the time window of interest. However, as mentioned in the previous paragraph, in our case study the transitions are fast enough to have a negligible effect, and this is also likely the case for other SLAs with

Table 8.3: Parameter values and sensitivity ranges

| Parameter | | baseline | sensitivity |
|---|---|---|---|
| name | description | value | range |
| $t_r$ | recovery time | 3 (hours) | {3;5;10;72} |
| $p_d$ | detection probability | 0.3 | [0;1] |
| $c_s$ | regular server cost | \$185 | \$185 |
| $c_{as}$ | alt. server cost | $1.05 \cdot c_s$ | $[c_s;10 \cdot c_s]$ |
| $X_{req}$ | required throughput | $X_{req} = 0.9 \cdot X_{ref}$ | - |
| $\beta$ | SLA disruption-time limit | 0.001 | {0.001, 0.01} |
| customers* | | 260 | - |
| penalty cost | | \$50 per customer | {50;100;150} |

*Each customer has continuous demand of the system, i.e. there are 260 jobs circulating in the QN.

the commonly used monthly commitment period.

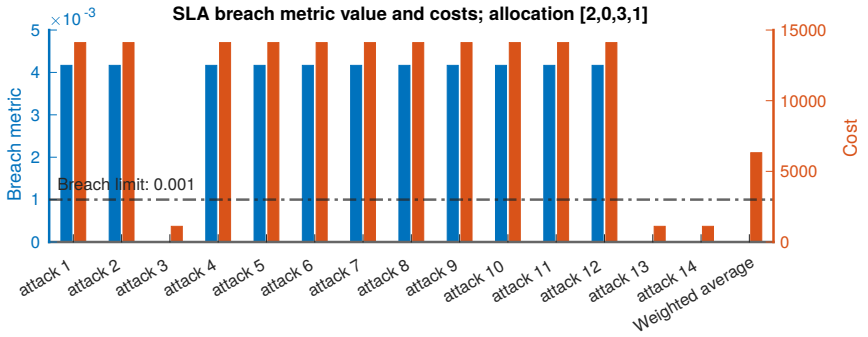
In cases where transition times are significant, a further stage could be added: the faster steady-state analysis using memoisation can be run first, with a transient analysis of the QNs (using simulation or the LINE fluid solver) applied in a region around the solution obtained from the steady-state analysis, to find if the optimal outcome is affected by the transitions. A neighbourhood for the solution could be estimated by applying perturbations to the disruption-time limit $\beta$ so as to capture the possible errors from omitting the transitions, finding optima for these perturbed versions of the problem, and then evaluating over this neighbourhood using the transient solution method.

## 8.6 Evaluation

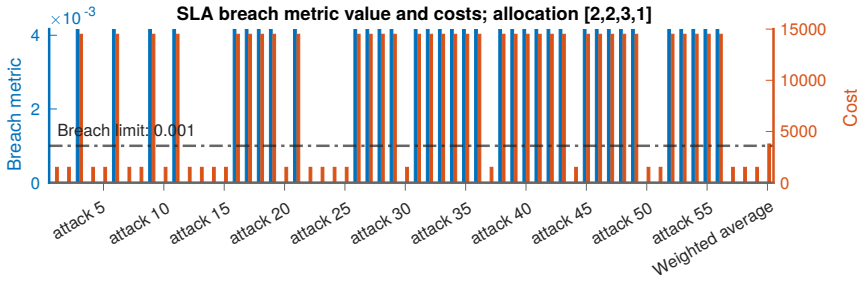
Our evaluation focuses on the optimal choices for redundancy in our case study. Sec. 8.6.1 discusses the results using our baseline values for model parameters; Sec. 8.6.2 examines how the different parameters impact the results; Sec. 8.6.3 investigates long-term maintenance costs using our TUL metric, and costs from reputation loss. Table 8.3 lists the parameter values and their ranges.

### 8.6.1 Baseline results

Fig. 8.4 shows results for the optimal server allocations in the "standard" non-diverse case (Fig. 8.4a), and with diversification (Fig. 8.4b). The optimal server allocation is given as a tuple, e.g. [2,0,3,1],




(a) Non-diversified specification




(b) Diversified specification

Figure 8.4: Optimal allocations in the case study, baseline, $\beta = 0.001$

where the first element denotes the number of application servers (in DG node $S_A$) of the regular type, and the second element with alternative servers (for diversity). The third and fourth elements are the numbers of database servers (in $S_{DB}$) and databases (in $DB$), respectively.

The figures depict the breach metric value for a given attack outcome with blue bars (left $y$-axis) and the costs during the outcome using orange bars (right $y$-axis). The number of attack outcomes depends on the number of exploits that can be detected, which increases with the number of servers in the allocation, and with diversification. In Fig. 8.4a there are 14 different outcomes to the three attack scenarios, while the diversified case in Fig. 8.4b leads to 59 outcomes.

Diversification helps bring down the expected cost of the attack scenario by around 40%. An indication of how this occurs can be gained from looking at the number of outcomes without breach. While under the non-diversified configuration only three of the attack outcomes (21%) avoid an SLA breach, diversification leads to 24 outcomes (41%) without a breach. Although the likelihoods of the different outcomes are not equal but vary as explained in Sec. 8.5.3, this is still indicative of how diversification mitigates costs: reducing the likelihood of outcomes causing a penalty.

### 8.6.2 Sensitivity to parameter changes

The modelling contains several parameters whose values can affect whether redundancy can provide benefits during attacks. We now analyse the sensitivity of the results to variations in the speed of recovery and $\beta$, the probability of detecting a move step by an attacker, and the cost of diversification.

#### 8.6.2.1 Time to recover and breach tolerance $\beta$

*The relationship between the time to recover servers ($t_r$) and the level of disruption tolerance in the SLA ($\beta$) is a key determinant of the effectiveness of redundancy strategies*. As attackers cause a disruption only when they have acquired all the privileges they require, the speed of recovery determines the duration of disruption. Whether this duration is long enough to breach the SLA depends on $t_r$ in relation to $\beta$.

Fig. 8.5 shows the effect of different recovery times for a given server allocation, other parameters being kept constant at their baseline levels. The figure shows how the impact of one specific attack outcome changes when $t_r$ is varied among $\{3,5,10,72\}$. To keep the figure clear, we only show one particular outcome (full success of Scenario 1), omitting outcomes with partial successes. The upper panel of Fig. 8.5 shows the average throughput of the system over time when $t_r$ is varied, illustrating the large impact the recovery time has on performance over time when an attack takes place. The lower panel is formatted similarly to Fig. 8.4, but focuses on one attack outcome (full success of Scenario 1) and varying $t_r$. We see that the higher recovery times of 10 and 72h cause breaches of the SLA. Note that, as the SLA penalty does not vary with the extent of the disruption, once the SLA is breached the duration of the recovery no longer affects the cost incurred.

Table 8.4 shows results when $t_r$ varies between 3 and 72 hours. The table considers two SLA disruption limits: up to 1% of the 1-month SLA window (7h 12min) in columns three to five ($\beta$ = 0.01), and up to 0.1% of the time window (43min) in the last three columns ($\beta$ = 0.001). It shows the optimal allocations for each recovery time in both the diversified (Div.) and non-diversified (N.D.) configurations. For each allocation, the expected overall cost is shown along with the cost savings obtained from diversification relative to the optimum in the non-diversified case. Due to the higher cost of the

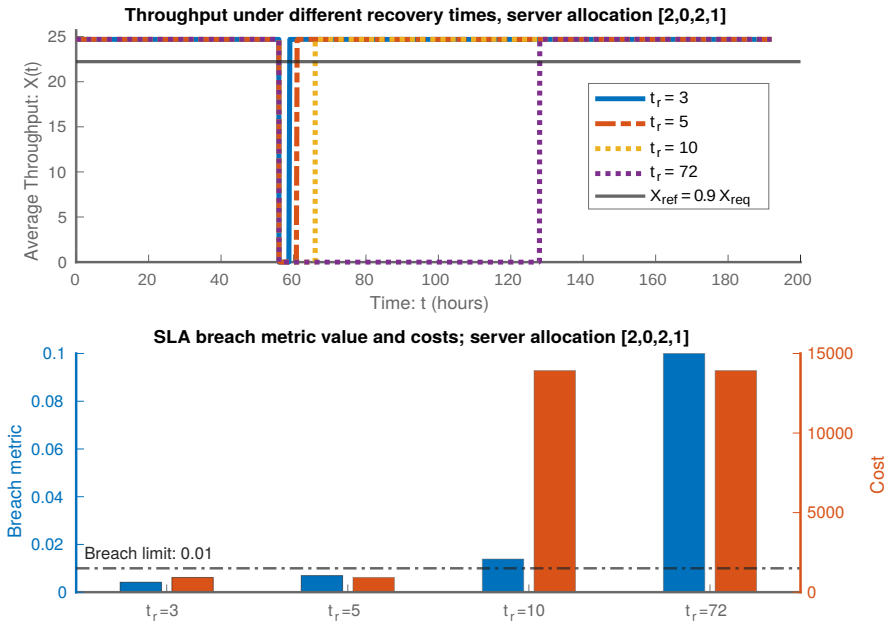



Figure 8.5: Impact of varying recovery time $t_r$ in the case study

alternative servers, the allocation cost in the diversified case is often higher than in the non-diversified one. However, the non-diversified configuration has a lower overall expected cost than the diversified one only when $\beta = 0.01$ and $t_r$ is 3 or 5. In these cases, the disruption from the attacks is not large enough to breach the SLA. In contrast, when the recovery time is longer or the disruption limit $\beta$ is lower, the attacks cause an SLA breach. In these cases, *diversification provides savings of 41.1% in the cost of the attack relative to the non-diversified optimum*, and 45.2% relative to the reference.

Note that the results obtained are the same in all cases where the SLA is breached. This is due to: a) the compensation being based on credit levels instead of varying in proportion to the extent of performance reduction, and b) the SLA having a single credit level. Thus $t_r$ and $\beta$ only impact whether the contract is breached or not, not the extent of the penalty. Often, in real cases, SLAs can have multiple credit levels, so the penalty increases in steps until the maximum credit level is reached. After that is reached, higher disruption duration no longer increases the penalty, as in our results.

The key finding from Table 8.4 is that the benefit from redundancy depends on whether the speed of recovery is faster than the SLA disruption tolerance $\beta$. When $t_r$ is short enough to be within the disruption tolerance, diversity is not beneficial as no penalty is incurred. When $t_r$ is longer than the disruption tolerance, a penalty is possible under some attack outcomes, and redundancy can yield cost savings. Moreover, diversification (Div.) provides benefits beyond those from redundancy with-

Table 8.4: Sensitivity to recovery time $t_r$

| | | $\beta$ = 0.01 | | | $\beta$ = 0.001 | | |
|---|---|---|---|---|---|---|---|
| $t_r$ | **Alloc. type** | **Optimum allocation** | **Exp. cost** | **% diff.** | **Optimum allocation** | **Exp. cost** | **% diff.** |
| 3;5 | Div. | $[1, 1, 2, 1]$ | 934 | +1.0 | $[2, 2, 3, 1]$ | 3795 | -41.1 |
| | N.D. | $[2, 0, 2, 1]$ | 925 | - | $[2, 0, 3, 1]$ | 6447 | - |
| | Ref. | $[2, 0, 2, 1]$ | 925 | - | $[2, 0, 2, 1]$ | 6931 | +7.5 |
| 10;72 | Div. | $[2, 2, 3, 1]$ | 3795 | -41.1 | $[2, 2, 3, 1]$ | 3795 | -41.1 |
| | N.D. | $[2, 0, 3, 1]$ | 6447 | - | $[2, 0, 3, 1]$ | 6447 | - |
| | Ref. | $[2, 0, 2, 1]$ | 6931 | +7.5 | $[2, 0, 2, 1]$ | 6931 | +7.5 |

Notes: $p_d$ = 0.3, $c_{as}/c_s$=1.05

out diversification (N.D.). Diversity gives greater flexibility to reduce the expected penalty costs by avoiding the penalty in some cases, and by reducing the probability of successful attacks. Thus, *when planning for resilience, attack impacts can be reduced by improving robustness to the attack by adding redundancy with diversity, or by improving recovery speed.* Alternatively, if the company can insist on an SLA with a loose disruption tolerance $\beta$, they have no cost-incentive to invest in mitigating attack impacts, as attacks do not breach the SLA.

#### 8.6.2.2 Detection probability

The probability of detection of attack steps $p_d$ affects the relative likelihoods of the attack outcomes. As such, it only impacts the optimal allocation if some attack outcome breaches the SLA. If an optimal allocation can avoid penalties altogether, varying $p_d$ has no impact on the optimum. We investigated the impact of $p_d$ in both Case 1 and Case 2 defence assumptions, described in Section 8.5.3. As discussed above, the values of $\beta$ and $t_r$ determine if a penalty is possible. Thus, we discuss only cases when an SLA breach can occur; the no-penalty cases always favour keeping the reference allocation.

The results are shown in Table 8.5a (Case 1) and 8.5b (Case 2). For each $p_d$ (1st column), these tables give the results for three types of allocation: diversified optimum (Div.), non-diversified optimum (N.D.), and a reference allocation (Ref.) that has not been optimised. Column 2 gives the respective allocation type, whilst column 3 shows the optimal server allocation tuple. The expected costs are shown in the 4th column, whilst the 5th and 6th columns show cost differences relative to N.D. arising from penalty and allocation costs, respectively. Finally, we show the percentage difference of the expected cost relative to the optimum without diversification (N.D.) in column 7.

Table 8.5: Sensitivity to detection probability $p_d$

| Detection prob. ($p_d$) | Alloc. type | Optimum allocation | Exp. cost | Cost difference | | % diff. |
|---|---|---|---|---|---|---|
| | | | | penalty | alloc. | |
| 0 | Div. | $[2,2,2,1]$ | 11425 | -2889 | 759 | -15.7 |
| | N.D. | $[1,0,1,1]$ | 13555 | - | - | - |
| | Ref. | $[2,0,2,1]$ | 13925 | 0 | 370 | +2.7 |
| 0.1 | Div. | $[2,2,3,1]$ | 7934 | -3309 | 389 | -26.9 |
| | N.D. | $[2,0,3,1]$ | 10854 | - | - | - |
| | Ref. | $[2,0,2,1]$ | 11143 | 474 | -185 | +2.7 |
| 0.3 | Div. | $[2,2,3,1]$ | 3795 | -3041 | 389 | -41.1 |
| | N.D. | $[2,0,3,1]$ | 6447 | - | - | - |
| | Ref. | $[2,0,2,1]$ | 6931 | 669 | -185 | +7.5 |
| 0.5 | Div. | $[2,2,3,1]$ | 2119 | -2224 | 389 | -46.4 |
| | N.D. | $[2,0,3,1]$ | 3954 | - | - | - |
| | Ref. | $[2,0,2,1]$ | 4175 | 406 | -185 | +5.6 |
| 0.7 | Div. | $[2,2,2,1]$ | 1531 | -1317 | 389 | -37.7 |
| | N.D. | $[2,0,2,1]$ | 2459 | - | - | - |
| | Ref. | $[2,0,2,1]$ | 2459 | 0 | 0 | 0 |
| 0.9 | Div. | $[2,2,2,1]$ | 1322 | -434 | 389 | -3.3 |
| | N.D. | $[2,0,2,1]$ | 1367 | - | - | - |
| | Ref. | $[2,0,2,1]$ | 1367 | 0 | 0 | 0 |

Notes: $c_{as}/c_s$=1.05, $P(A|R) = 0.5$, $\beta = 0.001$, $t_r = 3$, Case 1

(a) Case 1 defence

| Detection prob. ($p_d$) | Alloc. type | Optimum allocation | Exp. cost | Cost difference | | % diff. |
|---|---|---|---|---|---|---|
| | | | | penalty | alloc. | |
| 0 | Div. | $[2,2,2,1]$ | 11425 | -2889 | 759 | -15.7 |
| | N.D. | $[1,0,1,1]$ | 13555 | - | - | - |
| | Ref. | $[2,0,2,1]$ | 13925 | 0 | 370 | +2.7 |
| 0.1 | Div. | $[4,2,3,10]$ | 6152 | -39 | -722 | -11.0 |
| | N.D. | $[11,0,3,9]$ | 6913 | - | - | - |
| | Ref. | $[2,0,2,1]$ | 11143 | 7560 | -3330 | +61.2 |
| 0.3 | Div. | $[2,2,3,4]$ | 2911 | 105 | -352 | -7.8 |
| | N.D. | $[6,0,3,4]$ | 3158 | - | - | - |
| | Ref. | $[2,0,2,1]$ | 6931 | 5253 | -1480 | +119.5 |
| 0.5 | Div. | $[2,2,3,2]$ | 1939 | -101 | 19 | -4.1 |
| | N.D. | $[4,0,3,2]$ | 2021 | - | - | - |
| | Ref. | $[2,0,2,1]$ | 4175 | 2894 | -740 | +106.6 |
| 0.7 | Div. | $[2,1,2,1]$ | 1377 | -40 | 9 | -2.2 |
| | N.D. | $[3,0,2,1]$ | 1408 | - | - | - |
| | Ref. | $[2,0,2,1]$ | 2459 | 1236 | -185 | +74.6 |
| 0.9 | Div. | $[2,1,2,1]$ | 1129 | -1 | 9 | +0.7 |
| | N.D. | $[3,0,2,1]$ | 1121 | - | - | - |
| | Ref. | $[2,0,2,1]$ | 1367 | 431 | -185 | +21.9 |

Notes: $c_{as}/c_s$=1.05, $P(A|R) = 0.5$, $\beta = 0.001$, $t_r = 3$, Case 2

(b) Case 2 defence

Table 8.5a shows that, with Case 1 defence, there are no benefits to adding identical (i.e. non-diverse) redundant servers to a cluster which can be a final target of an attack, which is the case for $S_A$ and $DB$. However, there is benefit to redundancy for intermediate servers, which are only used as a stepping stone in the attack. For example, for server $S_{DB}$ one redundant server ensures two functional copies remain during an attack.[3] Consequently, when the attack scenarios can lead to SLA penalties, having three DB servers is optimal when $p_d \in \{0.1, 0.3, 0.5\}$. Additionally, when diversification is beneficial, the optimal allocation is one where there are as many alternative servers as regular ones, thus ensuring that the alternative servers cluster can meet the performance requirements on its own.

Table 8.5b shows the sensitivity to detection probability $p_d$ with the Case 2 defence model. When the attacks can lead to a penalty (when $\beta$ is tight relative to $t_r$), there is extensive scope for benefiting from redundancy, both with diversity (Div.) and without (N.D). The optima without diversity suggest adding a very large number of redundant services, with the most being 9 redundant application servers (11 in total) when $p_d = 0.1$. This is very different from the situation in Table 8.5a, where redundancy to the application server was never useful if it was done without diversity. The difference arises from the differing assumptions on the granularity of defence actions – while in Case 1 the application servers of a given type (regular or alternative) would all become unavailable if an attacker managed

[3]This is because the attacker needs only one server as a stepping stone, and will not risk detection to obtain additional ones.

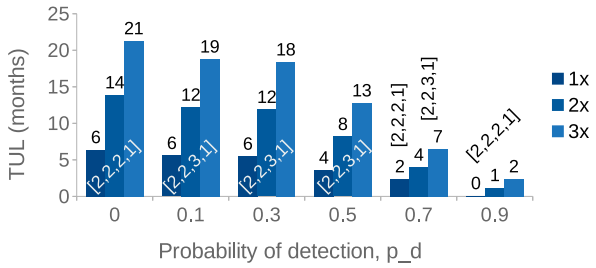
to breach one of them (as defence would be unable to stop the further compromises), with Case 2 all redundancy additions increase the likelihood of catching and stopping the attacker before they affect enough servers. It is also notable that with lower detection probabilities, diversification of the application service leads to a lower expected cost than redundancy without diversity, with a much smaller number of servers required. However, with high detection probability, the added maintenance cost from diversification can overcome the other benefits relative to redundancy without diversity. In the table, the 5% cost premium assumed to exist on the alternative server types means redundancy without diversity is marginally more cost efficient than with diversity when $p_d$ = 0.9.

We also find a large impact on the optimum allocation from the probability of detection. There appear to be two effects – first, redundancy overall becomes less beneficial with a higher probability of detection, although even with at $p_d$ = 0.9 redundancy still yields a cost saving of over 20% relative to the reference allocation. Second, redundancy in services that are only compromisable via long attack paths ($S_{DB}$ and $DB$, third and fourth elements of the allocation tuples) disappears completely when $p_d$ increases beyond 0.5, while some redundancy in the application server remains even at $p_d$ = 0.9.

To summarise, *redundancy with diversity is beneficial whenever SLA penalties cannot be fully avoided.* Additionally, the probability of detection affects the benefits obtained from redundancy. *A higher detection probability reduces the expected costs from attacks, and thus the benefit obtained from redundancy.* This suggests that companies could benefit from increasing their detection capabilities when cost effective, although this can be difficult to achieve in practice due to the cost of dealing with high levels of false positive alerts.

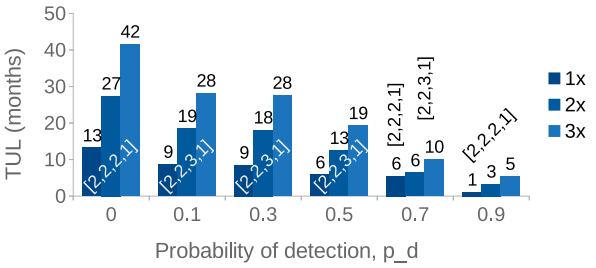
#### 8.6.2.3 Cost of diversification

We analysed the sensitivity of our results to different levels of diversification costs, varying the $c_{as}/c_s$ ratio, i.e. the per-unit costs of the alternative servers relative to the regular ones. The detailed results are shown in Appendix A.3. In summary, we find that the level of diversification cost does not impact which allocation is found optimal unless the cost difference between the server types is large, which in our analysis occurred once the cost ratio $c_{as}/c_s$ reached 9, i.e. when the alternative server is 9 times as expensive as the regular type. However, the benefit from diversification is affected, as the excess




Case 1, β = 0.001; ref. alloc. [2,0,2,1]; penalty multip. 1x-3x

(a) Minimum-cost reference allocation [2,0,2,1]




Case 1, β = 0.001; ref. alloc. [3,0,2,1]; penalty multip. 1x-3x

(b) Alternative reference allocation [3,0,2,1]

Figure 8.6: Months until cumulative maintenance cost exceeds attack-time benefit (TUL). Labels are optimal diverse allocations, and bar colours show different multipliers on the baseline SLA penalty.

maintenance costs of the alternative servers need to be balanced against the benefits observed during attacks. The impact of costs over time is explored further below.

### 8.6.3 Long-term maintenance costs

The analysis thus far has considered the expected cost during an attack, but not the costs during normal operation when no attacks occur. However, choosing a redundant allocation over one that is cheaper depends on the expected attack likelihood/frequency, and the excess maintenance cost. Thus, the choice depends on the expected cost of the allocations over a longer period mostly spent without attacks. The key parameter in this evaluation is the frequency of attacks in the chosen time frame.

We estimate the longer term benefit of the redundancy approaches by estimating the longest time between attacks before the cumulative excess maintenance costs exceed the benefit during attack periods. For this we use the TUL metric we introduced in (8.3). Its advantage is enabling accounting for the long term without requiring an estimate of attack frequency as model input before analysis.

The TUL value of an allocation is sensitive to the various model parameters, the key ones being the penalty for breaching the SLA and the reference allocation. Fig. 8.6 shows how TUL varies with $p_d$ and the SLA penalty. We can see that higher $p_d$ correlates with lower TUL because the benefit from diversity gets smaller with $p_d$, as shown in Sec. 8.6.2.2, reducing the numerator in (8.3). Similarly, higher levels of penalty correspond to higher TUL estimates. This is because the increased penalty makes the expected loss from an attack higher, which provides greater scope for cost savings from

diversity, and increases the numerator of TUL.

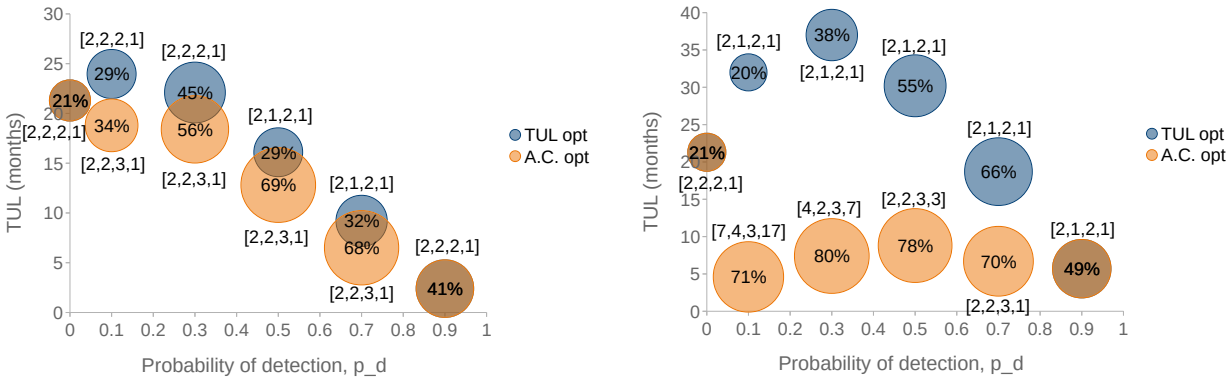
The reference allocation has a sizeable impact on TUL, because of its effect on the excess maintenance cost in the denominator of (8.3). We tested this by varying our reference in Fig. 8.6. In our case study, the minimum-cost allocation yielding the required performance during normal times is $[2, 0, 2, 1]$. However, the baseline used by [KB03] was $[3, 0, 2, 1]$. Using the latter as the reference, in Fig. 8.6b, leads to significantly larger TUL estimates. Thus, *organisations must choose their reference point carefully*: the cheapest allocation may not reflect the need for redundancy, and may bias the results.

To put the TUL values of Fig. 8.6 into perspective, [BLDC20] found that there were on average 1.37 successful attacks per year with significant ("very high profile") impact and 5.94 with moderate impact, for large international companies who were not among the top security performers. This amounts to *one significant breach every nine months, and a moderate breach every two months*. For top security performers, moderate breaches occurred once every 13 months, while significant ones were rare (0.05 per year). This shows that attack frequencies vary across companies, and having a reasonable estimate is important to determine if server diversification is worthwhile.

#### 8.6.3.1 TUL optimisation

The results in the previous sections aim to minimise cost during attack periods i.e., during the month of the attack. However, this ignores the excess maintenance paid during months when attacks do not occur. This is why Section 8.5.8.1 introduced a second optimisation problem: maximising the TUL metric. This section focuses on how the results from maximising TUL differ from minimising attack incident costs. Apart from the differences pointed out here, the results for the two optimisation problems are qualitatively similar. Thus, tables for TUL optimisation results are left in Appendix A.5.

The comparison of the two optimisation approaches is shown in Fig. 8.7. The key observation is that TUL maximisation typically leads to the use of fewer servers than attack cost minimisation, as this saves on maintenance costs and allows for a higher TUL. For Case 1 defence, in Fig. 8.7a, the difference in TUL is 5.2 months for $p_d = 0.1$, and decreases to 2.7 months for $p_d = 0.7$. This increased TUL comes with a higher during-attack cost, with the saving relative to the reference allocation re-




(a) Case 1 defence

(b) Case 2 defence

Figure 8.7: TUL and cost savings when minimising attack costs (A.C. opt, orange bubbles) or maximising TUL (blue). Bubble sizes represent attack cost saving relative to the reference allocation [2,0,2,1]. Labels show the server allocation. The results shown are for SLA credit level of \$150 per customer (3x multiplier), with $\beta = 0.001$ and $t_r = 3$.

duced by 5-40%-points compared to attack cost optimisation. In general the two optimisations show the extremes between which a suitable approach can be chosen based on a desired TUL level or preference for savings during attacks.

The results for Case 2 defence, in Fig. 8.7b tell a similar tale, but with more extreme differences between the two approaches. This is due to the defence being able to stop attacks between exploits within clusters of servers of the same type. In this context, attack cost minimisation leads to very high redundancy when detection probability is small but non-zero. This results in large attack cost savings but limited TUL. The TUL maximisation approach finds optimal allocations with an even lower number of servers than with Case 1 defence, with a higher TUL but often lower attack cost savings than in Fig. 8.7a.

Taking the two subfigures together, we see that the extent to which benefits from diversity can be realised depends on how frequent attacks are expected to be. If the detection capability is very high, $p_d = 0.9$, diversity is only beneficial if attacks are very frequent (on average every 2.4 months for Case 1, and 5.7 months for Case 2). Such frequencies may appear excessive, but do occur [BLDC20]. With lower levels of detection, a high TUL is possible, so diversity is viable for a wide range of attack frequencies. Moreover, when the expected attack frequency is between the TUL values provided by the two optimisation approaches, there is scope to choose an allocation between the two optima, trading-off some TUL for added cost savings during attack.

#### 8.6.3.2 Other costs from attacks

Underestimating the costs from an attack will lead to underestimating the TUL and thus underinvestment in redundancy. In the sections above, we have considered that attack incident costs arise only from SLA breaches. However, costs related to regulatory penalties or business lost due to impacts on reputation can behave very differently. First, such added costs can occur after an attack even if the SLA is not breached. Second, these costs can increase with the length of disruption even if the SLA penalty is capped. Third, even if these costs occur only when an SLA is breached, they would add to the cost of the incident. Thus, excluding them from the analysis means the TUL estimate is biased down, underestimating the incentive to reduce attack impacts. Here, we consider the case of loss of business to show how costs can be added to our analysis, and how the TUL results are affected.

We use loss of business as an approximation for reputational costs. The loss of revenue due to a cyber attack was estimated by [BLDC19] to represent 20% of all costs of a cyber attack among large international companies. However, as this estimate is relative to the total cost of an attack, which is not clear in advance and thus hard to use as a model input, we approximate the impact based on loss of customers. Based on a study of the cost of data breaches of large international companies, [Pon18] find the average abnormal churn in customers after an attack to be 3.4% of customers. As this is an estimate of loss of customers from data breaches, which cause larger than average loss of revenue than general attacks, we scale it down using loss of revenue estimates. The share of total attack cost arising from loss of revenue was 40% across all firms in [Pon18] (data breaches only), but 20% across all cyber attack types in [BLDC19]. Thus, we obtain an estimate for loss of customers from attacks that do not lead to a data breach by halving the [Pon18] churn rates. That is, 1.7% on average.

We test two approaches, one where the loss of customers is binary, occurring at a constant rate of 1.7% for any successful attack, and a second one where the customer loss varies based on the extent of disruption following a logistic function. We model the latter with the following function:

$$f(x) = \frac{0.034}{1 + e^{-500(x-0.01)}} \tag{8.5}$$

where $f(x)$ is the rate of loss of customers as a function of the size of the disruption $x$, measured

by the share of time when system performance is below the required level ("disruption-time share"). This provides a smooth sigmoid shape with values between 0 and 0.034, where the midpoint of 0.017 is reached at a disruption-time share of $0.01$.

A key parameter here is the length of time for which the number of customers will be dampened. We assume that the impact of the attack on customer numbers will be temporary, so eventually numbers will return to the level that would have been expected in the absence of the attack. We consider durations of 18 and 36 months, and estimate the loss as the duration (in months) times the monthly loss based on the revenue that would have been made from the customers. We discount the losses during future years (from month 13 onward) at a rate of 3% per year.

Tables 8.6a and 8.6b show results for our redundancy optimisation when we add this cost arising from loss of customers, using the fixed customer loss share of 1.7% in the former and the variable loss share approach in the latter table. The tables show the diversified allocation results for both the attack cost minimisation (shown as "Div. (a)") and TUL maximisation ("Div. (t)"). The "N.D" allocations are the same with both optimisation approaches, so are shown in the tables only once for each case.

Table 8.6: Sensitivity to loss of customers

| Loss months | Alloc. type | Optimum allocation | Exp. cost | Cost difference incid. | Cost difference alloc. | % diff. | TUL |
|---|---|---|---|---|---|---|---|
| $\beta$ = **0.01** | | | | | | | |
| 18 | Div. (a) | $[2,2,3,1]$ | 2196 | -924 | 389 | -19.6 | 1.0 |
| | Div. (t) | $[2,1,2,1]$ | 2490 | -250 | 9 | -8.8 | 1.3 |
| | N.D. | $[2,0,3,1]$ | 2731 | - | - | - | - |
| | Ref. | $[2,0,2,1]$ | 2749 | 203 | -185 | +0.7 | - |
| 36 | Div. (a) | $[2,2,3,1]$ | 2863 | -1808 | 389 | -33.1 | 2.8 |
| | Div. (t) | $[2,2,2,1]$ | 3106 | -1380 | 204 | -27.5 | 3.6 |
| | N.D. | $[2,0,3,1]$ | 4282 | - | - | - | - |
| | Ref. | $[2,0,2,1]$ | 4495 | 398 | -185 | +5.2 | - |
| $\beta$ = **0.001** | | | | | | | |
| 18 | Div. (a) | $[2,2,3,1]$ | 4493 | -3964 | 389 | -44.3 | 7.4 |
| | Div. (t) | $[2,1,2,1]$ | 7001 | -1076 | 9 | -13.2 | 9.0 |
| | N.D. | $[2,0,3,1]$ | 8068 | - | - | - | - |
| | Ref. | $[2,0,2,1]$ | 8755 | 872 | -185 | +8.5 | - |
| 36 | Div. (a) | $[2,2,3,1]$ | 5160 | -4848 | 389 | -46.4 | 9.3 |
| | Div. (t) | $[2,1,2,1]$ | 8312 | -1316 | 9 | -13.6 | 11.3 |
| | N.D. | $[2,0,3,1]$ | 9619 | - | - | - | - |
| | Ref. | $[2,0,2,1]$ | 10501 | 1067 | -185 | +9.2 | - |

Notes: $c_{as}/c_s$=1.05, $P(A|R) = 0.5$, $t_r$ = 3, $p_d = 0.3$, penalty cost \$50, Case 1

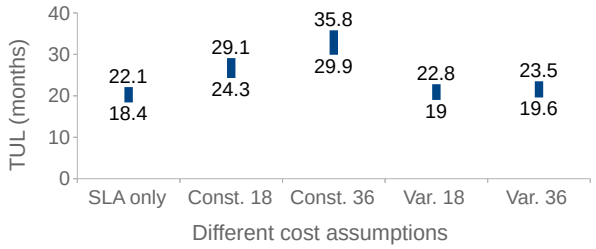
(a) Fixed loss share

| Loss months | Alloc. type | Optimum allocation | Exp. cost | Cost difference incid. | Cost difference alloc. | % diff. | TUL |
|---|---|---|---|---|---|---|---|
| $\beta$ = **0.01** | | | | | | | |
| 18 | Div. (a) | $[1,1,2,1]$ | 1122 | 1 | 9 | +0.9 | -1.1 |
| | Div. (t) | $[2,0,2,1]$ | 1112 | - | - | - | - |
| | N.D. | $[2,0,2,1]$ | 1112 | - | - | - | - |
| | Ref. | $[2,0,2,1]$ | 1112 | - | - | - | - |
| 36 | Div. (a) | $[1,1,2,1]$ | 1301 | 0 | 9 | +0.7 | -1 |
| | Div. (t) | $[2,0,2,1]$ | 1292 | - | - | - | - |
| | N.D. | $[2,0,2,1]$ | 1292 | - | - | - | - |
| | Ref. | $[2,0,2,1]$ | 1292 | - | - | - | - |
| $\beta$ = **0.001** | | | | | | | |
| 18 | Div. (a) | $[2,2,3,1]$ | 3867 | -3136 | 389 | -41.5 | 5.7 |
| | Div. (t) | $[2,1,2,1]$ | 5771 | -852 | 9 | -12.7 | 6.9 |
| | N.D. | $[2,0,3,1]$ | 6614 | - | - | - | - |
| | Ref. | $[2,0,2,1]$ | 7118 | 689 | -185 | +7.6 | - |
| 36 | Div. (a) | $[2,2,3,1]$ | 3935 | -3227 | 389 | -41.9 | 5.9 |
| | Div. (t) | $[2,1,2,1]$ | 5906 | -876 | 9 | -12.8 | 7.2 |
| | N.D. | $[2,0,3,1]$ | 6614 | - | - | - | - |
| | Ref. | $[2,0,2,1]$ | 7298 | 710 | -185 | +7.8 | - |

Notes: $c_{as}/c_s$=1.05, $P(A|R) = 0.5$, $t_r$ = 3, $p_d = 0.3$, penalty cost \$50, Case 1

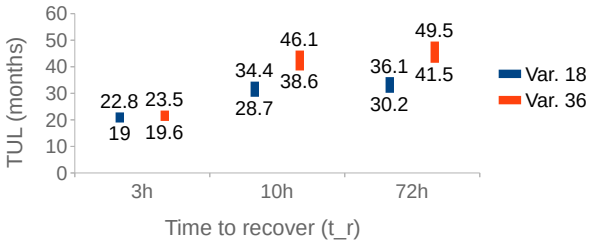
(b) Variable loss share

The main observation is that the added costs increase the incentive to invest in diverse redundancy. The largest difference relative to our earlier results occurs when $\beta = 0.01$, a case for which no redundancy was found preferable in our previous cases, as shown in Fig. 8.5 and the accompanying discussion. As we can see from the top part of table 8.6a with $\beta = 0.01$, with a fixed loss of business




Case 1; penalty multiplier 3x; p_d = 0.3; ref. alloc. [2,0,2,1].
Top of line: TUL maximization; lower end: A.C. minimization.

(a) TUL ranges, varying assumptions on customer loss




Case 1; penalty multiplier 3x; p_d = 0.3; ref. alloc. [2,0,2,1].
Top of line: TUL maximization; lower end: A.C. minimization.

(b) TUL ranges varying $t_r$; variable customer loss

Figure 8.8: TUL ranges, varying customer loss assumptions and $t_r$; $\beta$ = 0.001

due to attack incidents, redundancy could lessen the expected costs from attacks even when they do not lead to breach in the SLA conditions. However, with the parameter values used in Table 8.6a, the TUL is low, so the longer-term maintenance cost of the redundancy makes investing in it unwise unless attacks are expected very frequently. By comparison, with the variable incident costs shown in Table 8.6b, when $\beta = 0.01$, the added cost from the loss of customers is so small that diversification is loss-making, as was the case in previous sections where we only considered SLA penalty costs.

Apart from the slight changes in the expected costs and TUL values, the results are qualitatively similar to those in previous sections. The key impact of ignoring these additional costs would be an underestimation of the benefits of redundancy, as the cost impact and TUL would be underestimated. Accordingly, we now investigate how the TUL estimates are affected by the two types of added cost.

Fig. 8.8a shows the ranges in TUL using the two optimisation approaches when the cost assumptions are changed. The lower ends of the columns show the TUL metric when attack impact cost is minimised (Eq. 8.2), and the top ends are the values with TUL maximisation (Eq. 8.4). The five categories are "SLA-only" with attack costs only arising from SLA penalties, "Const. 18" and "Const. 36" show cases with a constant loss of customers in addition to SLA penalty, and "Var. 18" and "Var. 36" are with the customer loss varying with the disruption. The 18 and 36 in the labels refer to the duration of the customer loss in months. The situation pictured is for $3x$ penalty multiplier, $p_d = 0.3$, $t_r = 3$ and $\beta = 0.001$.

It is clear from Fig. 8.8a that the TUL estimates are affected by the inclusion or exclusion of added cost estimates. The impact is especially large with the constant loss assumption, as this causes a large

cost impact from loss of customers if any level of disruption is experienced due to a cyber attack. This added cost enables large savings on attack incident costs relative to the reference allocation, yielding large TUL values (up to 29.1 and 35.8 months for 18-month and 36-month customer losses, respectively). The impact from the variable customer loss ("Var. 18" and "Var. 36") is not as large, as the extent of disruption is at the low part of our logistic function (Eq. 8.5) due to the 3h recovery time, which keeps the loss of customers small.

Section 8.6.2.1 showed that in the case where attack costs are only from SLA penalties, varying $t_r$ only impacted the results if it causes a move between the classes of SLA tolerance and penalty levels. The same is true if other costs are added which are invariant to disruption level, such as with the constant loss of customers assumption, but not when the additional costs vary with the extent of disruption. Figure 8.8b shows how the results with our variable customer loss assumption are impacted by $t_r$. The graph has two categories of columns, one for 18-month and another for 36-month loss of customers, with the three categories of time to recover $t_r$ on the x-axis. We observe that when the costs from disruption vary according to our logistic function, the room to benefit from redundancy can increase dramatically if the speed of recovery is slow.

To summarise the findings of the analysis in this section, the key results are: a) *Including loss of business costs can increase TUL estimates significantly*: in our experiments, increases were up to 63% when a constant loss of customers (1.7% of customers for three years) was assumed (as seen in Fig. 8.8a), and up to 124% when loss of customers varied with disruption length (Fig. 8.8b); b) *If business loss varies with the length of disruption, speed of recovery has a large impact*: the TUL increase ranged from 3% with $t_r = 3$ up to 124% with $t_r = 72$ (Fig. 8.8b).

### 8.6.4 Scalability

Earlier on, we have illustrated our method and evaluated its output on a small scale case study. We built our case-study on this particular system as it is well known [Sta02], and its performance characteristics and model are understood and validated [KB03]. It is difficult to implement much larger case studies because the attack and system information required to build attack graphs are usually

confidential and not publicly available. However, our method can scale in practice to much larger systems for reasons pertaining to the granularity of the model, the theoretical complexity of the impact evaluation algorithm, and our observed computational time in the analysis reported here.

**Model granularity:** Each node in our graph represents either a set of privileges acquired by the attacker (in the attack progression model) or a component with an arrival queue in the production model. Thus, each node can represent components at various levels of aggregation, e.g. an individual process on a computer, a cluster of servers, or even an entire sub-network. The same model (with different parameters) can therefore be used to represent a system where a server is an entire cluster of servers and a database is a distributed database. This can be taken advantage of where the attack progression allows, e.g. if the attack spreads and impacts servers in groups of 10, then modelling at the level of individual servers is not needed.

**Theoretical complexity:** The most computationally expensive part of our impact estimation is the performance evaluation. While the fluid approximation used in the QN evaluations itself scales well, it needs to be done numerous times as a performance estimate is required for each different attack outcome under each candidate allocation considered during the optimisation. Consequently, the complexity of our optimisation depends mainly on two factors: 1. The solution space determined by the upper and lower bounds given to the optimisation; 2. The number of attack outcomes to evaluate for each candidate allocation.

The number of the attack outcomes to investigate depends substantially on whether attacks can be detected (and prevented) when moving laterally from one server to another. Servers that do not share vulnerabilities require the attacker to use new attack steps, and thus the attack can be potentially detected and prevented. For servers that share the same vulnerabilities, in particular identical servers, we distinguish two cases: **Case 1** where the attack cannot be detected or prevented when moving from one server to the other – this results in all identical servers being treated as a cluster, they are either all compromised or not; and **Case 2** where we assume that it is possible to detect and prevent the attack when moving from one server copy to another. Case 2 requires accounting for attack steps being detected or not detected for every server instance. The complexity therefore increases combinatorially when this assumption is considered for server allocations at all layers (application, processing,

databases). Complexity wise, this is a worst-case assumption that rarely occurs in practice. If an attack cannot be detected when it compromises an instance of a server, it will typically not be possible to detect it on identical copies of that server as the attack steps can be replicated identically. The only exception are alerts that are triggered when a certain volume of events has been reached, for example, connections to unopened ports or failed authentication attempts. Repeating these actions too often may result in an alert, whilst a small number of such events may be considered benign. However, such circumstances are rare, and even then each individual server instance would not make a sufficient difference. A more realistic view would be to group servers according to the alert thresholds. This would be, in essence, a case somewhere between Case 1 and Case 2 depending on the size of the groups. Our complexity analysis shows that Case 1 scales well to larger systems, Case 2 becomes difficult to scale, and solutions in-between depend on the size of the server groups considered. More detail is provided in Appendix A.1.

**QN model accuracy:** Large-scale models pose challenges in achieving high accuracy in performance prediction. While simulations may be carried out for increased precision, they are often too slow for a single scenario evaluation. The use of fluid solvers offers a solution to this issue by the mean-field approximation theory from [PC17], which guarantees that as the system scale grows (in terms of the total number of jobs and servers, i.e. processing units), the model precision also grows: asymptotically, the sample paths of the exact model converge to the fluid approximation [PC17, Thm. 1]. Since the number of ordinary differential equations does not change with increasing number of jobs and servers, the fluid solution approach to QNs can scale efficiently. However, if more stations are added to the model, the number of equations will grow linearly. In models with many equations, lumping approaches may be applied to control the cost of the model solution [CTTV16].

**Observed run time:** The time required to compute the system performance under a given "effective allocation" of servers in our example was on average 3.08s, with the QN model fluid approximation. The bounds analysis used in the first stage of the optimisation took 0.0056s for the same calculation, per effective allocation. This was achieved on a standard desktop computer with an Intel i7-6700 CPU with 4 cores (8 threads) at 3.4GHz clock speed, and 16GB of RAM. Note that our analysis is run "off-line", i.e. it does not need to be run in real time during the attack, as it is intended to form a part of preparing the system to withstand attacks.

## 8.7 Conclusion

Amongst our findings, two stand out as the most important for using redundancy (and diversity) for attack mitigation. First, when the losses are based on SLA penalties, there is a cut-off point for when redundancy can be beneficial and when it cannot, depending on the speed of recovery relative to the disruption tolerance of the SLA. A sufficiently fast recovery can avoid SLA penalties. However, when penalties can occur in response to attacks, redundancy with diversity can mitigate losses.

Second, choosing a server allocation (with and without diversity) depends on both the cost during an attack, and the added maintenance costs over time. We have introduced the Time Until Loss (TUL) metric to track the balance of these two types of costs. It measures the frequency of attacks required for a given server allocation to be financially viable in comparison to a reference allocation. The size of the potential loss from an attack has a large impact on the TUL values observed. Large losses make diversification viable even at lower attack frequencies. Further, TUL enables informed choices between focusing on keeping maintenance costs low or emphasising reductions to attack related costs: The solutions to our two optimisation problems, minimising attack time costs and maximising TUL, can provide a range of (diverse) redundancy allocations among which to choose based on their TUL values relative to the organisation's expectation about the frequency of attacks.

Detection probability has a significant impact. Increases in $p_d$ essentially tighten the limits of affordability for diversity, reducing expected costs under all allocations and thus limiting the scope for diversification to yield sufficient attack cost savings to justify the added maintenance costs.

Finally, the various parameters can impact the results in different directions. There are no clear rules-of-thumb and there is a paucity of tools and models to analyse the economic impact from cyber attacks, which can lead to ad-hoc decisions on security investment and limited provisioning of systems for resilience to attacks. Our work provides a first such methodology and analysis leading the way for other tools and models to be developed. While the parameter values used in this work are drawn from the available literature, they may differ across industry sectors and organisations. Conducting the analysis in specific contexts would be relatively straightforward by using context specific parameters, and the methodology can be adapted to accommodate variations in the cost or attack models.

# Chapter 9

# Applications: Countermeasure selection

This chapter, based on our work in [SML19], shows how our resilience impact assessment methodology can be applied to the problem of reactive countermeasure selection during attacks. Specifically, we propose an attack countermeasure selection approach based on cost impact analysis that takes into account the impacts of actions by both the attacker and the defender.

We consider a networked system providing services whose functionality depends on other components in the network. Using our impact assessment methodology, we model the costs and losses to service availability from compromises and defensive actions to the components. We show that while containment of the attack can be an effective defence, it may be more cost-efficient to allow parts of the attack to continue further whilst focusing on recovering services to a functional state. Based on this insight, we build a countermeasure selection method that uses our impact assessment methodology to choose the most cost-effective action based on its impact on expected losses and costs over a given time horizon. We evaluate our countermeasure selection approach using simulations in synthetic graphs representing network dependencies and vulnerabilities, and performs well, in terms of overall cost-effectiveness, in comparison to two alternatives.

## 9.1 Introduction

Organisations providing services across the Internet, or otherwise connected to an external network, must invest in cyber security to protect their services, both to lower the risk of attacks, and to reduce the impact when they occur. However, determining the correct application of this investment is not as straightforward as attempting to secure everything fully. The literature on cyber security investment has shown that a company should never invest to the extent as to cover all potential vulnerabilities or weaknesses [GL02], and that a company should retain some part of the security investment budget for when an attack event has taken place [GLL03, CPG18]. Furthermore, given the existence of unknown vulnerabilities and exploits [NKK+17], not all attacks can be fully stopped from occurring. Additionally, security actions can cause limitations to availability, e.g. reductions to communication between systems or users, or loss of compatibility between software applications due to patching.

Given the possibility that an attack against an organisation's system will occur, some cyber-security investment should be allocated into improving the ability of the system to cope during an attack, and recover to normal functionality. The actions organisations take to mitigate the effects of attacks is the subject of the literature on attack countermeasure selection, reviewed by [NPMK17]. In contrast to existing work on countermeasure selection, we take a longer term view by focusing on cyber-resilience, forcing the countermeasure selection to consider impacts over time, and recovery dynamics.

In this chapter, we propose an approach to countermeasure selection based on cost impact assessment of both attacker and defender actions, in a medium-to-long-run setting including recovery dynamics. The impacts of actions are estimated using an implementation of our impact assessment methodology. The particular implementation used here builds on the attack impact analysis approach by [AJ17, AJPS11], but in contrast to their work we asses attack impacts on system output performance over time, and use the assessment for choosing countermeasures. In addition, and in line with the methodology proposed, our approach implements a cost model, specifying costs for loss of node availability, the countermeasures, and recovery actions. This extends the impact assessment beyond attack impact alone, including the cost impact of defender's actions alongside that from attack steps. The structure enables dynamic semi-automatic countermeasure selection based on the overall costs of alternative defender actions. This allows our method to make more nuanced countermeasure decisions,

balancing attack containment and focus on recovery according to the situation to yield cost-efficient countermeasure strategies.

Our main contribution on countermeasure selection, beyond the use of our impact assessment methodology, is an approach to evaluating the effectiveness of each countermeasure based on its expected impact on the system. This involves the effects of the countermeasure on the possible evolution of the attack, and on the network's service provision, both immediately and in a longer term. We achieve this by using the system attack graph to form expectations of possible attack paths and their likelihoods, and employing our impact assessment methodology to estimate the cost impact of the different states the system could enter within a few steps from the current state.

We test our approach by simulations on a small sample graph and on synthetic graphs, against two alternative methods. The results suggest that our approach provides more cost-efficient countermeasure selection than the alternatives tested, especially when attacks are detected with a delay. We made comparisons to an alternative countermeasure selection that employs the principles of the attack impact assessment proposed in [AJ17, AJPS11]. The comparisons show that, for countermeasure selection, our approach adds considerable improvements to cost effectiveness and average service performance.

In summary, our contributions include: 1. Extending the attack impact assessment model by [AJ17, AJPS11] into a more general model for impact assessment including defender actions, via modelling service losses and the costs of the countermeasures, and by enabling the modelling of recovery, which the original formulation does not support; 2. Introducing an approach to countermeasure selection based on estimating the expected impacts of actions; 3. Showing that considering recovery and the costs of actions over time can yield a more efficient countermeasure selection.

The rest of this chapter is structured as follows: Sec. 9.2 discusses related work relevant to this application; Sec. 9.3 introduces our impact analysis model, including concepts and definitions from [AJ17, AJPS11] which we build upon; Sec. 9.4 introduces our countermeasure selection approach; Sec. 9.5 evaluates the method against two alternatives, using simulations; and Sec. 9.6 concludes.

## 9.2 Related work

In addition to works discussed with respect to their relation to our overall methodology in Chapter 2, there are parts of literature related to this specific application not discussed earlier, but which are relevant here. In addition, how the application discussed here relates to works in the resilience literature needs to be briefly clarified separately, as it involves reactive actions during an attack event.

Our work described in this chapter differs from most of the existing literature on resilience by focusing on the actions and investment choices during an ongoing event (attack) and the recovery phase, instead of preparatory planning and capability investment. This focus is intended to address the evolving nature of systems, and adaptation to conditions such as loss of confidentiality or network unavailability. Most cyber resilience works have focused on the planning and design stage, such as [SHÇ+10, GMG+16, SC17]. Additionally, papers considering reactive response and recovery apply to narrow settings unrelated to our work. For example, the approaches by [CRC+15, YWR18] apply to settings where a control action to correct for a deviation from desired performance is easy to determine in advance, and to apply automatically using controllers.

In the cyber security literature, *countermeasure selection* refers to approaches to choose actions to counter security events (cyber attacks) [NPMK17]. These methods focus on defence against attack events, and as such mainly involve the stages before and during an attack: modelling the system, the potential attacks and countermeasures; identifying the attack and choosing a countermeasure. We believe that by introducing longer-term resilience considerations and using impact assessment, countermeasure choices can be made more cost-efficient. Our proposed approach has similarities to the countermeasure selection techniques in [KD15] and [SSLHC16] in the use of AGs and DGs, using costs to quantify attack impact, and basing countermeasure decisions on costs. While these works focus on stopping attacks or reducing the risk of them reaching pre-determined goals, we aim to find a strategy that is efficient over longer time, considering the cost of recovery to previous functionality. Additionally, we model time dynamics explicitly, and approach cost and component valuation via the impact on the final services as opposed to giving each component an intrinsic value.

## 9.3 Impact analysis modelling

### 9.3.1 Attack and dependency graphs

The attack graphs used here follow the definition of Vulnerability Dependency Graph from [AJ17], a compact AG representation used in their work on attack impact assessment, where the nodes represent vulnerabilities, and edges are security conditions that enable the exploit of such vulnerabilities. We gave this as Definition 4.2 in Chapter 4, the attack modelling chapter. We chose this particular AG type here as it enabled us to make more direct comparisons to the work in [AJPS11, AJ17] that is relevant here.

The DGs in this work represent the **availability** dependencies of services provided by software applications in a network. The DG nodes are applications that provide services across the network, such that the functionality of applications elsewhere in the network depends on them. At a given time $t$, the availability of a given node is measured as the service level it provides at time $t$ as proportion of the level normally expected of it, so that 1 represents full availability, and 0 means the node is unavailable. Dependencies between services are represented by directed edges between the DG nodes.

In this application, we estimate performance impacts using the status function approach by [AJPS11, AJ17], discussed in Sec. 5.2.1, and the dependency graph definition associated with it. Thus, for the DG, we use the definition of Generalised Dependency Graph from [AJ17]. We gave this in Sec. 5.2.1 as Definition 5.2, and the related dependency functions ($f_r$, $f_d$, $f_s$) in equations (5.1)-(5.3).

### 9.3.2 Attack impact analysis

Our impact assessment approach used for this application builds on the *impact assessment graph* (IAG) proposed in [AJ17], with combined use of an AG and a DG for analysing attack impacts. We propose a slight simplification relative to [AJ17]. Specifically, in the IAG we use an AG without the compromise time-windows feature in the original, as the separate handling of time in our model makes this feature unnecessary.

**Definition 9.1** ***(Impact Assessment Graph, modified from [AJ17])*** *Given a vulnerability dependency graph $A = (V, R)$ and a generalised dependency graph $D = (H, Q, \phi)$, an impact assessment graph is a 4-tuple $(A, D, F, \eta)$ where: $F \subseteq V \times H$; $\eta : F \to [0, 1]$ is a function that associates with each pair $(v, h) \in F$ a real number in the $[0, 1]$ interval representing the percentage reduction in the availability of network component $h$ caused by vulnerability exploit $v$.*

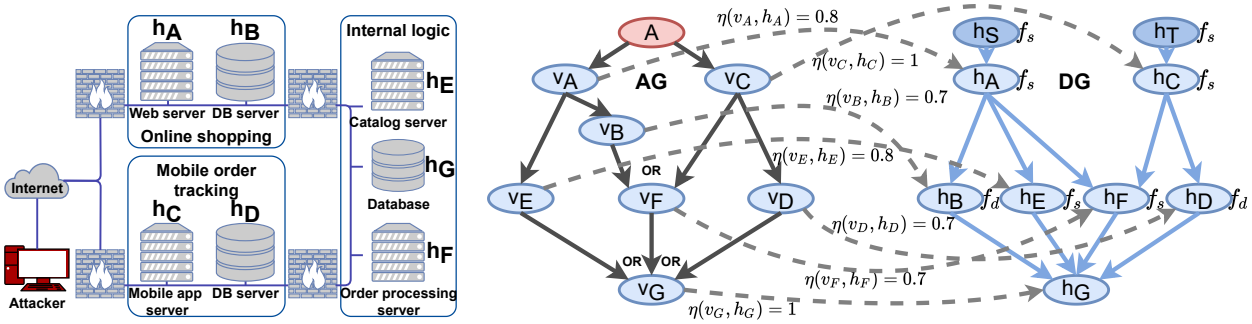
Effectively, the IAG consists of both the AG and the DG in full, and connections in between them in the form of the function $\eta$ that describes how the availability of the components in the DG are affected by vulnerabilities in the AG.

In this work, we define how the availability status of a node evolves over time with relation to the availabilities of its dependencies using the *network status function* approach by [AJPS11, AJ17]. We described this approach in Sec. 5.2.1, and gave the network status function definition, Definition 5.3. We shall not repeat the definition and general discussion, but focus on how it is applied to the model here. The functional form that we use for the network status function (repeat of Equation 5.4) is:

$$s(h, t) = f(s(\dot{h}, t)) \prod_{v \in V_{e,t}} (1 - \eta(v, h)) \tag{9.1}$$

where $V_{e,t}$ is the set of AG nodes that are in an exploited state at time $t$; $\eta(v, h) \in [0, 1]$ is the availability effect that the exploit of vulnerability $v$ has on component $h$ (0 for no effect, 1 for fully unavailable); $\dot{h}$ is the set of components that $h$ is dependent on; and $f(s(\dot{h}, t))$ is the availability effect on $h$ from its dependencies. Note that a component recovery is represented as the removal of a vulnerability $v$ from the exploited set $V_{e,t}$, while patching vulnerability $v$ removes it the from the full set $V$. The status of network component $h$ is composed of two effects on availability: 1. Compromise effect (direct): $(1 - \eta(v, h))$, the effect of compromise (vulnerability exploit) in the AG node $v$ which corresponds to component $h$; 2. Dependency availability effect (indirect): $f(s(\dot{h}, t))$, the effect of unavailabilities in the components that $h$ is dependent on, of type $f_r$ (redundancy), $f_d$ (degradation) or $f_s$ (strict dependence).

The final key component for attack impact analysis using the IAG is the utility derived from the components in the dependency graph: $\forall h \in H, u(h)$ gives the utility for node $h$. The utility is,




(a) Topology of the sample network

(b) Impact assessment graph for the sample network

Figure 9.1: Sample system network topology, and the IAG

in effect, the value of the service provided by a given node at each time unit. The work in [AJPS11, AJ17] assumes that all the DG nodes are given a utility value which remains static during the analysis. By contrast, we believe that intermediate services provide value only when at least a part of the final service to which they contribute is online (available). Therefore, we only set utility values for the final service nodes (e.g. $h_S$ and $h_T$ in the system in Fig. 9.1b below), with the value of the other nodes only reflecting the impact they have on the value arising from the final services. In this way, we use the dependency structure to dynamically determine the value of intermediate services.

We use the example system from [AJPS11, AJ17] to demonstrate our approach, and how it differs from what they proposed. Fig. 9.1a recreates the network topology of the sample system from [AJ17, Fig. 1]. The sample represents the network of a small organisation with two final services, an online shopping web service ($h_A$) and a mobile order tracking app ($h_C$), their local cache databases ($h_B$ and $h_D$ for online shopping and order tracking, respectively), and a separate subnetwork for the internal logic ($h_E$ and $h_F$) and a central database ($h_G$) powering the services.

The dependency structure and potential attack paths in the sample network from [AJPS11, AJ17] are shown in the IAG in Fig. 9.1b. The AG is on the left, and DG on the right. The AG nodes represent vulnerabilities in the network, with the attacker entry points shown as edges from node $A$. The DG nodes are network components: $h_A$ to $h_G$ provide intermediate services (internal services), while $h_S$ and $h_T$ are customer-facing services (the product of the organisation). The exploit of a vulnerability $v_i$ affects the availability of the corresponding component $h_i$, and the availability of components which are dependent on $h_i$. The dashed lines in Fig. 9.1b show the availability impact of a vulnerability, $\eta(v_i, h_i)$, for example $\eta(v_F, h_F) = 0.7$ means $h_F$ will lose 70% of its availability when $v_F$ is exploited

(unless there is also a dependency availability effect on $h_F$).

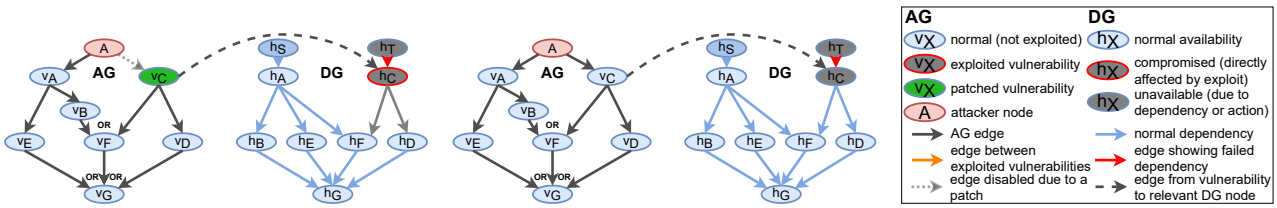
In contrast to [AJ17, AJPS11], we propose a comprehensive cost model that takes into account the costs of node unavailability, costs of different countermeasures, and recovery costs. Doing so, we move to a more general impact assessment by including the estimation of the impacts of defensive actions as well as those of the attacker. Countermeasure decisions can then be made based on expected costs over time, instead of only relying on the projected impacts of attacker actions. Our work uses the impact analysis framework as part of an approach to countermeasure selection, employing the impact estimates to determine which countermeasure is the most cost effective at a given attack situation. For this purpose, leaving out the analysis of costs of the countermeasure actions could lead to inefficient choices of countermeasures, as the application of a countermeasure can reduce service availability.

### 9.3.3 Performance measurement and resilience

We measure the performance of the networked system with an overall service provision status, *service performance* (SP), which we introduced as equation (5.5) in Chapter 5. It is the weighted sum of the statuses of the client-facing services (the "product" of the organisation; nodes $h_S$ and $h_T$ in our case study here), weighted by their relative utilities for the organisation. To limit repetition, we omit the equation from here, and refer the reader to equation (5.5) in Section 5.2.1 for the details. When observed over time, this metric can be used to measure the system's resilience based on a performance-curve approach to resilience in the spirit of [BCE+03], as in Fig. 2.1.

### 9.3.4 Attacks, countermeasures and recovery

As discussed in Sec. 4.2.1, we model **attacks** as sequences of attack steps, i.e. atomic exploits of a single vulnerability in an AG, with potential to lead to further steps. They enter the network via particular *entry nodes* which are directly exploitable by the attacker. For example, in Fig. 9.1b, an attack can start by an exploit of $v_A$ or $v_C$. The potential next steps are then determined by the AG structure, e.g. if $v_A$ was exploited, the possible next steps are $v_B$ and $v_E$. In the analysis done in this




(a) Defender patches $v_C$ ($h_C$ not cleaned automatically). Takes $t_P$ time units to come to effect.

(b) Defender recovers $h_C$. $h_C$ remains temporarily unavailable. After $t + t_R$, $h_C$ and $h_T$ are up again.

(c) Graph legend

Figure 9.2: Patching and recovery actions and their effects on the system

work, we assume that each exploit also leads to a disruption of the DG node they relate to (shown by the hyphenated arrows from AG to DG nodes), so there are no distinct move and disruption steps.

We make the following assumptions on how an attack step proceeds. The time to exploit a vulnerability $v_i$ (for $i \in V$) is $t_{v_i} = 1$ for all vulnerabilities, so only one attack step can occur during a time unit $t$. The probability of an attacker taking an attack step at a given time period is $p_{step}$. Additionally, the parameter $p_{fast-step}$ reflects the chance that a countermeasure/recovery application is slower than the attacker's next step, so the attacker's next step gets executed just before the defender's one. This provides some uncertainty to the timing of events, to avoid limiting us to the case where the defence always beats the attacker to the next step, which seems unrealistic.

**Countermeasures** (CMs) are actions aimed at reducing the impact of an attack. In general these could be of two types, ones affecting the network's security capabilities which can be taken before an attack but not during one (capability changes: redundancy additions, back-ups), and others which can be taken at any time (dynamic countermeasures). In this work, we consider a type of dynamic countermeasure, *patching* vulnerabilities. The effect of patching is to remove a vulnerability node from the AG, restricting potential attack paths. A patching action requires $t_P$ units of time to implement, and comes at a cost consisting of a direct cost and a service impact, introduced in Sec. 9.3.5 below. Fig. 9.2a illustrates the case where $v_C$ is patched after an exploit. Note that patching does not clear the related DG node $h_C$ from compromise, as recovery is handled separately.

**Recovery** refers to actions that the network owner uses to recover the functionality of components (DG nodes) compromised by vulnerability exploits. To simplify the analysis, we consider the case where there is only one type of recovery method, akin to component replacement. This, in effect,

replaces a compromised component with a clean and working instance, with the same functionality (and vulnerabilities) as before, assumed to take $t_R$ time, and cost $c_R$. Both the time and cost values are assumed equal across components. In terms of the graphs, this corresponds to making a DG node fully functional and the corresponding AG vulnerability not exploited (although it will remain in the AG, so may be re-exploited). Fig. 9.2b shows the case of recovery of $h_C$ after compromise by exploit of $v_C$. More detail on recovery modelling is provided in Sec. 9.3.6.

### 9.3.5 Costs of actions

Our modelling of costs contains direct costs for each action, i.e. node recovery cost $c_R$ and patching cost $c_P$, in addition to their impact on the system production, i.e. the availability of final services, due to dependencies. While the direct costs do matter, the key element of our cost modelling is the loss to the provision of final services caused by component unavailability.

**Service loss due to unavailability of DG node $h$ at time $t$:**

$$g(h,t) = \sum_{h_j \in H_S} \Big( u(h_j) \cdot \big( f_s(s(\dot{h_j}, t) \mid s(h,t) = s(h,t-1)) - f_s(s(\dot{h_j}, t) \mid s(h,t)) \big) \Big) \tag{9.2}$$

where we have used $s(h_j, t) = f_s(s(\dot{h_j}, t))$ for $h_j \in H_S$, which follows from (9.1) and the observation that the customer-facing service nodes $h_S$ and $h_T$ do not have direct exploits (so only the dependency effect counts for them). The function $g(h,t)$ is the impact of $h$'s deviation, at time $t$, from its previous observed availability level onto the availability of services.

The **costs of the countermeasure and recovery actions** consist of two parts: the direct cost for the action ($c_R$ for recovery of a node, $c_P$ for patching a node, $c_D$ for disabling), and the cost of unavailability of the network components that are directly impacted by the action. For example, the observed (after the fact) cost of patching vulnerability $v_i$ at time $t$ is given by:

$$c_{patch}(v_i, t) = c_P + \sum_{h_j \in M(v_i)} \left( \sum_{\tau=t}^{t+t_P} g(h_j, \tau) \right) \tag{9.3}$$

where $v_i$ is a node in the AG, $c_P$ is the direct cost of patching. $M(v_i)$ represents the set of elements in

the DG that are adjacent to the AG node $v_i$, that is, the components directly affected by the vulnerability $v_i$. In other words, these are the software where the vulnerability exists, and where the patching of $v_i$ takes place. The current time period is $t$, and $t_P$ represents the time units required for the patching. The inner summation adds together the cost of component $h_j$ being unavailable from $t$ to $t + t_P$. The observed costs due to disabling and node recovery work in a similar manner.

Note that (9.3) shows the calculation of the *observed cost* when we know the path of any attack steps and defender actions and therefore know $g(h_j, \tau)$ for $\tau = [t, t + t_P]$. For estimating the cost of an action beforehand, we require an expectation of the state of the model in terms of attacker steps and defender actions during the periods in question, as we discussed in Sec. 7.2. In practice, our approach is to estimate the benefit of an action in terms of expected trajectories for the system state, as we explain below in Sec. 9.4.

### 9.3.6 Recovery process

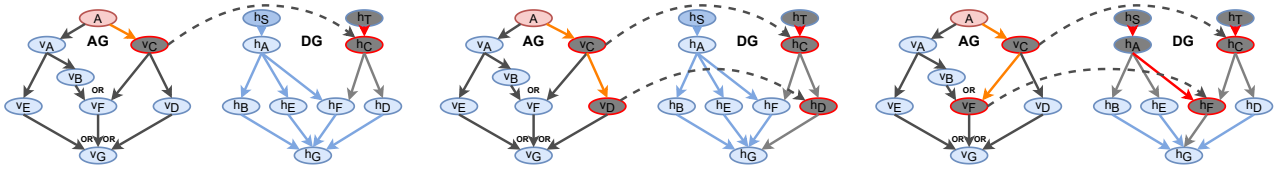
We evaluate our model in a setting with automatic recovery decisions, where the choice of whether to recover a node or not is based on the likely benefit versus the costs of the recovery strategy. We keep this recovery process separate from countermeasure choices, which simplifies expectation formation for CM selection. Node recovery is done if it leads to a reduction in losses exceeding the recovery cost. The loss reduction from recovering node $h$ at time $t$ is:

$$LR(h, t) = Loss_{\neg R}(h, t) - Loss_R(h, t) \tag{9.4}$$

where $Loss_R(h, t)$ is the loss with recovery, which is:

$$Loss_R(h, t) = \sum_{\tau=t}^{t_{max}} E_t(g(h, \tau | s(h, \tau) = 0)) + \sum_{\tau=t_{max}+1}^{t_{horizon}} E_t(g(h, \tau | s(h, \tau) = 1)) \tag{9.5}$$

where $t_{max} = t + t_R + t_P$ is the time it would take to recover and patch the node (if necessary), and $t_{horizon}$ is the last time period in the time horizon considered. This metric consists of the loss from service unavailability from time $t$ until $t_{max}$ when the recovery is finished, and from time $t_{max} + 1$




(a) $v_C$ exploited: $h_C$ and $h_T$ down (b) Attacker exploits $v_D$: $h_D$ goes down (c) Attacker exploits $v_F$: both services down

Figure 9.3: Sample attacker moves and their effects on the system

onward when the node will be assumed recovered (and not re-compromisable). Loss without recovery, $Loss_{\neg R}(h,t)$, is given by:

$$Loss_{\neg R}(h,t) = \sum_{\tau=t}^{t_{horizon}} E_t(g(h,\tau|s(h,\tau)=0)) \tag{9.6}$$

### 9.3.7 Sample impact analysis for CM selection

Choosing countermeasures by focusing solely on stopping the progress of an attack means ignoring: 1. the cost of countermeasure actions (direct and indirect), and 2. recovery of the network toward a desired state. Disregard of these aspects can lead to choices that are not efficient in the longer term. To demonstrate, let us assume that the network from Fig. 9.1b experiences an attack exploiting $v_C$ at $t = 0$; this situation is shown in Fig. 9.3a. As $v_C$ is exploited, $h_C$ becomes unavailable, and so does $h_T$ due to its dependency on $h_C$. While the exploit of $v_C$ already causes considerable damage in terms of service losses, once an attacker has exploited $v_C$, it can further exploit $v_D$ (Fig. 9.3b) or $v_F$ (Fig. 9.3c). While the exploit of $v_D$ only affects $h_D$ (as $h_C$ is down) resulting in no change in either cost or SP, an attack on $h_F$ takes down the remaining service $h_S$, causing full loss of service and cost due to service losses of $u(h_T) + u(h_S)$ per time period.

When applied repeatedly at each step, the marginal impact analysis approach from [AJ17, AJPS11] would choose to patch the most high-impact component that could be affected by an attack next. Given the compromise of vulnerability $v_C$, at the following time step $t = 1$, the next attack step could be $v_F$ or $v_D$, affecting components $h_F$ and $h_D$, respectively. Choosing based only on the potential impacts of these attack steps, the choice would be to patch $v_F$. Given this choice, at time $t = 1$

the attacker could proceed to exploit $v_D$, which was not patched. The appropriate reaction to this would be to patch $v_G$, given another round of marginal impact calculation. Finally, to return to full availability of services, nodes $v_D$ and $v_C$ need to be recovered ($v_c$ with patching, $v_D$ without).

While the above approach is sensible if we want to guarantee that the highest impact nodes $h_F$ and $h_G$ are never compromised, it may not be cost efficient over time. The costs of the actions, or recovery, are not considered by the above countermeasure strategy. However, the choice of actions and their order can have a large impact on the costs, especially those arising from service losses. Note that, as the service impact cost $g(h, t)$ for node $h$ is nonzero only when the node causes a change in the status of the services, the service impact of a given node $h$ can change over time. For example, the unavailability of $h_G$ only impacts services when $h_F$ is available. This changing loss impact can make a great difference on the overall cost of a countermeasure strategy. For example, if there is a low probability that the attack will have successfully moved to a different node before the next time step, it may be better to **not** take a countermeasure that contains the attack, but focus on recovering and patching $v_C$. In the worst case, the attacker has been able to move fast enough to exploit $v_F$ before $h_C$ goes offline for patching and recovery. If so, all services go down at $t = 1$, and $h_F$ will have to be recovered – but this happens without additional availability impact, as services will be down already. Again, in the worst case the attacker may move fast enough to compromise $h_G$ before the system $h_F$ is taken down for recovery, so $h_G$ will require recovery at $t = 3$. Even in this worst case the overall costs could be lower than with the approach from earlier, if the time to recover is lower than to patch $t_R < t_P$. However, this approach also benefits from there being a chance that the attacker will not successfully make another step before the defender reacts, meaning in the best case only $h_C$ has to be recovered and patched.

This sample highlights that, depending on the situation, it can be more cost effective to act reactively and rely on recovery capabilities, while sometimes proactively containing the attack is better. We built our CM selection approach on cost impact analysis to be able to find the approach that works best in a given situation.

**Algorithm 1** Choosing a countermeasure to implement

**Precondition:** $A$ is an AG, $D$ a DG. sortD($l$) sorts a list $l$ to a descending order of the values of elements in $l$. For graph $G \in \{A, D\}$, $G$.children($i$) returns a list of the children of node $i$ in graph $G$. The parameter $s$ represents the current model state, including AG node compromise and DG node availability statuses, and the current time $t$.

```
1: function CHOOSECOUNTERMEASURE(A, D, s)
2:     all_cms := {}
3:     nodes_list := nodes_of_interest(A, s)      ▷ list of AG nodes against which CMs are considered, in state s
4:     while nodes_list.size() > 0 do
5:         i := nodes_list.pop()
6:         cms_i := listCountermeasures(A, i, s)      ▷ list countermeasures for i at state s
7:         all_cms.append(cms_i)      ▷ append the list cms_i into all_cms
8:     end while
9:     sortD(all_cms)      ▷ sort the list in decreasing order of benefit
10:    return all_cms.pop()      ▷ return the highest-benefit CM
11: end function

12: function LISTCOUNTERMEASURES(A, i, s)      ▷ List the best countermeasures for a given AG node i
13:    countermeasure_list := {}
14:    nodes_list := A.children(i)
15:    while nodes_list.size() > 0 do
16:        j := nodes_list.pop()
17:        cm_benefit := B(j, s)      ▷ benefit (cost reduction) from patching j at state s
18:        countermeasure_list.push(j, cm_benefit)
19:    end while
20:    sortD(countermeasure_list)      ▷ sort the list in decreasing order of cm_benefit
21:    return countermeasure_list
22: end function
```

## 9.4 Countermeasure selection

Our approach to countermeasure selection relies on cost analysis of the impacts of defender actions and expected attack steps. We shall refer to this method as cost-impact countermeasure selection, *CICM* for short. Our CM selection algorithm, described in pseudocode in Algorithm 1, estimates the impact of all CMs that apply to AG nodes of interest (nodes that have been exploited) given the system state, and lists them in descending order of their overall benefit relative to what would be expected if no countermeasure was applied. The highest benefit CM is implemented, if its benefit is positive.

The effectiveness of each countermeasure is evaluated by comparing the expected costs of the CM to the benefits it is expected to yield, in the manner discussed in Sec. 7.3.1. This is represented by the function call $B(j, s)$ on line 17 of Algorithm 1. The benefit considers the direct cost of the action, and system production under two potential evolutions ("trajectories") of the system: the "baseline trajectory" reflects how an attack would be expected to proceed within the network in the absence of the countermeasure, and measures what the impact would be in terms of costs due to both availability

loss and recovery actions. The "deviating trajectory" given a CM action measures the expected system production impact when the countermeasure is applied. As a countermeasure initially requires a network component to be taken offline temporarily, we also make a distinction between the immediate and the longer-run impacts of the countermeasure. Therefore, our measure of the benefit arising from a countermeasure is given by[1]:

$$B(v_i, s) = trajD_{curr}(v_i, s) + eaf(v_i) \cdot (t_{horizon} - t) \cdot trajD_{LR}(v_i, s) - c_{cm} \tag{9.7}$$

where the first part is the trajectory difference currently (until the CM has been successfully applied), and the second part is an estimate of benefit in future time periods, consisting of the long-run trajectory difference $trajD_{LR}(v_i, s)$ multiplied by the expected frequency of future attacks exploiting $v_i$, $eaf(v_i)$, and the time periods left until the end of the horizon. Finally, $c_{cm}$ is the direct CM cost.

The expected attack frequency to node $v_i$, $eaf(v_i)$, is an estimate of the probability that the attacker will attempt to exploit node $v_i$ again. This is not based on the current compromise state, but on the probability to (re-)obtain the privileges for exploiting $v_i$ in the future via any path. We estimate $eaf(v_i)$ by approximating the probability of the shortest viable (not patched) path from the attacker node $A$ to $v_i$. This we calculate as the step probability $p_{step}$ to the power of the number of edges on the shortest viable attack path from $A$ to $v_i$. This approximation is by no means fully accurate, but captures the behaviour we want, and is considerably simpler and faster than the calculation of the exact probability. The benefit from the exact calculation would be limited given the randomness in the model and arising from its environment, and uncertainty over real attacker behaviour and intent.

The current and long-run trajectory differences are given by:

$$\begin{aligned} trajD_{curr}(v_i, s) &= devTraj_{curr}(v_i, s) - blTraj(v_i, s) \\ &= E(V(X_{v_i,s})) - E(V(X_{0,s})) \end{aligned} \tag{9.8}$$

$$\begin{aligned} trajD_{LR}(v_i, s) &= devTraj_{LR}(v_i, s) - blTraj(v_i, s) \\ &= E(V(X^{LR}_{v_i,s})) - E(V(X_{0,s})) \end{aligned} \tag{9.9}$$

[1]The notation here differs somewhat from that used in Sec. 7.3.1, to offer a simpler form in this specific application. However, equation (9.7) can easily be rewritten in the form of equation (7.14).

As explained in Sec. 7.3.1, $E(V(X_{v_i,s}))$ is the expected monetary value of system output arising from the outcomes of $X_{v_i,s}$, i.e. attack outcomes given a patching countermeasure is applied to AG node $v_i$ at system state $s$. Substituting these trajectory difference definitions into (9.7) yields the form of the countermeasure benefit introduced in equation (7.14) of Sec. 7.3.1.

The trajectories $blTraj(v_i, s)$, $devTraj_{curr}(v_i, s)$ and $devTraj_{LR}(v_i, s)$ are calculated by formulating an expected value for availability impact and costs in the next $k$ time steps. For $blTraj(v_i, s)$, this proceeds as follows: 1. From the AG, estimate what paths the attacker could follow in the next $k$ time steps starting from the current head of the attack, given system state $s$. Estimate the impact of each of these potential paths on service availability and on recovery costs. 2. Formulate expected values for the performance impact and costs in each of the time steps from $t$ to $t + k$, using the paths calculated in step 1 with the probabilities for each exploit, the probability for attack step being completed in each time step. An illustration of this is provided in Fig. 9.4, for the case where the vulnerability $v_C$ in the sample graph has been exploited at time period $t$. The first panel shows the AG situation at time $t$, the middle panel shows the possible attack paths in the next $k = 2$ time steps. The last panel shows the calculation of the expected values for the time periods $t, t + 1, t + 2$, where $V(s)$ refers to the value of system performance (in monetary units) at system state $s$.[2] While the state includes information on the unavailable DG nodes in addition to exploited AG nodes, for brevity, in the figure the states are labelled with only the exploited nodes listed, e.g. $s_{compr:\{v_C,v_D\}}$ stands for the state where the set of exploited nodes is $\{v_C, v_D\}$. The "deviating trajectory" estimates given a countermeasure $cm$ are calculated similarly to $blTraj(v_i, s)$, but assuming that a CM is applied. An additional difference is that when calculating the expected values (step 2 above), we further consider the possibility that the CM is not applied in time before the next step by the attacker, represented by the probability parameter $p_{fast-step}$. We calculate two deviating trajectories as the application of a CM causes temporary unavailability at first, before the longer-run effect is obtained.

While we calculate probabilities of various attack paths to formulate the trajectories, we do not solve for the whole network or the full time horizon, as that would not scale to realistic network sizes. Instead, the method focuses on "neighbourhoods" of the current attack, by looking at potential states

[2]Thus, in this work the expected value $E(V(X))$ is calculated for each time period separately, resulting in a time-series of expected costs, instead of for complete performance curves over time.

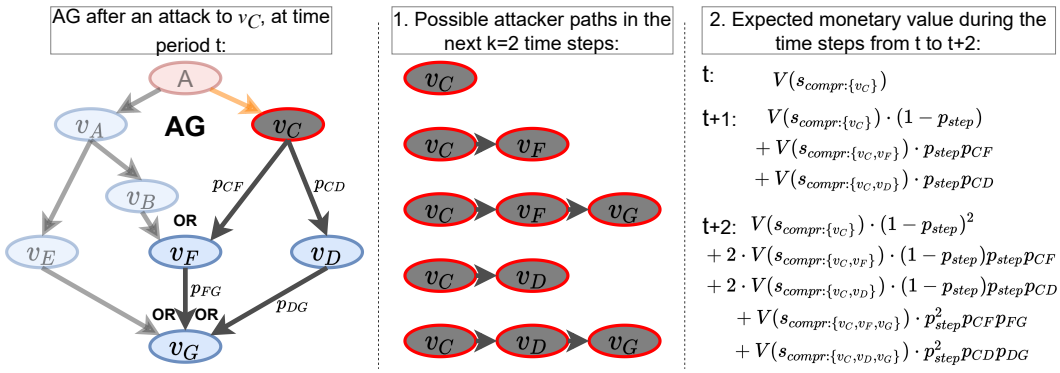



Figure 9.4: The stages of the baseline trajectory calculation

a few time-steps forward from the current boundary of the attack. To simplify the problem further, we do not consider patching at all nodes, but focus on those nearest to the entry (if patchable) [for future attacks], and those on the attack boundary [for containing the ongoing attack, and future attacks].

## 9.5 Evaluation

We used simulations to investigate the usefulness of our methodology for countermeasure selection, by comparison to two alternative strategies. The first comparison point is what we call the *attack impact approach* (*AIA*), which we built by adapting the attack impact assessment approach from [AJ17] to use in automated countermeasure selection. While their original work was not proposed for this specific purpose, we built a CM selection approach for patching actions using the main principles of their attack impact assessment method, using their marginal impact metric for patching choices. To use this in a setting where we care about performance over time, we also required the method to handle recovery, and thus AIA uses our version of the node status calculation (9.1). AIA can be considered a containment-focused approach, as it chooses patching actions that apply to the vulnerabilities exploitable next by the attacker, not ones that apply to already exploited vulnerabilities. This has the potential for stopping the attacker from compromising important nodes, but applying CMs on healthy components will lead to temporary availability losses, which can be costly.

The second comparison is to a strategy where a patch is always applied to the latest exploited vulnerability, without considering costs or alternative actions. We call this *PLE*, for "*patch latest exploit*". Instead of a containment focus, this approach favours blocking the last used attack steps from being

exploitable in the future, limiting future exposure while accepting current risk. PLE benefits from limiting the availability loss from CM actions, as actions are only applied to nodes that already suffer from reduced availability. While these two comparison points represent extremes in terms of containment and treatment focus, our approach is intended to be able to choose a cost-efficient approach in between these extremes based on the attack situation. The same model of the recovery process, described in section 9.4, is used with all of the approaches.

We ran tests in two settings: the sample graph in Fig. 9.1b, and randomly generated synthetic graphs. The sample graph provides a useful basis for comparisons as the attack impact assessment method by [AJ17, AJPS11] was demonstrated on it, while the generated graphs enable further testing in a larger number of situations and graphs of varying sizes.

The generated graphs are directed acyclic graphs of a specified size (in terms of nodes). For simplicity, the number of AG nodes was restricted to match that of the DG nodes. To control the structure, we have restricted the maximum number of parents of a node (nodes dependent on the node) to three. The connections between the AG and DG nodes are chosen at random, meaning that the vulnerabilities in the AG correspond to random system components (in DG), and the attack paths on the AG can be considerably different from paths in the dependency structure. Furthermore, in the DG, the number of dependencies (children) of each node are chosen at random, as are the dependency functions for each node. For each DG, two nodes are allocated as service nodes. In the AG, the number of children of a given node (number of further vulnerability exploits made possible by a given exploit) is drawn at random. To introduce attack entry points, we add a node representing the attacker's starting point, and its children, drawn among the other nodes, are the entry nodes. Additionally, probability values for the AG edges are set, representing the ease of exploiting a node, drawn from a distribution corresponding to that of the access complexity metric of CVSS scores [FIR16].

We simulated randomly generated attacks into each graph considered. The attacks follow a path towards a goal node, which is picked from among the AG nodes with the highest availability impact on the final services (the single highest impact one, or drawn among the shared highest impact nodes). Each chosen edge is picked from those along the paths to the goal, based on a draw between viable candidates, where the distribution is based on the edge probability values (representing access com-

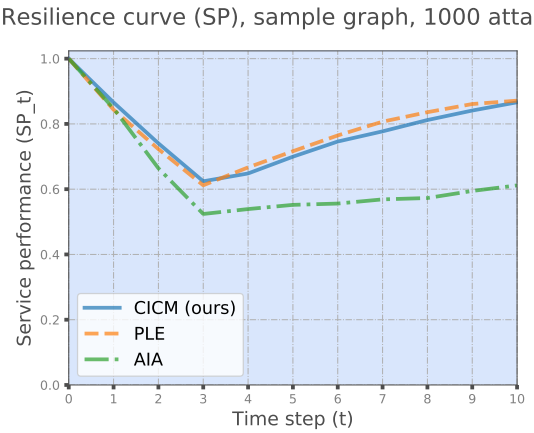
plexity). This creates variety across simulations, approximating different attacker choices based on e.g. different skill levels. One vulnerability exploit is allowed per time period. If a given step is not possible, due to a CM action, the attacker attempts another exploit that is on a path to the goal. If the goal becomes unreachable, the attack will stop.

Unless otherwise stated, the simulations use the following parameter values: one unit of time to compromise a vulnerability $t_{v_i} = 1$, two units to patch a vulnerability ($t_P = 2$), and one to recover a node $t_R = 1$. The direct costs are $c_P = 2$ for patching, and $c_R = 3$ for node recovery. There is a 30% probability that an attacker takes an attack step at a given time step ($p_{step} = 0.3$), and $p_{fast-step} = 0.3$, so there is a 30% chance that the application of a given CM/recovery action is slower than the attacker's next step.

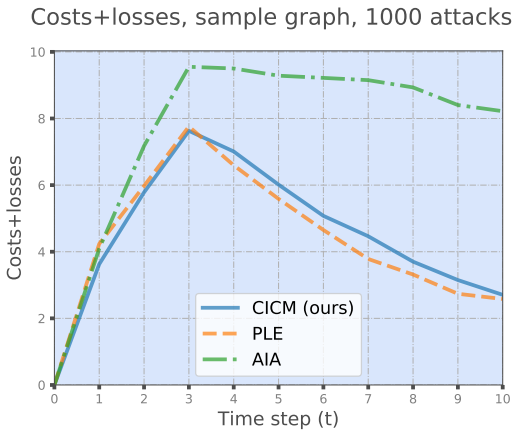
### 9.5.1 Results for the sample graph

The simulation results on the sample graph are shown in Fig. 9.5. The figure shows a comparison of our method to the two alternative CM schemes, both for the resilience curve (SP metric), and for costs and service losses over time. The curves represent the mean values for the metrics (SP, overall costs) across 1000 simulated attacks; the solid curve indicates our method (CICM), the dashed one is for PLE and the dash-dot line for AIA. The two upper panels show simulations where the attacks are detected immediately at the first step, and the defender actions (countermeasures, recovery) can be started immediately. By comparison, on the lower two panels, there is a delayed detection of the initial attack step, in which the attacker has already done one step before a step is detected (that is, the second step overall) and the defender actions start. We believe that this case is closer to reality, as attacks can remain undetected in a network for long periods of time.

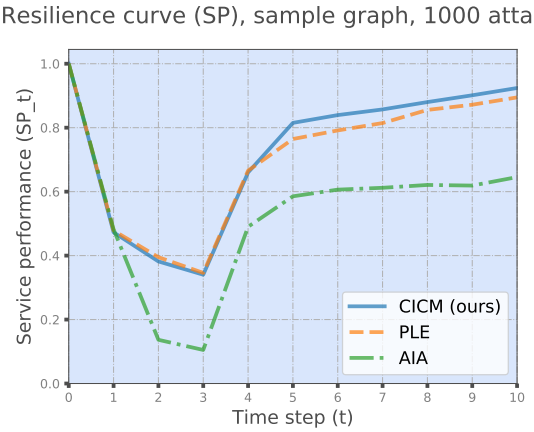
Comparing the results for our approach to those from AIA, we notice our method outperforming AIA both with regard to service performance over time, and the overall costs. The difference between our method and AIA becomes large early on and fails to recover afterwards. This is true whether we look at the case with immediate attack detection (upper panels), or the case with delayed detection (lower panels). Using our approach, the mean SP was over 20% better (23% for immediate detection, 27%



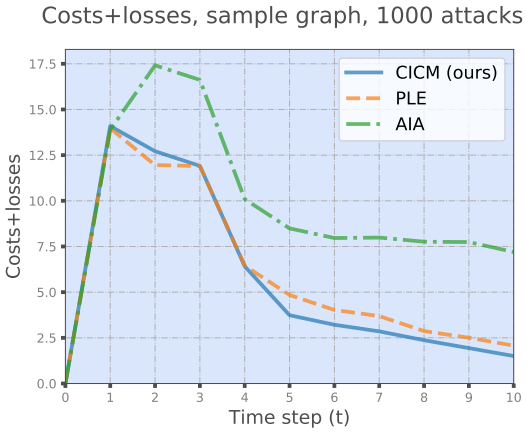

(a) Resilience curve (SP); fully detected steps




(b) Costs and service losses over time; fully detected steps




(c) Resilience curve (SP); 1 undetected step




(d) Costs and service losses over time; 1 undetected step

Figure 9.5: Simulation results for the sample graph

for delayed), and the overall costs were 42% lower on average than for AIA (for both immediate and delayed detection). The Wilcoxon signed-rank test rejected the hypothesis of the difference being zero for SP and overall costs in both immediate and delayed case, suggesting that the performance difference is statistically significant. The result appears to be due to reduced service availability, which causes the similarity in the patterns of SP and overall costs (of which service losses is a part). As Fig. 9.5 shows average curves over randomised attacks, some of the detail related to the individual runs is averaged out, but it seems that AIA has a harder time purging the attack from the system than the other approaches, so the difference to other approaches remains until the end of the time window shown.

The PLE approach exhibits performance roughly matching ours, both when the detection of the attack is instant (upper panels), and when there is a delay (lower panels). In fact, the difference between the

Table 9.1: Comparing mean values of performance metrics, our approach (CICM) vs AIA; immediate detection

| | **CICM** | | | **Difference: CICM - AIA** | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **DG size** | 10 | 20 | 50 | 10 | | 20 | | 50 | |
| | | | | diff. | # +/- | diff. | # +/- | diff. | # +/- |
| SP | 0.862 (0.301) | 0.904 (0.243) | 0.946 (0.169) | 0.168 (0.352) | 95/3 | 0.132 (0.298) | 95/2 | 0.077 (0.224) | 100/0 |
| p-value | - | - | - | **0.00** | - | **0.00** | - | **0.00** | - |
| Cost | 3.49 (7.14) | 2.71 (5.92) | 1.84 (4.32) | -3.91 (8.15) | 4/96 | -3.25 (7.00) | 0/100 | -2.17 (5.33) | 0/100 |
| p-value | - | - | - | **0.00** | - | **0.00** | - | **0.00** | - |

Notes: 100 graphs per size, 100 attack simulations each; "# +/-": count of positive/negative diffs.

two approaches was found statistically insignificant using the Wilcoxon test. For this particular graph, this result can be expected. As discussed in Sec. 9.3.7, to obtain good CM selection performance, the initial exploit will always be patched in this sample graph as the services are directly and fully dependent on the entry nodes. Additionally, the small size of the sample graph limits the variety in attacker steps, and thus on the CM choices of the defender.

### 9.5.2 Results for randomly generated graphs

Simulations were run on graphs of different sizes, and varying the detection delay between no delay and a delay of two steps. The size classes vary the number of nodes in the DG (10, 20 and 50-node DGs), with a corresponding number of AG nodes. We generated 100 graphs of each size, and ran 100 random attacks on each graph.

Table 9.1 shows the results for comparisons to the AIA approach in the case of immediate detection. The results for delayed detection are overall very similar, so we omitted them to save space. We found a considerable benefit for CICM in terms of both the SP metric and overall cost. While the size of the difference in overall costs between the approaches gets smaller with the size of the graphs, due to a general improvement of both of the approaches, the relative cost saving for CICM is around 53-54% across graph sizes for immediate, and 46-51% for delayed detection. Using the Wilcoxon signed-rank test, we find that CICM demonstrated statistically significantly better results than AIA for both of the metrics.

Table 9.2: Comparing mean values of performance metrics, our approach (CICM) vs PLE

**Immediate attack detection**

| | **CICM** | | | **Difference: CICM - PLE** | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **DG size** | 10 | 20 | 50 | 10 | | 20 | | 50 | |
| | | | | diff. | # +/- | diff. | # +/- | diff. | # +/- |
| SP | 0.862 (0.301) | 0.904 (0.243) | 0.946 (0.169) | 0.001 (0.112) | 42/19 | 0.000 (0.110) | 38/30 | 0.000 (0.080) | 29/38 |
| p-value | - | - | - | **0.01** | - | 0.17 | - | **0.00** | - |
| Cost | 3.49 (7.14) | 2.71 (5.92) | 1.835 (4.32) | -0.03 (2.69) | 34/58 | -0.03 (2.59) | 43/56 | 0.04 (2.23) | 36/64 |
| p-value | - | - | - | **0.00** | - | **0.00** | - | **0.02** | - |

**Delayed attack detection, two undetected steps**

| | **CICM** | | | **Difference: CICM - PLE** | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **DG size** | 10 | 20 | 50 | 10 | | 20 | | 50 | |
| | | | | diff. | # +/- | diff. | # +/- | diff. | # +/- |
| SP | 0.801 (0.361) | 0.852 (0.302) | 0.901 (0.235) | 0.044 (0.235) | 78/19 | 0.034 (0.197) | 71/21 | 0.020 (0.154) | 68/25 |
| p-value | - | - | - | **0.00** | - | **0.00** | - | **0.00** | - |
| Cost | 5.04 (8.86) | 4.16 (7.60) | 3.33 (6.32) | -1.09 (5.60) | 11/86 | -0.96 (4.85) | 8/91 | -0.80 (4.08) | 6/94 |
| p-value | - | - | - | **0.00** | - | **0.00** | - | **0.00** | - |

Notes: 100 graphs per size, 100 attack simulations each; "# +/-": count of positive/negative differences.

Comparisons to the PLE strategy are displayed in Table 9.2. When there is no delay in attack detection, CICM performs, on average, almost identically to PLE in terms of the performance metric SP. The difference values are practically zero, and the Wilcoxon test fails to reject the null hypothesis of equal means for SP for the 20-node case. On the side of overall cost, the difference magnitudes are also small. While the simulations show CICM slightly outperforming PLE on average for 10 and 20-node graphs, with 50-node graphs the results are mixed. Overall, CICM was more cost-efficient in 64% of the 50-node graphs simulated, but the mean difference over all graphs is in favour of PLE (2% cost difference (1.835/1.795)). The main message here is: if the attack is detected immediately at the point of entry, there is little difference between CICM and PLE. This makes sense, as patching the first node stops the attacker from regaining access to the network after the attack is purged elsewhere, and immediate detection makes this very effective.

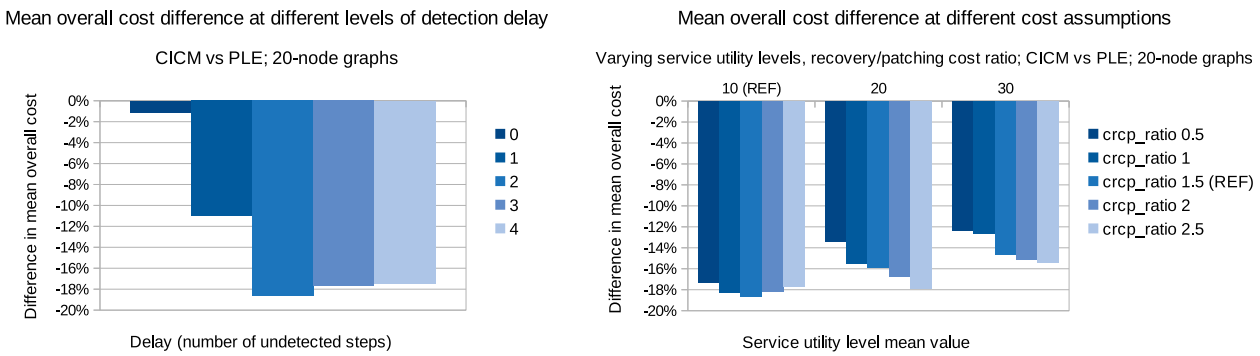
When there is a delay in the detection of the attack, the benefits of CICM over PLE become clear. There is some evidence of a benefit on the average service performance relative to PLE (amounting to between 0.2% and 0.5% of the overall value of the output of the services, on average). Regardless, the main impact is on the overall cost side, where our method provides average cost-efficiency im-

provements over PLE amounting to 18% for 10-node, 19% for 20-node and 20% for 50-node graphs. Importantly, the cost savings are consistent, with benefits obtained in 90% of all the graphs tested (see the +/- counts in the lower part of Table 9.2: 86% of 10-node graphs had a negative sign for the cost difference, 91% of 20-node graphs, and 94% of 50-node). The fact that the difference shows up in cost instead of SP makes sense, as our method chooses actions based on the overall cost, not on SP. The results also show that this effect grows with the size of the graph, suggesting that larger graphs provide more room for choices that improve cost efficiency.

Summarising the findings from Table 9.2, we conclude that while a straightforward patching strategy like PLE can yield good results in terms of service performance, our method provides considerable cost savings when attacks are detected with a delay.

We investigated the sensitivity of the results to the length of delay in detection. We varied the number of attack steps that go undetected before the defender starts their CM selection and recovery processes, with the other parameter values held constant at the levels mentioned above. For this, we used the 20-node generated graphs (100 graphs, 100 simulated attacks for each graph), comparing to PLE. The results, in Fig. 9.6a, show that the cost savings from CICM relative to PLE increase drastically when moving from immediate to delayed detection, with the saving jumping from 1.1% for immediate detection (0 undetected steps) to 11% for one and 19% for two undetected steps. However, further undetected steps provide no additional advantage to our approach, with the relative cost decreasing slightly to 18% and 17% for 3 and 4 undetected steps, respectively. The reason for the initial jump is clear, as the attacker holds more ground in the network and has more possibilities to pursue, and the PLE strategy loses its edge when there are choices to be made. The reduction to the benefit at higher number of undetected steps is due to an increase in average overall cost faced by both approaches, so the difference is smaller relative to this level.

Sensitivity of the results to different assumptions about the cost structure was also tested. The tests consisted of varying two different settings relating to costs: the utility obtained from services per time unit, which affects the costs arising from node unavailability; the ratio of the direct cost of recovery to the direct cost of patching, which can affect the cost-effectiveness of patching relative to recovery actions. These tests were run for the 20-node graphs, and for delayed detection with 2 undetected




(a) Sensitivity to different levels of detection delay

(b) Sensitivity to different cost parameter levels

Figure 9.6: Sensitivity to detection delay and cost assumptions

steps, which provided the highest cost impact in the delay sensitivity tests.

Fig. 9.6b shows the results for the cost parameter sensitivity. We can see two broad patterns: First, the magnitude of the cost difference between the approaches is reduced as the average service utility level increases. Second, in most cases a higher cost of recovery (higher $c_R/c_P$ ratio) leads to an increase in the cost savings provided by CICM compared to PLE, other things equal.

The first observation suggests that the higher the potential loss from unavailability, the less room there is to find benefit from actions that proactively contains the attack as opposed to treating the latest compromise. Therefore, the cost-efficient approach becomes more similar to PLE, which not only treats the latest event but also patches only nodes that are already down, avoiding additional unavailability costs. However, the difference between the approaches is still sizeable, with the smallest difference in Fig. 9.6b suggesting a 12% average saving using CICM relative to PLE.

The second observation suggests that the higher the cost of recovery is relative to patching, the more room there is for CICM to find cost savings by deviating from treating the latest compromise, as the relative cost of taking proactive patching steps is reduced. The average cost levels of both approaches increases with the $c_R/c_P$ ratio, but the rise in the costs incurred is smaller when using CICM than PLE, leading to an increasing benefit relative to PLE.

## 9.6 Conclusion

In this chapter, we proposed an implementation of our resilience impact analysis methodology as a basis for an approach to automated countermeasure selection that aims for a cost-effective approach to maintaining service functionality. This approach takes advantage of the methodology to determine the over-time cost impacts of different actions, so countermeasures can be chosen based on what action yields the lowest overall costs over time.

The method was demonstrated via examples in a sample network, and an evaluation of its countermeasure selection performance was conducted using simulations. The results suggest that our method outperforms an alternative approach to choosing countermeasures based on the attack impact assessment method by [AJ17, AJPS11], both in terms of average service performance and overall costs over a given time window. Comparisons against a straightforward patching approach showed that, while average service performance was a close match, our method found more cost-efficient ways to achieve the goal.

While the evaluations were done considering patching and recovery only, the approach should be extendable to more countermeasures, as the principles of how the methodology would be applied are not specific to patching. That is, other countermeasures, such as containment via limiting connectivity, have similar concerns in terms of their availability impacts, and should therefore benefit from the use of our methodology of impact estimation over a longer time. The current approach could estimate the impact of such connectivity limiting actions, but additional topological information would need to be processed to determine what connections are impacted by each such action, and then map their effect onto the components in the attack and dependency graphs.

# Chapter 10

# Applications: Mission viability analysis

This chapter demonstrates how our methodology can be applied to mission viability analysis, shown in the context of multi-UAV survey missions. The material in this chapter is based on our work presented in [SPL21].

With advanced video and sensing capabilities, unoccupied aerial vehicles (UAVs) are increasingly being used for numerous applications that involve the collaboration and autonomous operation of teams of UAVs. However, such vehicles can be affected by cyber attacks, impacting the viability of their missions. Building on our methodology for resilience impact analysis, we propose an approach to conduct mission viability analysis under cyber attacks for missions that employ a team of several UAVs that share a communication network. The characteristics of this application setting impose a departure from the way we implement the attack progression modelling, relative to the approach used in the applications in the previous two chapters. Specifically, whereas in the previous chapters the attack progression model was built around an attack graph, which the defence and attacker behaviour assumptions used and interacted with, here we model attack progression using a type of Petri net, a stochastic well-formed net (SWN).

This change in the implementation of the attack progression model owes itself primarily to two points that make an AG less suitable for the problem here: the inter-UAV network consists only of vehicles that are functionally identical, but can be adapted during the mission to take on different roles, changing the impact their compromise has on the system (i.e. the impact map is dynamic); the attack

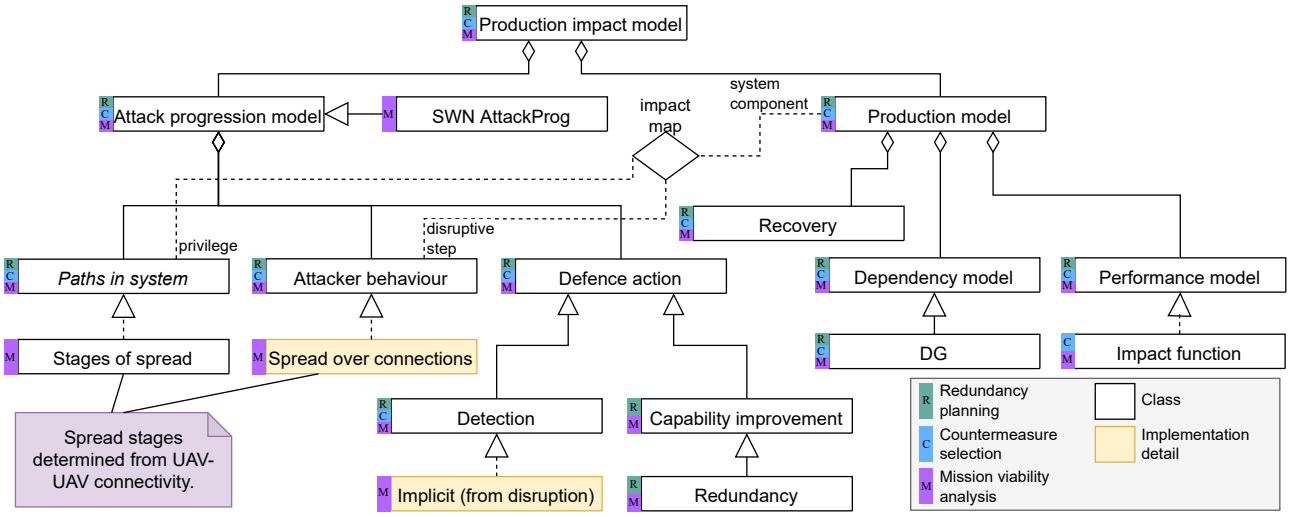



Figure 10.1: Modelling components in the mission viability analysis application

behaviour is to spread and disrupt components widely instead of targeting specific individual nodes. These features mean that, while an approach with an AG and DG could be used, the adaptable nature of the group of UAVs would require implementing repeated changes to the graphs and on the links between them. At the same time, an SWN model provides a suitable way to model an attack spreading among identical components that can take on different roles, and perform stochastic analysis on it to report the impacts over time. It is thus very appropriate for implementing our impact analysis on this specific type of system and problem. Our SWN is, in effect, used as a *method for solving* the attack progression model for the metrics of interest, and achieves the desired analysis result using an existing Petri net formalism and tool. This choice was also, in part, made to demonstrate how our methodology can be implemented with Petri net techniques, which are familiar for distributed systems modelling. The implementation also highlights what is required of a particular Petri net formalism to be used within the context of our attack impact analysis.

Figure 10.1 shows the structure of the modelling implementation used here, in the context of the overall methodology as in Fig. 3.1. The key differences from the implementations in Chapters 8 and 9 are the following: **1.** The attack paths in the system, and attacker behaviour, are not based on the AG for the system, but inferred from the UAV-UAV network topology and the assumption that the attack goal is to spread infection across the vehicles to maximise disruption. **2.** The attack progression model is represented as an SWN, instead of on the system AG. These choices were both made due to the dynamics of the AG and the impact mapping to the DG, as mentioned above. Further, *monetary*

*cost modelling is not implemented*, for two reasons: For firefighting, safety concerns are assumed to override cost considerations, and new investments during the wildfire would not arrive in time.

We apply this mission viability analysis approach to a case study of a survey mission in a wildfire firefighting scenario. Within this context, we show how our approach can help quantify the expected mission performance impact from an attack and determine if the mission can remain viable under various attack situations. This analysis method can be used both in the planning of the mission and for decision making during mission operation. Further, the approach to modelling attack progression and impact analysis with Petri nets can be more broadly applicable to other settings that involve multiple resources which can be used interchangeably towards the same objective.

## 10.1 Introduction

Among the various uses for unoccupied aerial vehicles (UAVs)[1], several applications involve the operation and collaboration of teams of multiple UAVs, for example search and rescue, delivery of goods, providing wireless coverage, and fire survey [SSAF+19]. In a firefighting context, the use of UAVs enables effective survey of the area whilst reducing human exposure to danger. UAVs are currently used for this purpose in, for example, Latvia and the Netherlands [SMADB+20]. However, UAVs are also vulnerable to various cyber attacks, including ones that enable UAV hijacking and allow the attacker to gain complete control of the vehicle. Several works, such as [NBM+21, YS20, BST+20], analyse UAV security and survey the different attacks demonstrated so far on UAVs. As the capabilities of UAVs increase, so does their attack surface [BST+20], and new types of attacks can be anticipated.

For the successful running of missions with multiple UAVs, it is important to be able to assess the impact of a cyber attack on mission performance, to determine mission viability after attack and to enable planning missions with appropriate levels of redundancy. To this end, we propose an approach to modelling and reasoning about attacks and mission impacts on a group of UAVs using our methodology for resilience impact assessment. In this instance, we implement the attack propagation model

[1] In this work we use the term UAV to refer to the vehicle, regardless of whether it is autonomous or remotely controlled by a human pilot.

using a Petri net, while system production (i.e., mission performance) is estimated using a production function. We demonstrate our approach through a case study of fire survey missions.

Here, the use case of interest is estimating the impact of an attack that can spread within the team of UAVs, taking into account the system configuration, defensive capabilities in terms of containment and recovery, and the expected attack behaviour. Petri nets are appropriate here, as they are a framework for modelling distributed systems that is especially useful for modelling the behaviour of systems with multiple components that exhibit symmetry in how they function as part of the system. Further, stochastic formalisms such as SWNs offer tools to estimate the effect of changes to a system over time, using transient analysis. While Petri nets have been used for modelling cyber attacks by e.g. [MT04, LLC+18, DW06, CH19], our approach differs from them in employing the Petri net model as part of a performance impact evaluation model, in the attack modelling specifics, and the application to UAV missions.

The key contributions of the work in this chapter are: **a)** We propose a novel approach to modelling and reasoning about attack impacts for UAV missions, using our resilience impact assessment methodology. **b)** We show that the system and attacks to be modelled form a special case that can be complicated to model using an AG-based approach, but is suitable to be translated into an SWN to be solved for our metrics of interest. The attack propagation model approach used here is particularly suitable for modelling attacks that can spread across a team of UAVs, as the Petri net formalism enables a compact and effective modelling of impacts to systems with multiple interchangeable components which can be attacked in a similar manner. **c)** We demonstrate how this approach can be used both during mission planning and as a decision support tool during mission operation. The approach helps to quantify things such as: i) The expected mission performance if an attack occurs at different times during the mission, given the mission configuration and redundancy; ii) The number of redundant UAVs required for a desired likelihood of success; iii) How the number of vehicles required changes with the speed of containment relative to attack speed. **d)** We formalise and analyse in detail a case study of wildfire surveys, explaining the use and benefits of the methodology.

This chapter is structured as follows: Section 10.2 discusses related work. Our case study and threat model are introduced in Section 10.3. Our approach to mission viability analysis is described in Sec-

tion 10.4, and its implementation in our case study is shown in Section 10.5. Section 10.6 discusses our analysis results, and Section 10.7 concludes.

## 10.2 Related work

UAV security from the point of view of individual vehicles has been studied by e.g. [YS20, BST+20, NBM+21]. Additionally, [JSDA12] proposed an approach to model and analyse the threats to a system of several UAVs. While their approach is a framework for estimating risk from various specific attacks on a system, ours is a high level model intended for estimating mission viability in response to attacks that impact the availability of vehicles, with a focus on the performance of the overall mission.

As in the other applications of our methodology in Chapters 8 and 9, we evaluate the impacts of cyber attacks on system performance over time and on mission viability, employing a resilience-inspired view on attack impacts as a basis for decision making. In doing so, our approach here is similar to what we used for resilience planning (Chapter 8) and to select countermeasures to attacks (Chapter 9), but our focus here is on a different kind of system consisting of a group of UAVs, and on making decisions over the appropriate mission setup and on mission continuation in the face of attacks. We model attack impacts with a combination of a model of attack progression and a model of system performance in producing its output. This relates to methods that approach attack impact assessment via a combination of an attack graph (AG) and a service dependency graph, such as those by [AJPS11, AJ17], and our work in the previous chapters. In contrast to such an approach, we use a Petri net for modelling attack progression, finding it particularly suited to this in our application context. While the use of AGs for describing attack progression works well when the attacker's goal is reaching a specific privilege, cases where the attacker aims to reduce system performance and availability by spreading across components are more conveniently modelled using Petri nets. Similarly, our application area contains multiple interchangeable components that share the same vulnerabilities, which is more difficult to represent using AGs, but suitable for Petri nets.

Petri net formalisms have been used before for attack modelling in works such as [MT04, LLC+18, DW06, CH19], as we discussed earlier in Section 2.2.2.2. Several works have taken the approach

of creating a Petri net that is essentially an attack graph of the attack steps, with places and related transitions for each individual attack step, for example in [MT04, DW06, CH19]. A weakness of this type of approaches is that they yield a complex representation using a high number of places and transitions. We take a different route, enabled by the problem we are trying to solve: instead of creating separate places and transitions for all components to compromise, we build a Petri net that represents the process of an attack spreading among functionally identical UAVs, and employ token colours to separate between different UAV roles. In complex missions containing multiple functionally equivalent components, our method for describing attacks leverages these symmetries to generate more compact models that are easier to analyse and use than if each component was individually represented by a place in the net.

Li et al. [LLC+18] use a GSPN to model the impact of coordinated topology attacks to smart grids, using a Petri net that represents the system instead of the attack steps. Their use of a GSPN for attack impact modelling in the presence of countermeasures is similar to our evaluation of expected attack impacts. However, a key difference in the attack modelling is how the attack progresses: the propagation of the compromise from a UAV to another is a key feature of our model, while they model attacks to different components as individual attacks without a direct link between compromises, i.e., their model only represents whether a particular attack has occurred, not what stages the attack would need to progress over.

## 10.3 Scenario and Threat Model

Our case study is a survey mission involving a team of UAVs in a wildfire scenario. To give context on the scale of the issue of wildfires, in European countries on average 444 000 ha were burnt by forest fires annually over the last decade (2010-2019), according to data from the European Forest Fire Information System (EFFIS) [Eur00]. For economic perspective, in Germany the direct economic damage from forest fires was estimated to be €819 per hectare burnt in 2019 [SMADB+20].

We focus on wildfires that are large enough to benefit from multiple UAVs to map the potential fire area. Depending on the equipment used and the speed achievable, this will likely mean fires

where the required area coverage is over a hundred hectares. For reference, according to data from [SMADB+20], within Europe there were over a hundred fires exceeding 100 ha burnt area in 2019, with the largest single fires covering thousands of hectares.

Our scenario has the following features: The *mission* contains a team of surveying vehicles (UAVs) flying in a loose formation across a given area to identify wildfires by collecting data. The data are passed to a centralised mission control (MC), which deals with processing the data and coordinating the overall mission. The *data collected* by the UAVs contains camera feed of the area (UAVs with video sensors), and temperature of fires (temperature sensors), and maps based on these. A subset of the UAVs are directly connected to the mission control via radio/line of sight communication, while communication between the UAVs within a team is ensured by some of the UAVs acting in a *relay* role. The data thus flows to the MC via the inter-vehicular network across relays, and via vehicles with a direct link to the MC.

**Threat model:** In our model we consider attacks that disable the vehicles' sensing capabilities, via "manual" or automated attacks, including malware. The key difference between the attack types is the speed at which the attack spreads to other vehicles. We do not model the modes of attack in detail, but reflect different attack speeds by varying parameter values.

As the focus of our work is on the overall mission viability, we do not detail how the attack proceeds inside an individual UAV. That is, we assume that attack actions within a UAV will either succeed and compromise the UAV (and possibly propagate to other UAVs), or the attack on that UAV will fail.

We make the following assumptions on attacks: The *attacker's goal* is to disrupt the mission by disabling UAV capabilities or the UAVs themselves, using *methods* such as malware, vulnerability exploits (disabling the UAV or a sensor). Any UAV can serve as the entry node for the attack. We consider attacks that can propagate from vehicle to vehicle, i.e. a compromised UAV can be used to compromise other UAVs in the network. The *impact* of a compromise is the capabilities of the UAV becoming fully disabled, although the model could be extended to represent partial degradation of function for the UAVs by considering the roles they can fulfil in the mission. We consider the capabilities independently from the vehicle, which could enable cases where a UAV with one compromised capability could be used in another role that it still had a capability for, if the compromise was isolated

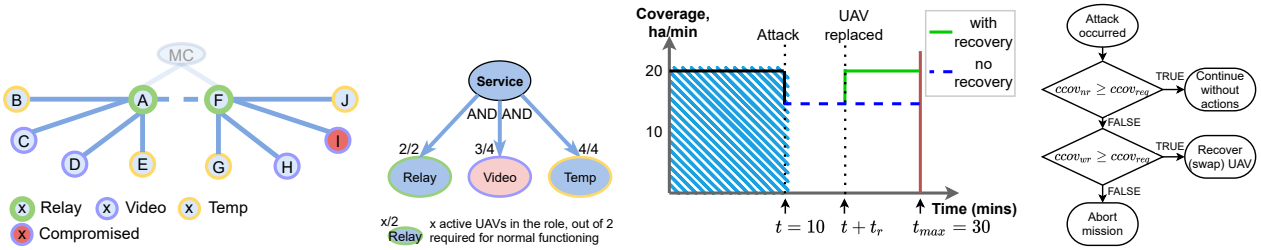
from affecting it. Here we simplify by not allowing compromised UAVs to remain in the mission, but the model could be extended to consider scenarios where they could be reassigned to other roles.

## 10.4 Modelling approach

Within our fire survey scenario, the *objective* is to track mission progress, making decisions on whether the mission can no longer succeed and should be aborted, given the impacts of cyber attacks. In this setting, the components of our methodology do the following: The *attack progression model* focuses on the *availability* of UAV capabilities, given the threats of malware or attacks to sensors. The model describes the progression of the attacker (behaviour and time to progress) within the team of UAVs. This is modelled using a Petri net (SWN). The *production model* describes how the tasks (UAV roles) combine to produce the mission output (e.g. coverage of a certain area). We model this with a Leontief production function[2], where a unit of coverage rate is achieved by a strict ratio of UAVs in specific roles. We choose this form as relay roles are typically not substitutes for sensor roles and vice versa, although in a different setting a more general type of function could be used. To measure *performance*, we set a *target for area coverage* given mission goals, and measure *performance as the rate of the area covered per a unit of time*, estimated based on the UAV type information and assuming effective control of flight paths. The modelling components require input data from the *mission plan* and the *UAV connectivity* status. The production model is based on the mission plan, and the specifications and capabilities of the UAVs taking part in the mission. The connectivity among the participant UAVs impacts both the attack progression and the production model, as connectivity is required to transmit the mission data.

The *decisions* are to continue mission if current and expected progress meet the target, otherwise decide between using redundancy (with existing redundant capacity, or by adding UAVs) or aborting the mission. The choice of actions is made based on current progress and expectations over what could happen during the time remaining before mission end. The decision process takes inputs from the various model components, and leads to changes in the states of the models when actions are

[2] An introduction to different production functions, including the Leontief input-output model, can be found in e.g. [MWG95].




(a) Mission connectivity graph (b) Production graph (c) Effect on mission coverage (d) Decision

Figure 10.2: Sample analysis, attack that does not spread

taken. For example, adding a UAV will add capacity to the production model, but also more paths for an attack to progress. These changes can affect the performance estimates and expected mission progress. We consider the key decision to be over continuing the mission or aborting: *Continue without changes* if the mission can be completed without adaptations; *Take recovery actions* (replace a UAV with another) to improve coverage rate if this allows successful mission completion; *Abort mission* if it cannot be completed successfully even with recovery actions, or if no recovery is possible.

To illustrate the decision making required, Fig. 10.2 shows a sample analysis for the simple case of an attack that does not spread. For this illustration, we have used the following assumptions for the mission, and the attack:

- Task: cover an area of 560 ha, within a maximum time of $t_{max} = 30$ minutes
- Coverage rate function: $cov\_rate = 20 * min(\frac{x_{rel}}{2}, \frac{x_{vid}}{4}, \frac{x_{tmp}}{4})$ $(ha/min)$, where $x_{rel}, x_{vid}$ and $x_{tmp}$ are the number of UAVs active in the relay, video and temperature roles, respectively. During normal operation the rate of coverage with this example function is 20 $ha/min$.
- The attack occurs at $t = 10$ (min), taking down one video sensor. As a consequence, the coverage rate drops to 15 $(ha/min)$, as $\frac{x_{vid}}{4} = 3/4$.

Fig. 10.2a shows that the attack at $t = 10$ disrupts node $I$ in the mission connectivity graph, a UAV in a video capture role. The production graph in Fig. 10.2b shows that this impacts the video service, taking the number of UAVs in the video role below the standard requirement of 4 (shown above the "Video" node of the graph). Fig. 10.2c shows how this affects the mission performance in terms of coverage, with the coverage rate dropping to 15 $ha/min$ at $t = 10$. To make a decision on what to do, we must consider what could happen in the remaining time before $t_{max}$. As the attack will not

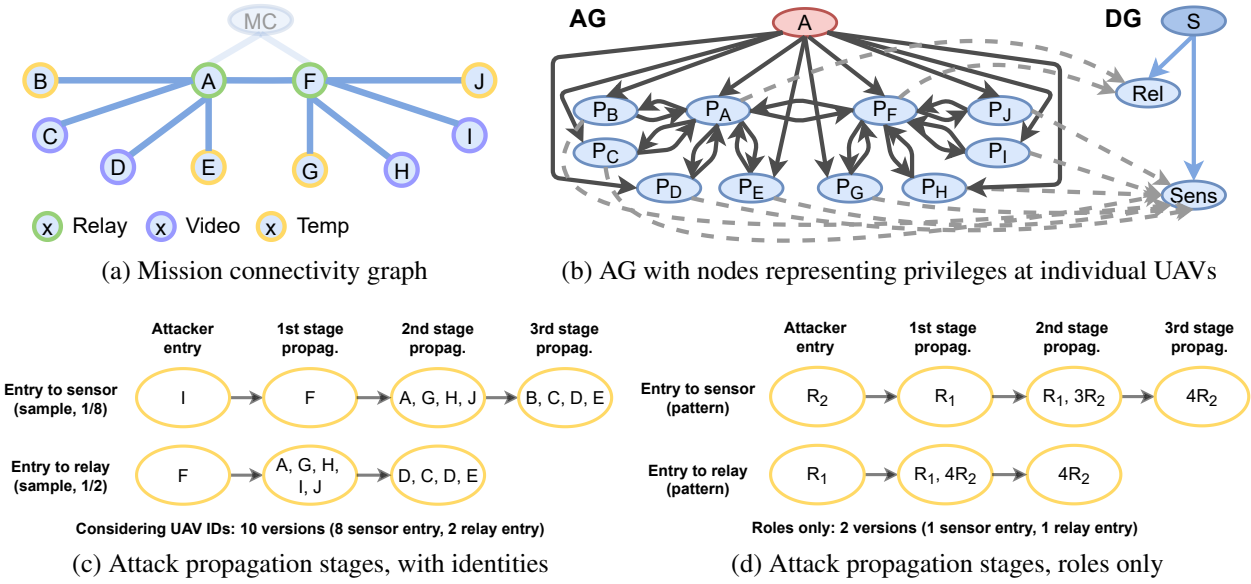
spread (in this example), there are two cases to consider. Without a recovery action, the coverage rate remains at the level of 15 $ha/min$ until the end of the mission time. On the other hand, recovery could be done in $t_r$ time by replacing the compromised UAV with another UAV (without the same vulnerability), so performance would return to 20 ha/min at time $t = 10 + t_r$. Finally, Fig. 10.2d shows the decision to be made: **a)** Continue without recovery if cumulative coverage without recovery $ccov_{nr}$ meets the required level $ccov_{req}$ – this is not the case here, as $ccov_{nr} = 500 \ngeq 560 = ccov_{req}$; **b)** Recover by swapping UAVs if coverage with recovery, $ccov_{wr}$, exceeds $ccov_{req}$ – this holds in the example if $t_r < 8$; **c)** Abort the mission if $ccov_{wr} < ccov_{req}$ – here if $t_r > 8$.

The example above gives a flavour of the analysis required, but in practice we expect that the attack can spread to other vehicles, or that there can be several attacks. Our attack propagation model is intended for analysing such situations. The analysis employs a Petri net formalism, specifically SWN, to evaluate the effects of an attack. This provides a stylised representation of the attack progression and defensive actions (containment and recovery), with a specific net instantiated based on the attack propagation model (showing attack paths), the attacker entry point, and assumptions on the propagation speed, defence capabilities and speed. As the attack stages, containment, and recovery actions can interleave in various ways and with different time profiles, causing different outcomes in terms of production impact, the SWN formalism provides a way to determine the mean behaviour of the system considering this interleaving and timing of actions.

## 10.5 Model implementation

### 10.5.1 Connectivity and attack progression

In addition to enabling data transfer from sensors to the mission control, connectivity also leads to the existence of potential attack paths across vehicles. Fig. 10.3 shows how the connectivity of the surveying mission can lead to specific attack paths. In our case study we model connectivity as a hubs and spokes model, as in Fig. 10.3a, where all sensor UAVs are connected to relay UAVs, which then pass the information to the MC. This type of connectivity is typical of such missions. In this context,




(a) Mission connectivity graph

(b) AG with nodes representing privileges at individual UAVs

(c) Attack propagation stages, with identities

(d) Attack propagation stages, roles only

Figure 10.3: From connectivity to attack propagation stages

the *attack behaviour* is as follows: the attack spreads in a straightforward sequence of steps, with differences arising based on which node is attacked first, and whether the transmission of the attack follows a one-hop transmission or if all reachable UAVs can be infected in one step.

While the attack progression could be modelled using an AG, as we did in the applications in Chapters 8 and 9, this would be unnecessarily complex for modelling attacks of this type. Fig. 10.3b illustrates the complexity of the AG if the privileges that enable the compromise of UAVs were represented as separate AG nodes for each UAV. The connectivity and dynamic structure of the group of UAVs means that any UAV can be an entry point, so the attacker node $A$ has an edge to all privileges. Further, the connectivity is bidirectional, so each UAV can send and receive communications with its relay UAV, and the attack can spread both ways. This means that there are numerous possible attack orderings, even with this simple network topology. Further, as the vehicles can switch roles, the impact map (the edges from AG to DG nodes) can also change. Considering also the recovery actions that can be taken, and when the attack is contained, all the dynamism in the system means enumerating all possible attack outcomes would be complex, even though the attack behaviour is straightforward.

Instead, Fig. 10.3c illustrates what happens if we represent the attack paths as set of propagation stages, given that the attack can spread to all adjacent UAVs at the same time. This simplifies the paths to sets of stages, consisting of attack entry and propagation, yielding up to four stages in the

case of the sample topology. Two instances of such attack paths are shown in Fig. 10.3c: one where the attack enters the sensor node $I$, then spreading in stages to nodes adjacent to compromised nodes, starting from relay $F$; the second type shown is where entry occurs at relay $F$. However, as the UAVs differ in their roles but are otherwise functionally identical, many of the possible attack path instances differ in the ordering of the specific UAV identities but are otherwise symmetric. Thus, as shown in Fig. 10.3d, by ignoring the UAV identities we can map the attack path instances to representative patterns, here yielding one pattern with entry to a sensor UAV, and one where entry is to a relay.

The situation shown in 10.3d is essentially what we do in our SWN model, we ignore the identities and take advantage of the symmetries, and only follow how many UAVs there are in given roles, using coloured tokens. The impact map is implicit in these token colours, and swapping between tokens of different colours represents the dynamism of the mapping. The SWN model provides an efficient approach to stochastic evaluation of the metrics of interest to us, over time. By enabling drawing transition timings from (exponential) distributions, the SWN allows representing various time profiles of the attack progression, considering the different interleavings of the attack and defence actions.

### 10.5.2 Actions to mitigate attacks

We consider the following remedial actions as part of possible responses to an attack:

1. **Contain the attack** by disabling a compromised part of the fleet (e.g. return to base). This leads to losing the UAVs in question for the duration of the mission.
2. **Swap roles/tasks between UAVs**. Some mission time is lost before the swap is completed.
3. **Introduce a replacement UAV** to the mission. The replacement arrives after some delay.

Swapping roles acts as a recovery mechanism for performance, allowing to continue tasks at the previous coverage level, if the relevant capabilities exist in other UAVs. Introducing a new UAV into the mission achieves the same goal but takes a longer time on average, as the new UAVs need to fly in from the dispatch point.

The effectiveness of swapping tasks between UAVs (or bringing in replacements) depends on the

level of similarity between the UAVs in terms of exploitability: if it is easy to replicate the attack on the UAV swapped in, then the failure of that task is still likely. We assume that the UAV to be swapped in does not have a vulnerability to the same attack, for example as it is a UAV of a different type, although we are aware that this is a strong assumption. Future work should aim to relax this assumption and extend the model to study the impact of several successive attacks, regardless on whether they resulted from the same or different vulnerabilities.

### 10.5.3 Petri net (SWN) model

Petri nets are commonly applied when considering resources, and are especially useful for settings with multiple components which exhibit individual behaviour that is symmetric across the components, but which takes place concurrently and can have interactions or joint behaviour. Thus they are particularly suited to the case of teams of UAVs, as they are similar in terms of their tasks, and are assumed to respond to attacks in a similar way. However, we distinguish between the different *roles* (relay, sensor [video, temperature]) the UAVs have in the mission, which we model using the Petri net extension of coloured tokens. We also require time to be represented in our model, which leads us to use the timed transition extension to Petri nets, specifically the version introduced in generalised stochastic Petri nets (GSPNs) [MBC+95]. The need for these two extensions made us opt for the stochastic well-formed net (SWN) [CDFH93] formalism for our model, as it provides both colours and stochastic transitions[3].

Fig. 10.4 shows the SWN model of the attack propagation and defence in our fire survey scenario. It follows standard notation for GSPNs and SWNs, and was made with GreatSPN [ABB+16]. In the remainder of this section we describe the broad aspects of the model and the basic flow of events. A detailed introduction to Petri nets and the notation can be found in e.g. [MBC+95]. Table 10.1 lists the parameters used in the model, with brief explanations.

As shown in Fig. 10.4, the network is composed of three sub-models demarcated with gray frames: the *Attack* sub-model represents the different stages of the spread of an attack within the network of vehicles; *Containment* models the process of containing the attack to stop further spread; *Recovery*

[3]Technically the SWN colours can be unfolded to represent our net as a GSPN, but that is visually more complex.

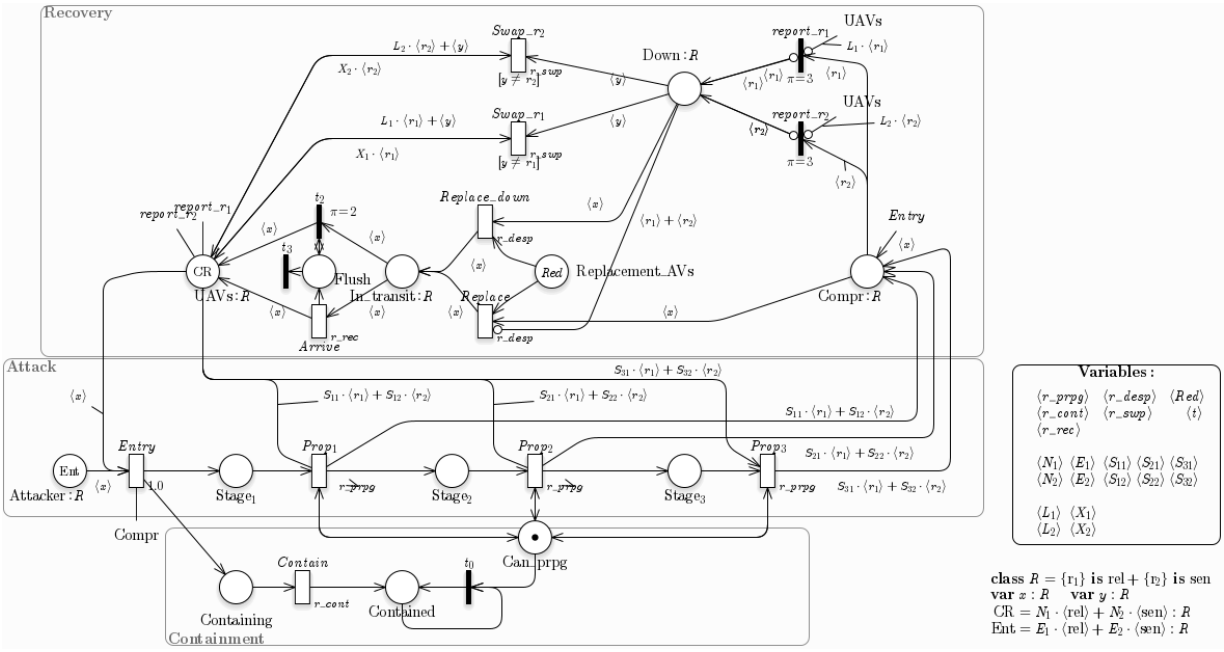



Figure 10.4: Stochastic well-formed net for attack stages

Table 10.1: Parameters in the SWN model

| | |
|---|---|
| $r_{prpg}$: rate of attack propagation (1/propag. delay) | $r_{cont}$: rate of attack containment (1/cont. delay) |
| $r_{desp}$: rate of dispatch of replacements (1/desp. delay) | $r_{rec}$: rate of recovery via replacement (1/repl. delay) |
| $r_{swp}$: rate of recovery via swap (1/swap delay) | $t$: model time, seconds |
| $Red$: number of replacement UAVs available for recovery | $N_i$: number of UAVs in role $i$ taking part in the mission |
| $E_i$: the UAV role $i$ that attack entry occurs in, $E_i \in [0, 1]$ | $S_{si}$: number of UAVs in role $i$ compromised at attack stage $s$ |
| $L_i$: UAV count in role $i$ below which recovery is needed | $X_i$: excess/redundancy in role $i$ (1 over limit) $X_i = L_i + 1$ |
| $R$: set of vehicle roles (token colours), $R = \{r_1, r_2\}$ | $x, y$: token colour (vehicle role) variables, $x, y \in R$ |

contains the role swaps and vehicle replacements that can be used to recover mission performance. The sub-models interact at several transitions and places, most importantly via their impact on the count of vehicle roles that are currently fulfilled and those unfulfilled due to compromise, represented by the places *UAVs* and *Compr*, respectively. Coloured tokens represent vehicles in different roles: a token of colour *r1* is a UAV in the relay role, while colour *r2* is a sensor role. This enables tracking the count of vehicles in different roles, and those in each role that are compromised by an attack.

An **attack** is represented as follows: the initial entry to the system is denoted by the *Entry* transition, which takes a token of colour *r1* or *r2* from the places *Attacker* and *UAVs*, and passes a token of the same colour to *Compr*, and one uncoloured token each to *Stage1* and *Containing*. The subsequent spread of the attack is represented by the firing of the timed propagation transitions (*Prop1*,*Prop2*,*Prop3*): e.g. firing of *Prop1* takes the token from *Stage1* and one from *Can_prpg* (and returns it back), and coloured tokens from *UAVs* representing the number of vehicles of each role that

the attack can spread to in the first stage of propagation (i.e. the vehicles that the compromised UAVs can communicate with). The number of coloured tokens of colour $r$ taken in step $s \in \{1, 2, 3\}$ of an attack is given by the parameters $S_{sr}$; these parameters are set based on the number of vehicles in different roles reachable at a given stage of an attack based on the connectivity of the mission. The same number of coloured tokens of each type is passed to *Compr* as were taken from *UAVs*, and an uncoloured token is passed to the place representing the next stage of the attack, e.g. *Stage2*. For example, in Fig. 10.3c where entry occurs at the sensor UAV $I$, the firing of *Prop1* takes one token of colour *r1* from *UAVs* to represent propagation to the relay $F$, and passes one *r1* token to *Compr*. Next, the firing of *Prop2* takes three tokens of colour *r2* (sensors $G$,$H$,$J$), and passes three *r2* to *Compr*. As in this case the relays are connected to each other, *A* can be compromised at the second stage with *Prop2*, spreading to further sensor vehicles with *Prop3*.

This model can be made to apply to different attacks and configurations of the inter-vehicular network by providing the parameters $S_{sr}$ that describe how many of vehicles in each role are impacted at each stage of the attack. While in Fig. 10.4 the the number of spread stages (at most three) in this sub-model is specific to our scenario, this is straightforward to extend by adding the places, transitions and arcs for any further stages needed.

**Containment** is modelled as a process that can remove the possibility of the attack propagating to other vehicles. Specifically, we include a place *Can_prpg* representing the ability of the attack to propagate (i.e. not being contained), and containment occurs when the token in it is removed. The timed transition *Contain* becomes available after attacker entry as a token is passed to *Containing*, and when it fires a token is passed to *Contained*, which causes the firing of the immediate transition *t1* that removes the token from *Can_prpg*, disabling the propagation transitions.

The containment action in the SWN is treated as stopping the attack from proceeding to the next stage (to the UAVs adjacent to the attack boundary), so no further vehicles are compromised by the same attack. While this representation may appear to avoid the aspect of the containment action itself leading to loss of production from the contained UAVs, the model can be interpreted in a manner where the loss of vehicles is equivalent whether they were successfully compromised or contained. That is, we view the attack stages in the SWN as representing the extent of spread which the attack

*could have reached* in the time conditional on the previous stage being successful. With this interpretation the SWN model counts the vehicles at a given attack stage as lost, whether it is due to them being certainly compromised or being contained due to becoming untrustworthy as the attack could conceivably have reached them. This simplifies the modelling.

**Recovery** in the model can occur in two ways, either by replacing compromised vehicles with others not currently partaking in the mission (e.g. because they arrived to the mission after the original set of UAVs were dispatched), or by swapping the roles of UAVs already in the mission, when appropriate.

The recovery model has a multi-stage structure. When a vehicle with role $x$ has become compromised by an attack, a token of colour $x$ gets added to the place *Compr*. If the compromise would lead to service $x$ (e.g. the relay service) performing below the minimum requirement, a token of colour $x$ is added to *Down* (by the firing of *report_r1* or *report_r2*), which has a higher recovery priority than *Compr*. If *Down* contains tokens, then a role swap between UAVs currently partaking in the mission can be used as a recovery mechanism to enable faster return to acceptable performance for the service.

When a token is added to *Compr* and *Down* is empty, the timed transition *Replace* can fire, representing the step of preparing a replacement vehicle to take over the role of a compromised one. *Replace* cannot fire if *Down* has tokens, as services that are fully down take priority in recovery over those that are simply degraded. This constraint is represented by the inhibitor arc from *Down* to *Replace*. If a token exists in *Down*, the service flagged down can be recovered by replacing it with a new vehicle by firing *Replace_down*, or by a swap with another vehicle partaking in the mission. The swap is done with either *Swap_r1* or *Swap_r2* depending on the role to be swapped, which swaps a vehicle from a role with a redundant number of vehicles to the desired one. Once *Replace* or *Replace_down* fires, a token is placed into *In_transit* to represent that the replacement has been dispatched but is yet to arrive to the desired location. When the transition *Arrive* fires, the tokens that were in *In_transit* get passed to *UAVs*. The structure involving *Flush* and immediate transitions *t2* and *t3* is used to pass all the remaining tokens from *In_transit* into *UAVs* once *Arrive* has fired. This is used to represent simultaneous flight of the UAVs in transit, instead of having them queue to fire "Arrive" one at a time.

This model of recovery can apply in UAV scenarios where the replacement of vehicles and swapping of roles is relevant. In a more generic setting recovery could be modelled similarly to how we model

Table 10.2: Area coverage over time for an individual vehicle, ha/min

| **Speed (m/s)** | 5 | | | | 10 | | | | 15 | | | | 20 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **AOV** | 40° | 80° | 95° | 150° | 40° | 80° | 95° | 150° | 40° | 80° | 95° | 150° | 40° | 80° | 95° | 150° |
| **Altitude** | | | | | | | | | | | | | | | | |
| 50m | 1.1 | 2.5 | 3.3 | 11.2 | 2.2 | 5.0 | 6.5 | 22.4 | 3.3 | 7.6 | 9.8 | 33.6 | 4.4 | 10.1 | 13.1 | 44.8 |
| 100m | 2.2 | 5.0 | 6.5 | 22.4 | 4.4 | 10.1 | 13.1 | 44.8 | 6.6 | 15.1 | 19.6 | 67.2 | 8.7 | 20.1 | 26.2 | 89.6 |
| 120m | 2.6 | 6.0 | 7.9 | 26.9 | 5.2 | 12.1 | 15.7 | 53.7 | 7.9 | 18.1 | 23.6 | 80.6 | 10.5 | 24.2 | 31.4 | 107.5 |

Notes: AOV - angle of view of the camera

replacement, with a given recovery capacity (of components or repairers) and places and transitions representing the recovery process.

We solve the model for the average number of roles being fulfilled over time, i.e. the number of tokens of each colour in the place *UAVs*, using the GreatSPN [ABB+16] transient solver. The time unit in the model is seconds, $t \in [0, 1200]$, and the transient solution is evaluated at intervals of 30 time units. The timed transitions in our model use exponentially distributed firing rates, as used by GSPN and SWN. More general distributions are enabled by some extended Petri net formalisms, but these require simulations to analyse the model.

### 10.5.4 Mission performance modelling

Mission performance is quantified as the rate of area covered by sensor vehicles. We approximate the expected coverage rate of an individual vehicle based on capabilities of currently available vehicles in terms of their speed and the maximum area coverage of cameras with different angles of view at plausible flight altitudes during operation. These are shown in Table 10.2. We have assumed speeds during operation to be between 5 and 20 m/s. For the angle of view (AOV) of UAV cameras, the reported AOV values are between 40° and 95° for current thermal cameras[4], and normal high-definition cameras are between 80° and 150°. We have assumed operational altitudes between 50 and 120m.[5] The resulting approximations may overestimate the coverage reachable by vehicles in practice, but provide useful direction. In practical implementations the coverage rates could be evaluated using data from previous missions.

The coverage rate of a group of UAVs on a mission is estimated by a production function, which takes

[4]E.g. the DJI Zenmuse H20T thermal camera has 40.6° DFOV [DJI20], and Flir VUE TZ20 has 95° FOV [Fli20].
[5]The EU limit for operation in the 'open' category is 120m [Eur19].

into account the required number of vehicles in different roles. Here we model this as:

$$cov\_rate = IC * x_{vid}^{req} * min(\frac{x_{rel}}{x_{rel}^{req}}, \frac{x_{sen}}{x_{sen}^{req}}) \tag{10.1}$$

where $IC$ is individual coverage (as in Table 10.2), $x_{rl}$, is the number of UAVs active in the role $rl \in \{rel, sen\}$ (relay, sensor [incl. video and temperature]), and the required number of UAVs in role $rl$ is denoted by $x_{rl}^{req}$. The multiplier $IC * x_{vid}^{req}$ represents our assumption that the number of video sensors is a key determinant of the extent of mission coverage, while relays and temperature UAVs offer support and added services to the mission.

With the coverage rate function we can convert the SWN analysis results (numbers of UAVs in different roles) into coverage rates, evaluate mission performance, and make decisions based on the expected total coverage.

The expected total coverage achievable during a mission that is attacked is evaluated as follows: **1.** The time period from the start of the mission until the attack occurs is assigned the full coverage achievable with the mission setup, e.g. if the attack occurs 5 minutes in, this is $5 * cov\_rate$. In during-mission analysis, this could instead be the observed coverage. **2.** The coverage from the attack moment onward is evaluated based on the results from the Petri net model, converted to coverage values using a production function such as (10.1). This is applied until the end of the evaluation time window of the Petri net model, unless it would exceed the overall mission time. **3.** The mission time that remains after the evaluation time window of the Petri net model is assigned the coverage rate estimate reached at the end of the Petri net evaluation.

### 10.5.5 Pre-planning and during mission use

Our modelling provides a structured approach for mission planning support and viability evaluation. When planning for a mission, it can help determine details such as the appropriate number of vehicles to be used for a mission with a given coverage requirement, or the number of replacement vehicles for a given mission. It can also quantify the impact of various parameters on the coverage achievable, such as attack containment or recovery (UAV replacement and/or role swap) speed. When an attack

occurs during a mission, our model can be used to evaluate the extent to which the mission can be expected to succeed despite the attack.

#### 10.5.5.1 Pre-planning

When planning a mission, the SWN component of our model enables answering questions such as: *What is the impact of an attack if no containment approach is used, vs with containment?* Answering this is achieved by comparing the performance of two SWN versions, with and without containment. *How many replacement vehicles are required to ensure recovery to a specific number of vehicles after an attack?* This can be answered by running the analysis varying the number of replacement UAVs. *Can varying the formation of the vehicles/partitioning the network reduce the chances of a disruption due to an attack?* Such analysis would involve varying the SWN input parameters to reflect different assumptions on vehicle connectivity and formations, and comparing the results.

The overall mission analyses, conducted with our full modelling approach that includes the coverage model, also require consideration for mission-level parameters outside of the SWN model, such as the maximum mission duration, UAV specifications, and the time of the attack occurrence. The full model enables the estimation of mission success metrics and consequently the evaluation of mission-level issues such as: *What is the overall coverage reachable after an attack, and how does it change with the maximum mission length or the time of attack? How does the loss of a UAV to compromise affect the mission coverage if the vehicle in question was acting in a sensor role, or in a relay role?*

Although we have focused here on the availability of UAVs over time, the metrics obtainable with the SWN modelling are not limited to this. Other metrics that can be readily obtained from the SWN analysis, using the GreatSPN tool, include the likelihood of system states of interest, such as the loss of N sensor vehicles to compromise, or the likelihood that at least the minimum number of relays are online to ensure mission information is passed to the MC. Such metrics could be used to answer questions such as: *How many replacement UAVs are required to keep the probability of mission failure due to an attack below a certain threshold?* Additionally, in future work we plan to explore multiple and/or repeat attacks in the model. This would enable evaluating the impact of repeat attacks on the number of redundant replacement vehicles required and the time it takes to recover the mission.

From the mission performance perspective it would allow estimating the number of attacks that can be sustained before the mission can no longer be fulfilled.

#### 10.5.5.2 Analysis during an attack

During an attack, the focus is on the likelihood of mission success and degree of mission completion. Our modelling can help answer questions such as: *Can the mission continue, or should it be aborted?* This is answered by estimating the expected mission coverage given the attack using the SWN model, and comparing the result to the mission requirement. This also involves considering the remaining mission time, as part of the overall mission coverage estimation. We assume such analysis will be performed by the mission command based on data received from the vehicles. *What share of the coverage requirement can be achieved given the attack?* The current progress towards the goal and estimate of the coverage reachable in the remaining time can be a useful metric for mission planners and operators, especially if the mission goal is flexible. This is a direct output of our mission analysis.

## 10.6 Analysis results

This section shows the results of the analysis for our case study. We describe them in two parts. First, we discuss the impact on vehicle counts in different roles, evaluated using the SWN model. This shows the expected impact of an attack on the vehicles participating in the mission, in a time window starting from the attack occurrence. These are results obtainable from the SWN model on its own, without the other components of our mission viability evaluation method. Second, we show results for the overall mission analysis, combining the SWN model results with the coverage rate function to obtain mission coverage, and show how the results are affected by parameters relating to the mission.

### 10.6.1 Analysis using the SWN model

We report here the results of the impact analysis for sensor vehicles only (i.e. without discussing relays), as they are the basis for overall mission coverage, and because the insights obtained from the

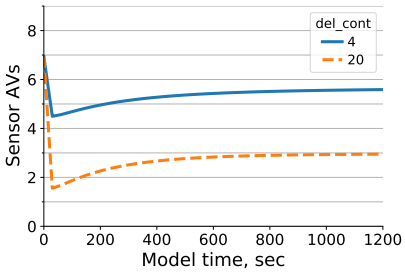
Table 10.3: Parameter values used

| | | |
|---|---|---|
| $del_{cont}$: containment delay, $del_{cont} \in \{4, 20\}$, $(del_{cont} = 1/r_{cont})$ | $Red$: replacement (redundant) UAVs, $Red \in \{0, 3, 6, 9\}$ | $N_i$: $N_1 = 2$, $N_2 = 8$ |
| $del_{prpg}$: propagation delay, $del_{prpg} \in \{2, 5\}$, $(del_{prpg} = 1/r_{prpg})$ | $X_i$: excess/redundancy in role (1 over limit), $X_i = L_i + 1$ | $L_i$: $L_1 = 2$, $L_2 = 4$ |
| $del_{rec}$: replacement delay, $del_{rec} \in \{300, 599, 901\}$, $(del_{rec} = 1/r_{rec})$ | $t$: model time, seconds, $t \in [0, 1200]$ | $S_{si}$: $S_{11} = 1$, $S_{12} = 0$; |
| $del_{swp}$: swap delay, $del_{swp} \in \{40, 150, 300\}$, $(del_{swp} = 1/r_{swp})$ | $r_i$: UAV with role $i$, $r_1$: relay, $r_2$: sensor (temp/video) | : $S_{21} = 1$, $S_{22} = 3$; |
| $del_{desp}$: despatch delay, $del_{desp} = 25$, $(del_{desp} = 1/r_{desp})$ | $E_i$: entry UAV role: $E_1 = 0$, $E_2 = 1$ | : $S_{31} = 0$, $S_{32} = 4$ |

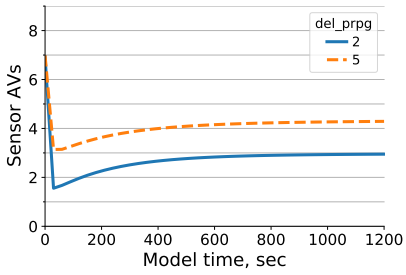
results for relay vehicles are qualitatively similar. The mission analysis in the next section combines both the sensor and relay vehicles into the mission coverage rate. Table 10.3 shows the parameter values for our SWN model used in the analysis of the mission considered in our case study.

Fig. 10.5 shows how the attack impacts the number of vehicles taking part in the mission over time, excluding those that have been compromised by the attack. The figure depicts the expected number of sensor vehicles, i.e. tokens representing sensor vehicles in the place *UAVs* of the Petri net, evaluated over the model time window using the transient solution provided by the GreatSPN solver [ABB+16]. The various panels of the figure show the effect on the expected number of sensor vehicles from varying different parameters values. When a parameter is not varied, it is kept at its baseline level; the baseline parameter values used are: del_cont=20; del_prpg=2; del_rec=599; del_swp=40; Red=3.

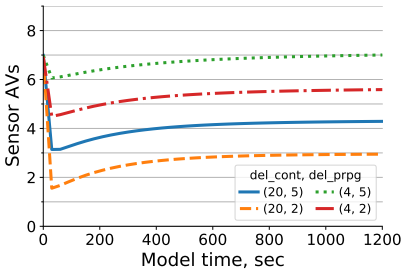
We can observe in Fig. 10.5 that: **1.** *Containment delay has a large impact* on the effectiveness of the defense: early on there is a difference of around three sensor vehicles between the slower and faster containment cases (Fig. 10.5a). This effect can persist until the end of the evaluation time if there are not enough replacement UAVs to replace all the compromised ones. **2.** A longer propagation delay reduces the impact on sensor vehicle numbers, as it increases the likelihood that containment occurs before propagation (Fig. 10.5b). **3.** Replacement delay causes some difference initially, with longer delays reducing the number of sensors in use, but this closes out toward the end of the evaluation time frame (Fig. 10.5d). **4.** Variations in the swap delay, i.e. the time to swap UAV roles, have a minor impact on the expected number of sensor UAVs available in our case study (Fig. 10.5e). **5.** *The number of replacement vehicles has a huge impact if containment is slow.* With a large number of replacement vehicles, the impact of an attack is eventually mitigated even if it propagates to the whole network, while a system with faster containment can manage with fewer replacements (Fig. 10.5h).



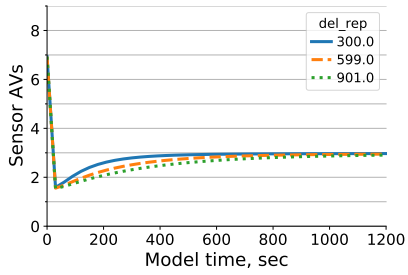

(a) Containment delay {4,20}



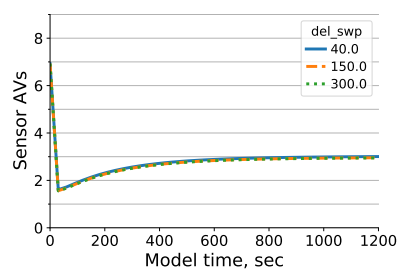

(b) Propagation delay {2,5}



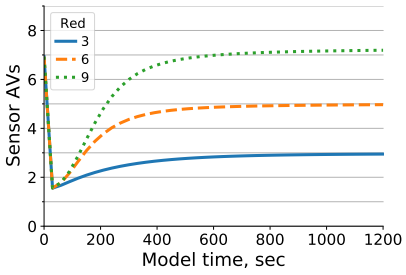

(c) Containment and propagation delays



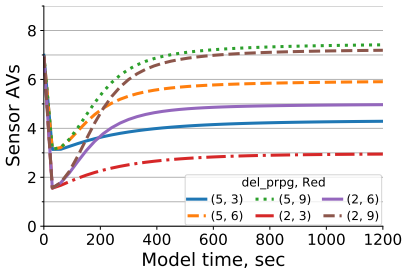

(d) Replacement delay {300,599,901}



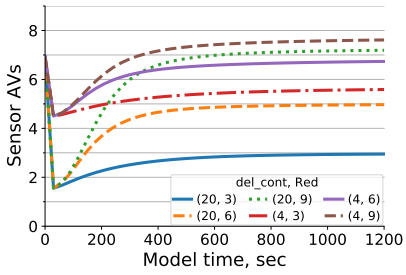

(e) Swap delay {40,150,300}



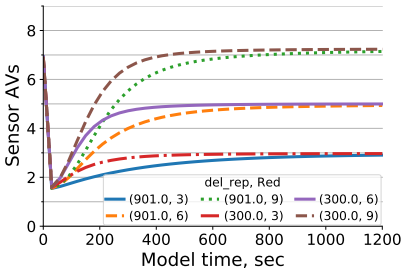

(f) Count of replacement UAVs {3,6,9}




(g) Propagation delay and replacement UAVs




(h) Containment delay and replacement UAVs



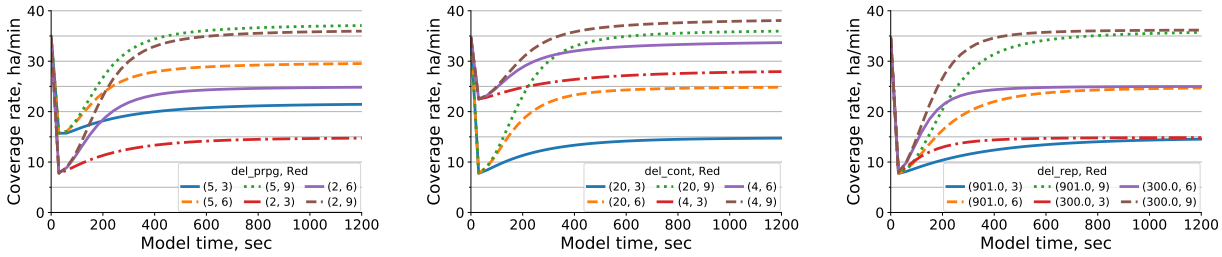

(i) Replacement delay and replacement UAVs

Figure 10.5: Expected number of sensor vehicles online. Baseline parameter values: del_cont=20; del_prpg=2; del_rec=599; del_swp=40; Red=3

### 10.6.2 Mission success analysis

Using the coverage rate function (10.1) to convert the SWN analysis results to mission coverage, Fig. 10.6 shows the coverage impacts from attacks. The patterns are similar to those in the sensor numbers plots, but with somewhat deeper dips due to the impact compromised relays have on the coverage rate.

Table 10.4 shows the total expected coverage reachable during a mission when sustaining a cyber attack, using the approach explained in Section 10.5.4. The table illustrates the impacts of individual vehicle's coverage rate (as in Table 10.2), maximum mission time, and the time of attack. The table




(a) Propagation delay and replacement UAVs

(b) Containment delay and replacement UAVs

(c) Replacement delay and replacement UAVs

Figure 10.6: Coverage rates during attack. Baseline parameter values: del_cont=20; del_prpg=2; del_rec=599; del_swp=40; Red=3

Table 10.4: Total expected mission coverage, ha

| | UAV coverage, ha/min | 1 | | | 5 | | | 10 | | | 15 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Max time, min | 20 | 30 | 40 | 20 | 30 | 40 | 20 | 30 | 40 | 20 | 30 | 40 |
| Replacements | Attack time, min | | | | | | | | | | | | |
| 3 | 5 | 41.1 | 54.6 | 69.7 | 205.7 | 273.2 | 348.4 | 411.4 | 546.5 | 696.7 | 617.1 | 819.7 | 1045.1 |
| | 15 | 66.7 | 81.1 | 94.6 | 333.4 | 405.7 | 473.2 | 666.8 | 811.4 | 946.5 | 1000.2 | 1217.1 | 1419.7 |
| | 25 | 80.0 | 106.7 | 121.1 | 400.0 | 533.4 | 605.7 | 800.0 | 1066.8 | 1211.4 | 1200.0 | 1600.2 | 1817.1 |
| 9 | 5 | 67.4 | 99.6 | 135.4 | 337.2 | 498.1 | 677.1 | 674.3 | 996.2 | 1354.1 | 1011.5 | 1494.3 | 2031.2 |
| | 15 | 72.7 | 107.4 | 139.6 | 363.4 | 537.2 | 698.1 | 726.8 | 1074.3 | 1396.2 | 1090.3 | 1611.5 | 2094.3 |
| | 25 | 80.0 | 112.7 | 147.4 | 400.0 | 563.4 | 737.2 | 800.0 | 1126.8 | 1474.3 | 1200.0 | 1690.3 | 2211.5 |

Notes: A setup with 2 relays, 8 sensor UAVs; SWN parameter values: del_cont=20; del_prpg=2; del_rec=599; del_swp=40; "Replacements" refers to Red.

is calculated for the sample mission setup (as in Fig. 10.2a-10.2b and Fig. 10.3a) with two interconnected relays and eight sensor UAVs, four of which are in the video role, and the baseline SWN parameter values. The same SWN attack evaluation is applied to all cases. Table 10.4 shows that **a)** the same attack occurring earlier in the mission time significantly limits the overall expected coverage achieved relative to a later attack if the number of replacement vehicles is low (Red=3), but a high redundancy reduces this effect (Red=9); **b)** The total area that can be covered within a mission time varies greatly with the individual UAV coverage rate.

## 10.7 Conclusion

In this chapter, we applied our resilience impact assessment methodology to propose an approach for evaluating the impacts of cyber attacks on the viability of a mission involving the deployment of multiple UAVs that can communicate with each other. It provides a structured approach for mission planning support and viability evaluation, helping to determine: a) how many vehicles are needed for a mission with a given coverage requirement, and how this varies with parameters such as mission

connectivity configuration, maximum mission time, vehicle specifications, and the occurrence time of a possible attack; b) how many redundant vehicles should be available to act as replacements in case of an attack; c) how different parameters impact the total coverage reachable within a mission time, which enables a better understanding of the impact of an attack and can help in prioritising different capabilities.

We have shown the use of this approach in a wildfire survey case study scenario informed by actual data on fires and UAV capabilities. The results highlight the importance of the speed of containment and appropriate capacity for recovery in the form of redundant vehicles. While their importance as the key determinants of overall mission performance during attacks is to be expected, our method provides a way to quantify their impact and to estimate their relative importance where a trade-off is possible, as faster containment can make up for fewer redundant vehicles and vice versa.

Although we have discussed an implementation to a specific setting, our approach can be applied more generally. Most importantly, the approach to modelling attack progression and impact analysis with SWNs is applicable to other settings that involve several components that are similar in how they function and how an attack can laterally proceed across them. This applies e.g. in other multi-UAV deployments, or self organising robotic swarms. We also demonstrated how the SWN model of attack progression can be used in conjunction with the production modelling part of our methodology to conduct analysis of mission-level success, via the use of a performance metric relating to mission progress. We believe that the estimated coverage metric we used can be applied to other distributed (aerial) missions where area coverage is important, such as search and rescue or disaster assessment [SSAF+19], with only small adjustments. For other domains a different metric might be required, but the base approach would still apply.

# Chapter 11

# Conclusion

For evaluating the resilience of a system to cyber attacks, and for making choices over ways in which it could be improved, a way to evaluate the impact of cyber attacks and defensive actions is required. This thesis proposes such a methodology, for attacks that can disrupt system output production.

Specifically, our resilience impact assessment methodology focuses on quantifying the system output performance impact, over time, from attacks and defensive actions, and expressing these impacts in terms of monetary values. While related work and approaches exist along many aspects of the modelling, such as on cyber resilience and attack impact assessment, existing approaches lack features required to meet our aims on resilience, attack modelling, and cost-effectiveness evaluation. Thus, our methodology implementations represent the first impact assessment approach for cyber resilience evaluation that considers the full duration of detailed attack events and provides cost evaluations.

This chapter summarises the content of the previous chapters, discusses the contributions made in the thesis, explains possible applications for our methodology, and suggests areas for future work.

## 11.1 Summary

In this thesis, we have introduced a modelling methodology for impact assessment and cyber resilience evaluation of systems, based on their output performance during attacks. The main purpose

of the methodology is to enable estimating the benefits of actions that can improve resilience to cyber attacks, and the cost-effectiveness of alternatives.

To achieve this, the methodology consists of three key elements: an *attack progression model*, a *production model*, and a *cost model*. The combination of attack progression and production modelling is used to evaluate attack impacts, in terms of changes to output production. The cost modelling is used to express the production impacts in monetary terms, and to consider the costs of defensive actions and resilience improvements. These modelling components were explained, each in turn, in Chapters 4-7, including the different implementations of them that we used when applying the methodology to our use-cases.

In addition to describing this methodology and setting out its structure, our key contributions relating to the methodology are the following:

*Providing an attack progression modelling approach suitable for evaluating expected impacts of possible attack scenarios.* The impact assessment aims, and resilience focus, impose several requirements to our attack modelling that are not fully satisfied by previous approaches that we know of. For example, while risk assessment approaches using attack graphs enable assigning probabilities for certain system compromises becoming compromised in case of an attack, our evaluations require a different type of analysis, where full attack outcomes are assigned probabilities. As a consequence, we have provided an approach that not only considers the possible paths for attacks, but also takes into account attacker behaviour and defensive actions to form the outcomes that a given attack scenario can yield.

As part of our production modelling, *we propose a way to evaluate the impact of attack outcomes using QNs for detailed performance modelling*. For this, we provided a description for how the stages of attack impact are represented as QNs showing the effect on the different production model components, enabling the analysis of the performance impact from several stages of attack and recovery. As far as we are aware, this has not been done in the context of general multi-stage attacks (although QNs have been used for analysing DDoS cases, these are special cases of attacks where the attack occurs in only one stage), and can thus inspire further work.

In our cost model, *in addition to describing how the approach to calculating the costs related to*

*impacts on production, we provided approaches to relating the costs that occur during an attack event to costs in future time periods*. First, we set out a cost model for relating attack mitigation benefits from resilience improvements to the cost of the investment to the improvement, and provided a metric, TUL, to simplify assessing this benefit/cost relation by setting it in terms that can be directly compared to the expected frequency of attacks. The use of this metric was shown in the context of the redundancy planning use case in Chapter 8. Second, we proposed a method to compare the short-term costs and benefits of reactive actions with the longer-term ones (using possible "trajectories" of how an attack evolves), implemented in the case of the countermeasure selection work in Chapter 9.

*We applied our resilience-impact assessment methodology in three studies*, on problems where cyber resilience and attack impact assessment are of interest. In Chapter 8, we apply the methodology to determining a cost-effective allocation of redundancy (with and without diversity) to the components of a system, evaluating the expected cost impact of attack scenarios under different redundancy designs. As part of this, we provide an algorithm for generating impact comparisons for system capability/architectural design changes, such as redundancy and diversity as in the use-case considered, and describe an optimisation problem for choosing the cost-efficient redundancy allocation. Additionally, the case study we use illustrates how our cost modelling approach applies to systems that are subject to SLAs. The results of our case-study evaluations showed that redundancy with diversity can be effective at reducing the production impact of attack scenarios, and the costs related to SLA penalties, but determining the cost-effectiveness of the redundancy investments requires comparing this benefit to the excess maintenance costs arising from the redundancy. As various parameters can impact this cost-benefit balance, it is useful to be able to model and quantify these costs and benefits, which our methodology provides a solution to.

Chapter 9 shows our work on applying the methodology to the problem of attack countermeasure (CM) selection, where we choose defensive actions based on their expected impacts on the costs arising from an attack event. In addition to methodological contributions already mentioned above, the key contributions specific to this work are the following: a) Proposing and implementing a CM selection approach based on cost effectiveness, using a resilience approach to impact estimation over time and providing an algorithm for choosing CMs based on that; b) Using simulated attack events, both in a case study setting and on random synthetic graphs, to evaluate our CM selection approach,

we showed that our approach can be more cost efficient in making patching choices than an alternative based on an existing impact assessment approach, and another alternative based on a rule-of-thumb; c) At a more general level, the results we obtained suggest that the longer-run thinking about the impacts and costs (that is inherent to our methodology) could bring about cost savings, and for this there is a need for modelling, as the cost-effective solution does not follow a simple rule-of-thumb. Further, the results imply that with a good recovery mechanism in place, it can be more cost-effective to try to react to the attack in a way that causes less disruption than by stopping systems, which supports the view that response strategies like we suggested do indeed have their place.

Finally, Chapter 10 presents the use-case of evaluating the viability of a multi-UAV mission if an attack occurs, where our methodology is used to estimate the impact of a spreading attack given the mission design, component redundancy and recovery. Our mission analysis can be used for designing the appropriate levels of redundancy and recovery capacity to ensure meeting the mission requirements in the event of an attack, or when an attack is taking place, to evaluate the expected level of completion of the mission and determine if the mission should be aborted. This work showed how our methodology could be applied to analysing a mission, showing how the concept of 'production' applies to such systems based on the mission aims, and how they are fulfilled, and provides an alternative way to conduct the analysis of attacks, using an SWN model to do the stochastic analysis.

## 11.2 Applications for our methodology

As a way to evaluate the impact of cyber attacks on a system, our methodology could be used wherever such impact assessment is beneficial, such as: 1. As part of the approaches used for *planning the design for a system*, or a mission, and for determining capability improvements, component replacements and upgrades. 2. As part of *incident management*, to assess impacts of attacks and countermeasures to them, to formulate a response strategy or choosing between reactive responses. 3. For deciding an appropriate service-level agreement (SLA) to offer to customers, by assessing the impact of different SLAs on the expected amount of penalties to be paid due to attack scenarios. 4. With further adaptation, the methodology could be used to assess whether companies are adequately

incentivised to invest in cyber resilience. For example, using an approach similar to ours in Chapter 8, regulators could conduct analysis on whether the structure and terms of SLAs offered by companies are too effective at shielding them from penalties, and therefore limiting their appetite to invest in improving their cyber security and resilience.

When applying the methodology, the specific use case, system, and attack characteristics can pose specific requirements to the implementation of parts of the methodology. The largest methodological difference in our application examples occurs with relation to the implementation of the attack progression model. In the mission viability analysis for a team of UAVs in Chapter 10, it is important to be able to evaluate the different interleavings of attack stages and the recovery/swap and attack containment actions, to formulate a view of the average number of roles being filled over time. As a consequence, we opted for modelling the attack propagation using a Petri net formalism, namely SWNs, as this enables stochastic analysis over time using the transient solution of a continuous-time Markov chain (CTMC). By comparison, the two other applications we introduced do not require the same detail for evaluating how attack and defence stages can interleave: In the redundancy planning application of Chapter 8, there is no such complex interleaving of attack and defence occurring, but detection and containment simply occur after certain number of stages have been taken, given a probability of detection. Modelling of more complex interleaving is not required, as the attack launches the impact on one go, once detected. In the countermeasure selection work in Chapter 9, the use case is about running the CM selection analysis during an attack event, so the interleavings are assumed to be more limited, although they are evaluated a few steps ahead. Additionally, it cannot be modelled using a SWN in the same way, because the attack behaviour is different and not pre-defined, and the defensive actions are chosen based on impacts, thus not pre-determined either.

## 11.3 Future Work

The research presented in this thesis focused on determining how an approach to resilience-oriented assessment of cyber-attack impacts should work, how the different modelling components should work together, and showing how it can be implemented and applied in different use cases. Therefore,

the research has been about building and testing the foundations for our methodology, and the results so far have emphasised showing the usefulness of the methodology in theoretical, synthetic systems and attacks, albeit informed by real world data. Conducting evaluations of the methodology in real systems and with real data forms an important direction of future work on our methodology, even though such data may be hard to obtain due to its business sensitivity. The results of such evaluations can then help guide further development of the methodology and its implementations.

The applications and methodology implementations provided would benefit from evaluations that compare them directly to alternative techniques. Such evaluations were not done, except to a limited extent in the CM selection work, as we are not aware of approaches that are similar enough that they could be compared directly in a publicly available testing environment or case study description. While we can make arguments over e.g. how we differ from other impact assessment approaches in [AJPS11, AJ17, SSL17, CYS+18, HSKG20], direct comparisons of the relative usefulness of different approaches in practical use in real systems and attacks would require access to system and attack information that system owners would prefer to keep private. Thus such comparisons would likely require a publicly available testing environment, or be conducted within organisations themselves. In the absence of such detailed testing data, we have chosen applications and implementations that we believe highlight the usefulness of the approach in cases that are relatively general and common. Hence our resilience planning work focused on redundancy and diversity, as these are widely applicable mechanisms to improve the resilience of systems.

On techniques to improve resilience, such as the architectural techniques discussed by [Gol10, BG11], we have focused on redundancy and diversity, instead of building sample implementations of all such techniques. Enabling the remaining techniques to be assessed with our modelling approach would require creating sample models of architectures that implement the different techniques. However, we believe that the approaches that we propose for analysing decisions would be valid and useful for analysing other techniques as well, once modelled. For example, the approach we use for comparing the effectiveness of redundancy allocations could apply to various techniques that impact the architecture of the system or mission, as long as their impact on the attack progression and production models can be modelled. The work of building implementations and case studies for the other techniques, and possible extensions to the methodology that may be required, is likely best done within organisations

with access to the architectures and techniques in question.

There are also some modelling-specific extensions that we would like to make. First, in the attack progression modelling, we considered cases where disruption on the system is launched at the attacker's discretion (in the redundancy planning work, Chapter 8), and where all exploits used for move steps cause disruption in the component (the CM selection and mission viability applications, Chapters 9,10). We have not yet considered cases where some exploits are disruptive and others not. Such a case could mean the existence of paths that require a disruptive exploit, which would be easily detectable, but could be shorter than an alternative. Addressing the issue of choosing between such paths could be done by adding priorities to the attacker behaviour model, likely in a probabilistic manner instead of completely banning lower priority paths. Adding such priorities in the attack progression modelling would therefore affect the probabilities of attack outcomes, and thus the expected impact calculations, but not the set of possible attack outcomes themselves.

Second, to extend the countermeasure selection work, the most direct future direction is modelling the impact of limiting connections or shutting down system components to contain the attack while maintaining some productive services. Also, before extending the CM selection modelling further, more detailed investigations should be made, on real system environments and with realistic attack scenarios, to determine if the actions that would be considered are fast enough to implement, to justify them over simpler actions that the system has been prepared to take quickly.

Third, adding explicit attack detection into the SWN model of attack progression used for the UAV mission analysis in Chapter 10. In that study, detection was implicit, as it was assumed that the attack steps cause clear disruption that leads to detection. However, if the steps would not cause such disruption, detection could be modelled explicitly, perhaps by adding transitions representing attack detection into the SWN. Such a transition could launch purely based on time, or have some behaviour based on how many stages have occurred, so the likelihood of it firing increases with every stage etc.

# Appendix A

# Redundancy planning appendices

## A.1 Complexity: Attack outcomes to evaluate, and variant cases

When evaluating the effectiveness of different redundancy allocations in Chapter 8, for each server allocation (e.g. [2,0,2,1]), the expected impact of cyber attacks is calculated based on the losses due to performance impacts across the different possible outcomes of the attack scenarios. That is, for each candidate server allocation, the performance of the system (and related SLA losses, if any) must be evaluated for each attack outcome in each attack scenario considered. Consequently, the computational complexity of the cost impact evaluation depends largely on the number of different attack outcomes that must be tested. In this section we explain in more detail what this means, and also how the memoisation approach we use speeds things up by reducing the number of times a performance calculation is conducted.

### A.1.1 Variant attack cases for attack scenarios

Adding a diverse implementation of a given server leads to added variant cases for the attack scenarios, accounting for the attacker being able to exploit both the existing ("regular") and the alternative server, or only one of them (or neither). The likelihood of each of them is determined based on the assumptions about attacker capabilities. In this section we focus on the number of cases created,

as that contributes to computational complexity, while the probabilities assigned to each variant are discussed in Section A.2 below.

Thus, each case of splitting a server cluster into two types (regular and alternative) yields four variant cases for an attack scenario. For example, Table A.1 shows the variants arising from diversification for attack Scenario 1. This also includes two sub-variants (1a and 1b) for the case where the attacker exploits both the regular and alternative servers, where the order of the exploits is changed. This different ordering may not affect the impact or likelihood of attacks, but is considered for completeness.

Table A.1: Variants to attack Scenario 1 due to diversification

| Variant | **Non-diversified case** | Variant | **Diversified case** |
|---|---|---|---|
| 1 | $[A, P_1, P_2]$ | 1a | $[A, P_1, P_2, P_{1A}, P_{2A}]$ |
| 2 | $[A]$ | 1b | $[A, P_{1A}, P_{2A}, P_1, P_2]$ |
| | | 2 | $[A, P_1, P_2]$ |
| | | 3 | $[A, P_{1A}, P_{2A}]$ |
| | | 4 | $[A]$ |

Where the variants yield the same attack impact matrix $M_{attack}$, with memoisation the performance impact gets evaluated only once for variants with equivalent impacts. Thus the addition of variants only increases computational complexity in terms of computing the probabilities to be assigned to each case, which is computationally cheap.

### A.1.2 Attack outcomes to evaluate for a given server allocation

The number of outcomes that must be evaluated for a given scenario (or its variant) depends on: a) the number of steps in the attack path for the scenario; b) number of copies of privileges the attacker needs to obtain for the full success of the attack; c) assumptions on attack progression and defence (e.g. Case 1 vs Case 2 defence, the order in which the attacker conducts repeated exploits, etc.).

The simplest determinant is the number of steps there are in the path describing the success outcome of a scenario (or its diversified variant). This is, however, typically not very relevant for the computational complexity of our optimisation, as many outcomes will only consist of stepping stone exploits and do not have a system performance impact. Therefore only a subset of the outcomes, those that have a performance impact, require a performance evaluation with separate $M_{attack}$ matrices, while "no impact" outcomes get grouped together and assigned a $M_{attack}$ matrix without an attack stage.

**Algorithm 2** Evaluation of SLA penalties for each candidate allocation

**Precondition:** $m$ is a candidate server allocation; $scenarios$ a list of scenario objects; $params$ is a list of other parameters, including $clients$ (the number of clients), $p_c$ (the SLA breach cost per client) and $breach\_limit$ (breach limit, $\beta$).

```
1:  function SLAPENALTIES(m, scenarios, params)
2:      SLApenalty := clients * p_c
3:      outcomes := gen_outcomes(m, scenarios, params)
4:      att_mats := gen_att_mats(m, outcomes, params)
5:      E_pen := 0                                          ▷ Expected penalty
6:      for each am in att_mats do
7:          breach_metric := perf_eval(am, m, params)
8:          if breach_metric > breach_limit then
9:              E_pen := E_pen + am.weight * SLApenalty
10:         end if
11:     end for
12:     return E_pen
13: end function
```

The pseudocode Algorithm 2 shows the expected SLA penalty calculation, which must be run for each candidate allocation tried by the optimisation. Consequently, it contains the key parts impacting the complexity of the approach. The attack outcomes (line 3) must be calculated for each allocation based on the allocation $m$ and the attack scenarios, and the attack matrices ($M_{attack}$) have to be generated from these (line 4). After this, the system performance must be evaluated for each attack outcome matrix (line 7), in order to calculate the expected penalty from SLA breaches (line 9). The number of rows in the $M_{attack}$ matrix determines how many performance values are required. However, memoisation can drastically limit the number of performance evaluations run, as we explain next.

### A.1.3 Worst case complexity: Optimisation bounds and attack outcomes

The number of times the evaluation is conducted during an optimisation depends on the size of the solution space and the number of attack outcomes to consider, in addition to the effectiveness of the optimisation algorithm. In the worst case the optimisation will simply attempt all candidate allocations in the search space. Due to the way server numbers in an allocation affect attack outcomes and their impact, many of the same outcomes and effective allocations occur across multiple candidate server allocations. Here the memoisation of solutions based on $M_{attack}$ matrices helps.

Further, the "effective allocations" resulting from attack outcomes have a lot of repetition across different allocations and attack scenarios. This can occur also within a given scenario when not all attack steps cause server impacts, as can be seen in Table A.2. Due to this, we can use another type of memoisation: saving the performance values relating to a given effective allocation.[1] As the time to reach these effective allocations depends on the number of attack steps and will differ, we memoise the performance levels corresponding to the allocations, and then apply these levels for the duration appropriate for the attack outcome in question. With this memoisation approach, *the number of times we evaluate the performance is capped by the number of distinct combinations of servers in the search space, plus the number of distinct effective allocations from attack outcomes.*

Table A.2: Effective allocations from attack outcomes, Scenario 1 attack variant 1a

| # | Attack outcome | Functioning servers |
|---|---|---|
| | **Case 1 defence, allocation [2,2,2,1]** | |
| 1 | $[A, P_1, P_1, P_2, P_2, P_{1A}, P_{1A}, P_{2A}, P_{2A}]$ | $[0, 0, 2, 1]$ |
| 2 | $[A, P_1, P_1, P_2, P_2, P_{1A}, P_{1A}]$ | $[0, 2, 2, 1]$ |
| 3 | $[A, P_1, P_1, P_2, P_2]$ | $[0, 2, 2, 1]$ (repeat) |
| 4 | $[A, P_1, P_1]$ | ~~$[2, 2, 2, 1]$~~ (no impact) |
| 5 | $[A]$ | ~~$[2, 2, 2, 1]$~~ (no impact) |
| | **Case 1 defence, allocation [2,3,2,1]** | |
| 1 | $[A, P_1, P_1, P_2, P_2, P_{1A}, P_{1A}, P_{1A}, P_{2A}, P_{2A}, P_{2A}]$ | $[0, 0, 2, 1]$ |
| 2 | $[A, P_1, P_1, P_2, P_2, P_{1A}, P_{1A}, P_{1A}]$ | $[0, 3, 2, 1]$ |
| 3 | $[A, P_1, P_1, P_2, P_2]$ | $[0, 3, 2, 1]$ (repeat) |
| 4 | $[A, P_1, P_1]$ | ~~$[2, 3, 2, 1]$~~ (no impact) |
| 5 | $[A]$ | ~~$[2, 3, 2, 1]$~~ (no impact) |
| | **Case 2 defence, allocation [2,2,2,1]** | |
| 1 | $[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}]$ | $[0, 0, 2, 1]$ |
| 2 | $[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}]$ | $[0, 1, 2, 1]$ |
| 3 | $[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}]$ | $[0, 1, 2, 1]$ (repeat) |
| 4 | $[A, P_1, P_2, P_1, P_2, P_{1A}]$ | $[0, 2, 2, 1]$ |
| 5 | $[A, P_1, P_2, P_1, P_2]$ | $[0, 2, 2, 1]$ (repeat) |
| 6 | $[A, P_1, P_2, P_1]$ | $[1, 2, 2, 1]$ |
| 7 | $[A, P_1, P_2]$ | $[1, 2, 2, 1]$ (repeat) |
| 8 | $[A, P_1]$ | ~~$[2, 2, 2, 1]$~~ (no impact) |
| 9 | $[A]$ | ~~$[2, 2, 2, 1]$~~ (no impact) |

| # | Attack outcome | Functioning servers |
|---|---|---|
| | **Case 2 defence, allocation [2,3,2,1]** | |
| 1 | $[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}, P_{1A}, P_{2A}]$ | $[0, 0, 2, 1]$ |
| 2 | $[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}, P_{1A}]$ | $[0, 1, 2, 1]$ |
| 3 | $[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}]$ | $[0, 1, 2, 1]$ (repeat) |
| 4 | $[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}]$ | $[0, 2, 2, 1]$ |
| 5 | $[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}]$ | $[0, 2, 2, 1]$ (repeat) |
| 6 | $[A, P_1, P_2, P_1, P_2, P_{1A}]$ | $[0, 3, 2, 1]$ |
| 7 | $[A, P_1, P_2, P_1, P_2]$ | $[0, 3, 2, 1]$ (repeat) |
| 8 | $[A, P_1, P_2, P_1]$ | $[1, 3, 2, 1]$ |
| 9 | $[A, P_1, P_2]$ | $[1, 3, 2, 1]$ (repeat) |
| 10 | $[A, P_1]$ | ~~$[2, 3, 2, 1]$~~ (no impact) |
| 11 | $[A]$ | ~~$[2, 3, 2, 1]$~~ (no impact) |
| | **Case 2 defence, allocation [3,3,2,1]** | |
| 1 | $[A, P_1, P_2, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}, P_{1A}, P_{2A}]$ | $[0, 0, 2, 1]$ |
| 2 | $[A, P_1, P_2, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}, P_{1A}]$ | $[0, 1, 2, 1]$ |
| 3 | $[A, P_1, P_2, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}]$ | $[0, 1, 2, 1]$ (repeat) |
| 4 | $[A, P_1, P_2, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}]$ | $[0, 2, 2, 1]$ |
| 5 | $[A, P_1, P_2, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}]$ | $[0, 2, 2, 1]$ (repeat) |
| 6 | $[A, P_1, P_2, P_1, P_2, P_1, P_2, P_{1A}]$ | $[0, 3, 2, 1]$ |
| 7 | $[A, P_1, P_2, P_1, P_2, P_1, P_2]$ | $[0, 3, 2, 1]$ (repeat) |
| 8 | $[A, P_1, P_2, P_1, P_2, P_1]$ | $[1, 3, 2, 1]$ |
| 9 | $[A, P_1, P_2, P_1, P_2]$ | $[1, 3, 2, 1]$ (repeat) |
| 10 | $[A, P_1, P_2, P_1]$ | $[2, 3, 2, 1]$ |
| 11 | $[A, P_1, P_2]$ | $[2, 3, 2, 1]$ (repeat) |
| 12 | $[A, P_1]$ | ~~$[3, 3, 2, 1]$~~ (no impact) |
| 13 | $[A]$ | ~~$[3, 3, 2, 1]$~~ (no impact) |

The **defence assumptions** on how many server copies are impacted by an attack step play a large role in the complexity of the model if the number of servers in a cluster is large. We considered two cases of this, representing the extremes, introduced in Section 8.5.3. The "**Case 1**" assumption, where the defence cannot stop an attack step that affects one copy of servers in a cluster from affecting all other identical copies in the same cluster, yields only two outcomes for attack steps on a given server cluster: either all server copies in the cluster are impacted, or none of them are. So only one additional "effective allocation" resulting from an attack needs to be evaluated, in addition to the

[1] Note that the use of this particular memoisation relies on the transitions between states being fast relative to recovery times, and could lead to an evaluation error if transition times between states are slow. If transition times are significant enough to cause an error, another evaluation stage could be added to counter such error, as explained in Section 8.5.8.2.

fully functioning candidate allocation that is evaluated in any case. Thus the number of evaluations required is not affected directly by the number of servers, only the number of distinct combinations in the search space. For "**Case 2**", where it is assumed that the attack (and its impact) can be stopped between exploits of copies of identical servers, the number of attack outcomes to evaluate increases with the number of servers. Table A.2 illustrates this difference between the two cases, showing that Case 2 yields more distinct cases of effective allocations to evaluate.

A generalised version capturing the Case 1 and Case 2 defence assumptions and the situations in between is the following: Assume a server cluster with $m$ servers, and that attacks to the cluster can succeed in groups of $G \in [1, m]$ servers. Denoting $K = \lceil m/G \rceil$, there can then be $K + 1$ distinct attack outcomes from attack steps affecting that cluster. In Case 1 $G = m$ and $K = 1$, while in Case 2, $G = 1$ and $K = m$. This extends to a whole server allocation as follows: for an allocation vector with $n$ elements, $m_n$ representing the number of servers in the $n$th cluster and $K_n$ the distinct outcomes to evaluate for the cluster, the number of distinct attack outcomes to evaluate is $\prod_{i=1}^{n}(K_i + 1)$. With Case 1, this is always $2^n$, and in Case 2 the worst case depends on the upper bound of solution space, so $\prod_{i=1}^{n}(u_i + 1)$, where $u_i$ is the upper bound given for the number of servers in cluster $i$.

For example, if the lower bound for servers in candidate allocations was given by $l = [2, 2, 2, 2]$, upper bound $u = [5, 5, 5, 5]$ and $G_i = m_i \forall i$ (Case 1), then at worst, performance would be evaluated $\prod_{i=1}^{n}(u_i - l_i + 1) + \prod_{i=1}^{n}(K_i + 1) = 4^4 + 2^4 = 272$ times (in fact some of the effective allocations are the same as some candidate allocations, but this overlap is of little consequence for the overall complexity). This case scales well, as the number of evaluations depends on the upper and lower bound, not the number of servers. However, with Case 2 $K_i = m_i \forall i$, and at worst we would have to evaluate all combinations between the upper bound $u$ and $[0, 0, 0, 0]$, as the effective allocations in the attack outcomes include each number of servers between 0 and $K_i$ for each cluster $i$. In that case, we would estimate performance in the worst case $\prod_{i=1}^{n}(u_i + 1) = 6^4 = 1296$ times. With the Case 2 assumption, increasing the number of servers in the clusters makes the number of performance evaluations required explode quickly, e.g. with $u = [20, 20, 20, 20]$ the number of evaluations would be $160\,000$. Using the fluid solver used in our 2nd optimisation stage, this would take over 130 hours to evaluate based on an average of around 3s per performance evaluation.

As apparent from the previous paragraph, different values of $K_i$ make a great difference to the complexity of the algorithm. In the intermediate cases where the success of an attack on a cluster $j$ progresses at the level of subgroups $G_j > 1$, the number of evaluations required can still be reasonable even for high numbers of servers. For example, if we assume that a cluster of 100 servers could be defended in four equal compartments so when detected the defence could contain the attack from affecting the remaining non-infected compartment, we would have $K_j + 1 = 5$ distinct outcomes from attacks on the cluster, with 0, 25, 50, 75 and 100 functioning servers. In this case the complexity of the algorithm would still be manageable, if the bounds given to the optimisation problem were also reasonable. Let's assume $u = [105, 105, 105, 105]$, $l = [100, 100, 100, 100]$ and $K_i = 4 \forall i$. Then the estimation count in the worst case is $\prod_{i=1}^{n}(u_i - l_i + 1) + \prod_{i=1}^{n}(K_i + 1) = 6^4 + 5^4 = 1296 + 625 = 1921$.

Although we have included Case 2 to illustrate the extreme where defence can contain each server copy individually, we believe that in practice, defence is more likely to be of the type in Case 1, for two reasons: 1. If an exploit has gone unnoticed by a detection system, it is likely that the same exploit applied to another copy of an identical server is also not noticed by the detection system; 2. Containing the effects of an attack at the level of individual copies (or sub-groups) of servers seems unrealistic in a cluster of identical servers, as the exploits could continue until the attacker is purged, while containing the attacker by blocking communication can yield the same performance impact.

**Model granularity and scaling:** When the number of servers is high, it may not make sense to consider changes in numbers of individual servers, but groups of servers. Using a one-server granularity is not necessary in such cases, and scaling could reduce the optimisation complexity. For example, with $G_i = 25$, considering upper and lower bounds $u = [125, 125, 125, 125]$, $l = [100, 100, 100, 100]$ without scaling would cause a high number of cases to consider ($\prod_{i=1}^{n}(u_i - l_i + 1) = 456\,976$), but only a few of them are relevant if server additions are considered in groups of $G_i = 25$. In such cases we can scale the representation in the model, so that each unit in cluster $i$ represents a group of servers of size $G_i$. In the above example with $G_i = 25$, we have $u = [5, 5, 5, 5]$, $l = [4, 4, 4, 4]$ and $K_i = 4$ to obtain $\prod_{i=1}^{n}(u_i - l_i + 1) + \prod_{i=1}^{n}(K_i + 1) = 2^4 + 5^4 = 641$ performance evaluations in the worst case.

Note that the scaling is limited by the defence assumption that determines $G_i$, as we want to consider all distinct attack outcomes appropriately. That is, if we have a given $G_i$, we must scale in proportion

to it so that no attack outcomes are lost. So with $G_i = 25$, we can use a $1/25$ or $1/5$ scale, but no others. For Case 1, where $G_i = m$, we can scale with any value $S < m$ as long as $m/S$ is an integer.

**Observed evaluation times:** In our evaluation runs with the QN fluid solution method, in the 2nd stage of our optimisation, the average time to evaluate the performance of the system for a given row of an $M_{attack}$ matrix was 3.08s, while the average time to process a memo hit was 0.00055s. The bounds approximation in the 1st stage of the optimisation, used to quickly evaluate a large search space to find a closer region for the fluid solution, is considerably faster, having taken 0.0056s on average to evaluate one row of an $M_{attack}$ matrix. This was achieved on a standard desktop computer with an Intel i7-6700 CPU with 4 cores (8 threads) at 3.4GHz clock speed, and 16GB of RAM.

## A.2 Attacker capabilities

While diversity increases robustness to attacks that exploit a particular vulnerability, when adding diversity to the system, additional vulnerabilities may be added, thus increasing the threat surface. In our analysis we assume that both server types involved, the regular and the alternative one, have vulnerabilities, and the attacker has capabilities to exploit these vulnerabilities with certain probabilities. We denote by $P(R)$ the probability that the attacker has the capability to exploit a vulnerability in the regular server type, and by $P(A)$ the probability that the attacker is capable of exploiting a vulnerability in the alternative type that is used for adding diversity. This means that, when considering diversity, we are interested in four joint probabilities, $P(R \cap A)$, $P(R \cap A')$, $P(R' \cap A)$, and $P(R' \cap A')$, where event $R'$ denotes the complement of event $R$, i.e. when the attacker does not have the exploit for the vulnerability in the regular server type. These four cases determine the probabilities of scenario variants, which represent the full attack to both server types, partial attacks to type $R$ only and to type $A$ only, and no attack to either server type. For example, for attack Scenario 1 $[A \rightarrow P1, P1 \rightarrow P2]$, the diversified case leads to the scenario variants shown in Table A.3. Although shown in the table for completeness, when setting the relative weights of the variants in our impact evaluation we do not consider the case $(R' \cap A')$ where the attacker lacks the capability to compromise either server type. Consequently, the weights relating to the three other cases are rescaled to sum to 1, as shown in the

"Weight" column of the table.

Table A.3: Scenario variants for attack Scenario 1

| **Non-diversified case** | | | | **Diversified case** | | | |
|---|---|---|---|---|---|---|---|
| **#** | **Variant path** | **Probability** | **Weight** | **#** | **Variant path** | **Probability** | **Weight** |
| 1 | $[A, P_1, P_2]$ | $P(R)$ | 1 | 1a | $[A, P_1, P_2, P_{1A}, P_{2A}]$ | $0.5 \cdot P(R \cap A)$ | $0.5 \cdot P(R \cap A)/\zeta$ |
| | | | | 1b | $[A, P_{1A}, P_{2A}, P_1, P_2]$ | $0.5 \cdot P(R \cap A)$ | $0.5 \cdot P(R \cap A)/\zeta$ |
| | | | | 2 | $[A, P_1, P_2]$ | $P(R \cap A')$ | $P(R \cap A')/\zeta$ |
| | | | | 3 | $[A, P_{1A}, P_{2A}]$ | $P(R' \cap A)$ | $P(R' \cap A)/\zeta$ |
| 2 | $[A]$ | $P(R')$ | 0 | 4 | $[A]$ | $P(R' \cap A')$ | 0 |

Note: The scaling parameter $\zeta = P(R \cap A) + P(R \cap A') + P(R' \cap A)$

To make comparisons to the case with no diversification, where only event $R$ is relevant, we are most interested in the probability $P(R \cap A)$ relative to the probability $P(R)$. Here we test the impact of assuming that the events $R$ and $A$ are not independent, which could arise if attackers are more likely to be able to exploit a vulnerability if they already know how to exploit a similar vulnerability. We are thus interested in how the conditional probability $P(A|R)$ affects the benefits from diversification.

Table A.4: Sensitivity to conditional probability of exploits $P(A|R)$

$\beta$ = **0.001,** $t_r$ = **3**

| $P(A\|R)$ | **Optimum allocation** | **Exp. cost** | **Cost difference** penalty | alloc. | **% diff.** | $P(R \cap A)$ |
|---|---|---|---|---|---|---|
| $p_d = 0.3$ | | | | | | |
| 1 | $[2, 2, 3, 1]$ | 4910 | -1926 | 389 | -23.8 | 0.5 |
| 0.9 | $[2, 2, 3, 1]$ | 4606 | -2230 | 389 | -28.6 | 0.45 |
| 0.8 | $[2, 2, 3, 1]$ | 4352 | -2484 | 389 | -32.5 | 0.4 |
| 0.7 | $[2, 2, 3, 1]$ | 4138 | -2698 | 389 | -35.8 | 0.35 |
| 0.6 | $[2, 2, 3, 1]$ | 3954 | -2882 | 389 | -38.7 | 0.3 |
| 0.5 | $[2, 2, 3, 1]$ | 3795 | -3041 | 389 | -41.1 | 0.25 |
| **Comparison point – Non-diversified optimum** | | | | | | |
| N/A | $[2, 0, 3, 1]$ | 6447 | - | - | - | N/A |
| **Comparison point – reference allocation** | | | | | | |
| N/A | $[2, 0, 2, 1]$ | 6931 | 669 | -185 | +7.5 | N/A |
| $p_d = 0.5$ | | | | | | |
| 1 | $[2, 2, 3, 1]$ | 2514 | -1829 | 389 | -36.4 | 0.5 |
| 0.9 | $[2, 2, 3, 1]$ | 2406 | -1937 | 389 | -39.2 | 0.45 |
| 0.8 | $[2, 2, 3, 1]$ | 2317 | -2026 | 389 | -41.4 | 0.4 |
| 0.7 | $[2, 2, 3, 1]$ | 2241 | -2102 | 389 | -43.3 | 0.35 |
| 0.6 | $[2, 2, 3, 1]$ | 2176 | -2167 | 389 | -45.0 | 0.3 |
| 0.5 | $[2, 2, 3, 1]$ | 2119 | -2224 | 389 | -46.4 | 0.25 |
| **Comparison point – Non-diversified optimum** | | | | | | |
| N/A | $[2, 0, 3, 1]$ | 3954 | - | - | - | N/A |
| **Comparison point – reference allocation** | | | | | | |
| N/A | $[2, 0, 2, 1]$ | 4175 | 406 | -185 | +5.6 | N/A |

Notes: $c_{as}/c_s$=1.05, $P(R)$=0.5, $P(R \cap A') = P(R' \cap A)$

(a) Case 1 defence

$\beta$ = **0.001,** $t_r$ = **3**

| $P(A\|R)$ | **Optimum allocation** | **Exp. cost** | **Cost difference** penalty | alloc. | **% diff.** | $P(R \cap A)$ |
|---|---|---|---|---|---|---|
| $p_d = 0.3$ | | | | | | |
| 1 | $[5, 1, 3, 4]$ | 3167 | 0 | 9 | +0.3 | 0.5 |
| 0.9 | $[4, 2, 3, 4]$ | 3140 | -37 | 19 | -0.6 | 0.45 |
| 0.8 | $[3, 2, 3, 4]$ | 3078 | 87 | -167 | -2.5 | 0.4 |
| 0.7 | $[3, 2, 3, 4]$ | 3023 | 32 | -167 | -4.3 | 0.35 |
| 0.6 | $[3, 2, 3, 4]$ | 2977 | -14 | -167 | -5.7 | 0.3 |
| 0.5 | $[2, 2, 3, 4]$ | 2911 | 105 | -352 | -7.8 | 0.25 |
| **Comparison point – Non-diversified optimum** | | | | | | |
| N/A | $[6, 0, 3, 4]$ | 3158 | - | - | - | N/A |
| **Comparison point – reference allocation** | | | | | | |
| N/A | $[2, 0, 2, 1]$ | 6931 | 5253 | -1480 | +119.5 | N/A |
| $p_d = 0.5$ | | | | | | |
| 1 | $[3, 1, 3, 2]$ | 2030 | 0 | 9 | +0.4 | 0.5 |
| 0.9 | $[2, 2, 3, 2]$ | 2012 | -28 | 19 | -0.4 | 0.45 |
| 0.8 | $[2, 2, 3, 2]$ | 1989 | -51 | 19 | -1.6 | 0.4 |
| 0.7 | $[2, 2, 3, 2]$ | 1970 | -70 | 19 | -2.5 | 0.35 |
| 0.6 | $[2, 2, 3, 2]$ | 1953 | -87 | 19 | -3.4 | 0.3 |
| 0.5 | $[2, 2, 3, 2]$ | 1939 | -101 | 19 | -4.1 | 0.25 |
| **Comparison point – Non-diversified optimum** | | | | | | |
| N/A | $[4, 0, 3, 2]$ | 2021 | - | - | - | N/A |
| **Comparison point – reference allocation** | | | | | | |
| N/A | $[2, 0, 2, 1]$ | 4175 | 2894 | -740 | +106.6 | N/A |

Notes: $c_{as}/c_s$=1.05, $P(R)$=0.5, $P(R \cap A') = P(R' \cap A)$

(b) Case 2 defence

Tables A.4a and A.4b show how the results from diversification are affected by different assumptions with regard to the conditional probability $P(A|R)$, in the case where detection applies to all attack steps, including to replica privileges. The tables assume $P(R) = 0.5$. We also change the conditional

probability $P(A|R')$ so that we maintain $P(R \cap A') = P(R' \cap A)$, i.e. both the attacks with partial capabilities are equally likely. This last assumption is made to simplify the amount of parameters to change, while the focus is on $P(A|R)$ and its impact on $P(R \cap A)$.

From Tables A.4a and A.4b we can see that for the case study system, while the cost impact found is lower if the events $R$ and $A$ are dependent than in the independent case, the benefit from applying diversity remains sizable even with high levels of dependence.

In the analysis included in Chapter 8, we assume that the probability that an attacker has a capability to exploit a given vulnerability is independent from the probability of them having the capability to exploit a different one. This is because vulnerabilities can be very specific to a particular component, and often require specific conditions to be exploitable. Another reason for this assumption is that we are not aware of a database that would provide information on the relations in exploitability between vulnerabilities,[2] which would be required to test to what extent the exploitability of different vulnerabilities is independent or not.

## A.3 Sensitivity to diversification cost $c_{as}$

We ran tests with different levels of diversification costs. Tables A.5a and A.5b show the results of the optimisation for different levels of the ratio $c_{as}/c_s$ of per-unit costs for the alternative servers used for redundancy with diversification, in addition to the regular ones. The two tables differ in the SLA disruption-time limit $\beta$, i.e. how high a share of disruption is accepted before the SLA is breached and customers are compensated. Table A.5a shows the looser SLA disruption-time limit of $\beta = 0.01$ (disruption tolerated for up to 1% of the time in a month), and A.5b is with the tighter limit $\beta = 0.001$ (0.1% of time). The tables are structured as follows. The cost ratio $c_{as}/c_s$ is shown in the first column, and the second column shows the optimal server allocations for each cost ratio. In the server allocation tuples, the first two elements of the tuple relate to the number of application servers

[2]MITRE ATT&CK® [SAM+18] provides the closest approximation to such information which we know, as the joint occurrence of specific attack techniques in known attacker groups' arsenal could be analysed. However, this is based on known high-profile attacker groups, and on their capabilities that they have been known to exhibit in previous attacks, which mean the data would be unlikely to represent attacker groups more generally, or even the latest level of capabilities of the known attacker groups.

(in DG node $S_A$), with regular application servers in the first and alternative application servers (for diversity) in the second element, while the third and fourth elements are the numbers of database servers (in node $S_{DB}$) and databases (in $DB$), respectively. The third column of the table includes the expected costs under our attack scenarios, the fourth and fifth columns show cost differences relative to N.D. arising from penalty and allocation costs, respectively. Finally, the sixth column shows the percentage difference in the expected cost relative to the optimum without diversification (N.D.). The optimum without diversification is included below the results for the diversified optimisation, specified by the label "N.D." in the first column of the table. The non-diversified optimum does not vary with the cost of the alternative servers, so one comparison point covers all cost-ratios shown.

Table A.5: Sensitivity to diversification cost levels

| Cost ratio ($c_{as}/c_s$) | **Optimum allocation** | **Exp. cost** | **Cost difference** | | **% diff.** |
|---|---|---|---|---|---|
| | | | penalty | alloc. | |
| 1 | $[1,1,2,1]$ | 925 | 0 | 0 | 0 |
| 1.05 | $[1,1,2,1]$ | 934 | 0 | 9 | +1.0 |
| 1.1 | $[1,1,2,1]$ | 944 | 0 | 19 | +2.1 |
| 1.2 | $[1,1,2,1]$ | 962 | 0 | 37 | +4.0 |
| 1.5 | $[1,1,2,1]$ | 1018 | 0 | 93 | +10.1 |
| 2.0 | $[1,1,2,1]$ | 1110 | 0 | 185 | +20.0 |
| N.D. | $[2,0,2,1]$ | 925 | - | - | - |

Notes: $p_d$ = 0.3, $t_r$ = 3, Case 1 defence, $\beta$ = 0.01

(a) $\beta = 0.01$

| Cost ratio ($c_{as}/c_s$) | **Optimum allocation** | **Exp. cost** | **Cost difference** | | **% diff.** |
|---|---|---|---|---|---|
| | | | penalty | alloc. | |
| 1.0 | $[2,2,3,1]$ | 3776 | -3041 | 370 | -41.4 |
| 1.05 | $[2,2,3,1]$ | 3795 | -3041 | 389 | -41.1 |
| 1.1 | $[2,2,3,1]$ | 3813 | -3041 | 407 | -40.9 |
| 1.2 | $[2,2,3,1]$ | 3850 | -3041 | 444 | -40.3 |
| 1.5 | $[2,2,3,1]$ | 3961 | -3041 | 555 | -38.6 |
| 2.0 | $[2,2,3,1]$ | 4146 | -3041 | 740 | -35.7 |
| 5.0 | $[2,2,3,1]$ | 5256 | -3041 | 1850 | -18.5 |
| 9.0 | $[2,1,3,1]$ | 6592 | -1520 | 1665 | +2.2 |
| 10.0 | $[2,1,3,1]$ | 6777 | -1520 | 1850 | +5.1 |
| N.D. | $[2,0,3,1]$ | 6447 | - | - | - |

Notes: $p_d$ = 0.3, $t_r$ = 3, Case 1 defence, $\beta$ = 0.001

(b) $\beta = 0.001$

In Table A.5a, there is no difference in the optimal allocation chosen in the diversified cases, only the cost for the allocation. The non-diversified allocation has the same amount of servers, but as the servers for the application service are all of the cheaper type, the diversified cases are more expensive. There is no breach of the SLA conditions, so no penalty is applied.

In Table A.5b, we again see no differences to the optimal allocation choice due to diversification cost, until the cost of the alternative server type reaches 9 times that of the regular type. Before this level, the cost change is not enough to lead to a different optimal choice. However, the relative benefit of diversification changes. The change in the cost could be enough to impact the choice of whether or not to diversify the system, given the maintenance costs of the redundant servers need to be balanced against the benefits during an attack.

## A.4 Sensitivity to attack scenario weights

In the baseline situation, we have set the weights of the three different attack scenarios (their relative probabilities of occurrence) as equal, $1/3$ each, as explained in Section 8.5.4. In Table A.6 we show the impact of varying them, showing results for attack cost minimisation using the baseline parameter values, varying the weights assigned to each of the three scenarios.

Table A.6: Sensitivity to attack scenario weights, attack cost minimisation

| **Scenario weights** | | | **Alloc.** | **Optimum** | **Exp.** | **Cost** | **TUL** |
|---|---|---|---|---|---|---|---|
| $p_{S1}$ | $p_{S2}$ | $p_{S3}$ | **type** | **allocation** | **cost** | **diff. %** | **(mths)** |
| $1/3$ | $1/3$ | $1/3$ | Div. | $[2,2,3,1]$ | **3795** | -41.1 | **5.5** |
| | | | N.D. | $[2,0,3,1]$ | 6447 | - | 2.6 |
| | | | Ref | $[2,0,2,1]$ | 6931 | +7.5 | - |
| 0 | 0.25 | 0.75 | Div. | $[2,2,3,1]$ | **3990** | -15.7 | 2.4 |
| | | | N.D. | $[2,0,3,1]$ | 4733 | - | **3.5** |
| | | | Ref | $[2,0,2,1]$ | 5384 | +13.8 | - |
| 0 | 0.5 | 0.5 | Div. | $[2,2,3,1]$ | **4200** | -8.0 | 2.1 |
| | | | N.D. | $[2,0,3,1]$ | 4566 | - | **4.4** |
| | | | Ref | $[2,0,2,1]$ | 5384 | +17.9 | - |
| 0 | 0.75 | 0.25 | Div. | $[2,1,3,1]$ | 4404 | +0.1 | 2.6 |
| | | | N.D. | $[2,0,3,1]$ | **4399** | - | **5.3** |
| | | | Ref | $[2,0,2,1]$ | 5384 | +22.4 | - |
| 0.25 | 0 | 0.75 | Div. | $[2,2,3,1]$ | **3581** | -42.5 | **5.2** |
| | | | N.D. | $[2,0,3,1]$ | 6228 | - | 1.7 |
| | | | Ref | $[2,0,2,1]$ | 6544 | +5.1 | - |
| 0.25 | 0.25 | 0.5 | Div. | $[2,2,3,1]$ | **3791** | -37.4 | **4.8** |
| | | | N.D. | $[2,0,3,1]$ | 6060 | - | 2.6 |
| | | | Ref | $[2,0,2,1]$ | 6544 | +8.0 | - |
| 0.25 | 0.5 | 0.25 | Div. | $[2,2,3,1]$ | **4001** | -32.1 | **4.4** |
| | | | N.D. | $[2,0,3,1]$ | 5893 | - | 3.5 |
| | | | Ref | $[2,0,2,1]$ | 6544 | +11.0 | - |
| 0.25 | 0.75 | 0 | Div. | $[2,2,3,1]$ | **4211** | -26.5 | 4.1 |
| | | | N.D. | $[2,0,3,1]$ | 5726 | - | **4.4** |
| | | | Ref | $[2,0,2,1]$ | 6544 | +14.3 | - |
| 0.5 | 0 | 0.5 | Div. | $[2,2,3,1]$ | **3383** | -55.2 | **7.5** |
| | | | N.D. | $[2,0,3,1]$ | 7555 | - | 0.8 |
| | | | Ref | $[2,0,2,1]$ | 7705 | +2.0 | - |
| 0.5 | 0.25 | 0.25 | Div. | $[2,2,3,1]$ | **3592** | -51.4 | **7.2** |
| | | | N.D. | $[2,0,3,1]$ | 7388 | - | 1.7 |
| | | | Ref | $[2,0,2,1]$ | 7705 | +4.3 | - |
| 0.5 | 0.5 | 0 | Div. | $[2,2,3,1]$ | **3802** | -47.3 | **6.8** |
| | | | N.D. | $[2,0,3,1]$ | 7220 | - | 2.6 |
| | | | Ref | $[2,0,2,1]$ | 7705 | +6.7 | - |
| 0.75 | 0 | 0.25 | Div. | $[2,2,3,1]$ | **3184** | -64.1 | **9.9** |
| | | | N.D. | $[2,0,2,1]$ | 8865 | - | N/A |
| | | | Ref | $[2,0,2,1]$ | 8865 | 0 | - |
| 0.75 | 0.25 | 0 | Div. | $[2,2,3,1]$ | **3394** | -61.1 | **9.5** |
| | | | N.D. | $[2,0,3,1]$ | 8715 | - | 0.8 |
| | | | Ref | $[2,0,2,1]$ | 8865 | +1.7 | - |
| 1 | 0 | 0 | Div. | $[2,2,2,1]$ | **2800** | -72.1 | **18.6** |
| | | | N.D. | $[2,0,2,1]$ | 10025 | - | - |
| | | | Ref | $[2,0,2,1]$ | 10025 | 0 | - |
| 0 | 1 | 0 | Div. | $[1,1,3,1]$ | 4241 | +0.2 | 5.9 |
| | | | N.D. | $[2,0,2,1]$ | **4231** | - | **6.2** |
| | | | Ref | $[2,0,2,1]$ | 5384 | +27.2 | - |
| 0 | 0 | 1 | Div. | $[2,2,3,1]$ | **3780** | -22.9 | **2.8** |
| | | | N.D. | $[2,0,3,1]$ | 4900 | - | 2.6 |
| | | | Ref | $[2,0,2,1]$ | 5384 | +9.9 | - |

Notes: $c_{as}/c_s$=1.05, $P(A|R) = 0.5$, $\beta$ = 0.001, $p_d$ = 0.3, $t_r$ = 3, Case 1 defence

The main observation relating to the results in Table A.6 is that *the optimal allocation is rather stable in face of changes to the scenario weights.* In 14/16 of the cases tested for the table, the optimal allocation was found to be $[2,2,3,1]$ (using attack cost minimisation). In the remaining two cases, where a high weight was given to Scenario 2 ($p_{S2} = 1$, and [$p_{S2} = 0.75$, $p_{S3} = 0.25$]), the optimum was the non-diversified allocation $[2,0,3,1]$.

## A.5 TUL optimisation result tables

Tables A.7a and A.7b show results for TUL optimisation; the former assumes attack penalty multiplier 1x ($50 per customer), while the latter uses penalty multiplier 3x ($150 per customer).

Note that, due to the condition in the TUL maximisation problem (equation (8.4) in Section 8.5.8.1) to consider only allocations with more servers than in the reference allocation, in cases where adding redundancy is not beneficial the optimisation may fail to find a solution that is reasonable, i.e. the chosen allocation has a higher cost than the reference. In such cases the reference allocation should be chosen instead. This situation occurs in Tables A.7a and A.7b for the non-diversified cases with $p_d = 0$ and $p_d = 0.9$, and for $p_d = 0.7$ in Table A.7a.

Table A.7: Sensitivity to detection probability $p_d$, TUL optimisation, Case 1 defence

| **Detection prob.** ($p_d$) | **Alloc. type** | **Optimum allocation** | **Exp. cost** | **Cost difference** | | **% diff.** |
|---|---|---|---|---|---|---|
| | | | | penalty | alloc. | |
| $\beta$ **= 0.001,** $t_r$ **= 3** | | | | | | |
| 0 | Div. | $[2,2,2,1]$ | 11425 | -2889 | 389 | -18.0 |
| | N.D. | $[2,0,2,1]$ | 13925 | - | - | - |
| | Ref | $[2,0,2,1]$ | 13925 | 0 | 0 | 0 |
| 0.1 | Div. | $[2,1,2,1]$ | 9721 | -1142 | 9 | -10.4 |
| | N.D. | $[2,0,3,1]$ | 10854 | - | - | - |
| | Ref | $[2,0,2,1]$ | 11143 | 474 | -185 | +2.7 |
| 0.3 | Div. | $[2,1,2,1]$ | 5630 | -826 | 9 | -12.7 |
| | N.D. | $[2,0,3,1]$ | 6447 | - | - | - |
| | Ref | $[2,0,2,1]$ | 6931 | 669 | -185 | +7.5 |
| 0.5 | Div. | $[2,1,2,1]$ | 3263 | -700 | 9 | -17.5 |
| | N.D. | $[2,0,3,1]$ | 3954 | - | - | - |
| | Ref | $[2,0,2,1]$ | 4175 | 406 | -185 | +5.6 |
| 0.7 | Div. | $[2,1,2,1]$ | 1995 | -658 | 194 | -18.9 |
| | N.D. | $[2,0,2,1]$ | 2459 | - | - | - |
| | Ref | $[2,0,2,1]$ | 2459 | 0 | 0 | 0 |
| 0.9 | Div. | $[2,2,2,1]$ | 1322 | -434 | 389 | -3.3 |
| | N.D. | $[2,0,2,1]$ | 1367 | - | - | - |
| | Ref | $[2,0,2,1]$ | 1367 | 0 | 0 | 0 |

Notes: $c_{as}/c_s$=1.05, $P(A|R) = 0.5$, cc=$50

(a) 1x penalty multiplier ($50 per customer)

| **Detection prob.** ($p_d$) | **Alloc. type** | **Optimum allocation** | **Exp. cost** | **Cost difference** | | **% diff.** |
|---|---|---|---|---|---|---|
| | | | | penalty | alloc. | |
| $\beta$ **= 0.001,** $t_r$ **= 3** | | | | | | |
| 0 | Div. | $[2,2,2,1]$ | 31647 | -8667 | 389 | -20.7 |
| | N.D. | $[2,0,2,1]$ | 39925 | - | - | - |
| | Ref | $[2,0,2,1]$ | 39925 | 0 | 0 | 0 |
| 0.1 | Div. | $[2,2,2,1]$ | 22270 | -8276 | 204 | -26.6 |
| | N.D. | $[2,0,3,1]$ | 30342 | - | - | - |
| | Ref | $[2,0,2,1]$ | 31579 | 1422 | -185 | +4.1 |
| 0.3 | Div. | $[2,2,2,1]$ | 10362 | -6963 | 204 | -39.5 |
| | N.D. | $[2,0,3,1]$ | 17121 | - | - | - |
| | Ref | $[2,0,2,1]$ | 18943 | 2007 | -185 | +10.6 |
| 0.5 | Div. | $[2,1,2,1]$ | 7552 | -2098 | 9 | -21.7 |
| | N.D. | $[2,0,3,1]$ | 9641 | - | - | - |
| | Ref | $[2,0,2,1]$ | 10675 | 1219 | -185 | +10.7 |
| 0.7 | Div. | $[2,1,2,1]$ | 3746 | -1607 | 9 | -29.9 |
| | N.D. | $[2,0,3,1]$ | 5344 | - | - | - |
| | Ref | $[2,0,2,1]$ | 5527 | 368 | -185 | +3.4 |
| 0.9 | Div. | $[2,2,2,1]$ | 1337 | -1286 | 204 | -44.7 |
| | N.D. | $[2,0,2,1]$ | 1082 | - | - | - |
| | Ref | $[2,0,2,1]$ | 1082 | 0 | 0 | 0 |

Notes: $c_{as}/c_s$=1.05, $P(A|R) = 0.5$, cc=$150

(b) 3x penalty multiplier ($150 per customer)

# Bibliography


[AAM+10] Benjamin A Allan, Robert C Armstrong, Jackson R Mayo, Lyndon G Pierson, Mark D Torgerson, and Andrea Mae Walker. The theory of diversity and redundancy in information system security: LDRD final report. No. SAND2010-7055, Sandia National Laboratories, Albuquerque, NM, and Livermore, CA, `https://doi.org/10.2172/992781`, 2010.

[ABB+16] Elvio Gilberto Amparore, Gianfranco Balbo, Marco Beccuti, Susanna Donatelli, and Giuliana Franceschinis. 30 Years of GreatSPN. *Principles of Performance and Reliability Modeling and Evaluation*, pages 227–254, 2016.

[AC77] Algirdas Avižienis and Liming Chen. On the implementation of N-version programming for software fault tolerance during execution. In *Proc. 1st IEEE-CS Int. Comput. Software Appl. Conf., COMPSAC 77*, pages 149–155, 1977.

[AJ17] Massimiliano Albanese and Sushil Jajodia. A Graphical Model to Assess the Impact of Multi-Step Attacks. *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, pages 1–15, 2017.

[AJPS11] Massimiliano Albanese, Sushil Jajodia, Andrea Pugliese, and V. S. Subrahmanian. Scalable Analysis of Attack Scenarios. *Atluri V., Diaz C. (eds) Computer Security – ESORICS 2011. Lecture Notes in Computer Science*, 6879:416–433, 2011.

[Ale16] Mohammed J.F. Alenazi. Graph resilience improvement of backbone networks via node additions. *8th Int. Workshop on Resilient Networks Design and Modeling, RNDM 2016*, pages 231–237, 2016.

[AMS+16] Andy Applebaum, Doug Miller, Blake Strom, Chris Korban, and Ross Wolf. Intelligent, automated red team emulation. *ACM International Conference Proceeding Series*, 5-9-December-2016:363–373, 2016.

[AMS+17] Andy Applebaum, Doug Miller, Blake Strom, Henry Foster, and Cody Thomas. Analysis of automated adversary emulation techniques. *Simulation Series*, 49(9):169–180, 2017.

[AS15a] Mohammed J.F. Alenazi and James P.G. Sterbenz. Comprehensive comparison and accuracy of graph metrics in predicting network resilience. *11th Int. Conference on the Design of Reliable Communication Networks, DRCN 2015*, pages 157–164, 2015.

[AS15b] Mohammed J.F. Alenazi and James P.G. Sterbenz. Evaluation and comparison of several graph robustness metrics to improve network resilience. *7th Int. Workshop on Reliable Networks Design and Modeling, RNDM 2015*, pages 7–13, 2015.

[AS15c] Mohammed J.F. Alenazi and James P.G. Sterbenz. Evaluation and Improvement of Network Resilience against Attacks using Graph Spectral Metrics. *2015 Resilience Week (RWS)*, pages 206–211, 2015.

[ASP12] Mohammed A AlZain, Ben Soh, and Eric Pardede. A new approach using redundancy technique to improve security in cloud computing. In *Int. Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*, pages 230–235. IEEE, 2012.

[BB18] Hiba Baroud and Kash Barker. A Bayesian kernel approach to modeling resilience-based network component importance. *Reliability Engineering and System Safety*, 170:10–19, 2018.

[BCE+03] Michel Bruneau, Stephanie E Chang, Ronald T Eguchi, George C Lee, Thomas D O'Rourke, Andrei M Reinhorn, Masanobu Shinozuka, Kathleen Tierney, William A Wallace, and Detlof Von Winterfeldt. A framework to quantitatively assess and enhance the seismic resilience of communities. *Earthquake spectra*, 19(4):733–752, 2003.

[BDB11] Ran Bhamra, Samir Dani, and Kevin Burnard. Resilience: the concept, a literature review and future directions. *International Journal of Production Research*, 49(18):5375–5393, 2011.

[BG11] Deborah Bodeau and Richard Graubart. Cyber Resiliency Engineering Framework. Technical Report MTR110237, The MITRE Corporation, Bedford, MA, 2011. [Online] Available: `https://apps.dtic.mil/sti/citations/AD1108457`, Accessed: 2022-10-09.

[BG13] Deborah Bodeau and Richard Graubart. Intended Effects of Cyber Resiliency Techniques on Adversary Activities. *IEEE Int. Conference on Technologies for Homeland Security, HST 2013*, pages 7–11, 2013.

[BGDMT06] Gunter Bolch, Stefan Greiner, Hermann De Meer, and Kishor S Trivedi. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[BJY+21] Hamed Badihi, Saeedreza Jadidi, Ziquan Yu, Youmin Zhang, and Ningyun Lu. Diagnosis and Mitigation of Smart Cyber-Attacks on an Offshore Wind Farm Network Operator. pages 479–484, 2021.

[BK02] Falko Bause and Pieter S Kritzinger. *Stochastic Petri Nets: An Introduction to the Theory*. Vieweg Verlag, Braunschweig/Wiesbaden (Germany), 2002. Available: `https://ls4-www.cs.tu-dortmund.de/download/typo3/de/home/bause/spnbook2/index.html`, Accessed: 2022-10-09.

[BLDC19] K Bissell, RM LaSalle, and P Dal Cin. The Cost of Cybercrime—Ninth Annual Cost of Cybercrime Study. Technical report, Accenture, 2019.

[BLDC20] K Bissell, RM LaSalle, and P Dal Cin. Innovate for Cyber Resilience—Third Annual State of Cyber Resilience. Technical report, Accenture, 2020.

[BST+20] Katharina L Best, Jon Schmid, Shane Tierney, Jalal Awan, Nahom M Beyene, Maynard A Holliday, Raza Khan, and Karen Lee. How to Analyze the Cyber Threat from

Drones. Technical Report RR-2972-RC, RAND Corporation, Santa Monica, CA, United States, 2020. URL: `www.rand.org/t/RR2972`, Accessed: 2022-10-09.

[BWJS18] Daniel Borbor, Lingyu Wang, Sushil Jajodia, and Anoop Singhal. Surviving unpatchable vulnerabilities through heterogeneous network hardening options. *Journal of Computer Security*, 26(6):761–789, 2018.

[BWJS19] Daniel Borbor, Lingyu Wang, Sushil Jajodia, and Anoop Singhal. Optimizing the network diversity to improve the resilience of networks against unknown attacks. *Computer Communications*, 145:96–112, 2019.

[CAC+10] Richard A Caralli, Julia H Allen, Pamela D Curtis, David W White, and Lisa R Young. CERT Resilience Management Model, Version 1.0. (CMU/SEI-2010-TR-012), 2010. URL: `https://apps.dtic.mil/sti/pdfs/ADA522534.pdf`, Accessed: 2022-10-09.

[CAH21] Juan Francisco Carias, Saioa Arrizabalaga, and Josune Hernantes. Cyber Resilience Strategic Planning and Self-assessment Tool for Operationalization in SMEs. In Y Murayama, D Velev, and P Zlateva, editors, *Information Technology in Disaster Risk Reduction*, pages 259–273. Springer, 2021.

[CALH21] Juan Francisco Carias, Saioa Arrizabalaga, Leire Labaka, and Josune Hernantes. Cyber Resilience Self-Assessment Tool (CR-SAT) for SMEs. *IEEE Access*, 9:80741–80762, 2021.

[Cas20] Giuliano Casale. Integrated Performance Evaluation of Extended Queueing Network Models with LINE. In *Winter Simulation Conference (WSC) 2020*. ACM, 2020.

[CBL+20] Juan Francisco Carías, Marcos R.S. Borges, Leire Labaka, Saioa Arrizabalaga, and Josune Hernantes. Systematic Approach to Cyber Resilience Operationalization in SMEs. *IEEE Access*, 8:174200–174221, 2020.

[CBL+21] Juan Francisco Carias, Marcos R.S. Borges, Leire Labaka, Saioa Arrizabalaga, and Josune Hernantes. The Order of the Factors DOES Alter the Product: Cyber Re-

silience Policies' Implementation Order. In *Advances in Intelligent Systems and Computing AISC*, volume 1267, pages 306–315. Springer, 2021.

[CCD+21] Silvia Colabianchi, Francesco Costantino, Giulio Di Gravio, Fabio Nonino, and Riccardo Patriarca. Discussing resilience in the context of cyber physical systems. *Computers & Industrial Engineering*, 160(July):107534, 2021.

[CCR+19] Pin Yu Chen, Sutanay Choudhury, Luke Rodriguez, Alfred O. Hero, and Indrajit Ray. Toward cyber-resiliency metrics for action recommendations against lateral movement attacks. *Advances in Information Security*, 75:71–92, 2019.

[CDFH93] Giovanni Chiola, Claude Dutheillet, Giuliana Franceschinis, and Serge Haddad. Stochastic Well-Formed Colored Nets and Symmetric Modeling Applications. *IEEE Transactions on Computers*, 42(11):1343–1360, 1993.

[CH19] David H Collins and Aparna V Huzurbazar. Petri net models of adversarial scenarios in safety and security. *Military Operations Research*, 24(3):27–48, 2019.

[CKN+13] Binbin Chen, Zbigniew Kalbarczyk, David M Nicol, William H Sanders, Rui Tan, William G Temple, Nils Ole Tippenhauer, An Hoa Vu, and David KY Yau. Go with the flow: Toward workflow-oriented security assessment. In *2013 New Security Paradigms Workshop*, pages 65–76, 2013.

[CPG18] Michail Chronopoulos, Emmanouil Panaousis, and Jens Grossklags. An options approach to cybersecurity investment. *IEEE Access*, 6:12175–12186, 2018.

[CRB10] Gian Paolo Cimellaro, Andrei M Reinhorn, and Michel Bruneau. Framework for analytical quantification of disaster resilience. *Engineering Structures*, 32(11):3639–3649, 2010.

[CRC+15] Sutanay Choudhury, Luke Rodriguez, Darren Curtis, Kiri Oler, Peter Nordquist, Pin-Yu Chen, and Indrajit Ray. Action Recommendation for Cyber Resilience. *Workshop on Automated Decision Making for Active Cyber Defense, SafeConfig '15*, pages 3–8, 2015.

[CRDP17] Andrew Chaves, Mason Rice, Stephen Dunlap, and John Pecarina. Improving the cyber resilience of industrial control systems. *International Journal of Critical Infrastructure Protection*, 17:30–48, 2017.

[CTTV16] Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Efficient syntax-driven lumping of differential equations. *Lecture Notes in Computer Science*, 9636:93–111, 2016.

[CYS+18] Chen Cao, Lun Pin Yuan, Anoop Singhal, Peng Liu, Xiaoyan Sun, and Sencun Zhu. Assessing attack impact on business processes by interconnecting attack graphs and entity dependency graphs. *Lecture Notes in Computer Science*, 10980:330–348, 2018.

[DCC+02] Daniel D. Deavours, Graham Clark, Tod Courtney, David Daly, Salem Derisavi, Jay M. Doyle, William H. Sanders, and Patrick G. Webster. The Möbius framework and its implementation. *IEEE Trans. Softw. Eng.*, 28(10):956–969, 2002.

[DJI20] DJI. DJI Zenmuse H20T specs, 2020. [Online] `https://www.dji.com/uk/zenmuse-h20-series/specs`, Accessed: 2021-06-03.

[DK15] Elena Doynikova and Igor Kotenko. Countermeasure Selection Based on the Attack and Service Dependency Graphs for Security Incident Management. *LNCS 9572, CRiSIS 2015*, 9572:107–124, 2015.

[Dun18] Nick Dunn. The economics of defensive security. Technical report, NCC Group, 2018.

[DW06] Ole Martin Dahl and Stephen D. Wolthusen. Modeling and Execution of Complex Attack Scenarios Using Interval Timed Colored Petri Nets. In *4th IEEE Int. Workshop on Information Assurance, IWIA 2006*, pages 157–168, 2006.

[Eur00] European Union. European Forest Fire Information System (EFFIS), 2000. `https://effis.jrc.ec.europa.eu/applications/data-and-services`, Accessed: 2021-04-27.

[Eur19] European Commission. Commission Delegated Regulation (EU) 2019/945 of 12 March 2019 on unmanned aircraft systems and on third-country operators of unmanned aircraft systems, C/2019/1821, 2019. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32019R0945, Accessed: 2021-06-03.

[FFM+19] Kelly Finnerty, Sarah Fullick, Helen Motha, Jayesh Navin Shah, Mark Button, and Victoria Wang. *Cyber Security Breaches Survey 2019*. Department for Digital, Culture, Media & Sport, April 2019.

[FGS16] Angelo Furfaro, Teresa Gallo, and Domenico Saccà. Modeling Cyber Systemic Risk for the Business Continuity Plan of a Bank. In *CD-ARES 2016, LNCS 9817*, pages 158–174. 2016.

[FIR16] FIRST. Common Vulnerability Scoring System. [Online] https://www.first.org/cvss, Accessed: 2016-08-17, 2016.

[FKL+13] Michael D. Ford, Ken Keefe, Elizabeth Lemay, William H. Sanders, and Carol Muehrcke. Implementing the ADVISE security modeling formalism in Möbius. *Int. Conference on Dependable Systems and Networks*, 2013.

[Fli20] Flir. Flir VUE TZ20 specs, 2020. [Online] https://www.flir.co.uk/products/vue-tz20/, Accessed: 2021-06-03.

[FPM+16] Andrew Fielder, Emmanouil Panaousis, Pasquale Malacaria, Chris Hankin, and Fabrizio Smeraldi. Decision support approaches for cyber security investment. *Decision Support Systems*, 86:13–23, 2016.

[FPZ16] Yi Ping Fang, Nicola Pedroni, and Enrico Zio. Resilience-Based Component Importance Measures for Critical Infrastructure Network Systems. *IEEE Transactions on Reliability*, 65(2):502–512, 2016.

[FW08] Marcel Frigault and Lingyu Wang. Measuring network security using Bayesian network-based attack graphs. In *32nd Annual IEEE Int. Computer Software and Applications Conference*, pages 698–703. IEEE, 2008.

[GBB16] Jianxi Gao, Baruch Barzel, and Albert László Barabási. Universal resilience patterns in complex networks. *Nature*, 530(7590):307–312, 2016.

[GGQ+14] Yongqiang Gao, Haibing Guan, Zhengwei Qi, Tao Song, Fei Huan, and Liang Liu. Service level agreement based energy-efficient resource management in cloud data centers. *Computers & Electrical Engineering*, 40(5):1621–1633, 2014.

[GKK17] Mengmeng Ge, Huy Kang Kim, and Dong Seong Kim. Evaluating security and availability of multiple redundancy designs when applying security patches. In *47th Annual IEEE/IFIP Int. Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 53–60. IEEE, 2017.

[GL02] Lawrence A Gordon and Martin P Loeb. The economics of information security investment. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):438–457, 2002.

[GLL03] Lawrence A Gordon, Martin P Loeb, and William Lucyshyn. Information security expenditures and real options: A wait-and-see approach. *Computer Security Journal*, 19(2), 2003.

[GMG+16] Alexander A. Ganin, Emanuele Massaro, Alexander Gutfraind, Nicolas Steen, Jeffrey M. Keisler, Alexander Kott, Rami Mangoubi, and Igor Linkov. Operational resilience: Concepts, design and analysis. *Scientific Reports*, 6:1–12, 2016.

[GMH20] Ramin Giahi, Cameron A. MacKenzie, and Chao Hu. Design optimization for resilience for risk-averse firms. *Computers and Industrial Engineering*, 139:106122, 2020.

[GMP11] Harriet Goldman, Rosalie McQuaid, and Jeffrey Picciotto. Cyber Resilience for Mission Assurance. *IEEE Int. Conference on Technologies for Homeland Security, HST 2011*, pages 236–241, 2011.

[Gol10] Harriet G Goldman. Building secure, resilient architectures for cyber mission assurance. *Secure and Resilient Cyber Architectures Conference SRCA 2010*, pages 1–18, 2010. URL: `https://apps.dtic.mil/sti/pdfs/AD1108588.pdf`.

[HBRM16] Seyedmohsen Hosseini, Kash Barker, and Jose E Ramirez-Marquez. A review of definitions and measures of system resilience. *Reliability Engineering & System Safety*, 145:47–61, 2016.

[HCA11] Eric Hutchins, Michael Cloppert, and Rohan Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *6th Int. Conf. on Information Warfare and Security, ICIW 2011*, pages 113–125, 2011.

[HGG+14] Rui Han, Moustafa M Ghanem, Li Guo, Yike Guo, and Michelle Osmond. Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. *Future Generation Computer Systems*, 32:82–98, 2014.

[HSK19] Md Ariful Haque, Sachin Shetty, and Bheshaj Krishnappa. Modeling Cyber Resilience for Energy Delivery Systems Using Critical System Functionality. *Resilience Week, RWS 2019*, pages 33–41, 2019.

[HSKG20] Md Ariful Haque, Sachin Shetty, Charles A. Kamhoua, and Kimberly Gold. Integrating Mission-Centric Impact Assessment to Operational Resiliency in Cyber-Physical Systems. *2020 IEEE Global Communications Conference, GLOBECOM*, 2020.

[HWIL09] Ye Hu, Johnny Wong, Gabriel Iszlai, and Marin Litoiu. Resource provisioning for cloud computing. In *Conference of the Center for Advanced Studies on Collaborative Research*, pages 101–111. IBM Corp., 2009.

[HWWC17] Hongchao Hu, Jiangxing Wu, Zhenpeng Wang, and Guozhen Cheng. Mimic defense: a designed-in cybersecurity defense framework. *IET Information Security*, 12(3):226–237, 2017.

[ISO12] ISO 22301:2012 Societal security — Business continuity management systems — Requirements. Standard, International Organization for Standardization, Geneva, CH, May 2012.

[ISO19] ISO 22301:2019 Security and resilience — Business continuity management systems — Requirements. Standard, International Organization for Standardization, Geneva, CH, October 2019.

[Jak11] Gabriel Jakobson. Mission cyber security situation assessment using impact dependency graphs. *14th Int. Conference on Information Fusion*, pages 1–8, 2011.

[JF18] Mark Jackson and John S Fitzgerald. Towards Resilience-Explicit Modelling and Co-simulation of Cyber-Physical Systems. *SEFM 2017 Workshops, LNCS 10729*, 10729:361–376, 2018.

[JSDA12] Ahmad Y. Javaid, Weiqing Sun, Vijay K. Devabhaktuni, and Mansoor Alam. Cyber security threat analysis and modeling of an unmanned aerial vehicle system. *IEEE Int. Conference on Technologies for Homeland Security, HST 2012*, pages 585–590, 2012.

[KAsR15] Yasir Imtiaz Khan, Ehab Al-shaer, and Usman Rauf. Cyber Resilience-by-Construction. *Workshop on Automated Decision Making for Active Cyber Defense - SafeConfig '15*, pages 9–14, 2015.

[KB03] Samuel Kounev and Alejandro P Buchmann. Performance modeling and evaluation of large-scale J2EE applications. In *Int. CMG Conference*, volume 11, 2003.

[KBP+09] Jonathan G Koomey, Christian Belady, Michael Patterson, Anthony Santos, and Klaus-Dieter Lange. Assessing trends over time in performance, costs, and energy use for servers. Technical report, Lawrence Berkeley National Laboratory, Stanford University, Microsoft Corporation, and Intel Corporation, 2009.

[KC13] Igor Kotenko and Andrey Chechulin. A Cyber Attack Modeling and Impact Assessment framework. *Int. Conference on Cyber Conflict, CYCON*, 2013.

[KCBCD10] Nizar Kheir, Nora Cuppens-Boulahia, Frédéric Cuppens, and Hervé Debar. A service dependency model for cost-sensitive intrusion response. *Lecture Notes in Computer Science*, 6345:626–642, 2010.

[KD14] Igor Kotenko and Elena Doynikova. Evaluation of computer network security based on attack graphs and security event processing. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 5(3):14–29, 2014.

[KD15] Igor Kotenko and Elena Doynikova. Countermeasure selection in SIEM systems based on the integrated complex of security metrics. *23rd Euromicro Int. Conf. Parallel, Distributed, and Network-Based Processing, PDP 2015*, pages 567–574, 2015.

[KD16] Igor Kotenko and Elena Doynikova. Dynamical Calculation of Security Metrics for Countermeasure Selection in Computer Networks. *24th Euromicro Int. Conf. Parallel, Distributed, and Network-Based Processing, PDP 2016*, pages 558–565, 2016.

[KGTL21] Alexander Kott, Maureen S. Golan, Benjamin D. Trump, and Igor Linkov. Cyber Resilience: by Design or by Intervention. *Computer*, pages 112–117, Aug 2021.

[KL12] Yousri Kouki and Thomas Ledoux. SLA-driven capacity planning for cloud applications. In *4th IEEE Int. Conference on Cloud Computing Technology and Science Proceedings*, pages 135–140. IEEE, 2012.

[KMH+16] MHR Khouzani, Pasquale Malacaria, Chris Hankin, Andrew Fielder, and Fabrizio Smeraldi. Efficient numerical frameworks for multi-objective cyber security planning. In *European Symposium on Research in Computer Security*, pages 179–197, 2016.

[Lam16] Wing Man Wynne Lam. Attack-prevention and damage-control investments in cybersecurity. *Information Economics and Policy*, 37:42–51, 2016.

[LB08] David John Leversage and Eric James Byres. Estimating a System's Mean Time-to-Compromise. *IEEE Security & Privacy*, 6(1):52–60, 2008.

[LDB20] Harjinder Singh Lallie, Kurt Debattista, and Jay Bal. A review of attack graph and attack tree visual syntax in cyber security. *Computer Science Review*, 35:100219, 2020.

[LFH20] Tingting Li, Cheng Feng, and Chris Hankin. Scalable Approach to Enhancing ICS Resilience by Network Diversity. In *50th Annual IEEE/IFIP Int. Conference on Dependable Systems and Networks (DSN)*, pages 398–410. IEEE, 2020.

[LFK+11] Elizabeth LeMay, Michael D Ford, Ken Keefe, William H Sanders, and Carol Muehrcke. Model-based security metrics using adversary view security evaluation

(ADVISE). In *8th Int. Conference on Quantitative Evaluation of SysTems*, pages 191–200. IEEE, 2011.

[LI05] Richard Paul Lippmann and Kyle William Ingols. An annotated review of past papers on attack graphs. Technical Report PR-IA-1, Massachusetts Inst of Tech Lexington Lincoln Lab, 2005. Available: `https://apps.dtic.mil/sti/citations/ADA431826`, Accessed: 2022-10-09.

[LJZY20] Zhi Li, Hai Jin, Deqing Zou, and Bin Yuan. Exploring New Opportunities to Defeat Low-Rate DDoS Attack in Container-Based Cloud Environment. *IEEE Trans. Parallel Distrib. Syst.*, 31(3):695–706, 2020.

[LK19] Igor Linkov and Alexander Kott. Fundamental concepts of cyber resilience: Introduction and overview. In *Cyber Resilience of Systems and Networks*, pages 1–25. Springer, 2019.

[LKL21] Alexandre Ligo, Alexander Kott, and Igor Linkov. How to Measure Cyber-Resilience of a System with Autonomous Agents: Approaches and Challenges. *IEEE Engineering Management Review*, 2021.

[LLC+18] Beibei Li, Rongxing Lu, Kim-Kwang Raymond Choo, Wei Wang, and Sheng Luo. On Reliability Analysis of Smart Grids under Topology Attacks: A Stochastic Petri Net Approach. *ACM Trans. Cyber-Phys. Syst.*, 3(1), August 2018.

[LM05] Yu Liu and Hong Man. Network vulnerability assessment using Bayesian networks. In *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security*, volume 5812, pages 61–71. SPIE, 2005.

[LSO12] Chaya Losada, M. Paola Scaparra, and Jesse R. O'Hanley. Optimizing system resilience: A facility protection model with recovery time. *European Journal of Operational Research*, 217(3):519–530, 2012.

[LYGH21] Jian Li, De-Fu Yang, Yan-Chao Gao, and Xin Huang. An adaptive sliding-mode resilient control strategy in smart grid under mixed attacks. *IET Control Theory & Applications*, (April):1–16, 2021.

[LZGS84] Edward D Lazowska, John Zahorjan, G Scott Graham, and Kenneth C Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc., 1984.

[MAA+18] D Miller, R Alford, A Applebaum, H Foster, C Little, and B Strom. Automated adversary emulation: A case for planning and acting with unknowns. 2018. Available: `https://apps.dtic.mil/sti/citations/AD1108001`, Accessed: 2022-10-09.

[MBC+95] M. Ajmone Marsan, Gianfranco Balbo, Gianni Conte, Susanna Donatelli, and Giuliana Franceschinis. *Modelling With Generalised Stochastic Petri Nets*. J. Wiley & Sons Ltd., Chichester, 1995.

[MBFB06] Miles A McQueen, Wayne F Boyer, Mark A Flynn, and George A Beitel. Time-to-Compromise Model for Cyber Risk Reduction Estimation. In Dieter Gollmann, Fabio Massacci, and Artsiom Yautsiukhin, editors, *Quality of Protection*, volume 23, pages 49–64. Springer, Boston, MA, 2006.

[Mey80] John F Meyer. On Evaluating the Performability of Degradable Computing Systems. *IEEE Transactions on Computers*, C-29(8):720–731, 1980.

[Mey92] John F. Meyer. Performability: a retrospective and some pointers to the future. *Performance Evaluation*, 14(3):139–156, 1992.

[Mey13] J. F. Meyer. Model-based evaluation of system resilience. In *43rd IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 1–7, 2013.

[MGSBL17] Luis Muñoz-González, Daniele Sgandurra, Martín Barrère, and Emil Lupu. Exact inference techniques for the analysis of Bayesian attack graphs. *IEEE Transactions on Dependable and Secure Computing*, 2017.

[MGSPL17] Luis Muñoz-González, Daniele Sgandurra, Andrea Paudice, and Emil C Lupu. Efficient attack graph analysis through approximate inference. *ACM Transactions on Privacy and Security*, 20(3):30, 2017.

[MIT19] MITRE Corporation. MITRE CALDERA. [Online] Available: `https://caldera.mitre.org/`, Accessed: 2022-03-10, 2019.

[MMPP19] Andrea Marrella, Massimo Mecella, Barbara Pernici, and Pierluigi Plebani. A design-time data-centric maturity model for assessing resilience in multi-party business processes. *Information Systems*, 86:62–78, 2019.

[MSR06] Peter Mell, Karen Scarfone, and Sasha Romanosky. Common Vulnerability Scoring System. *IEEE Security & Privacy*, 4(6):85–89, 2006.

[MT04] B. B. Madan and K. S. Trivedi. Security modeling and quantification of intrusion tolerant systems using attack-response graph. *Journal of High Speed Networks*, 13(4):297–308, 2004.

[MWG95] Andreu Mas-Colell, Michael Dennis Whinston, and Jerry R Green. *Microeconomic Theory*. Oxford University Press, New York, 1995.

[Nat18] National Cyber Security Centre. Cyber Assessment Framework. [Online] Available: `https://www.ncsc.gov.uk/collection/caf`, Accessed: 2022-10-09, 2018.

[NBM+21] B Nassi, R Bitton, R Masuoka, A Shabtai, and Y Elovici. SoK: Security and Privacy in the Age of Commercial Drones. In *Proc. 2021 IEEE Symp. Security and Privacy (SP)*, pages 73–90, 2021.

[NIS] NIST. National vulnerability database. [Online] Available: `https://nvd.nist.gov/`. Accessed: 2016-06-09.

[NKK+17] Laurent L Njilla, Charles A Kamhoua, Kevin A Kwiat, Patrick Hurley, and Niki Pissinou. Cyber Security Resource Allocation: A Markov Decision Process Approach. In *High Assurance Systems Engineering (HASE)*, pages 49–52. IEEE, 2017.

[NPMK17] Pantaleone Nespoli, Dimitrios Papamartzivanos, Félix Gómez Mármol, and Georgios Kambourakis. Optimal countermeasures selection against cyber attacks: A compre-

hensive survey on reaction frameworks. *IEEE Communications Surveys & Tutorials*, 2017.

[NWJS13] William Nzoukou, Lingyu Wang, Sushil Jajodia, and Anoop Singhal. A unified framework for measuring a network's mean time-to-compromise. In *IEEE 32nd Int. Symposium on Reliable Distributed Systems*, pages 215–224. IEEE, 2013.

[NY17] Perri Nejib and Edward Yakabovicz. NATO Resilience by Design: Enhancing Resilience through Cyber Systems Engineering. In *NATO IST-153/RWS-21 Workshop on Cyber Resilience*, pages 1–7, 2017.

[OBM06] Xinming Ou, Wayne F Boyer, and Miles A McQueen. A scalable approach to attack graph generation. In *13th ACM Conference on Computer and Communications Security*, pages 336–345. ACM, 2006.

[Off21] Cabinet Office. *National Cyber Strategy 2022*. HM Government, December 2021.

[OGA05] Xinming Ou, Sudhakar Govindavajhala, and Andrew W Appel. Mulval: A logic-based network security analyzer. In *USENIX security*, 2005.

[Ora] Oracle. Oracle Cloud Service Level Agreement. [Online] Available: `https://cloud.oracle.com/iaas/sla`. Accessed: 2019-09-09.

[OS12] Xinming Ou and Anoop Singhal. *Quantitative security risk assessment of enterprise networks*. Springer, 2012.

[PC17] Juan F Pérez and Giuliano Casale. Line: Evaluating software applications in unreliable environments. *IEEE Trans. Reliab.*, 66(3):837–853, 2017.

[PDR12] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic security risk management using bayesian attack graphs. *Dependable and Secure Computing, IEEE Transactions on*, 9(1):61–74, 2012.

[Pon15] Ponemon Institute. The Cost of Denial-of-Service Attacks. Technical report, Ponemon Institute, March 2015, 2015.

[Pon18] Ponemon Institute. 2018 Cost of a Data Breach Survey: Global Overview. Technical report, IBM, 2018.

[RMEB19] Adam Rose, Noah Miller, Jonathan Eyer, and Joshua Banks. Economic effectiveness of mitigation and resilience. In Alexander Kott and Igor Linkov, editors, *Cyber Resilience of Systems and Networks*, pages 315–351. Springer, 2019.

[SA14] Ali Sedaghatbaf and Mohammad Abdollahi Azgomi. Attack modelling and security evaluation based on stochastic activity networks. *Security and Communication Networks*, 7(4):714–737, Apr 2014.

[SAM+18] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. Mitre ATT&CK: Design and philosophy. Technical report, MITRE Corporation, 2018.

[SBH12] Carlos Sarraute, Olivier Buffet, and Jörg Hoffmann. POMDPs make better hackers: Accounting for uncertainty in penetration testing. *Proc. National Conference on Artificial Intelligence*, 3:1816–1824, 2012.

[SC17] Tristan A Shatto and Egemen K Cetinkaya. Variations in Graph Energy : A Measure for Network Resilience. *Resilient Networks Design and Modeling (RNDM), 2017 9th Int. Workshop on*, pages 1–7, 2017.

[SCML22] Jukka Soikkeli, Giuliano Casale, Luis Muñoz-González, and Emil Lupu. Redundancy Planning for Cost Efficient Resilience to Cyber Attacks. In press, to appear in IEEE Transactions on Dependable and Secure Computing. URL: doi.org/10.1109/TDSC.2022.3151462, 2022.

[SHÇ+10] James PG Sterbenz, David Hutchison, Egemen K Çetinkaya, Abdul Jabbar, Justin P Rohrer, Marcus Schöller, and Paul Smith. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks*, 54(8):1245–1265, 2010.

[SLS19] Xiaoyan Sun, Peng Liu, and Anoop Singhal. Toward Cyberresiliency in the Context of Cloud Computing [Resilient Security]. *IEEE Security and Privacy*, 16(6):71–75, 2019.

[SM01] William H. Sanders and John F. Meyer. Stochastic Activity Networks: Formal Definitions and Concepts. *Lecture Notes in Computer Science*, 2090(9975019):315–343, 2001.

[SMADB+20] J. San-Miguel-Ayanz, T. Durrant, R. Boca, P. Maianti, G. Libertá, T. Artés Vivancos, D. Jacome Felix Oom, A. Branco, D. De Rigo, D. Ferrari, H. Pfeiffer, R. Grecchi, D. Nuijten, and T. Leray. *Forest Fires in Europe, Middle East and North Africa 2019*. EUR 30402 EN, Publications Office of the European Union, Luxembourg, 2020, JRC122115, 2020.

[SML19] Jukka Soikkeli, Luis Muñoz-González, and Emil Lupu. Efficient Attack Countermeasure Selection Accounting for Recovery and Action Costs. In *14th Int. Conference on Availability, Reliability and Security, ARES '19*. ACM, 2019.

[SPL21] Jukka Soikkeli, Cora Perner, and Emil Lupu. Analyzing the Viability of UAV Missions Facing Cyber Attacks. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 103–112. IEEE, 2021.

[SSAF+19] Hazim Shakhatreh, Ahmad H. Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. *IEEE Access*, 7:48572–48634, 2019.

[SSB20] Nallan Suresh, George Lawrence Sanders, and Michael J. Braunscheidel. Business Continuity Management for Supply Chains Facing Catastrophic Events. *IEEE Engineering Management Review*, 48(3):129–138, 2020.

[SSL17] Xiaoyan Sun, Anoop Singhal, and Peng Liu. Towards actionable mission impact assessment in the context of cloud computing. *Lecture Notes in Computer Science*, 10359 LNCS:259–274, 2017.

[SSLHC16] A. Shameli-Sendi, H. Louafi, W. He, and M. Cheriet. Dynamic optimal countermeasure selection for intrusion response system. *IEEE Trans. Depend. Sec. Comput.*, 15(5):755–770, 2016.

[Sta02] Standard Performance Evaluation Corporation. SPECjAppServer2002 Benchmark. [Online] Available: `http://www.spec.org/jAppServer2002/index.html`, Accessed: 2020-02-13, 2002.

[STM15] Navid Sahebjamnia, S Ali Torabi, and S Afshin Mansouri. Integrated business continuity and disaster recovery planning: Towards organizational resilience. *European Journal of Operational Research*, 242(1):261–273, 2015.

[SWP17] Huasong Shan, Qingyang Wang, and Calton Pu. Tail Attacks on Web Applications. In *Proc. 2017 ACM SIGSAC Conf. Computer and Communications Security*, pages 1725–1739, 2017.

[TCNFW16] Hiep Tran, Enrique Campos-Nanez, Pavel Fomin, and James Wasek. Cyber resilience recovery model to combat zero-day malware attacks. *Computers and Security*, 61:19–31, 2016.

[Ten] Tenable. Nessus. [Online] Available: `https://www.tenable.com/products/nessus`. Accessed: 2022-03-16.

[U.S12] U.S. National Academies. *Disaster Resilience: A National Imperative*. The National Academies Press, Washington, DC, 2012.

[VTC+14] An Hoa Vu, Nils Ole Tippenhauer, Binbin Chen, David M Nicol, and Zbigniew Kalbarczyk. Cybersage: A tool for automatic security assessment of cyber-physical systems. In *Int. Conference on Quantitative Evaluation of Systems*, pages 384–387. Springer, 2014.

[WDCO15] Dong Wei, Leandro Pfleger De Aguiar, Ben Collar, and Martin Otto. Improving control system resilience by highly coupling security functions with control. *Resilience Week, RWS 2015*, pages 195–198, 2015.

[WIL+08] Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, and Sushil Jajodia. An attack graph-based probabilistic security metric. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 283–296. Springer, 2008.

[WL16] Ziheng Wei and Jiamou Liu. Measuring Structural Resilience through a Carrier Network Model. *IEEE 14th Int. Conference on Dependable, Autonomic and Secure Computing, DASC 2016, IEEE 14th Int. Conference on Pervasive Intelligence and Computing, PICom 2016, IEEE 2nd Int. Conference on Big Data*, pages 184–189, 2016.

[WNJ06] Lingyu Wang, Steven Noel, and Sushil Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18):3812–3824, 2006.

[WRS20] Yi Wang, Anastasios Oulis Rousis, and Goran Strbac. On microgrids and resilience: A comprehensive review on modeling and operational strategies. *Renewable and Sustainable Energy Reviews*, 134:110313, 2020.

[WSD15] Wenjun Wang, W. Nick Street, and Renato E. DeMatta. Topological Resilience Analysis of Supply Networks under Random Disruptions and Targeted Attacks. *IEEE/ACM Int. Conference on Advances in Social Networks Analysis and Mining - ASONAM '15*, pages 250–257, 2015.

[YS20] Jean-Paul Yaacoub and Ola Salman. Security analysis of drones systems: Attacks, limitations, and recommendations. *Internet of Things*, 11:100218, 2020.

[YTGW14] Shui Yu, Yonghong Tian, Song Guo, and Dapeng Oliver Wu. Can We Beat DDoS Attacks in Clouds? *IEEE Trans. Parallel Distrib. Syst.*, 25(9):2245–2254, 2014.

[YWR18] Nita Yodo, Pingfeng Wang, and Melvin Rafi. Enabling Resilience of Complex Engineered Systems Using Control Theory. *IEEE Transactions on Reliability*, 67(1):53–65, 2018.

[YWZ17] Nita Yodo, Pingfeng Wang, and Zhi Zhou. Predictive Resilience Analysis of Complex Systems Using Dynamic Bayesian Networks. *IEEE Transactions on Reliability*, 66(3):761–770, 2017.

[ZBK15] Richard M. Zahoransky, Christian Brenig, and Thomas Koslowski. Towards a process-centered resilience framework. *10th Int. Conference on Availability, Reliability and Security, ARES 2015*, pages 266–273, 2015.

[ZHLB16] Richard Zahoransky, Julius Holderer, Adrian Lange, and Christian Brenig. Process analysis as first step towards automated business security. *24th European Conference on Information Systems, ECIS 2016*, pages 1–15, 2016.

[ZKA14] Richard M. Zahoransky, Thomas Koslowski, and Rafael Accorsi. Toward resilience assessment in business process architectures. *Lecture Notes in Computer Science*, 8696:360–370, 2014.

[ZKL+17] Yanbo Zhang, Rui Kang, Ruiying Li, Chenxuan Yang, and Yi Yang. Resilience-based component importance measures for complex networks. *2016 Prognostics and System Health Management Conference (PHM-Chengdu)*, pages 1–6, 2017.

[ZMSG18] Xiaoge Zhang, Sankaran Mahadevan, Shankar Sankararaman, and Kai Goebel. Resilience-based network design under uncertainty. *Reliability Engineering and System Safety*, 169:364–379, 2018.

[ZWC+19] Jianping Zeng, Shuang Wu, Yanyu Chen, Rui Zeng, and Chengrong Wu. Survey of Attack Graph Analysis Methods from the Perspective of Data and Knowledge Processing. *Security and Communication Networks*, 2019.

[ZWJ+16] Mengyuan Zhang, Lingyu Wang, Sushil Jajodia, Anoop Singhal, and Massimiliano Albanese. Network Diversity: A Security Metric for Evaluating the Resilience of Networks Against Zero-Day Attacks. *IEEE Transactions on Information Forensics and Security*, 11(5):1071–1086, 2016.