Scalable and Adaptable Security Modelling and Analysis

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Doctor of Philosophy
in the
University of Canterbury
by
Jin Bum Hong

Supervision and Examining Committee

Dr. Dong Seong Kim	Supervisor
Prof. Yang Xiang	External Examiner
Prof. Paul Watters	External Examiner

Department of Computer Science and Software Engineering
University of Canterbury
2015

Abstract

Modern networked systems are complex in such a way that assessing the security of them is a difficult task. Security models are widely used to analyse the security of these systems, which are capable of evaluating the complex relationship between network components. Security models can be generated by identifying vulnerabilities, threats (e.g., cyber attacks), network configurations, and reachability of network components. These network components are then combined into a single model to evaluate how an attacker may penetrate through the networked system. Further, countermeasures can be enforced to minimise cyber attacks based on security analysis. However, modern networked systems are becoming large sized and dynamic (e.g., Cloud Computing systems). As a result, existing security models suffer from scalability problem, where it becomes infeasible to use them for modern networked systems that contain hundreds and thousands of hosts and vulnerabilities. Moreover, the dynamic nature of modern networked systems requires a responsive update in the security model to monitor how these changes may affect the security, but there is a lack of capabilities to efficiently manage these changes with existing security models. In addition, existing security models do not provide functionalities to capture and analyse the security of unknown attacks, where the combined effects of both known and unknown attacks can create unforeseen attack scenarios that may not be detected or mitigated. Therefore, the three goals of this thesis are to (i) develop security modelling and analysis methods that can scale to a large number of network components and adapts to changes in the networked system; (ii) develop efficient security assessment methods to formulate countermeasures; and (iii) develop models and metrics to incorporate and assess the security of unknown attacks.

A lifecycle of security models is introduced in this thesis to concisely describe performance and functionalities of modern security models. The five phases in the lifecycle of security models are: (1) Preprocessing, (2) Generation, (3) Representation, (4) Evaluation, and (5) Modification.

To achieve goal (i), a hierarchical security model is developed to reduce the computational costs of assessing the security while maintaining all security information, where each layer captures different security information. Then, a comparative analysis is presented to show the scalability and adaptability of security

models. The complexity analysis showed that the hierarchical security model has better or equivalent complexities in all phases of the lifecycle in comparison to existing security models, while the performance analysis showed that in fact it is much more scalable in practical network scenarios.

To achieve goal (ii), security assessment methods based on importance measures are developed. Network centrality measures are used to identify important hosts in the networked systems, and security metrics are used to identify important vulnerabilities in the host. Also, new network centrality measures are developed to improvise the lack of accuracy of existing network centrality measures when the attack scenarios consist of attackers located inside the networked system. Important hosts and vulnerabilities are identified using efficient algorithms with a polynomial time complexity, and the accuracy of these algorithms are shown as nearly equivalent to the naive method through experiments, which has an exponential complexity.

To achieve goal (iii), unknown attacks are incorporated into the hierarchical security model and the combined effects of both known and unknown attacks are analysed. Algorithms taking into account all possible attack scenarios associated with unknown attacks are used to identify significant hosts and vulnerabilities. Approximation algorithms based on dynamic programming and greedy algorithms are also developed to improve the performance. Mitigation strategies to minimise the effects of unknown attacks are formulated on the basis of significant hosts and vulnerabilities identified in the analysis. Results show that mitigation strategies formulated on the basis of significant hosts and vulnerabilities can significantly reduce the system risk in comparison to randomly applying mitigations.

In summary, the contributions of this thesis are: (1) the development and evaluation of the hierarchical security model to enhance the scalability and adaptability of security modelling and analysis; (2) a comparative analysis of security models taking into account scalability and adaptability; (3) the development of security assessment methods based on importance measures to identify important hosts and vulnerabilities in the networked system and evaluating their efficiencies in terms of accuracies and performances; and (4) the development of security analysis taking into account unknown attacks, which consists of evaluating the combined effects of both known and unknown attacks.

Publications Arising from this Thesis

Much of the work contained in this thesis has been published or submitted in peer-reviewed conferences and journals listed below. Unpublished work and preprint versions are included in the appendicies. Each publications are noted in parentheses for the corresponding chapters.

1. **Hong, J.** and Kim, D.; "HARMs: Hierarchical Attack Representation Models for Network Security Analysis" in Proc. of the 10th Australian Information Security Management Conference in SECAU Security Congress (SECAU 2012), Dec. 2012. Perth, Australia, 74-81. (Chapters 2 and 3)

2. **Hong, J.** and Kim, D.; "Performance analysis of scalable attack representation models" in Proc. of the 28th IFIP TC-11 International Information Security and Privacy Conference (SEC 2013), Jul. 2013. Auckland, New Zealand, 330-343. (Chapter 4)

3. **Hong, J.** and Kim, D. and Takaoka, T.; "Scalable Attack Representation Model Using Logic Reduction Techniques" in Proc. of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2013), Jul. 2013. Melbourne, Australia, 404 - 411. (Chapter 2)

4. **Hong, J.** and Kim, D.; "Scalable Security Analysis in Hierarchical Attack Representation Model using Centrality Measures" in Proc. of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSNW 2013), Jun. 2013. Budapest, Hungary, 1 - 8. (Chapter 7)

5. **Hong, J.** and Kim, D.; "Scalable Security Model Generation and Analysis using k-importance Measure" in Proc. of the 9th International Conference on Security and Privacy in Communication Networks (SecureComm 2013), Sep 2013. Sydney, Australia, 270-287. (Chapters 3 and 6)

6. **Hong, J.** and Kim, D.; "Scalable Security Models for Assessing Effectiveness of Moving Target Defenses" in Proc. of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014), Jun 2014. Atlanta, USA, 515 - 526. (Chapters 2 and 5)

7. **Hong, J.** and Kim, D. and Haqiq, A.; "What Vulnerability Do We Need To Patch First?" in Proc. of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSNW 2014), Jun 2014. Atlanta, USA, 684 - 689. (Chapter 6)

8. **Hong, J.** and Eom, T. and Park, J. and Kim, D.;"Scalable Security Analysis using Partition and Merge Approach in an Infrastructure as a Service Cloud" in Proc. of the 11th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC 2014), Dec 2014. Bali, Indonesia. (Chapter 2)

9. **Hong, J.** and Kim, D.; "Security Assessment and Mitigation of Unknown Attacks" submitted to the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2015), Dec 2014 (Chapter 8, Appendix A)

10. **Hong, J.** and Kim, D.; "Performance Analysis of Graph-based Security Models" under revision for the Elsevier Special Issue of the Computers and Security Journal, Oct 2014 (Chapters 2 and 4, Appendix B).

11. **Hong, J.** and Kim, D.; "Assessing the Effectiveness of Moving Target Defense using Security Models" under revision for the IEEE Special Issue on Transactions of Dependable and Secure Computing (TDSC) Journal, Oct 2014 (Chapters 2 and 5, Appendix C).

12. **Hong, J.** and Chung, C. and Huang, D. and Kim, D.; "Survey of Graph-based and Tree-based Security Models" to be submitted to the IEEE Communications Surveys & Tutorials, Oct 2014 (Chapters 2 and 3, Appendix D)

Acknowledgments

I would like to express my gratitude to everyone who supported me throughout the period of creating this thesis in my life. It would have been impossible to complete this thesis without their support, encouragement, and guidance.

First of all, thank you to my supervisor, Dr. Dong-Seong Kim, for your guidance, support, encouragement, and wisdom throughout the research process. You have enlightened me with many groundbreaking ideas that form the skeleton of this thesis, and I most certainly could not have completed this project without you. Thanks also to Professor Tadao Takaoka, my co-supervisor, for your vast knowledge in algorithms, as well as for continuously checking my progress and providing moral support for all these years.

Thanks to senior research advisors of NRG and SRG, Professor Kryzsztof Pawlikowski, Dr. Andreas Willig, Dr. Greg Ewing, and Professor Don McNickle, for your guidance and technical discussions that further enriched the completeness of this thesis.

Thanks to my parents for raising me here in New Zealand, where I have learned to be open minded and independent through good times and rough times. Thanks to JK and Jen for being a lovely brother and sister, I definitely enjoy times talking about other (very random) topics than computers.

Thank you Hannah Vu, for supporting me during my research, for your efforts of comforting me when I am stressed, for your ever so delicious home-made food, and finally for all the good and rough times we went through together, as well as ones to come. I would have never been the same without you.

Thank you to my DS lab, as well as NRG and SRG colleagues (Fangcheng (T), Saima, Ehsan, and Amir) for many interesting talks and discussions.

Finally, thank you to all others that I received guidance, idea, and support from. Your valuable time spent has made a significant contribution to this thesis.

This thesis was partially funded by the Todd Foundation Awards for Excellence and the G B Battersby-Trimble Scholarship. Financial support for conference travel was partially provided by the Claude McCarthy Fellowship.

Table of Contents

List of Figures **vii**

List of Tables **xi**

Chapter 1: Introduction **1**
1.1 Research Goals 4
1.2 Research Contributions 4
1.3 Thesis Structure 6

I Introduction to Hierarchical Security Models **9**

Chapter 2: Hierarchical Security Model (HARM) **11**
2.1 Security Models and Their Developments 11
2.2 A Need for a Scalable and Adaptable Security Model 15
2.2.1 An Example Networked System 16
2.3 Formalism of the HARM 18
2.3.1 Formalism of the HARM structures 18
2.3.2 Formalism of the HARM transformations 21
2.4 Tools and Methods to Generate Security Models 24
2.5 Security Analysis using the HARM 25
2.5.1 Security Analysis of Graph-based Security Models 25
2.5.2 Security Analysis of Tree-based Security Models 27
2.6 Summary 28

II Comparative Analysis of Security Models **29**

Chapter 3: Complexity Analysis **31**
3.1 Complexity Analysis of the Generation Phase 32
3.1.1 Generating AG, AT, and HARM 32

3.1.2 Generating Other Security Models 33
3.2 Complexity Analysis of the Representation Phase 35
3.2.1 Representing AG, AT, and HARM 35
3.2.2 Representing Other Security Models 36
3.3 Complexity Analysis of the Evaluation Phase 36
3.3.1 Evaluating AG, AT, and HARM 36
3.3.2 Evaluating Other Security Models 37
3.4 Complexity Analysis of the Modification Phase 38
3.4.1 Modifying AG, AT, and HARM 38
3.4.2 Modifying Other Security Models 39
3.5 Structural Advantages of the HARM 39
3.6 Conclusions 40

Chapter 4: Scalability Analysis **41**
4.1 Key Questions to Compare Scalability 42
4.1.1 Generation Phase 43
4.1.2 Evaluation Phase 45
4.2 Simulation Results 46
4.2.1 Experiment 4A: Simple Network Topologies 46
4.2.2 Experiment 4B: Combined Network Topologies 50
4.3 Discussions 53
4.3.1 Scalability of Security Models in the Lifecycle Phases . . 53
4.3.2 Network Structure and Attack Scenarios 55
4.3.3 Real Testbed Experiments 56
4.3.4 Comparisons with Other Security Models 56
4.3.5 Differences between the AG and HARM 57
4.3.6 Security Evaluation and Overhead 57
4.4 Conclusions 57

Chapter 5: Adaptability Analysis **59**
5.1 Preliminaries 60
5.1.1 A Virtualised System 60
5.1.2 Categorising the MTD techniques 62
5.1.3 Securing attack paths 62

5.1.4 Computing the Importance Measures 64
5.2 *Shuffle* 65
5.2.1 Incorporating only Shuffle in the HARM 65
5.2.2 Assessing the Effectiveness of Shuffle 65
5.2.3 Experiment 5A: Analysing Shuffle 67
5.3 *Diversity* 69
5.3.1 Incorporating only Diversity in the HARM 69
5.3.2 Assessing the Effectiveness of Diversity 70
5.3.3 Experiment 5B: Analysing Diversity 73
5.4 *Redundancy* 78
5.4.1 Assessing the Effectiveness of Redundancy 79
5.4.2 Experiment 5C: Analysing Redundancy 80
5.5 Discussion 83
5.5.1 Validation using a Real System 83
5.5.2 Incorporating Various Vulnerabilities 83
5.5.3 Optimisation of the IMs w.r.t the MTD techniques 84
5.5.4 Optimisation between performance and security metrics . 84
5.5.5 Combining multiple MTD techniques 84
5.6 Related Work on MTD Techniques 85
5.7 Conclusions 87

III Importance Measure based Security Assessments 89

Chapter 6: Attacker Located Outside the Networked System 91
6.1 Computing IMs 92
6.1.1 Ranking Important Hosts 92
6.1.2 Ranking Important Vulnerabilities 94
6.1.3 Generating the HARM 94
6.2 Security Analysis using IMs 95
6.2.1 Risk Analysis using IMs 96
6.3 Combining the Importance Measures 97
6.3.1 Security Analysis using the ES Method 98
6.3.2 Security Analysis using the Combined IMs 101
6.4 Simulation Results 103

6.4.1 Experiment 6A: Effectiveness of IMs . . . 103
6.4.2 Experiment 6B: Effectiveness of Combined IMs . . . 106
6.5 Discussion . . . 110
6.5.1 Vulnerabilities without security metrics . . . 111
6.5.2 Categorised vulnerability ranking . . . 111
6.5.3 Network features for k_1 selection . . . 112
6.5.4 Attack on less important hosts and vulnerabilities . . . 112
6.5.5 Adjusting the weight between host and vulnerability IMs . 112
6.5.6 Order of vulnerabilities in the PSV . . . 113
6.6 Related Work on Scalable Security Analysis . . . 113
6.7 Conclusions . . . 114

Chapter 7: Attacker Located Inside the Networked System 117
7.1 Security Analysis with Existing IMs . . . 118
7.1.1 Comparing against the ES method . . . 118
7.2 Location-based IMs . . . 120
7.3 Simulation Results . . . 121
7.3.1 Experiment 7A: Attacker Outside the Networked System . 121
7.3.2 Experiment 7B: Attacker Inside the Networked System . . 123
7.3.3 Experiment 7C: Scalability of IMs . . . 125
7.4 Discussion . . . 126
7.4.1 Combinations of NCMs . . . 128
7.4.2 Combining NCMs with Vulnerabilities . . . 128
7.4.3 Multiple Target Hosts and Locations . . . 128
7.5 Related Work on Using Network Centrality Measures . . . 129
7.6 Conclusions . . . 129

IV Effects of Unknown Attacks 131

Chapter 8: Security Assessment of Unknown Attacks 133
8.1 Security Modelling of unVIP . . . 134
8.1.1 Classification of Unknown Attacks . . . 134
8.1.2 An Example Networked System and Attack Scenarios . . 136
8.1.3 Incorporate unVIP into the Security Model . . . 137

8.2 Security Analysis of Systems with unVIP 142
8.2.1 Security Analysis without unVIP 142
8.2.2 Security Analysis with unVIP 142
8.3 unVIP Mitigation Strategies 144
8.3.1 All Possible Attack Scenarios 144
8.3.2 Identification of Significant Hosts 144
8.3.3 Identification of Significant Vulnerabilities 148
8.4 Experimental Results 151
8.4.1 Security Analysis 152
8.4.2 Performance Analysis 157
8.5 Discussion 161
8.5.1 unVIP Mitigation Strategies and Effectiveness 161
8.5.2 Implementation in a Real Testbed 162
8.5.3 Security Metrics for Assessing Unknown Attacks 162
8.6 Related Work on Security Assessment of Unknown Attacks 162
8.7 Conclusions 164

V General Discussion and Conclusions 167

Chapter 9: Discussion 169
9.1 Research Objectives 170
9.2 Limitations and Future Work 171
9.2.1 Properties of the HARM 171
9.2.2 Scalability and Adaptability 174
9.2.3 Security Analysis using the HARM 176
9.2.4 Assessing the Effectiveness of Unknown Attacks 179
9.2.5 Unknown Attack Modelling 179

Chapter 10: Conclusions 181

References 183

Appendices 211

Appendix A: Submitted Publication: Security Assessment and Mitigation of Unknown Attacks 213

Appendix B: Draft Publication: Performance Analysis of Graph-based Security Models 227

Appendix C: Draft Publication: Assessing the Effectiveness of Moving Target Defenses using Security Models 253

Appendix D: Draft Publication: Survey of Graph-based and Tree-based Security Models 269

List of Figures

2.1 Development of Security Models 12
2.2 The Lifecycle of Security Models 15
2.3 An Example Networked System 17
2.4 2-HARM of the example networked system using AGs 19
2.5 2-HARM of the example networked system using ATs 20
2.6 Deploying MTD Techniques in the Networked System 22
2.7 An AG of the Networked System 25
2.8 System Risk Computation Algorithm Using the HARM 26
2.9 An AT of the Networked System 27

4.1 A Networked System for Experiment 4A 47
4.2 A Comparison between AG and HARM in the Generation Phase . 48
4.3 A Comparison between AG and HARM in the Evaluation Phase . 49
4.4 Scalability Difference of Network Topologies in the Evaluation Phase 49
4.5 Scalability Difference with Varying Number of Vulnerabilities . . 50
4.6 Generation of AG, 2-HARM, and 3-HARM 51
4.7 Performance of evaluating combined network topologies in the order of star-star, tree-star, ring-star and mesh-star topologies respectively 52
4.8 Performance of 2-HARM and 3-HARM Evaluating Various Network Topologies 53
4.9 Performance of AG, 2-HARM, and 3-HARM Evaluating Various Numbers of Vulnerabilities 54
4.10 Performance of AG, 2-HARM, and 3-HARM Evaluating Various Network Density 55

5.1 A Virtualised System for the Example Networked System 61
5.2 A Comparison of Securing Attack Paths and End Points 64

5.3 Migration of VMs in the Upper Layer of the HARM 66
5.4 A CloudBand Model for Simulation 68
5.5 HARM of the CloudBand Model with five VMs on Each Node . . 69
5.6 Performance comparison between AG and HARM when deploying a VM-LM 70
5.7 Comparison between the ES method and using the IMs when deploying a VM-LM 71
5.8 Possible Lower Layer ATs for VM_1 with OS Diversity 72
5.9 An Example Virtualised System for DAP 73
5.10 OS diversity assignments for our example 74
5.11 ECC and Risk Changes with Respect to the Number of Nodes . . 75
5.12 ECC and Risk Changes with Respect to the Network Density . . . 76
5.13 ECC and Risk Changes with Respect to the Number of Variants . 77
5.14 Deploying *Redundancy* Technique in the Virtualised System . . . 79
5.15 Various Probability of the Attack Success with Respect to *Redundancy* 81
5.16 Mean-Time-To-Attack with Respect to *Redundancy* 82
5.17 Risk and Availability with Respect to *Redundancy* 82

6.1 The 2-HARM of the Example Networked System 95
6.2 The ReHARM of the Example Networked System 96
6.3 ES Method using the Risk Metric 99
6.4 Performance of Security Analysis using k_1 values 105
6.5 Performance of Security Analysis using k_2 values 107

7.1 Performance of NCMs 122
7.2 A Networked System for the Second Simulation 123
7.3 Performance of NCMs with an Inside Attacker 124
7.4 Performance using AVNC Measures 125
7.5 Attack Scenario Covering Half of Networked system Hosts 126
7.6 Evaluation Time for NCMs 127

8.1 unVIP Attack Scenarios in the Example Networked System 138
8.2 HARM of the Example Networked System 139
8.3 Incorporating unVIP in the HARM 140

8.4 Pseudocode to Analyse all unVIP Scenarios 144
8.5 Identifying Significant Hosts Algorithm 145
8.6 Identifying Significant Hosts Approximation Algorithm 146
8.7 Security Analysis After Hardening a Host in the Networked System 148
8.8 Identifying Significant Vulnerabilities Algorithm 149
8.9 Identifying Known Vulnerabilities Approximation Algorithm . . . 150
8.10 The Networked System for Simulations 151
8.11 Security Analysis of the Networked System with only UV/UI . . . 153
8.12 unVIP assumed only in the IN Subnet 154
8.13 Effect of Patching Vulnerabilities in the Networked System 155
8.14 UVs in Respect to the System Risk 156
8.15 Effect of Varying Number of UVs 158
8.16 Effect of Varying Number of Known Vulnerabilities 160

List of Tables

2.1 Application of Metrics for Security Models 13
2.2 Security Models used in Various System Domains 14
2.3 OS used in Hosts 16
2.4 Windows 7 Vulnerabilities 17
2.5 Redhat Enterprise Linux Vulnerabilities 18

3.1 Computational Complexities of Phases in Security Model Lifecycle 40

4.1 Answers to Key Questions in the Generation Phase 42
4.2 Answers to Key Questions in the Evaluation Phase 44

5.1 Categories of MTD Techniques 63
5.2 NCMs and Ranking Important VMs 65
5.3 Total Attack Cost of deploying OS Diversity (in unit dollars) . . . 72
5.4 System Risk, Reliability and Probability using *Redundancy* 80

6.1 NCMs of Hosts in the Example Networked System 93
6.2 Vulnerability Rankings of W7 Hosts 94
6.3 Vulnerability Rankings of REL Hosts 94
6.4 Risk Analysis of Attack Paths 97
6.5 Risk analysis of attack paths using ReHARM 97
6.6 Risk-based PSV using the ES Method 99
6.7 Attack Cost based PSV using the ES Method 100
6.8 A List of PSV using the TD Method 101
6.9 A List of PSV using the BU Method 102
6.10 A List of PSV using the HB Method 103
6.11 Security Analysis using k_1 values ($k_2 = 10$) 104
6.12 Security analysis using k_2 values ($k_1 = 1000$) 106
6.13 Comparison of Naive and Optimal Solutions 108
6.14 Time to Compute the PSV (in seconds) 108

6.15 Set Coverage in Scenario 3, $k = 0.5$ 109
6.16 Set Coverage in Scenario 4, $k = 0.25$ 109
6.17 Set Coverage in Scenario 5 110

7.1 Occurrence of Hosts in All Possible Attack Paths 119
7.2 Occurrence of Vulnerabilities in Attack Paths 120
7.3 Ranking of Hosts based on AVC Measures 120

8.1 Details of Hosts 136
8.2 Possible Attack Scenarios 137
8.3 System Risk with Two UVs 143
8.4 Significant Hosts of the Example Networked System 147
8.5 Arbitrarily Assumed Vulnerabilities 150
8.6 Significant Vulnerabilities of the Example Networked System . . . 151

Chapter 1

Introduction

Cyber attacks have significant effects in our daily lives, where they target from critical infrastructures down to small home networks. Cyber attacks are becoming more complex (in terms of their attack patterns, types and methods), which makes harder to defend our networked systems against them. To fully understand the security of networked systems in depth, one must take into account not only the effects of individual vulnerabilities when they are exploited, but also the effects of cyber attacks exploiting vulnerabilities in different combinations. Hence, it is of paramount importance to achieve three security goals, which are *Confidentiality*, *Integrity*, and *Availability* (also known as the CIA triads) [80]. In-depth security assessments can identify critical attack scenarios and their associated vulnerabilities, which can be secured effectively by formulating mitigation strategies. For example, an Advanced Persistent Threat (APT) may violate confidentiality and integrity, and a Distributed Denial of Service (DDoS) attack violates the availability of the networked system. Hence, it is essential to carry out in-depth security assessments of networked systems to identify critical attack scenarios, and deploy effective mitigation strategies to minimise the impact of cyber attacks while ensuring the CIA of the networked system.

There are three major elements to assess the security of networked systems in depth, which are: (i) Security Measurement, (ii) Security Metrics, and (iii) Security Models. A security measurement is a process of collecting security related information in the networked system, such as identifying vulnerabilities and their associated threats. These information can be collected by means of using testbed measurements [181], emulations [143], and honeypots [11]. Security metrics are various qualitative and quantitative representations of cyber security associated with the networked system, and they are being developed from a wide range of disciplines [116] such as business community [66, 106], governments [3–6] and standards organisations [1, 2, 101]. Security models are generated based on se-

curity measurements, and they assess the security of a networked system using security metrics. Existing studies conduct security assessments using these three elements, and hardening the networked system (i.e., enhance security) can be done efficiently [99, 100, 158, 207]. However, modern networked systems are becoming large and dynamic, such as Cloud Computing systems that can scale to thousands of hosts, as well as dynamically adapting to traffic loads [140]. As a result, existing security models suffer from scalability and adaptability problems [88, 124].

Graph-based (e.g., an Attack Graph (AG) [182]) and tree-based (e.g., an Attack Tree (AT) [179]) security models are two mainly used approaches. Graph-based security models allow ones to assess the security in an acyclic graph that captures all possible attack paths [55, 79, 87, 88, 99, 124, 151, 158]. However, such approaches have an exponential computational complexity [124]. Tree-based security models express attack scenarios as a combination of attack events (e.g., exploits) [57, 58, 67, 81, 82, 135, 145, 169, 179]. For example, individual attack paths in the AG can be expressed as a series of events in the AT. However, it is a common practice to manually generate them (i.e., error prone and does not scale especially for a large sized networked systems), and automated generation methods create an exponential number of attack events [94, 191]. Moreover, there is a lack of knowledge about adaptabilities of existing security models, where taking into account regeneration of these models for each update in the networked system has exponential complexity [87].

To fully understand security posture of the networked system, analysing all possible attack scenarios provide all possible ways an attacker may penetrate through. However, it is infeasible to compute all possible attack scenarios to evaluate the security of the networked system [88, 124]. To resolve this problem, heuristic methods (e.g., simplification and approximation techniques) are developed [12, 43, 84, 100, 102, 173], but these methods are *model* and *metric* oriented (i.e., heuristic methods to be used only with specific security models and security metrics). As a result, provided functionalities and methods cannot be shared with other security models (i.e., there is no co-operability between existing security models). There are usually multiple network hardening choices available [12, 173, 200, 214], and the optimal solution depends on the security metrics used [93]. For example, incorporating a new firewall can minimise the system risk with a high investment cost, whereas patching vulnerabilities can minimise the in-

vestment cost but only a small reduction in the system risk. Therefore, efficient security assessment methods are needed that are not model or metric oriented, as well as an equivalent accuracy with respect to evaluating all possible attack scenarios.

Unknown attacks are difficult to detect and mitigate because one cannot generalise the properties of them (e.g., what is the cost of mitigation?) [30, 137, 166]. As a result, they are often not taken into account when hardening the networked system (i.e., hardening the networked system to mitigate only known attacks). Previous research assumed only the unknown vulnerabilities in security models, and analysed the security of networked system based on assumed security metrics [45, 99], or used new security metrics such as k-zero-day safety [14, 198, 199]. Unknown attacks consist of unknown vulnerabilities, unknown devices, and unknown attack paths, but their combined effects are not taken into account in the security analysis. It is of paramount importance to consider the combined effects of those unknown attacks when hardening the networked system, because sophisticated real-life cyber attacks utilise many unknown attacks, such as Stuxnet [63] and RSA SecurID breach [97].

This thesis aims to address the inefficiencies of security models and their analysis methods for modern networked systems. To do so, scalable and adaptable security modelling and analysis methods are developed, efficient security assessment methods are developed to formulate countermeasures, and the combined effects of unknown attacks are analysed to formulate mitigation strategies. A comparative analysis of hierarchical security model and existing security models taking into account their complexities and performances is presented, which showed significant improvements in scalability and adaptability using the hierarchical security model. Next, efficient security assessment methods based on importance measures (i.e., network-centric) are presented, which enhanced the operability (i.e., can be used in other security models) of the security evaluations without functional constraints of models and metrics. Finally, this thesis presents methods to incorporate unknown attacks into the hierarchical security model, analyse the combined effects of unknown attacks in the networked systems, and provide efficient algorithms to formulate mitigation strategies.

The following subsections formally define the research goals and methods to achieve them, contributions of the research, and outline of the thesis structure.

1.1 Research Goals

The goals of this thesis are to advance security assessment of large sized and dynamic networked systems and establish efficient and effective security assessment method. Four specific subgoals are described below.

1. *Develop security modelling and analysis methods to improve scalability and adaptability.* The outcomes of this goal are a new scalable and adaptable security model based on hierarchy developed, and a formal definition of its structures and functionalities.

2. *Compare the scalability and adaptability of security modelling and analysis for modern networked systems*. The outcomes of achieving this goal consist of a comparative analysis of security models, taking into account the complexities and performances of them in a large sized and dynamic networked system. The comparisons are made in terms of phases in the lifecycle of security models.

3. *Develop efficient and effective security assessment methods to enhance capabilities of formulating countermeasures*. The resulting outcomes of this goal are development of new importance measure based security assessment methods and algorithms, and evaluation of them taking into account different attack scenarios caused by cyber attackers located either outside or inside of the networked system.

4. *Analyse the combined effects of unknown attacks*. This goal will result in incorporation of unknown attacks into the framework of the hierarchical security model, developing new algorithms to identify significant hosts and vulnerabilities in the networked system to formulate effective mitigation strategies, and evaluation of these algorithms under various attack scenarios.

1.2 Research Contributions

Four primary contributions are made to formal security modelling and analysis in this thesis, which are:

1. A development of hierarchical security model to improve the scalability and adaptability of existing security models. The design of the hierarchical security model adopts different security models into different layers, which allows the inheritance of various functionalities of the existing security models.

2. A comparison of security models taking into account complexities and performances in terms of phases in the lifecycle of the security models. No such analyses are conducted at the time of writing, so this analysis is not only to compare complexities and performances for existing security models, but also to provide a basis for emerging security models to evaluate and compare their scalability and adaptability to qualify their practicability.

3. A development of importance measures based on important hosts and vulnerabilities. New network centrality measures are developed and existing security metrics are used to rank hosts and vulnerabilities respectively. Two attack scenarios are taken into account; (i) the attacker is located outside, and (ii) the attacker is located inside the networked system. Simulation results show that in case of the attacker outside the networked system, existing network centrality measures are capable of identifying important hosts accurately, whereas in case of the attacker inside the networked system, newly developed network centrality measures provide better accuracy of identifying important hosts.

4. Analysis of unknown attacks by incorporating them into the hierarchical security model. Unknown attacks are classified into unknown vulnerabilities, devices, and attack paths, and security of the networked system when they are introduced is analysed based on all possible attack scenarios. Mitigation strategies to minimise the effects of such unknown attacks are formulated based on new algorithms that compute significant hosts and vulnerabilities in the networked system. The experimental results show the effects on the security posture (e.g., system risk) varies depending on the location of unknown attacks assumed, and effective mitigation strategies can be formulated efficiently using algorithms to identify significant hosts and vulnerabilities.

1.3 Thesis Structure

The thesis is divided into five parts as follows:

- An introduction to hierarchical security model
- A comparative analysis of security models
- Efficient security assessment methods based on importance measures
- Security analysis and mitigation of unknown attacks
- An in-depth discussion of the work contained in this thesis, as well as the identification of limitations and future research directions.

The first four parts of the thesis are intended to address the four research goals described in Section 1.1, respectively.

Part I, addressing Research Goal 1, presents a hierarchical security model to improve the scalability and adaptability of existing security models. Chapter 2 introduces the hierarchical security model and its formal definitions.

Part II, addressing Research Goal 2, compares the complexities and performances of security models in terms of phases in the lifecycle of security models. Chapter 3 compares the complexities of security models on the basis of their structures and functionalities. Chapter 4 carries out performance analysis in terms of scalability of networked systems (i.e., performances of security assessment with respect to an increasing number of network components). Chapter 5 carries out performance analysis in terms of adaptability of networked systems (i.e., performances of security assessment with respect to changes in the networked system).

Part III, addressing Research Goal 3, develops and evaluates security assessments based on importance measures. Chapter 6 uses importance measures with existing network centrality measures and security metrics to assess the security of the networked system when the attacker is located outside the networked system. Chapter 7 develops new network centrality measures in case of an attacker located inside the networked system.

Part IV, addressing Research Goal 4, incorporates unknown attacks into the hierarchical security model and analyse the combined effects of them. Chapter

8 details classification of unknown attacks, incorporation of them into the hierarchical security model, analysis of their combined effects, and two new algorithms to compute significant hosts and vulnerabilities to formulate effective mitigation strategies.

Finally, Part V provides discussions and directions for future work of the thesis.

Part I

Introduction to Hierarchical Security Models

Chapter 2

Hierarchical Security Model (HARM)

This chapter provides an introduction to the hierarchical security model, or also known as a **H**ierarchical **A**ttack **R**epresentation **M**odel (HARM), which is a security model consisting of layers with other security models. Moreover, its formal definitions of structures and transformations are presented in Section 2.3, and the results of security analysis using the HARM, an Attack Graph (AG) [182], and an Attack Tree (AT) [179] are shown in Section 2.5 to yield that equivalent solutions are produced. The HARM forms a basis model structure for methods, functions and algorithms introduced in this thesis, as it is essential to utilise the properties of the HARM later in Chapters 3 through to 8.

First, this chapter introduces the development of security models over the past decades. Second, motivations for a need of scalable and adaptable security models are described. Third, this chapter provides a formal definition of the HARM for its structures and transformations. Fourth, tools and methods to generated security models are described. Finally, an example networked system is taken into account to demonstrate equivalent security analyses using the HARM, an AG [182], and an AT [179].

2.1 Security Models and Their Developments

Over the past two decades, many security models were developed to deal with the growing concern for security of networked systems. Also, as the networked systems become complex (e.g., topologies, protocols, functionalities), various security models are developed to adapt to these changes. Therefore, it is essential to understand how security models have adapted to modern networked systems, and also to understand which functions and methods are provided. Figure 2.1 overviews the relativity of existing security models and their successors, and further details of these models can be found in [85] and [120]. Two security models,

namely the AG and the AT, provided fundamental platforms for many security models. The details of various security models can be found in [85].

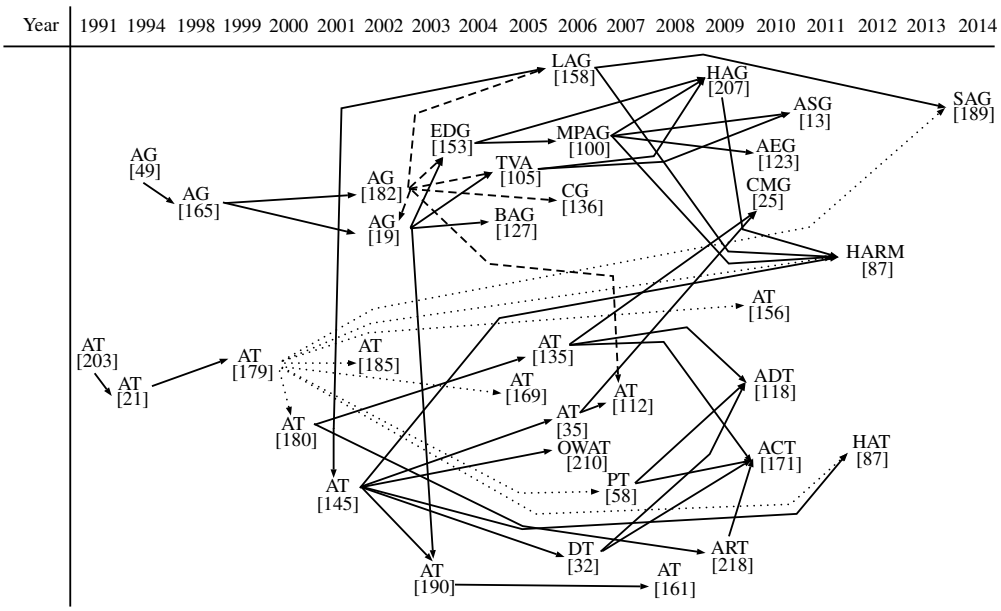


Figure 2.1: Development of Security Models

As a result of a diverse family of security models and security metrics, not all security models can provide functionalities to evaluate various security metrics. Table 2.1 shows security models and their support for various security metrics. Although it is difficult to provide functionalities to evaluate all security metrics, Table 2.1 shows that a few security models do provide functions to evaluate more than four or more different types of security metrics.

Also, as the number of networked systems are adopted to specialised platforms (e.g., e-commerce systems, power grid, and cyber-physical systems), security models are also modified to capture corresponding security information from various system domains. Table 2.2 shows the number of domains supported by security models. It clearly shows that security models are applicable for a wide range of system domains.

Many functionalities are provided to evaluate various security metrics using different security models as shown in Table 2.1. Also, security models are widely adopted in various system domains as shown in Table 2.2. However, due to the diversity of security models and functionalities, there is a lack of study to analyse how scalable and adaptable these security models are, especially when the size of

Table 2.1: Application of Metrics for Security Models

	Prob.	Attack Rate	Attack Cost	Risk	Attack Path	RoI/A	Intrusion Detection
AT	[36, 169]	[111, 114]	[56, 82, 169, 175]	[27,145, 169]			[78]
DT						[31, 32]	
OWAT	[210]		[210]				
PT	[57, 58]		[57, 58]				
ART			[218]				
ADT	[26]	[26]	[26,118, 119]	[118, 119]			
ACT	[171–173]		[171–173]	[171–173]	[171–173]	[171–173]	
HAT	[87]						
AG	[74,125, 126]	[50, 157,165, 186]	[12]	[154, 160]	[18, 107,108, 183]	[154]	[149, 150]
EDG			[29, 153]		[153]		
BAG	[68, 69, 144, 167]		[167]	[167]			[208]
TVA	[196]		[197, 200–202]	[154]		[154]	[162]
LAG	[176]		[213]	[16]	[159]		
MPAG			[100, 205]	[100]			[205]
CG		[136]					
HAG	[87, 207]			[88]			
CMG			[25]				
ASG	[13]	[13]					[13]
AEG					[122, 123]	[122, 123]	[122, 123]
SAG	[42, 189]						

Table 2.2: Security Models used in Various System Domains

System Domain	Model	Reference	Analysis Approach
E-commerce	AG	[65]	Intrusion detection and response
	PT	[57, 58]	Protection probability and cost
	AT	[175, 206]	Cost of attack and impact analysis
Network Monitor	AG	[149, 150]	Learn attacks from intrusion alerts
	TVA	[197]	Correlate, hypothesise and predict alerts
	BAG	[144]	Placement of intrusion detectors
	AT	[111]	Detection rate for Trojan horse
Power Grid	SAG	[42, 189]	Probabilistic analysis of availability and confidentiality
Online Games	CMG	[25]	Countermeasure selection via attack cost analysis
Software Development	AG	[23, 37]	Vulnerability elimination during software development
SCADA	AT	[145, 188]	Risk assessment
Phased-mission	AT	[141]	Reliability assessment
Electronic Copyright Management	AT	[81, 82]	Attack cost, risk, skill, accessibility, and impact
Mobile and Ad Hoc Network	AT	[114]	Assessment of packet drop rates
Airline	AT	[27]	Qualitative risk analysis
E-voting	AT	[36]	Probabilistic analysis
RFID	AT	[178]	Qualitative confidentiality, integrity, availability, and risk
Cyber-physical	EDG	[128, 129]	Analysis description of denial of sleep attack in the Smart Grid

the networked systems are growing rapidly, such as the Cloud. Hence, there is a need for analysing the scalability and adaptability of security models, which is further described in Section 2.2.

2.2 A Need for a Scalable and Adaptable Security Model

Security models are a widely adopted to analyse the security of networked systems, as they do not disrupt the system and also have low costs associated to implement and maintain. They are used to analyse not only individual vulnerabilities in the networked system, but also the security flaws associated with the complex relationship between network components and examine their potential threats. Moreover, security models can provide efficient network hardening solutions to manage those potential threats effectively [19, 53, 55, 175]. However, analysing all possible attack scenarios has a scalability problem [124], which becomes impractical to use security models in modern networked systems consisting of more than thousands of hosts.

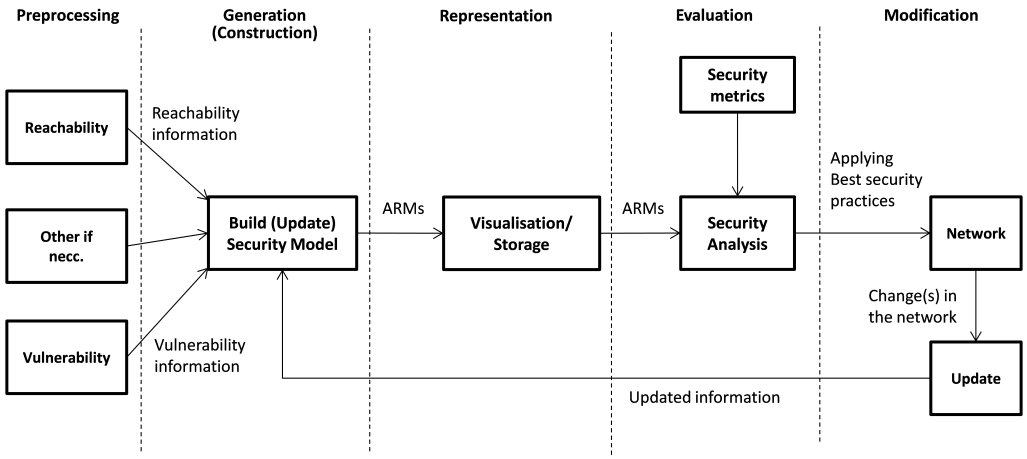


Figure 2.2: The Lifecycle of Security Models

The scalability problem becomes coherent in the terms of the security model lifecycle (denoted as lifecycle), which is first introduced in [90]. Figure 2.2 shows the procedural steps in the lifecycle when analysing the security of networked system, which consist of five phases: (i) Preprocessing, (ii) Generation, (iii) Representation, (iv) Evaluation, and (v) Modification. The preprocessing

phase gathers security information from the networked system (e.g., reachability, vulnerabilities), where they can be collected by means of using measurements in testbeds [181], emulations [143], honeypots [11], and network data flow monitoring [215]. The generation phase combines these information to construct the security model, and there are a wide range of available tools [24, 45, 98, 105, 122, 131,159,182,189,205]. The representation phase stores and visualises the security model, and the evaluation phase analyses the security of networked systems using the security model with given security metrics as inputs, such as ones developed from business communities [66, 106], governments [3–6] and standards organisations [1,2,101]. The modification phase updates the security model when there are changes in the networked system (e.g., adding or removing hosts, adaptive topology change to enhance the performance, and updating hosts with new operating systems that have different set of vulnerabilities), such as the Cloud [52, 140, 212] and Software Defined Networks [46, 104].

To address the scalability problem, two main approaches are used, namely structural modifications [100, 158, 207] and heuristic methods [43, 84, 167]. However, with the networked system becoming larger and dynamic, structural modification solutions still suffer the scalability problem [88]. On the other hand, heuristic methods (e.g., model simplifications and solution approximations) do not take into account all security information given. As a result, using heuristic methods may lose security information that may result in misguided security assessment. Therefore, it is of paramount importance to improve security modelling and analysis methods to overcome the scalability problem.

2.2.1 An Example Networked System

Table 2.3: OS used in Hosts

Host	Default OS	Backup OS
U_1	Windows 7	Windows Vista
U_2	Windows 7	Windows Vista
U_3	Redhat Enterprise Linux	Redhat Linux
U_4	Windows 7	Windows Vista
U_5	Redhat Enterprise Linux	Redhat Linux

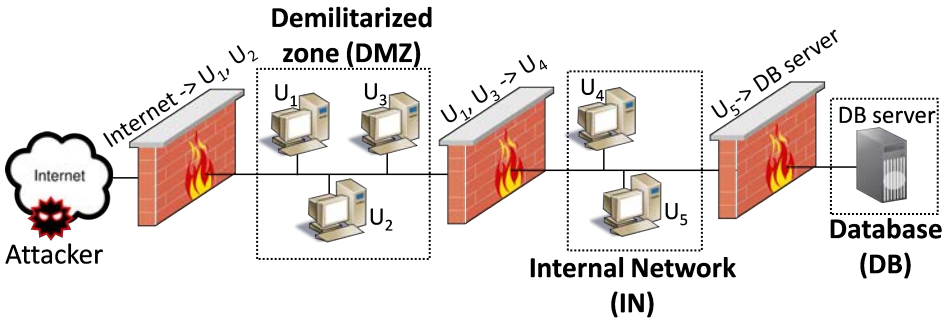


Figure 2.3: An Example Networked System

Table 2.4: Windows 7 Vulnerabilities

ID	CVE ID	CVSS BS	Impact	Exploitability
$W7_1$	CVE-2013-2556	7.5	6.4	10
$W7_2$	CVE-2013-2554	7.5	6.4	10
$W7_3$	CVE-2013-0013	5.8	4.9	8.6
$W7_4$	CVE-2012-0001	9.3	10	8.6
$W7_5$	CVE-2010-0494	4.3	2.9	8.6

Figure 2.3 is used as an example networked system that will be used throughout this thesis. An attacker is located outside the networked system that consists of three subnets (DMZ, IN, and DB). The goal of the attacker is to reach the DB server. Operating Systems (OSes) of hosts (denoted as U) are shown in Table 2.3, where Windows 7 is denoted as W7, Windows Vista is denoted as WV, Redhat Enterprise Linux is denoted as REL, and Redhat Linux is denoted as RL. Only the vulnerabilities found in OSes with properties to bypass firewalls and authentications via remote access are used in security models. However, other vulnerabilities (e.g., application vulnerabilities) can also be modelled. Table 2.4 and Table 2.5 show vulnerabilities associated with W7 and REL, respectively. Further, a common vulnerability scoring system base score (CVSS BS), impact values, and exploitability values are shown, where details of these metrics can be found in [70]. Exploiting any of these vulnerabilities are assumed to give the attacker (either in the state with guest or user privilege) the root privilege and to grant a full control of the host.

Table 2.5: Redhat Enterprise Linux Vulnerabilities

ID	CVE ID	CVSS BS	Impact	Exploitability
REL_1	CVE-2013-2051	2.6	2.9	4.9
REL_2	CVE-2012-4546	4.3	2.9	8.6
REL_3	CVE-2005-2700	10	10	10
REL_4	CVE-2005-0337	7.5	6.4	10
REL_5	CVE-2004-1145	5.0	2.9	10
REL_6	CVE-2004-0607	10	10	10

2.3 Formalism of the HARM

A HARM is developed to improve the scalability of existing security models [87]. The HARM captures security information in the networked system onto different layers, and in each layer a different security model can be implemented. For example, an AG [182] and an AT [179] are two most widely used security models [85,120], and they will mainly be incorporated into the HARM. Of course, other security models can be used in the HARM as well.

Hereon forth, symbols i, j, and k are used for numeric variables and does not present specific metrics or terms. Further, the symbol N represents the whole networked system, and $n \subseteq N$ is a component, which can also be denoted as a node, a host, or a state in the networked system.

2.3.1 Formalism of the HARM structures

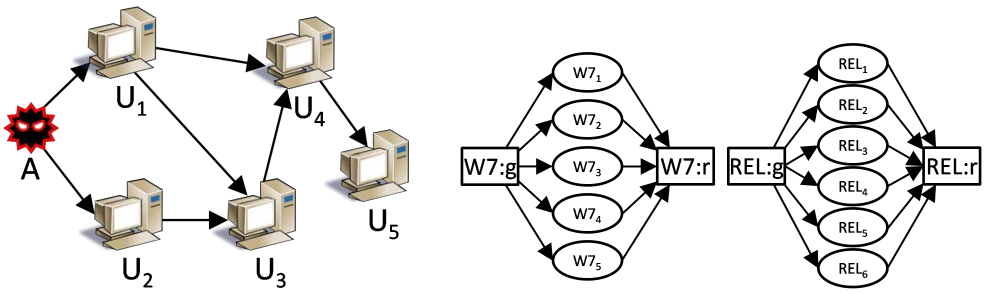
Definition 1. This is initially defined in [91]. The HARM has a 3-tuple $H = (h, M, C)$. h is the number of layers ($2 \leq h$). M is the set of all models used in each layer, $M_i \subseteq M$ is a subset of models in M at the ith layer, and $M_j^i \in M_i$ is the jth model at the ith layer in the HARM with $j = 1 \ldots |M_i|$. $C = \{n \leftrightarrow M_l^k \mid n \in M_j^i, j = 1 \ldots |M_i|, l = 1 \ldots |M_k|, i < k\}$ is the set of all mappings between a component n in a model M_j^i to its lower layer model M_l^k.

There are an arbitrary number of layers in the HARM, which is denoted as h-HARM for a h number of layers. The formalism of 2-HARM consisting of two layers with an AG in the upper layer and ATs in the lower layer is presented in [92], and the formalism of h-HARM consisting of only AGs in each layer is

presented in [91].

Definition 2. This is initially defined in [91]. An Attack Graph (AG) in the *i*th layer in the HARM is a directed graph $G^i = (N, E)$, where N is a finite set of network components or states and $E \subseteq N \times N$ is a set of edges where a pair of network components or states (n_j, n_k) is a mapping of $n_j \rightarrow n_k$ with $n_j, n_k \in N$ and $j \neq k$.

Definition 3. This is closely related to [92]. An Attack Tree (AT) in the *i*th layer in the HARM is defined as $T^i = (A, B, a_0)$, where A is a finite set of network components, states and gate (denoted as $gate$) with $gate \in [AND-gate, OR-gate]$, and $B \subseteq A \times A$ is a set of edges. An edge $(a_j, a_k) \in B$ represents a child node $a_j \in A$ attached to a parent node $a_k \in A$, and $a_0 \in A$ is the root node of the AT.



(a) Upper Layer of the HARM (b) Lower Layer of the HARM

Figure 2.4: 2-HARM of the example networked system using AGs

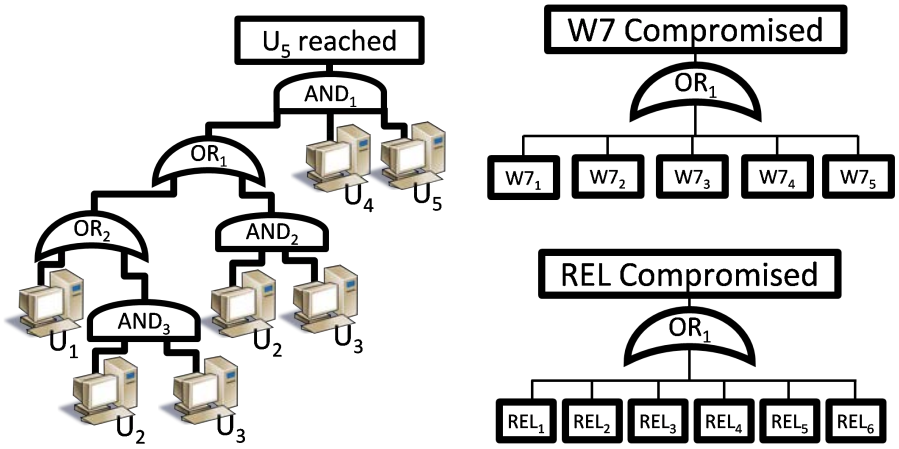
Figure 2.4 shows the 2-HARM of the example networked system using AGs only in both layers. The upper layer captures the reachability of the hosts, and the lower layer captures the exploit sequences required to compromise the OSes. The privilege state is shown as a rectangle in the lower layer, where there are two types of privileges shown; a guest privilege g, and a root privilege r (e.g., the root privilege of W7 is denoted as $W7 : r$). The examples are shown below:

Example 1. *The HARM using only AGs:* The 2-HARM in Figure 2.4 is $H^{ag} = (2, M^{ag}, C^{ag})$, where 2 is the number of layers, $M^{ag} = \{G_1^1, G_1^2, G_2^2\}$ for G_1^1 is an AG mapping the reachability of hosts in the upper layer, and G_1^2 and G_2^2 are AGs

for compromising W7 and REL, respectively. $C^{ag} = \{U_1 \leftrightarrow G_1^2, U_2 \leftrightarrow G_1^2, U_3 \leftrightarrow G_2^2, U_4 \leftrightarrow G_1^2, U_5 \leftrightarrow G_2^2\}$ is a one-to-one mapping of upper layer nodes (e.g., hosts) $\{U_1, U_2, U_3, U_4, U_5\}$ to its corresponding lower layer AGs.

Example 2. *Upper Layer HARM using an AG:* The AG in the upper layer of the HARM is $G_1^1 = (N_1^1, E_1^1)$. $N_1^1 = \{A, U_1, U_2, U_3, U_4, U_5\}$ is a set of all hosts. $E_1^1 = \{(A,U_1),(A,U_2),(U_1,U_3),(U_1,U_4),(U_2,U_3),(U_3,U_4),(U_4,U_5)\}$ is a set of all edges in G_1^1, where A is the attacker.

Example 3. *Lower Layer HARM using an AG:* The AG in the lower layer of the HARM to compromise W7 is $G_1^2 = (N_1^2, E_1^2)$. $N_1^2 = \{W7_1, W7_2, W7_3, W7_4, W7_5, W7 : g, W7 : r\}$ is a set of all vulnerabilities and host states for hosts using W7 as the OS. $E_1^2 = \{(W7 : g, W7_1),(W7 : g, W7_2),(W7 : g, W7_3),(W7 : g, W7_4),(W7 : g, W7_5),(W7_1, W7 : r),(W7_2, W7 : r),(W7_3, W7 : r),(W7_4, W7 : r),(W7_5, W7 : r),\}$ is a set of all edges in G_1^2.



(a) Upper Layer of the HARM (b) Lower Layer of the HARM

Figure 2.5: 2-HARM of the example networked system using ATs

Figure 2.5 shows the 2-HARM of the example networked system using ATs in both layers. The upper layer captures the combinations of hosts compromised, and the lower layer captures the combinations of vulnerabilities required to compromise the OS, and they are generated manually to optimise their representations. The examples are shown below:

Example 4. *The HARM using only ATs:* The 2-HARM in Figure 2.4 is $H^{at} = (2, M^{at}, C^{at})$, where 2 is the number of layers, $M^{at} = \{U_5^1, U_5^2, T_2^2\}$ for U_5^1 is an AT capturing the combinations of hosts compromised in the upper layer, and U_5^2 and T_2^2 are ATs for compromising W7 and REL, respectively. $C^{at} = \{U_1 \leftrightarrow U_5^2, U_2 \leftrightarrow U_5^2, U_3 \leftrightarrow T_2^2, U_4 \leftrightarrow U_5^2, U_5 \leftrightarrow T_2^2\}$ is a one-to-one mapping of upper layer nodes (e.g., hosts) $\{U_1, U_2, U_3, U_4, U_5\}$ to its corresponding lower layer ATs.

Example 5. *Upper Layer HARM using an AT:* The AT in the upper layer of the HARM is $U_5^1 = (A_1^1, B_1^1, a_{1_0}^1)$, where $A_1^1 = \{U_1, U_2, U_3, U_4, U_5, AND_1, AND_2, AND_3, OR_1, OR_2\}$ is a finite set of hosts and gates, $B_1^1 = \{(AND_1, a_{1_0}^1), (U_4, AND_1), (U_5, AND_1), (OR_1, AND_1), (OR_2, OR_1), (AND_2, OR_1), (U_2, AND_2), (U_3, AND_2), (U_1, OR_2), (AND_3, OR_2), (U_2, AND_3), (U_3, AND_3)\}$ is a set of edges. $a_{1_0}^1$ is a goal of the attacker (i.e., to reach U_5).

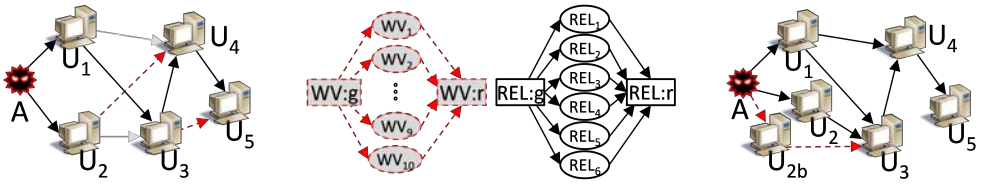
Example 6. *Lower Layer HARM using an AT:* The AT in the lower layer of the HARM to compromise W7 is $U_5^2 = (A_1^2, B_1^2, a_{1_0}^2)$, where $A_1^2 = \{W7_1, W7_2, W7_3, W7_4, W7_5, OR_1\}$ is a finite set of vulnerabilities and gates, $B_1^2 = \{(OR_1, a_{1_0}^2), (W7_1, OR_1), (W7_2, OR_1), (W7_3, OR_1), (W7_4, OR_1), (W7_5, OR_1)\}$ is a set of edges. $a_{1_0}^2$ is a goal of the attacker within hosts with W7 (i.e., to compromise the host by gaining the root privilege).

Any layers of the HARM can be exchanged in between (e.g., a 2-HARM consisting of AG in the upper layer as in Figure 2.4(a) with ATs in the lower layer as in Figure 2.5(b)). AGs can be generated in such a way described in [159], and ATs can be generated automatically as in [94].

2.3.2 Formalism of the HARM transformations

When there are changes in the networked system by means of network hardening (e.g., patching vulnerabilities, reconfiguration of the networked system), the HARM requires modifications accordingly to accurately evaluate the security. For example, a Moving Target Defense (MTD) continuously changes the attack surface of a networked system by means of *Shuffle*, *Diversity*, and *Redundancy* [92]. More descriptions on the MTD is presented in Chapter 5. Figure 2.6 shows examples of *Shuffle*, *Diversity*, and *Redundancy* in the HARM using AGs only. *Shuffle* based on changing the network topology shown in Figure 2.6(a) changes how

hosts are connected in the networked system, *Diversity* based on changing the OS shown in Figure 2.6(b) changes the vulnerability associated with hosts, and *Redundancy* based on replicating hosts are shown in Figure 2.6(c) changes the network topology and the number of hosts. These techniques cause variations in both hosts (i.e., by means of shuffle and redundancy) and vulnerabilities (i.e., by means of diversity), which are two major components in security models. Other network hardening techniques, such as patching vulnerabilities, and reconfiguring networked systems also affect how hosts and vulnerabilities are modelled. Hence, MTD techniques (i.e. *Shuffle*, *Diversity*, and *Redundancy*) are taken into account to demonstrate transformations of the HARM.



(a) Shuffle (b) Diversity (c) Redundancy

Figure 2.6: Deploying MTD Techniques in the Networked System

Shuffle: Shuffle changes the reachability of hosts in the networked system (e.g., network reconfigurations, virtual machine migrations in a virtualised systems), as shown in Figure 2.6(a). Hence, only the layers associated with the reachability information are modified. Such changes in the HARM are defined as follows:

Definition 4. Given an AG $G = (N, E)$ in a layer of the HARM affected by changes in reachability of hosts, an allowable modification (e.g., a *Shuffle*) transforms G to $G^* = (N, E^*)$, where $E^* \subseteq N \times N$ is any subset of edges.

Example 7. The upper layer AG in the HARM shown in Figure 2.6(a), is a transformation from $G_1^1 = (N_1^1, E_1^1)$ to $G_1^{1*} = (N_1^1, E_1^{1*})$, where $E_1^{1*} = \{(A, U_1), (A, U_2), (U_1, U_3), (U_2, U_4), (U_3, U_4), (U_3, U_5), (U_4, U_5)\}$.

Definition 5. Given an AT $T = (A, B, a_0)$ in a layer of the HARM affected by changes in reachability of hosts, an allowable modification (e.g., a *Shuffle*) transforms T to $T^* = \{A, B^*, a_0\}$, where $B^* \subseteq A \times A$ is any subnet of edges.

Example 8. The upper layer AT in the HARM is a transformation from $U_5^1 = (A_1^1, B_1^1, a_{1_0}^1)$ to $U_5^{1*} = (A_1^{1*}, B_1^{1*}, a_0^*)$, where $A_1^{1*} = \{U_1, U_2, U_3, U_4, U_5, AND_1, AND_2, AND_3, AND_4, OR_1, OR_2\}$, $B_1^{1*} = \{(AND_1, a_0^*), (U_5, AND_1), (OR_1, AND_1), (AND_2, OR_1), (AND_3, OR_1), (U_1, AND_2), (OR_2, AND_2), (U_2, AND_3), (U_4, AND_3), (U_3, OR_2), (AND_4, OR_2), (U_3, AND_4), (U_4, AND_4)\}$, and a_0^* is a new root node with an attack goal to compromise the WV.

Diversity: the reachability of the networked system is unchanged, as only a variant of the components is deployed with *Diversity*. Therefore, it only changes the layer using substitutions in the HARM. For example, substituting the W7 with WV (as shown in Figure 2.6(b) replaces the attack goal from compromising the W7 to compromising the WV. Such modifications in the HARM are defined as follows:

Definition 6. Given an AG $G = (N, E)$ in a layer of the HARM affected by changes in components of hosts or vulnerabilities, an allowable modification (e.g., a *Diversity*) transforms G to $G^* = (N^*, E^*)$, where N^* is a new set of nodes, $E^* \subseteq N^* \times N^*$ is any subset of edges.

Example 9. The lower layer AG in the HARM shown in Figure 2.6(b) is a transformation from $G_1^2 = (N_1^2, E_1^2)$ to $G_1^{2*} = (N_1^{2*}, E_1^{2*})$, where $N_1^{2*} = \{WV_1, WV_2, WV_3, WV_4, WV_5, WV_6, WV_7, WV_8, WV_9, WV_{10}, WV:g, WV:r\}$ and $E_1^{2*} = \{(WV:g, WV_1), (WV:g, WV_2), (WV:g, WV_3), (WV:g, WV_4), (WV:g, WV_5), (WV:g, WV_6), (WV:g, WV_7), (WV:g, WV_8), (WV:g, WV_9), (WV:g, WV_{10}), (WV_1, WV:r), (WV_2, WV:r), (WV_3, WV:r), (WV_4, WV:r), (WV_5, WV:r), (WV_6, WV:r), (WV_7, WV:r), (WV_8, WV:r), (WV_9, WV:r), (WV_{10}, WV:r)\}$.

Definition 7. Given an AT $T = (A, B, a_0)$ in a layer of the HARM affected by changes in components of hosts or vulnerabilities, an allowable modification (e.g., a *Diversity*) transforms T to $T^* = \{A^*, B^*, a_0^*\}$, where A^* is a new set of nodes, $B^* \subseteq A \times A$ is any subset of edges, and a_0^* is a new attack goal.

Example 10. The lower layer AT in the HARM is a transformation from $U_5^2 = (A_1^2, B_1^2, a_{1_0}^1)$ to $U_5^{2*} = (A_1^{2*}, B_1^{2*}, a_{1_0}^{1*})$, where $A_1^{2*} = \{WV_1, WV_2, WV_3, WV_4, WV_5, WV_6, WV_7, WV_8, WV_9, WV_{10}, OR_1\}$, $B_1^{2*} = \{(OR_1, a_{1_0}^{1*}), (WV_1, OR_1), (WV_2, OR_1), (WV_3, OR_1), (WV_4, OR_1), (WV_5, OR_1), (WV_6, OR_1), (WV_7, OR_1), (WV_8, OR_1), (WV_9, OR_1), (WV_{10}, OR_1)\}$, and $a_{1_0}^{2*}$ is a new attack goal to compromise WV.

Redundancy: Multiple replicas of network components are created with *Redundancy* (e.g., services, hosts or paths). As a result, capturing *Redundancy* may create new network components in the HARM. For example, U_2 is replicated to create U_{2b} as shown in Figure 2.6(c). Such modifications in the HARM are defined as follows:

Definition 8. Given an AG $G = (N, E)$ in a layer of the HARM affected by the additional network components, an allowable modification (e.g., a *Redundancy*) transforms G to $G^* = (N^*, E^*)$, where N^* is an extended set of nodes (i.e., $N \subseteq N^*$), and $E^* \subseteq N^* \times N^*$ is an extended subset of edges (i.e., $E \subseteq E^*$).

Example 11. The upper layer AG in the HARM shown in Figure 2.6(c) is a transformation from $G_1^2 = (N_1^2, E_1^2)$ to $G_1^{2*} = (N_1^{2*}, E_1^{2*})$, where $N_1^{2*} = N_1^2 \cup \{U_{2b}\}$, and $E_1^{2*} = E_1^2 \cup \{(A, U_{2b}), (U_{2b}, U_3)\}$.

Definition 9. Given an AT $T = (A, B, a_0)$ in a layer of the HARM affected by additional network components, an allowable modification (e.g., a *Redundancy*) transforms T to $T^* = \{A^*, B^*, a_0\}$, where A^* is an extended set of nodes (i.e., $A \subseteq A^*$), and $B^* \subseteq A \times A$ is a new subset of edges.

Example 12. The upper layer AT in the HARM is a transformation from $U_5^1 = (A_1^1, B_1^1, a_{1_0}^1)$ to $U_5^{1*} = (A_1^{1*}, B_1^{1*}, a_{1_0}^1)$, where $A_1^{1*} = A_1^1 \cup \{U_{2b}, OR_3, OR_4\}$, and $B_1^{1*} = \{(AND_1, a_{1_0}^1), (U_4, AND_1), (U_5, AND_1), (OR_1, AND_1), (OR_2, OR_1), (AND_2, OR_1), (U_2, OR_3), (U_{2b}, OR_3), (OR_3, AND_2), (U_3, AND_2), (U_1, OR_2), (AND_3, OR_2), (U_2, OR_4), (U_{2b}, OR_4), (OR_4, AND_3), (U_3, AND_3)\}$.

2.4 Tools and Methods to Generate Security Models

Various methods to generate graph-based security model are proposed in previous studies [100, 105, 158, 189]. Moreover, there are various commercial and publicly available tools, such as RedSeal [131], SkyBox [98], NuSMV [182], TVA [105], MulVAL [159], NetSPA [24], GARNET [205], NAVIGATOR [45], ADVISE [122], and CyberSAGE [189]. However, there is no well defined methods to generated tree-based security models (eg., an AT) efficiently, where available tools (e.g., SeaMonster [184], AttackTree+ [132], and SecuITree [130]) generate them manually. To improve this problem, a logic reduction technique can be

used, such as described in [94]. A comprehensive list of tools to generate security models is described in [85]. Most security models are generated on the basis of a single computer. However, a cloud computing resources can be also used to improve the performance of generating and evaluating security models [86].

2.5 Security Analysis using the HARM

Analysing the security of networked systems is shown in this section using the HARM defined in Section 2.3. System risk metric is used, and the example networked system shown in Figure 2.3 is analysed. The system risk, R_{system}, can be calculated as shown in Equation (2.1), which is an expression of a sum of risks associated with attack paths *path* in the networked system. The risks of each vulnerability are calculated based on their impact and exploitability as shown in Tables 2.4 and 2.5. Computing the equivalence of system risks using the 2-HARM and existing graph and tree based security models are shown in the following subsections respectively. Comparing the equivalence between existing security models and the HARM with more layers (e.g., a 3-HARM) can be found in [91].

$$R_{system} = \sum_{i \in path} R_i \tag{2.1}$$

2.5.1 Security Analysis of Graph-based Security Models

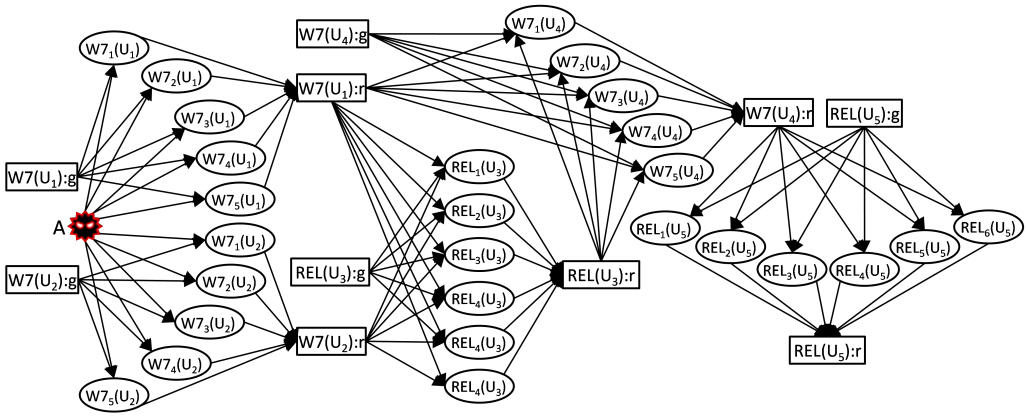


Figure 2.7: An AG of the Networked System

An AG representation of the example networked system is shown in Figure 2.7, where 2-HARM is previously shown in Figure 2.4. The AG express all hosts and vulnerabilities onto a single layer, which generates a complex view for users to comprehend when the size of the networked system is large. Even for a small sized networked system, there are total 1950 attack paths with the given AG. The risk calculated using the AG is 42683.62. For example, a risk associated with the attack path through exploiting $W7_3$ in U_1, $W7_5$ in U_4 and REL_4 in U_5 is the sum of risks associated with these vulnerabilities (i.e., $4.214 + 2.494 + 6.4 = 13.188$).

1: $HARM = \{h, M, C\}$
2: Risk R
3: No. of Attack Paths AP
4: **procedure** EVALUATE_HARM_RISK($HARM$)
5: **for** $i = 1 \ldots h$ **do**
6: **for all** $model \in M_i$ **do**
7: R_{model}, AP_{model} =solve_Risk($model$)
8: $R_{n_j} \leftarrow R_{model} \mid n_j \leftrightarrow M_i^{n_j} \in C$
9: $AP_{n_j} \leftarrow AP_{model} \mid n_j \leftrightarrow M_i^{n_j} \in C$
10: **end for**
11: **end for**
12: `Return` R_{M_h}, AP_{M_h}
13: **end procedure**
14:
15: **procedure** SOLVE_RISK(m)
16: `Compute All Possible Attack Paths` $path$ `of` $model$
17: $R_{model} \leftarrow$ sum of risk in $path$
18: $AP_{model} \leftarrow \mid path \mid$
19: `Return` R_{model}, AP_{model}
20: **end procedure**

Figure 2.8: System Risk Computation Algorithm Using the HARM

Using 2-HARM, a bottom-up approach is used as described in [91]. Figure 2.8 shows the algorithm to calculate system risks for graph-based HARMs. For example, the lower layer of U_3 is calculated as the sum of each vulnerability (i.e., $1.421 + 2.494 + 10 + 6.4 + 2.9 + 10 = 33.215$), and the upper layer is evaluated by taking into account the risk computed in the lower layers and the number of attack paths associated with them (i.e., there are five different attack paths exploiting

hosts with W7, and six different attack paths exploiting hosts with REL). So, an attack path through U_1, U_4, and U_5 equates to $(28.108 \times 5 + 28.108 \times 5) \times 6 + 33.215 \times 25 = 2516.855$. Similarly, the system risk computed using the 2-HARM is also 42683.62 (same with using the AG).

2.5.2 Security Analysis of Tree-based Security Models

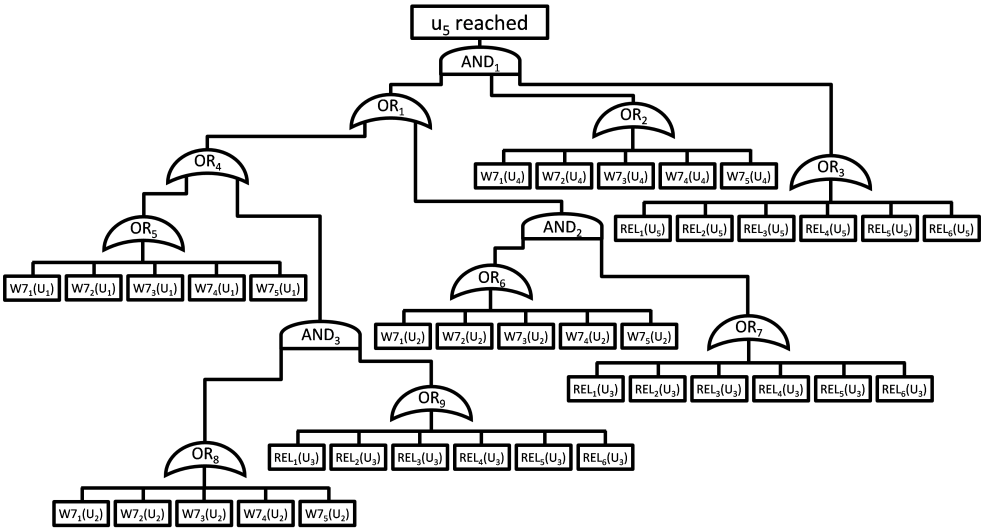


Figure 2.9: An AT of the Networked System

An AT representation of the example networked system is shown in Figure 2.9, where 2-HARM is previously shown in Figure 2.5. The AT (and other tree-based security models) expresses combinations of events in the networked system to achieve the attack goal specified (e.g., to reach U_5). Similarly with the AG, there are total 1950 possible attack events to achieve the given attack goal (e.g., there are 30 combinations at AND_3 gate, and so forth). The system risk based on the tree structure can be evaluated as in [172]. Because tree-based security models have different computational methods for security metrics compared to graph-based security models, resulting values are different. For example, the system risk of the example networked system using the AT equates to 37.2, which is significantly different to graph-based security models with 42683.62. However, computational methods can be integrated to calculate the same result as long as the same security information is given, but that is out of scope in this thesis.

The algorithm shown in Figure 2.8 can also be used to calculate system risks using tree-based HARMs. For example, the lower layer of U_3 is calculated to be 10.0 (the highest value is selected based on the $OR-gate$), and the upper layer is evaluated based on the risks computed for each host. So, events cumulated at OR_4 gate equates to 18.6 (OR_5 and OR_8 gates equate to 8.6, OR_9 equates to 10, and AND_3 gate equates to 18.6). Similarly, the system risk computed using the 2-HARM is also 37.2 (same with using the AT).

2.6 Summary

This chapter presented the HARM and its formalisms. The generation methods are described for both graph and tree based security models, and an equivalent security analysis is validated. This chapter concludes the background knowledge of the HARM, which forms the theoretical basis for the experiments and analyses presented throughout the remainder of the document.

Part II

Comparative Analysis of Security Models

Chapter 3

Complexity Analysis

The aim of this chapter is to analyse the computational complexities of security models in terms of phases in the lifecycle of security models shown in Section 2.2. Tasks performed in each phase of the lifecycle are different, so the complexities of methods used by different security models vary in each phase of the lifecycle. Because networked systems are growing, it is essential that security models are scalable. Conducting a complexity analysis hence provides a comparative indication for practicability of security models by analysing their theoretical performances. An early study on the scalability of security models based on complexity analysis for security models developed prior to 2005 are shown in [124], but it only focused on the generation and evaluation phases. Moreover, it only considered graph-based security models. For tree-based security models, there is a lack of efficient automated generation methods [94].

In this chapter, a comprehensive complexity analysis is conducted for each phase in the lifecycle of both graph-based and tree-based security models. We have chosen the AG and AT, and the analysis compares them with respect to the complexities of the HARM. An AG and an AT are chosen for representatives of graph-based and tree-based security models respectively, because they are (i) the most widely used security models, and (ii) the simplest form of any security models (i.e., other security models are extensions to an AG or an AT) [85]. In addition, a few security models, such as the Multiple Prerequisite Graph (MPG) [100], Logical Attack Graph (LAG) [158], and Two-Layer Attack Graph (TLAG) [207], are described with their complexities in terms of phases in the lifecycle. The complexity analysis focuses on the number of hosts and vulnerabilities, as they are the major scaling factors for security models in the networked system. In order to conduct the worst case analysis (i.e., to compute the upper bound complexity), the reachability of hosts in the network system is assumed with a fully connected topology (i.e. a complete graph). In addition, the preprocessing phase is not taken

into account in this analysis because it focuses on gathering security information that is usually a single step operation prior to generation of the security model.

3.1 Complexity Analysis of the Generation Phase

In this section, n and m represent the total number of hosts (nodes) and the average number of vulnerabilities in each host, respectively.

3.1.1 Generating AG, AT, and HARM

The structure of the AG is a series of connections between vulnerabilities in the networked system based on host reachability and conditions required to execute exploits (e.g., such as shown in Figure 2.7). Vulnerabilities in each host may have up to $m-1$ connections to other vulnerabilities within the host as a subsequent exploit (e.g., privilege escalations). As a result, there are total $m(m-1)$ number of possible vulnerability connections to other subsequent hosts in the networked system (i.e., edges between vulnerabilities from a host to all other vulnerabilities in other hosts). For each host, there are up to $n-1$ number of possible connections to other hosts. Hence, there are a total $n(n-1)$ number of possible host connections. Therefore, the computational complexity of the generation phase for the AG is $O(m^2n^2)$.

The AT consists of a set of grouped attacks (or grouped vulnerabilities that can be exploited) that satisfies the goal of the attacker (e.g., as shown in Figure 2.9). Since there is no defined automated generation method for ATs, a naive method that generated all possible attack paths to populate events in the AT are taken into account. Hence, for m number of vulnerabilities, there are $O(m!)$ number of possible exploit sequences (i.e., there is an exponential number of possible exploit sequences with given vulnerabilities). Similarly, for n number of hosts, there are $O(n!)$ number of possible attack paths. Therefore, the computational complexity of the generation phase for the AT is $O(m!n!)$.

The structure of the HARM is based on the security model used in its layers. First, if an AG is taken into account for both layers in HARM (e.g., a 2-HARM), the computational complexity of generating the upper layer is given by $O(n^2)$. For each host, there are up to $n-1$ connection(s) to other hosts, which results in $n(n-1)$ number of connections for all host pairs. In the lower layer, the computational

complexity is given by O(m^2n). For each vulnerability, there are up to $m-1$ number of other subsequent vulnerabilities, and this process is applied to n number of hosts. Therefore, the computational complexity of generating the HARM with AGs in its layers is O(m^2n+n^2).

Second, if the AT is taken into account for both layers in the HARM, the computational complexity of generating the upper layer is given by O($n!$). For n number of hosts, there is an O($n!$) number of possible attack paths. The computational complexity in the lower layer is given by O($m!n$). For m number of vulnerabilities, there is an O($m!$) number of possible exploit events for all hosts. Therefore, the computational complexity of generating the HARM with ATs in its layers is O($m!n+n!$).

3.1.2 Generating Other Security Models

MPG: Ingols *et al.* [99, 100] used a network security analysis tool named *Network Security Planning Architecture* (NetSPA) to evaluate the security of networked systems, but with an AG as the underlying security model could not analyse the security of more than 17 hosts in the networked system. MPG is then proposed and incorporated into the NetSPA, which improved the scalability of generating and evaluating the security. In the MPG, there are three types of nodes: (1) state, (2) prerequisite, and (3) vulnerability instance. A *state* node represents the access level of the attacker on a particular host, and hence it also represents hosts (i.e., n). A *prerequisite* node represents the reachability groups or credentials that capture the possible attack paths between hosts and vulnerabilities. A *vulnerability instance* node represents a particular vulnerability on a port of a particular host (i.e., m).

The structure of the MPG is defined such that all state nodes reach prerequisite nodes, prerequisite nodes reach vulnerability instance nodes, and vulnerability instance nodes reach state nodes. For a fully connected networked system with a single reachability group, there are n number of hosts reaching the prerequisite node, m number of outbound edges from the prerequisite node to all vulnerability instance nodes, and mn number of outbound edges from all vulnerability instance nodes to all state nodes. Therefore, generating the MPG has the computational complexity of O(mn). If there are more reachability groups (e.g., each prerequisite

node reaches a single host, resulting n number of prerequisite nodes), then the worst case will be equivalent with generating AGs with a complexity O(m^2n^2). Nevertheless, it is clear the structural form of the MPG enhances the performance of generating security models compared to AGs.

LAG: Ou *et al*. [158] used a LAG that generates a linearly proportional number of model components with respect to the networked system. It is generated with a polynomial computational complexity algorithm, where there are two types of nodes: (1) derivation nodes, and (2) fact nodes (where a fact node further categorised into a primitive fact node or a derived fact node). A *derivation* node describes how fact nodes can be realised (e.g., exploit vulnerabilities (derivation) to compromise a host (fact)). A *primitive fact* node provides the configuration information (i.e., existing vulnerabilities found). A *derived fact* node is computed from the configuration information that captures the attack path information (i.e., vulnerabilities used in a sequence). Hence, derivation nodes describe states, and fact nodes describe vulnerabilities in the networked system.

The structure of the LAG has derivation nodes satisfied by fact nodes, where derivation nodes directed from the fact nodes form a disjunction and the fact nodes directed from the derivation nodes form a conjunction. For a fully connected networked system, there are m number of inbound edges for all derivation nodes (e.g., state of hosts) from fact nodes (e.g., vulnerabilities), which has a computational complexity of O(mn^2). Also, there are n number of inbound edges for all fact nodes from all derivation nodes, which has a computational complexity of O(mn^2). Hence, generating the LAG has the computational complexity of O(mn^2), which has a better upper bound computational complexity than the AG.

TLAG: Xie *et al*. [207] used a Two-Layer Attack Graph (TLAG) that captures the host information in a form of an AG in the upper layer, and vulnerability information in a form of an AG in the lower layer. Many properties of the TLAG are similar with the 2-HARM with AGs in its layers, but one main difference is that the TLAG stores the lower layer information on the edges in the upper layer, whereas the HARM stores the lower layer information on the hosts in the upper layer. This difference is not observed in the generation phase, which has the same computational complexity as the HARM with O($m^2n + n^2$). However, this difference becomes apparent in the evaluation phase in the lifecycle.

3.2 Complexity Analysis of the Representation Phase

3.2.1 Representing AG, AT, and HARM

Visualising the security helps understanding about security level of the networked system. The complexity of the representation phase is determined by the number of components in the security model (i.e., by their sizes). Once an AG is generated, the simplest form consists of a linearly proportional number of network components (e.g., hosts and vulnerabilities). There are series of edges that reflect the relationships between these components, such as an example shown in Figure 2.7. Taking into account the number of hosts and vulnerabilities, the size complexity of the AG is O($n + nm$) (or O(nm) for AGs based only on vulnerabilities). On the other hand, an AT may consists of duplicated events, such as in Figure 2.9. Using the naive approach described in Section 3.1, there are an exponential number of events (i.e., vulnerabilities), which results in the size complexity of O($(nm)!$). Manually generating the AT, as well as using logic reduction techniques [94], may reduce the size of the AT.

In the case of the HARM that uses AG in its both layers, the size complexity is inherited from the AG. The resulting size complexity for the HARM is O($n+nm$), which is equivalent to the AG. Although it is not described in this analysis, there is a difference in the number of edges between the HARM and the AG. However, the number of edges is proportional to the number of nodes in security models, as it depends on relationships between those nodes. For the HARM using ATs in its layers, the same size complexity is inherited from the AT with O($n! + (nm)!$). Since the HARM also models hosts in the upper layer, it is larger than the AT that only captures the vulnerability information. A major difference between the HARM with AGs or ATs is the visualisation, where the HARM is a multi-layered model and, AGs and ATs are single-layered models. Consequently, visualising all components (such as AGs and ATs) are difficult to comprehend [84], whereas visualising in a structured layer provides a better view of the system and its security postures [151].

3.2.2 Representing Other Security Models

The numbers of components in the MPG, LAG, and TLAG are linearly proportional to the number of hosts and vulnerabilities in the networked system (i.e., MPG with the size complexity of O($n+nm$) in tools such as NAVIGATOR and GARNET [45, 205], LAG with the size complexity of O($n+nm$) in the MulVAL tool [159], and TLAG with the size complexity of O($n+nm$) representation in [207]). Both MPG and LAG are single-layered security models, where TLAG is a two-layered security model.

3.3 Complexity Analysis of the Evaluation Phase

3.3.1 Evaluating AG, AT, and HARM

There are various evaluation methods to analyse different security metrics. However, these methods are often model-centric (i.e., only applicable to the specific security model), or have different interpretations of the same security metrics (e.g., qualitative or quantitative system risk analysis?). Hence, the evaluation will be focused on all possible attack scenarios (e.g., all possible attack paths or events) in the networked system, which can be computed using any security models. In order to compute all possible attack scenarios, the attacker location is assumed to be outside of the networked system and a single target host is assigned (i.e., any host in the networked system, as it is a fully connected network).

First, an AG is taken into account. The number of paths between two hosts are in O($m!$), because there are m number of vulnerabilities and with the maximum attack path length of m (i.e., exploits all vulnerabilities) there are $m!$ number of choices to exploit those vulnerabilities. Such computations are carried out up to $n-1$ number of host pairs (i.e., all hosts are compromised), which results in O($m!^n$). This specifies the computational complexity of a single attack path, and there are total O($n!$) number of attack paths consisting of n hosts. Therefore, the computational complexity of the AG in the evaluation phase is O($m!^n n!$)

Second, an AT is taken into account. The evaluation of AT is given by the total number of events in the AT, because the structure of the AT already captures all possible attack events (i.e., the evaluation phase is dependent on the size of the AT). The size complexity of the AT is O($(nm)!$) and hence, evaluating the AT also

requires computing O($(nm)!$) number of events.

Lastly, the HARM is taken into account. The evaluation of the HARMs is divided into each layer (e.g., a bottom-up evaluation method). In the case of an AG in the layers of the HARM, evaluating upper and lower layers have computational complexities of O($n!$) and O($m!$) respectively. As a result, the computational complexity of the HARM in the evaluation phase is O($m!n!$). Further, if all hosts have the same lower layer information (e.g., homogeneous networked system), then it further simplifies the computational complexity to O($m!n+n!$), because the lower layer can be done only once for all hosts. In case of an AT in the layers of the HARM, evaluating upper and lower layers have computational complexities of O($n!$) and O($m!$) respectively, with the total computational complexity of O($m!n!$). Similarly with the HARM with an AG in its layers, computational complexity of O($m!n+n!$) is observed when the lower layer information is the same for all upper layer nodes (e.g., hosts).

3.3.2 Evaluating Other Security Models

To enhance the performance of evaluating the security of the networked system, different model-specific evaluation methods are proposed. MPG [100, 205] and LAG [84, 158] developed graph simplifications and clustering techniques to reduce the size complexity of their security models, while TLAG used adjacent matrix evaluation method to compute the overall security of networked system. However, evaluation using these specified methods limit the scope of security analysis specific to security metrics used (i.e., other security metrics cannot be evaluated using the same methods), as well as restricting the usage of other security models (e.g., cannot validate the result without other security models providing the same methods). The MPG and LAG have a similar structure as the AG, with the worst case performance (e.g., MPG with a single reachability group) of O($m!^n n!$). TLAG has a similar computational complexity as the HARM with AGs in its layers but since the lower layer models are stored in edges, the resulting computational complexity in the evaluation phase is O($m!n^2n!$) (i.e., there are O(n^2) number of edges in the upper layer). As a result, TLAG (O($m!n^2n!$)) is less efficient than the HARM (O($m!n!$)).

3.4 Complexity Analysis of the Modification Phase

3.4.1 Modifying AG, AT, and HARM

An update event (e.g., addition, reconfiguration and deletion of hosts or vulnerabilities) results in modifications in the networked system. The computational complexity in the modification phase calculates the number of changes made to security models. There may be multiple updates at the same time, but only a single update event is considered for simplicity.

First, the AG is taken into account. In the case of an updated vulnerability information, a vulnerability can be connected to m number of other vulnerabilities in a host, and there are $n-1$ number of possible hosts that can reach the vulnerability. The total number of components affected by the vulnerability update is in $O(m+n)$. In the case of an update in the host information, there are m number of vulnerabilities associated with a host, and there are $m(n-1)$ number of vulnerabilities reachable from the host. The total number of components affected by the host update is in $O(m+m(n-1))$. Hence, the worst case complexity of update in the networked system for the AG is in $O(mn)$ (i.e., $O(m+m(n-1)) \cong O(mn)$).

Second, the AT is taken into account. An update event in the AT requires affected event groups to be modified accordingly. On the basis of naively generated AT, the number of affected events in the AT is in $O(m!(n-1)!)$, because there are $m!$ number of vulnerability events associated with a host and these events are associated with $(n-1)!$ number of attack scenarios with the host present. In case of update in the vulnerability information, the number of affected events in the AT is in O((m-1)!n!), because there are $(m-1)!$ number of vulnerability events associated in a host, and there are $n!$ number of events associated with the host with updated vulnerability. Therefore, the total number of AT events affected by an update is in $O(m!n!)$ (i.e., $O(m!n!) \cong O(m!(n-1)!) \cong$ O((m-1)!n!)).

Lastly, the HARM is taken into account. The computational complexity in the modification phase for the HARM is an addition of computational complexities in each layer. If an AG is taken into account in the layers of the HARM, it results in $O(n+m)$ and $O(m)$ for computational complexities in upper and lower layers respectively. There are only n number of hosts affected, and vulnerability information is only affected for the host. For a vulnerability information update, only the lower layer is affected. Therefore, the total computational complexity of the

HARM with AGs is in O($n+m$). If ATs are taken into account in the layers of the HARM, the computational complexities are O($n!+m!$) and O($m!$) for upper and lower layers respectively. Similarly with AGs in the HARM, only updated components in each layer are affected with their corresponding lower layers. Therefore, the total computational complexity of the HARM with ATs is in O($n!+m!$).

3.4.2 Modifying Other Security Models

There is a lack of study on adaptability of existing security models when there are changes in the networked system. Consequently, regeneration is taken into account when there are changes in the networked system. Taking into account the worst case modification phase computational complexity for the AG is in O(mn), regeneration of MPG (O(m^2n^2)) or LAG (O(mn^2)) are significantly inefficient, and also with the HARM with O($n+m$) in comparison to regenerating TLAG (O(m^2n+n^2)).

3.5 Structural Advantages of the HARM

Each layer in the HARM can be generated in parallel. Using distributed systems, such as the Cloud, we can subdivide the HARM by each layer and generate them in parallel, and join them to generate the HARM as a single security model. Existing security models (e.g., MPG, LAG, TLAG) can also be generated in parallel (e.g., by dividing them into sub-models), but dividing and joining processes may require additional preprocessing to find independent components.

Layer-centric analysis can also be performed using the HARM. For example, computing characteristics of the network system (e.g., such as in [9]) only requires the network layer information. In addition, an additional analysis applicable in different layers can be performed using the HARM, such as a performance analysis on the network layer or severity analysis of vulnerabilities in the lower layer. Therefore, a wide range of constraints can be taken into account, not just security and costs, but as well as the performance and other requirements when analysing the security of networked systems.

3.6 Conclusions

Complexity analysis gives insight to possible performances of security models. This chapter focused on the complexity analysis of security models, which showed the worst case efficiency of these models in terms of their lifecycle. This analysis clearly shows that some of the existing security models are not capable of assessing the security of large sized networked systems due to their exponential complexities. Table 3.1 summarises the computational complexities of security models, namely the HARM, AG, AT, MPG, LAG, and TLAG. Qualitative complexity analysis for other security models can be found in [85]. MPG has the best computational complexity in the generation phase, but the HARM has the best computational complexities in the evaluation and modification phases. The AT has exponential complexities in all phases, which indicates the lack of efficient methods of using the AT. Chapter 4 therefore focuses on analysing the performance of graph-based security models.

Table 3.1: Computational Complexities of Phases in Security Model Lifecycle

	Generation	Representation	Evaluation	Modification
HARM (AG)	$O(m^2n+n^2)$	$O(n+mn)$	$O(m!n!)$	$O(m+n)$
HARM (AT)	$O(m!n+n!)$	$O(n!+(mn)!)$	$O(m!n!)$	$O(m!+n!)$
AG	$O(m^2n^2)$	$O(mn)$	$O(m!^n n!)$	$O(mn)$
AT	$O(m!n!)$	$O((nm)!)$	$O((mn)!)$	$O(m!n!)$
MPG	$O(mn)$	$O(n+mn)$	$O(m!^n n!)$	$O(mn)$
LAG	$O(mn^2)$	$O(n+mn)$	$O(m!^n n!)$	$O(mn^2)$
TLAG	$O(m^2n+n^2)$	$O(n+mn)$	$O(m!n^2n!)$	$O(m^2n+n^2)$

Chapter 4

Scalability Analysis

There is an emerging scalability problem with existing security models as the size of the networked systems becoming larger, especially when analysing all possible attack scenarios. Chapter 3 presented the worst case computational complexity analyses based on fully connected topology, but real life networked systems run on various network topologies, and other factors that affect the overall performances of security models. In this chapter, the scalability of existing security models is evaluated and compared with the HARM in realistic scenarios. Two main tasks in this chapter are (1) formulating key questions that need to be answered to assess the scalability of security models, and (2) evaluate and compare the scalability of security models using simulations.

Both graph-based and tree-based security models suffer the scalability problem, but there is also no automated generation method that captures all possible attack events for tree-based security models. As a result, it is difficult to assess the scalability of tree-based security models. Consequently, this chapter only focuses on assessing the scalability of graph-based security models, namely the AG, MPG, LAG, TLAG, and HARM with AGs in its layers. Previous studies proposed structural modifications [100, 158, 207] and heuristic methods [43, 84, 167]. But as described in Chapter 3, structurally modified security models suffer the scalability problem when the size of the networked system becomes very large [87, 91, 124, 151]. In the case of heuristic methods, such as graph simplifications [43, 84], there may be a loss of security information that may result in misguided security assessment. Therefore, performance analysis of security models with respect to their scalability is conducted for evaluating all possible attack scenarios.

4.1 Key Questions to Compare Scalability

Scalability is a growing concern for security assessment as it becomes difficult to manage the size of security models when the networked system becomes too large. Existing security models and their studies lack in comparative analysis to show the scalability of security models in various environments and attack scenarios. To address this problem, five key questions are formulated to compare the scalability of security models:

Q1 Was the computational complexity analysis performed?

Q2 Was the security model compared with other security models?

Q3 Were different network topologies considered?

Q4 Were the effects of variable number of vulnerabilities for hosts considered?

Q5 Were the different types of vulnerabilities (e.g., user and root) considered?

Scalability in the generation and the evaluation phases are taken into account for analysing the performance of security models, where answers to the key questions are shown in Table 4.1 and 4.2 for generation and evaluation phases respectively. The preprocessing phase generally requires the same efforts for all security models (i.e., information gathering in the networked system), and the representation phase does not involve any computational methods (i.e., only for storage and visualisation purposes). Performance of the modification phase will be given in Chapter 5.

Table 4.1: Answers to Key Questions in the Generation Phase

Security models	AG	TLAG	LAG	MPG	HARM
Q1	Yes	Yes	Yes	Yes	Yes
Q2	Yes	No	Yes	Yes	Yes
Q3	No	No	Yes	No	Yes
Q4	No	No	Yes	No	Yes
Q5	No	No	No	Yes	Yes

4.1.1 Generation Phase

The main task of the generation phase is to retrieve the network information and generating the relationships between network components specific to the requirements of security models (e.g., connecting a vulnerability node to its subsequent vulnerabilities or hosts based on the reachability, application, and port information).

Generating AG: Computational complexity of generating the AG has been conducted in [124], and it is compared against the HARM in [87]. However, various network topologies are not taken into account when generating AGs. Furthermore, the effect of variable number of vulnerabilities, as well as different types, are not considered.

Key properties of generating the AG is that the connections between vulnerabilities and hosts in the AG are independent. As a result, the computational complexity of generating the AG is greater than the HARM as shown in [87]. Moreover, because there are a larger number of edges, traversing the AG to compute all possible attack paths have worse computational complexity than the HARM [88].

Generating TLAG: Only the computational complexity of generating the TLAG is shown in [207]. There is no comparison with other security models in terms of performance, and only a fixed network topology was taken into account, which had a fixed number of vulnerabilities with the same properties (i.e., homogeneous).

Generating the TLAG is not described in [207], but if the same generation method as the HARM is assumed, it would have the same computational complexity as the HARM. However, the number of lower layer models is determined based on the number of host pairs (i.e., edges in the upper layer), which has the upper bound of $O(n^2)$. This is greater than the HARM with the upper bound of $O(n)$.

Generating LAG: Computational analysis of the LAG is conducted in [158], which is also compared against the AG in terms of generating both LAG and AG. Various network topologies are also taken into account, with a varying number of vulnerabilities. However, different types of vulnerabilities are not taken into account.

The LAG has a generation complexity of $O(\delta n)$, where n is the number of

hosts in the networked system and δ is the time to find the host in the lookup table [158]. All vulnerabilities are assumed to be remote to exploits. Since each derivation node is an *AND* node, repeated nodes are required for each exploit if there are multiple sources it could be exploited from. If the derivation nodes are allowed as *OR* nodes, the number of repeated nodes (and the size of the LAG in the representation phase) will be reduced. Therefore, the HARM (linear size) has better size complexity than the LAG (polynomial size).

Generating MPG: Computational complexity of the MPG is conducted in [100], which also compared its performance with an AG while taking into account different types of vulnerabilities. However, a fixed network topology was used throughout experiments and a fixed number of vulnerabilities was used for each host.

The MPG has the number of components linearly proportional to the number of hosts and vulnerabilities in the networked system [100]. Additionally, it also requires the use of *prerequisite* nodes, which decreases the number of independent connections between hosts and vulnerabilities, but increases the size of the MPG. However, their experimental results showed that the number of total nodes in the MPG is negligible compared to the AG when generating the MPG. Their performance in the simulation showed almost linear relationship between the computational time for generating the MPG with respect to the number of hosts. In [99], the client-side attacks using the reverse reachability calculations are captured in the MPG as an additional feature.

Table 4.2: Answers to Key Questions in the Evaluation Phase

Security models	AG	TLAG	LAG	MPG	HARM
Q1	Yes	Yes	Estimated	Estimated	Yes
Q2	Yes	No	No	Yes	Yes
Q3	No	No	No	No	Yes
Q4	No	No	No	No	Yes
Q5	No	No	No	Yes	Yes

4.1.2 Evaluation Phase

Computing all possible attack scenarios is taken into account in the evaluation phase for reasons described in Chapter 3. Existing methods of assessing the security can be used (e.g., graph simplification [84, 100] and heuristic methods [7, 102]), but only specific attack scenarios and the subset of all possible attacks are considered. Matrix evaluation can be used to compute the overall security of the networked system, but it lacks in detailed analysis of the individual attack path.

Evaluating AG: Computational complexity of evaluating the AG and comparing its performance against the HARM is shown in [87,124] and [88] respectively. However, previous studies on the AG did not take into account the performance of evaluating the AG with various network topologies and vulnerability information.

The AG generates edges between vulnerabilities, where there are total $O(n^2m^2)$ number of edges. Hence, evaluating such graph becomes $O((n^2m^2)!)$, which is highly exponential. In contrast, the HARM captures relationship between vulnerabilities in the upper layer, reducing the total number of edges to $O(nm^2 + n^2)$. This is shown in Section 4.2 via simulations.

Evaluating TLAG: Only the computational complexity of evaluating the TLAG is presented in [207] based on matrix evaluation. There is no comparison with other security models, and only fixed topology and number of vulnerabilities are used in the experiment.

The evaluation of the TLAG computes the overall security of the networked system using the matrix evaluation with probability of an attack. However, this method lacks in assessing different attack paths and their effects. It is shown in [207] that the number of host-pair attack graphs (i.e., the number of lower layer security models) was not linearly proportional to the number of hosts, which is larger than the HARM with a linearly proportional number of lower layer security models.

Evaluating LAG: Evaluating the LAG is not shown in [158], and graph simplification and approximation algorithms are used to evaluate the LAG in [84]. Moreover, the performance is not compared with other security models, as well as using a fixed network topology with a fixed number of identical vulnerabilities.

If all possible attack scenarios are computed using the LAG, then the compu-

tational complexity of the LAG is equivalent with the AG. Because each fact node (e.g., hosts) makes an independent connection to derivation nodes (e.g., vulnerabilities), the conceptual structure of the LAG is identical to the AG. The number of paths from each fact node increases exponentially as the number of choices increases in the attack path, which is the same property found in the AG.

Evaluating MPG: Performance of evaluating the MPG uses a graph simplification method, which has almost linear scalability performance in respect to the number of hosts. This is compared with the AG in [100]. However, the performance analysis did not consider different network topologies or variable number of vulnerabilities that may affect how reachability groups are formed.

Evaluating the MPG utilises graph simplification. As a result, their evaluation complexity is almost linear with respect to the number of nodes. But taking into account computing all possible attack scenarios, the worst case scenario of evaluating the MPG is equivalent to the AG (e.g., consisting of only a single reachability group). If there are multiple prerequisite nodes, then connections between hosts and vulnerabilities are grouped by the prerequisite nodes, and it reduces the complexity in the evaluation phase. However, computing the optimal number of reachability groups is out of scope in this chapter.

4.2 Simulation Results

The HARM improves the efficiency of the security model by reducing the number of independent connections between hosts and vulnerabilities, which is shown in complexity analysis. Further performance analysis is conducted in this section to validate the improvements achieved using the HARM. The experiment is divided into two parts: (i) performance analysis using simple network topologies and (ii) performance analysis using combined network topologies.

4.2.1 Experiment 4A: Simple Network Topologies

The performance of the HARM is compared with the AG for various cases. The attack scenario used in the simulation is similar to that in the experiment conducted in [100]. The networked system used in the simulation is shown in Figure 4.1. A network has four sites (i.e., four identical networked systems connected via firewalls), where each site has three subnets DMZ, Internal Networks, and

Database such as in Figure 2.3. In each DMZ, there are five hosts and five administrative LAN hosts, and each Internal Network is divided into ten subnets with hosts connected with a bus topology. The port information and the firewall rules are abstracted.

Ten remote-to-other vulnerabilities are assigned to half of the hosts in each subnet, and the other half with one remote-to-root and nine remote-to-other vulnerabilities. The attack scenario was to compromise a host in the DMZ, an administrative LAN host, and all hosts in the network that has a remote-to-root vulnerability. Hosts that are not directly reachable from the attacker are compromised using other hosts as a stepping stones. The number of hosts in each subnet was increased to compare the scalability between the AG and the HARM. The comparison of scalability is shown in Figures 4.2 and 4.3 for generation and evaluation phases respectively. The simulation is conducted using an automated network simulation tool named *Akaroa2* [62, 164], where the results are collected with the confidence level of 0.95 and the relative error of 0.05. The simulation program was coded using Python, and it was conducted in a Linux environment with Intel(R) Core2 Quad CPU 2.66GHz with 3.24GB of RAM.

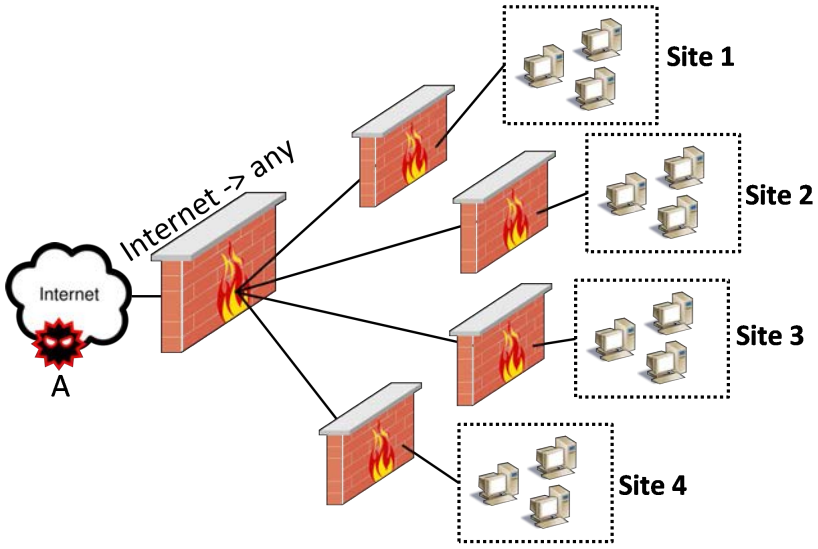
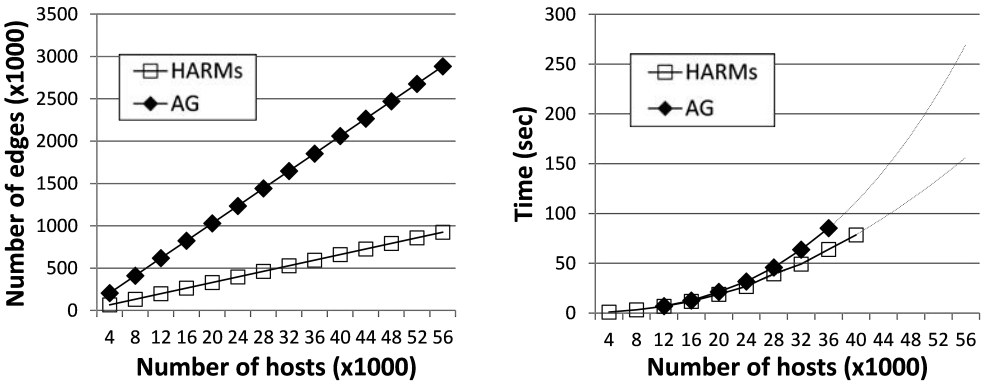


Figure 4.1: A Networked System for Experiment 4A

Performance Analysis with Fixed Variables: Figure 4.2 shows the performances of the AG and the HARM in the generation phase. This shows that the



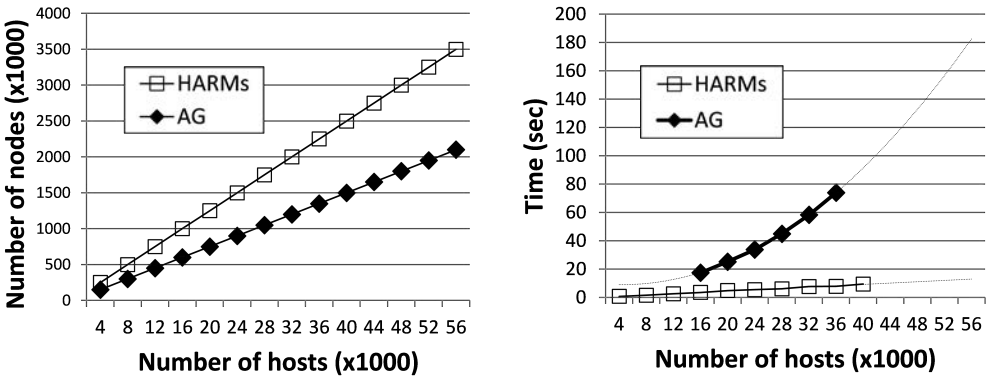
(a) Number of Edges (b) Generation Time Measurement

Figure 4.2: A Comparison between AG and HARM in the Generation Phase

number of edges in the AG increases quicker than the HARM. However, generation times for the AG and the HARM do not have a significant difference. This indicates that the number of edges has little influence on the generation time. Both security models have linear growth of the edge numbers, but the number of edges for the HARM was always less than that of the AG.

The trend observed from the simulation is comparable with the simulation result of the MPG [100]. The time comparison shows that the time for the evaluation increases rapidly for the AG, but almost linearly for the HARM as shown in Figure 4.3(b). In contrast, the number of nodes computed in the HARM is much greater than that of the AG. The AG constructs the attack paths using vulnerability sequences only, but the HARM also analyses the sequence of hosts. Therefore, an extra space of memory is required to store the information.

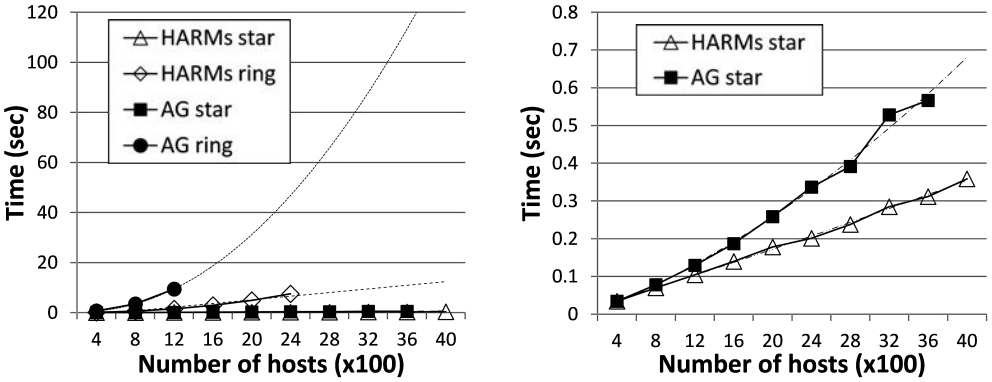
Performance Analysis with Non-fixed Variables: Various network topologies and variable number of vulnerabilities are taken into account. Bus (fully connected), ring, and star topologies are considered to connect hosts in each internal network, where the number of vulnerabilities for each host is varied from 10 to 150. The number of hosts is fixed at 1200 when simulating the variable number of vulnerabilities. For this experiment, the goal of the attacker is changed to compromise a single host selected in the one of the subnets in the Internal Network (e.g., a host in the 10th subnet in each internal networks). The bridging hosts (i.e., head hosts that connect to other subnets) are not selected as the target host. In order to



(a) Number of Nodes Computed (b) Evaluation Time Measurement

Figure 4.3: A Comparison between AG and HARM in the Evaluation Phase

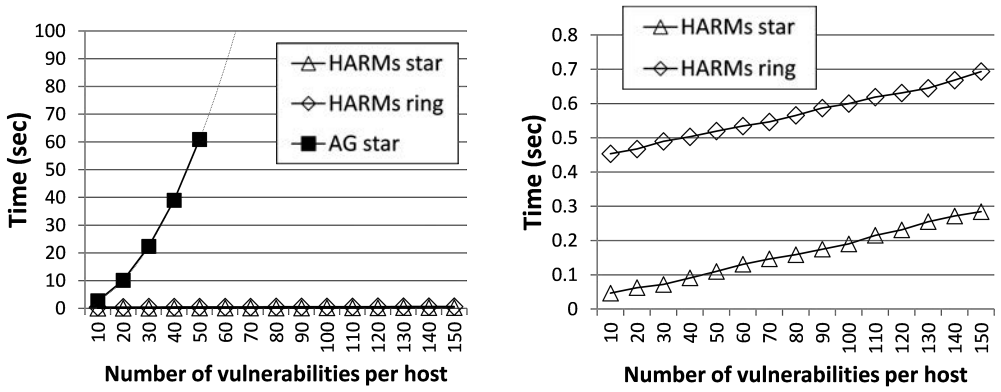
simulate different topologies, a single vulnerability to each host is assigned that is enough to gain the root access.



(a) Scalability of Different Network Topologies (b) Scalability of Star Topology

Figure 4.4: Scalability Difference of Network Topologies in the Evaluation Phase

The simulation result of different topologies is shown in Figure 4.4. Since the performances of generating the HARM and the AG are similar, only the differences in the evaluation phase are compared. Note that evaluating fully connected topology suffered from the scalability problem, where the evaluation of 400 hosts timed out (i.e., it took longer than three hours). However, the AG is significantly slower than the HARM when other topologies are taken into account.



(a) Evaluation Time (b) HARM Star and Ring Topologies

Figure 4.5: Scalability Difference with Varying Number of Vulnerabilities

The simulation result of varying the number of vulnerabilities is shown in Figure 4.5. The number of hosts was fixed at 1200. The fully connected topology for both security models could not be evaluated for 1200 hosts. In addition, the ring topology for the AG reached the time out during the simulation (i.e., it took longer than three hours to evaluate). The comparison in the evaluation phase shows that as the number of vulnerabilities increases, the growth rate of the AG is much greater than the HARM for all network topologies. The performance increase is almost linear using the HARM for all topologies, indicating the number of vulnerabilities is also a constant factor in the evaluation phase.

4.2.2 Experiment 4B: Combined Network Topologies

In this section, the scalability of AG, 2-HARM and 3-HARM in terms of generation and evaluation are compared using a networked system with various number of hosts, topologies, vulnerabilities and network densities. The same networked system shown in Figure 4.1 is used.

Complex Network Topologies:

The number of hosts in the DMZ and Internal Network was increased to compare the scalability between the AG, 2-HARM and 3-HARM. Figure 4.6 shows that generating these models are almost equivalent because their generating algorithm is the same, which generates linearly proportional number of nodes in respect to the number of hosts in the networked system. In case of an evalua-

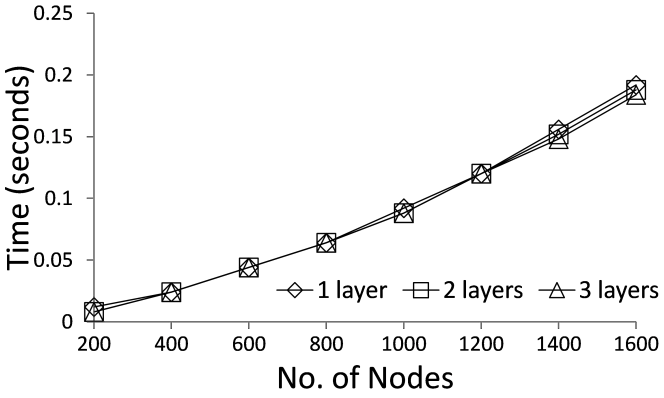


Figure 4.6: Generation of AG, 2-HARM, and 3-HARM

tion, four combined network topologies are considered: (i) star-star, (ii) tree-star, (iii) ring-star, and (iv) mesh-star, which are shown in Figure 4.7. Each combined topology creates a complex mesh structure as a result of mixing topologies. Figure 4.7(a) shows the performance of security models evaluating the star-star topology. It shows that all models have a similar trend in performance, but the 3-HARM outperforms the AG and 2-HARM. Similarly, in Figure 4.7(b), 4.7(c) and 4.7(d), the 3-HARM performs the best. Also, an observation is that the evaluation time is significantly increased for the ring-star and mesh-star topologies. Different performance of different network topologies are presented in Figure 4.8. It shows that as the network density increases, the time taken to evaluate the network increases. The network density is the measure of network connections between hosts in the networked system (i.e., a normalised degree centrality measure, which measures the number of edges).

Varying Number of Vulnerabilities:

An experiment is conducted to observe the effect of varying number of vulnerabilities as shown in Figure 4.9. Figure 4.9(a) shows that it affects the AG significantly compared to the HARM, where it is almost negligible to notice any changes in the HARM. Because the AG creates connections between individual vulnerabilities, the time taken to evaluate becomes significantly longer as the number of vulnerabilities is increased (as shown in Figure 4.5(a)). A larger number of vulnerabilities are used in Figure 4.9(b) to distinguish the performance difference between 2-HARM and 3-HARM. The result shows that the evaluation time for the

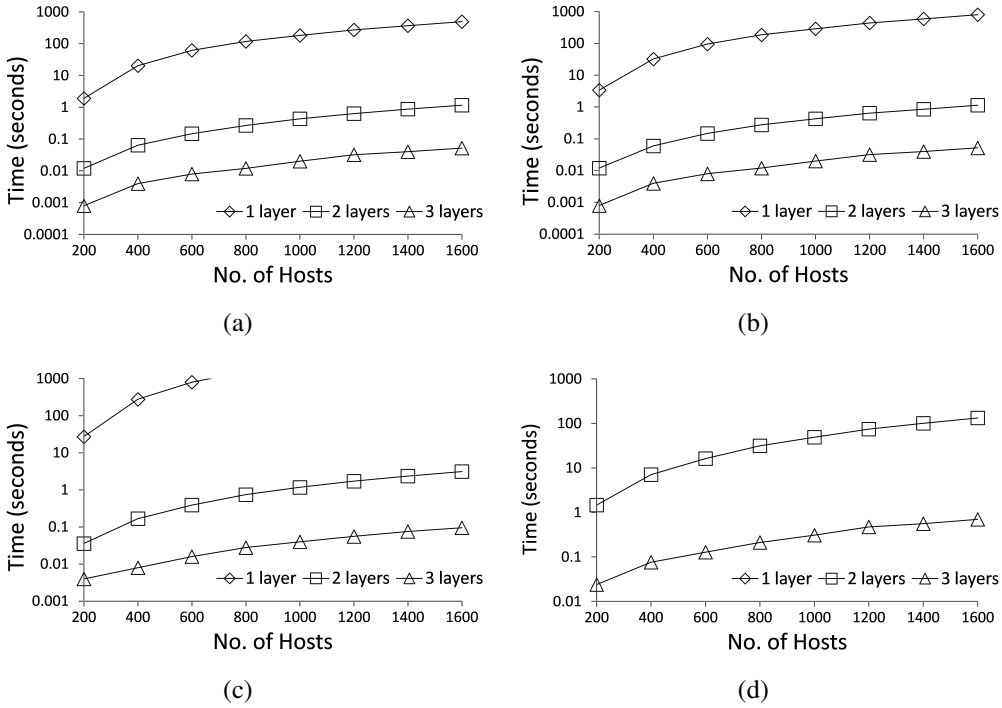


Figure 4.7: Performance of evaluating combined network topologies in the order of star-star, tree-star, ring-star and mesh-star topologies respectively

2-HARM is increasing gradually, whereas that of the increase of the 3-HARM is relatively constant.

Varying Network Density: A further simulation is conducted to investigate the effect of network density when analysing the security of networked systems. For this experiment, a mesh topology is used with various network density values. Figure 4.10 shows the performance of AG (i.e., 1 layer), 2-HARM (i.e., 2 layers), and 3-HARM (i.e., 3 layers) with respect to the network density. Figure 4.10(a) shows that the performances of the AG and 2-HARM are significantly worse than the 3-HARM. Figure 4.10(b) shows that both AG and 2-HARM converges to the worst case performance when the network density is greater than 0.4. The proportion represents the performance proportionality compared to when density equals to 1. Also, the proportionality of 3-HARM converges to the worst case performance is significantly slower than the AG and 2-HARM. For the network density at 0.5, the AG reached 99.7% of the worst case performance, while 2-HARM reached 98.8%, and 3-HARM only reached 90.0%. It shows that modelling with

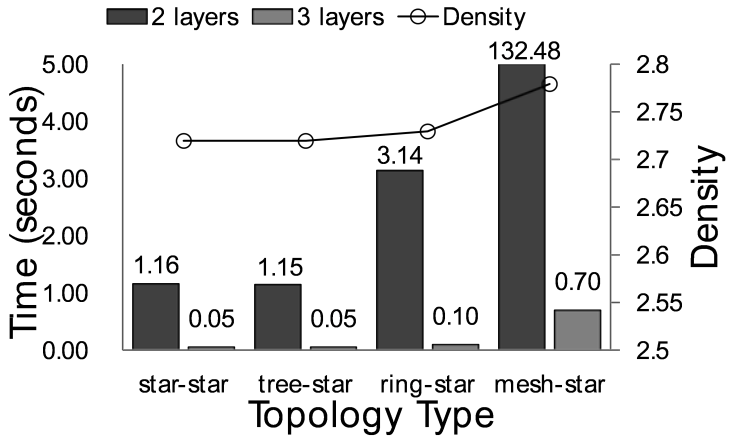


Figure 4.8: Performance of 2-HARM and 3-HARM Evaluating Various Network Topologies

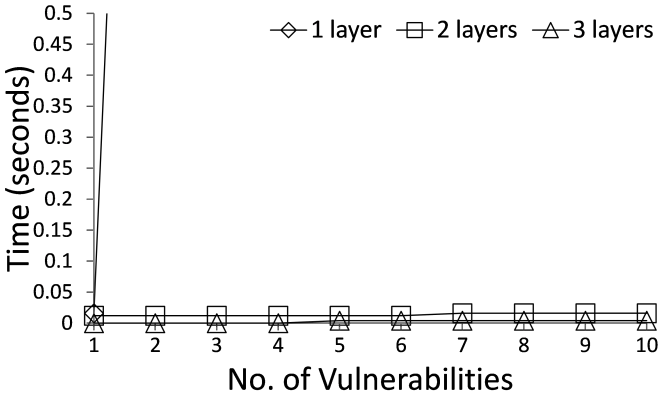
higher hierarchy may improve the performance of analysing the security of the networked system. However, finding the optimal number of layers for the HARM is out of scope in this chapter.

4.3 Discussions

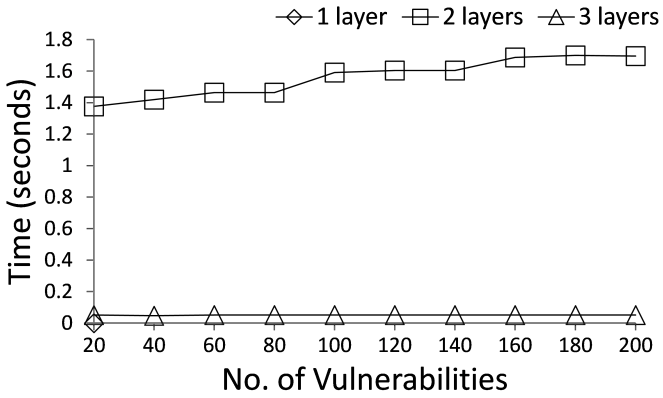
Key questions are listed to compare the scalability of security models, and the experimental results showed the efficiency of the HARM by answering these questions. The experimental results showed the efficiency of the HARM in comparison to the AG, especially the 3-HARM which is also much scalable than the 2-HARM (i.e., more scalable using more hierarchy). However, there is still a lack of understanding various attack scenarios which should be reflected in the security models. Some of these limitations are discussed in this section.

4.3.1 Scalability of Security Models in the Lifecycle Phases

Only a few studies conducted scalability analysis comparing the efficiency of various security models, and none of them considered the efficiency of security models in the modification phase. The similarity between the AG, LAG, and the MPG is that they are represented in a single layer. As a result, they suffer the scalability problem not only in the evaluation phase but also in the representation



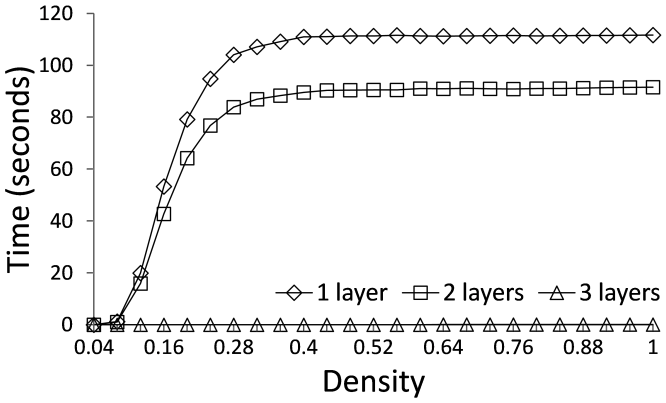
(a) 10 Max Vulnerabilities



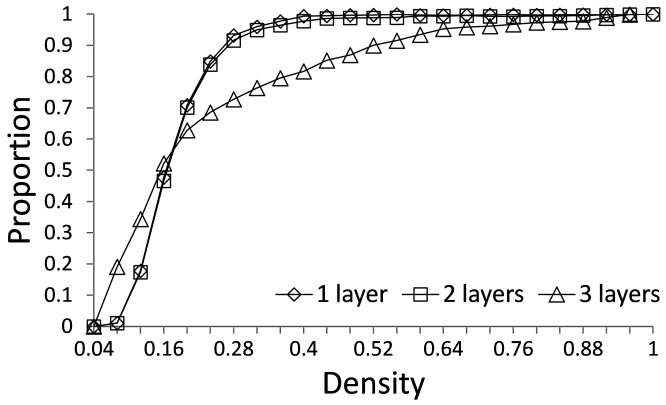
(b) 200 Max Vulnerabilities

Figure 4.9: Performance of AG, 2-HARM, and 3-HARM Evaluating Various Numbers of Vulnerabilities

phase (e.g., as shown in Figure 2.7). The AG suffered the scalability problem due to independent connections between the model components (e.g., vulnerabilities), where the representation of the AG had more edges compared with the HARM. The number of nodes was the same, but the number of edges was greater in the AG. On the other hand, security models using hierarchy (e.g., the HARM and TLAG) are less complex to represent. Moreover, in the case of updates in the networked system, single layered security models (e.g., AG, LAG and MPG) may affect many components in security models, whereas using hierarchy only affects specific layers and corresponding lower layers. However, there is a lack of scalability and adaptability analysis in the modification phase in the lifecycle of



(a)



(b)

Figure 4.10: Performance of AG, 2-HARM, and 3-HARM Evaluating Various Network Density

security models, which is presented in Chapter 3.

4.3.2 Network Structure and Attack Scenarios

In a real life networked system, network topologies are complex with many combinations of simple topologies (e.g., combinations of star, tree, ring and mesh topologies). The worst case complexity defined by analysing a fully connected topology gives the upper bound performance. Although a few complex network topologies are taken into account in the experiment, it is difficult to evaluate the scalability of security models. However, as shown in the density analysis in Section 4.2.2, the performance of security analysis can be estimated based on the

density of the networked system in respect to the worst case performance. Thus, the complexity analysis and experimental results are a reasonable estimation for the performances of using security models. Moreover, such assessment can be used to design networked systems that are secure as well as efficient to analyse the security to mitigate attacks.

4.3.3 Real Testbed Experiments

One of the limitations is a lack of experimenting on a real system. Although the observed performance in the simulation would be linearly proportional to the real systems, the complexities in real systems are difficult to estimate in a simulation. Also, other scalability factors are not taken into account (e.g., performance factors such as QoS, delay, and delivery rate), which may affect the performance of security analysis.

4.3.4 Comparisons with Other Security Models

The results obtained in the experiments were comparable with some of the existing security models and their analyses [100, 158]. The comparison between the HARM and the AG shows the performance of the HARM was always better than the AG. The variation of vulnerabilities affected the AG significantly, showing an almost exponential growth in the evaluation phase. In contrast, the HARM showed a linear growth of the evaluation time, which is practically computable for a large number of vulnerabilities. Because the underlying algorithms are the same (e.g., generation algorithm, full path search algorithm), the improvement of scalability comes from the structural advantages of the HARM.

A shortcoming is the lack of comparing the performances between the HARM and the TLAG. A network with homogeneous hosts will result in HARM and TLAG having the same number of upper and the lower layer components. However, the performances of these models are not analysed. A further comparison is required to distinguish the performance differences between the HARM and the TLAG, as well as MPG and LAG.

4.3.5 Differences between the AG and HARM

The experimental results show the HARM with better performances against the AG even when the same underlying security models and algorithms are used. Their performances in the generation phase were similar, but the AG showed that it created more edges than the HARM. Consequently, their performances in the evaluation phase showed that the AG had an exponential computational complexity while the HARM had a linear computational complexity with respect to the AG. The underlying algorithm to evaluate attack scenarios was the same, but the performance of the HARM is more efficient than the AG. Therefore, the structure of the HARM reduces the total number of edges in the security model, which resulted in fewer computations during the evaluation phase.

On the other hand, the number of memory space required in the representation phase for the HARM is greater than the AG, because the upper layer components of the HARM are also required in the evaluation. However, an extra memory space required by the HARM can be reduced with memory management.

4.3.6 Security Evaluation and Overhead

The complexity analyses and experimental results show a significant scalability improvement using the HARM over traditional security models (e.g., an AG). Using the hierarchy improves the performance because multiple computations can be grouped in the HARM, whereas a single layered security model does not have such properties. However, if an output of all possible attack scenarios is required, then there is an overhead associated with using the HARM, which requires a top-down correlation with lower layer components (i.e., mapping out all components into a single layer). This overhead is not taken into account, which may have a significant effect on security analysis.

4.4 Conclusions

Existing studies did not take into account analysing the scalability of these models in various network scenarios. As networked systems are becoming large and dynamic (e.g., a Cloud network), traditional solutions are facing scalability and adaptability problems. Structural modification solutions (e.g., LAG, MPG,

and TLAG) cannot evaluate all possible attack scenarios in a scalable manner, and heuristic solutions may lose security information. As a result, network and security administrators are facing difficulty determining the security posture to efficiently deploy defense strategies.

This chapter shows the scalability of security models used for a large sized networked system. A performance analysis was conducted to demonstrate how different security models, namely the HARM and the AG, performed in various network scenarios. The experimental results show that even when using the same algorithm to evaluate the security of networked systems, the performance of the HARM is much better than existing security models. Moreover, regardless of the network scenario, the HARM showed better or equal performance in terms of generation and evaluation phases. Chapter 5 therefore investigates the adaptability of security models in the modification phase.

Chapter 5

Adaptability Analysis

Adaptability of security models are not widely studied, since it has only been recently that networked systems are becoming highly dynamic, such as the Cloud [15, 17, 140] and Software Defined Networks [104, 121]. Consequently, it is uncertain how the security posture of networked systems is affected when there are changes in the networked system. Such changes in the networked systems are formalised in Section 2.3.2, and associated complexity is described in Chapter 3. However, the complexity analysis only indicates the performance in case of the worst-case scenario. Moreover, different characteristics of possible changes in the networked systems are not reflected as well. Therefore, it is important to investigate the adaptability of security models to understand how security is affected when there are changes in the networked system.

In this chapter, various scenarios of changes in the networked system are taken into account and their effects are analysed, such as changes in the attack surface [134]. These changes are based on Moving Target Defense (MTD) that continuously changes the attack surface of the networked system [48, 60, 133, 216, 217], where MTD techniques are widely used (e.g., dynamic networks [209], wireless sensor networks [39], and adaptive execution environment in a virtualised system [163]). More precisely, MTD techniques are classified into three categories: (i) Shuffle, (ii) Diversity, and (iii) Redundancy. *Shuffle* rearranges the system setting at various layers (e.g., address randomisation, migration, topology rearrangements) [22, 52, 104, 109, 155, 192, 212]. *Diversity* provides equivalent functions with different implementations (e.g., operating systems, variant input and interpreters, variant software stack components) [47,75,96,103,147,148,170,204]. *Redundancy* provides multiple replicas of network components (e.g., service, nodes, or paths) to make multiples of the same functions [10, 41, 76, 95, 117, 211]. These techniques are analysed on the basis of the security analysis and performance of security models.

5.1 Preliminaries

MTD techniques change the network environment (e.g., logical network topology) that may cause confusion to attackers, as well as to system administrators about how these changes affect the security of the system. The effectiveness of MTD techniques (e.g., changes in security postures) may vary depending on how they are implemented in the system. Those changes can be studied in a well-formed model so their effectiveness can be computed and compared. However, existing studies on MTD techniques do not rely on any formal security models. Previous work proved their effectiveness with simple metrics that are not comparable [209, 216, 217]. Hence, the HARM is utilised by incorporating MTD techniques as shown in Section 2.3.2.

The HARM is used instead of any existing security models because: (i) the HARM can adopt any security models in its layers (e.g., an AG in the upper layer and an AT in the lower layer), and (ii) it is more adaptable to changes in the networked system than single layer AGs (i.e., an AG). A comparative study in [88] showed that the HARM is more scalable and adaptable than the AG. However, the HARM still has a scalability problem when the number of nodes becomes large. To address this problem, importance measures (IMs) are also incorporated into the HARM [89, 90] to aid how to determine where to apply MTD techniques in the networked system. More details of IMs will be described in Chapters 6 and 7. The performance comparison of using the IMs and the exhaustive search (ES) method is conducted via simulations in each of subsequent sections.

First, the example networked system shown in Figure 2.3 is transformed into a virtualised system to enable dynamic changes in the networked system, which is shown in Section 5.1.1. Second, an overview on the classification of MTD techniques is shown in Section 5.1.2. Third, motivations and importance of assessing the MTD to secure attack paths is described in Section 5.1.3. Last, IMs are briefly described in Section ssec:5ims.

5.1.1 A Virtualised System

The example networked system shown in Figure 2.3 is transformed into a virtualised system as shown in Figure 5.1. Hosts are now represented as virtual machines (VM) with the same OSes as shown in Table 2.3 (e.g., $Host4$ in the example

networked system U_4 is represented as VM_4 with W7 OS). A few assumptions are made for the virtualised system: (i) the system hardware is trusted and checked using a hardware-based cryptographic verification (e.g., a trusted platform module), and the logic of operating-system-level virtualisation is trusted as in [155], (ii) the attacker exploits operating system (OS) vulnerabilities only as shown in Tables 2.4 and 2.5 (other vulnerabilities, such as application vulnerabilities, can also be modelled), and (iii) components of the virtualised system can be changed frequently due to security matters as well as VM creation/decease/migration. The virtualised system can be regarded as a basis example for modern networked systems, such as virtualised data centres and the Cloud.

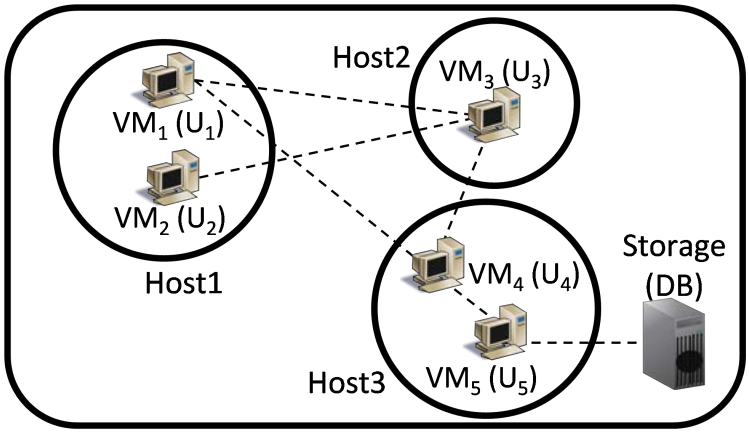


Figure 5.1: A Virtualised System for the Example Networked System

The following assumptions and configurations are used for the virtualised system, where these can be released in future work:

- All VMs remain active (up and running) at all times.
- Each host (i.e., a node) can hold up to two VMs only.
- $Host3$ cannot hold VM_1 or VM_2, because VM_1 and VM_2 are assumed as the entry points to the system (e.g., front-end servers).
- VM_5 remains on $Host3$ (i.e., cannot be removed) to process connection to the database (e.g., a back-end server).

Further, connection constraints (e.g., via IP addresses) of VMs (it can be obtained similarly as in [46]) are described as the following:

- Any VMs on $Host1$ can connect to any VMs on $Host2$, and one VM (whichever stayed longer) can connect to any VMs on $Host3$ excluding VM_5.
- VMs on $Host1$ are not interconnected.
- Any VMs on $Host2$ can connect to any VMs on $Host1$ and $Host3$ (excluding VM_5), and they are interconnected.
- Any VMs on $Host3$ are interconnected. Also, each VM has two available guest OSes.
- If there is only VM_5 on $Host3$, then any single VM from $Host2$ can connect to VM_5.

With the given information, the HARMs shown in Figures 2.4 and 2.5 can be generated.

5.1.2 Categorising the MTD techniques

Categories of MTD techniques implemented in various layers are described in this section. MTD techniques are mainly categorised into *Shuffle*, *Diversity* and *Redundancy*, and they are summarised in Table 5.1. *Shuffle* has been studied in various domains at different system layers (i.e., TCP/IP, Infrastructure and Application layers). Many studies have focused in the application layer for *Diversity* (e.g., software implementations, compiler based diversity, software mix, data diversity, address space partitioning and instruction set tagging), as well as in the topology layer (e.g., path diversity). *Redundancy* has also been focused in the application layer and the topology layer.

5.1.3 Securing attack paths

The focus of network hardening in this chapter is based on securing important nodes in the attack paths, such as in [147] (but not routing nodes here), rather than

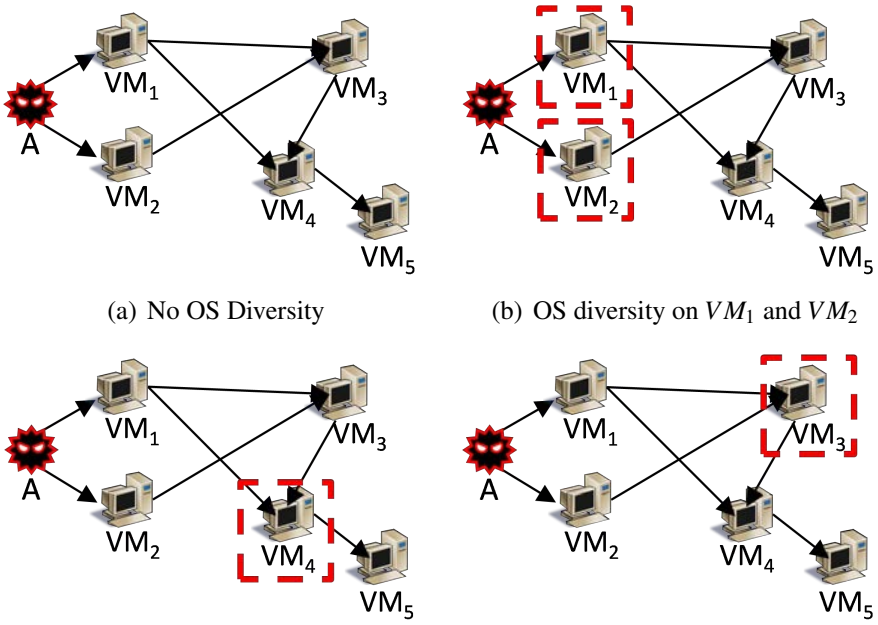
Table 5.1: Categories of MTD Techniques

	Subcategory	Layers	references
MTD	Shuffle	TCP/IP	[22, 104]
		Infrastructure	[52, 155, 212]
		Application	[109, 192]
	Diversity	Topology	[147, 170]
		Application	[47, 75, 96, 103, 148, 204]
	Redundancy	Topology	[10, 117]
		Application	[41, 76, 95, 211]

end points of an attack (i.e., initial attack points and target nodes) because of the following reasons: (i) in a virtualised system (e.g., cloud), visible components or nodes to the attacker (i.e., initial attack points in the attack surface) may change frequently (e.g., service updates), (ii) assets in the networked system change, and there may exist multiple assets (i.e., multiple target nodes), and (iii) target nodes are estimated with asset values but it is still difficult to specify them in an event of an attack.

A simple example based on the virtualised system is shown in Figure 5.2. The goal of the attacker is to compromise VM_5. An OS diversity technique is used to enhance the security (as specified in Section 5.1.1), where a dotted box highlights the VM(s) with the OS diversity technique deployed. The security goal is to ensure that the most (if not all) attack paths are affected with the OS diversity technique. Figure 5.2(a) shows an initial attack scenario (i.e., no OS diversity). Figure 5.2(b) shows that the OS diversity is applied to both initial entry points in the networked system VM_1 and VM_2. On the other hand, Figure 5.2(c) shows that the OS diversity is applied to VM_4, which satisfies the security goal (i.e., only requires a single implementation of the OS diversity technique). If the implementation of the OS diversity technique has an associated cost, then minimising the number of nodes to deploy the OS diversity technique is more cost effective.

Another key point is the different effectiveness in securing attack paths. If the OS diversity technique is deployed on VM_3 as shown in Figure 5.2(d), then only a subset of all possible attack scenarios is affected by the OS diversity technique. In



(a) No OS Diversity (b) OS diversity on VM_1 and VM_2

(c) OS diversity on VM_4 (d) OS diversity on VM_3

Figure 5.2: A Comparison of Securing Attack Paths and End Points

contrast, deploying the OS diversity technique on VM_4 covers all possible attack paths. More complex diversity assignment can be found in [147], and this will be shown in Section 5.3 to assess the effectiveness of these scenarios using the HARM.

5.1.4 Computing the Importance Measures

IMs are computed as described in [90]. Network centrality measures (NCMs) for the example (see Figure 5.1) is shown in Table 5.2. NCMs are used to rank upper layer hosts (e.g., VMs), and it is calculated based on degree, closeness and betweenness centrality measures. Other NCMs can also be used as in [59]. Vulnerabilities are ranked based on their CVSS BS (see Table 2.4 and 2.5).

Table 5.2: NCMs and Ranking Important VMs

NCM	VM_1	VM_2	VM_3	VM_4	VM_5
Degree	0.67	0.5	0.67	0.67	0.333
Closeness	0.71	0.56	0.71	0.71	0.45
Betweenness	0.6	0.3	0.5	0.7	0.3
Rank	2	4	3	1	5

5.2 Shuffle

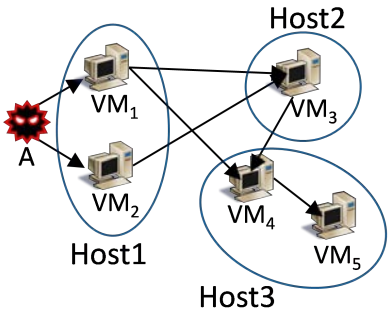
5.2.1 Incorporating only Shuffle in the HARM

Using the formalism given in Section 2.3.2, Figure 5.3 shows an example of taking into account a VM live migration (VM-LM) as the *Shuffle* technique deployed in the virtualised system. A VM-LM is applied to VM_1 and VM_4 with constraints enforced (as described in Section 5.1.1). It is believed that the other *Shuffle* based MTD techniques can be modelled in a similar way. Computing all VM-LM scenarios (and for any *Shuffle* based MTD techniques) is an NP-Hard problem, because there are an exponential number of host combinations that can be rearranged in the networked system. To deal with the scalability problem, only the next available migration step is modelled (i.e., only a single available VM is migrated) and the security of each scenario is analysed.

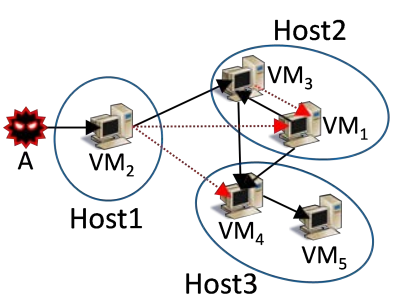
VM_1 or VM_2 can migrate from $Host1$ to $Host2$, or VM_4 can migrate from $Host3$ to $Host2$ under constraints specified in Section 5.1.1. Figure 5.3(b) shows the migration of VM_1 with an addition of three new edges between (VM_1, VM_3), (VM_1, VM_4), and (VM_2, VM_4). Newly created edges are highlighted by dotted arrow lines in red. Migration of VM_4 resulted in two new edges between (VM_4, VM_3) and (VM_2, VM_4), as shown in Figure 5.3(c).

5.2.2 Assessing the Effectiveness of Shuffle

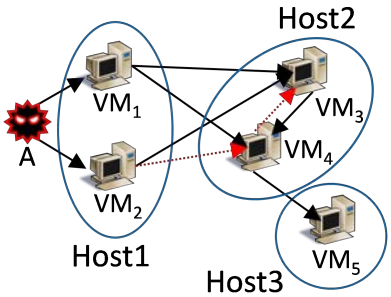
Using the Exhaustive Search (ES) method (i.e., the naive method), the system risk is computed for each migration scenarios. There are a total of three migration scenarios with VM_1, VM_2, and VM_4. Due to constraints, other VMs cannot be migrated. The system risk computation is the same as shown in Section 2.5, with



(a) No VM-LM



(b) VM-LM of VM_1



(c) VM-LM of VM_4

Figure 5.3: Migration of VMs in the Upper Layer of the HARM

security metrics shown in Table 2.4 and 2.5. The system risk associated with migrating VM_{id} (id represents the identification number of the VM, VM ID) is denoted as $R_{system}^{VM_{id}}$ and all possible attack paths associated with this migration is denoted as $path^{VM_{id}}$. For example, the system risks after migrating VM_1 and VM_4 are shown in equations (5.1) and (5.2) respectively. Equation (5.3) shows the risk calculation of an attack path $i = \{VM_1, VM_4, VM_5\}$. Here, nodes in attack paths are denoted by the VM ID for simplicity. The system risk of migrating VM_1 is much greater than migrating VM_4, which indicates that it is more ideal to migrate VM_4 to keep the system risk minimised.

$$R_{system}^{VM_1} = \sum_{i \in path^{VM_1}} PR_i = 298505.01 \tag{5.1}$$

$$R_{system}^{VM_4} = \sum_{i \in path^{VM_4}} PR_i = 45200.47 \tag{5.2}$$

$$PR_{145} = \sum_{j \in \{1,4,5\}} NR_j = 2516.86 \tag{5.3}$$

5.2.3 Experiment 5A: Analysing Shuffle

The scalability of deploying *Shuffle* in a large sized networked system is analysed in this section. Two methods, using the IMs and the ES method, are compared with respect to their performances. An abstracted CloudBand model [15] is used for the simulation, where the example virtualised system can be regarded as a small sized CloudBand model, and a larger model is created for the simulation. Two CloudBand nodes are setup and a resource node as shown in Figure 5.4. The HARM of this CloudBand model with each node hosting five VMs is depicted in Figure 5.5. The attack goal is to compromise the resource node, and each CloudBand node can hold up to 450 VMs which they are connected in a mesh topology with minimum number of edges of three. A VM-LM is taken as an example *Shuffle*, assuming that any VMs can migrate between the CloudBand nodes (one at any given time under the availability of space). When a VM migrates, the reachability of VMs changes and it affects the security posture of the networked system. As shown in Section 5.2.2, migration of different VMs result in various system risk values. Therefore, a VM that minimises the system risk is migrated.

The performance of the following three cases are analysed: (i) an AG using the ES method, (ii) a HARM (upper layer AG and lower layer AT) using the ES method, and (iii) a HARM using the IMs (based on the betweenness centrality). The betweenness centrality measure is used among others (e.g., degree and closeness) based on the result in [90]. The top 10% proportion of important VMs are chosen for security analysis, and choosing different percentages of important nodes is analysed in the next experiment. An assumption is that there are two vulnerabilities for each VM, and the attacker can exploit any of the two vulnerabilities to compromise that VM. More vulnerabilities can be modelled (with other

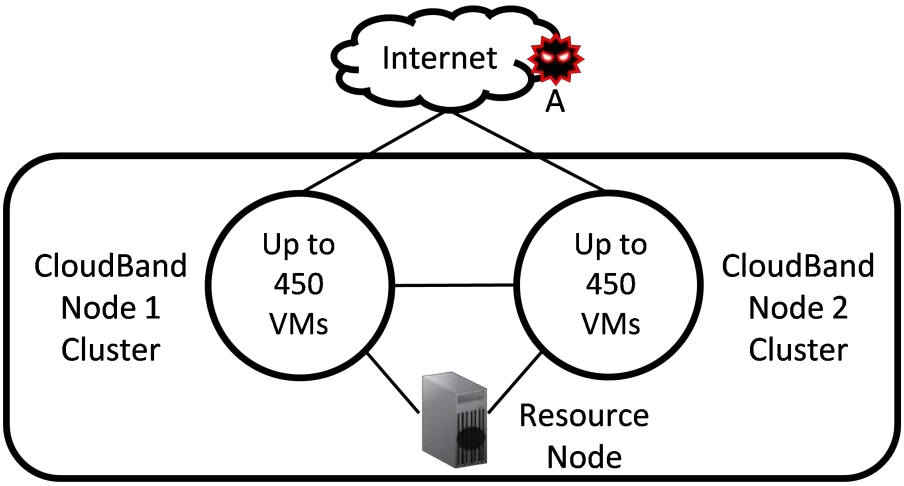


Figure 5.4: A CloudBand Model for Simulation

privilege types), but the number of vulnerabilities in the experiment is constrained due to the poor scalability of the AG.

First, Figure 5.6 shows the performance comparison of the three cases (i), (ii) and (iii) in the evaluation phase of HARM and AG. The time to evaluate the AG and the HARM increases exponentially, but the HARM is more scalable than the AG. Using the IMs performed the best with a proportional time in comparison to the HARM using the ES method. All these methods produced the same result in the experiment (i.e., which VM to migrate).

Second, the performance with respect to the different proportion of important VMs selected is analysed, which is shown in Figure 5.7. It shows that as the proportion of important VMs decreases (i.e., the number of selected important VMs decreases), the performance of security analysis increases. It also shows that taking into account all VMs using the IMs (i.e., equivalent to the ES method) performs worse than the ES method due to the overhead of computing the IMs. However, this overhead is almost negligible, as the complexity of computing IMs is in polynomial, whereas evaluating security with AG or HARM is in exponential.

In conclusion, the AG performed the worst, followed by the HARM using the ES method, and the HARM using the IM performed the best in terms of time. The simulation result showed that using the IMs for security analysis is efficient, which also computes equivalent (or nearly the same) solutions to the ES method

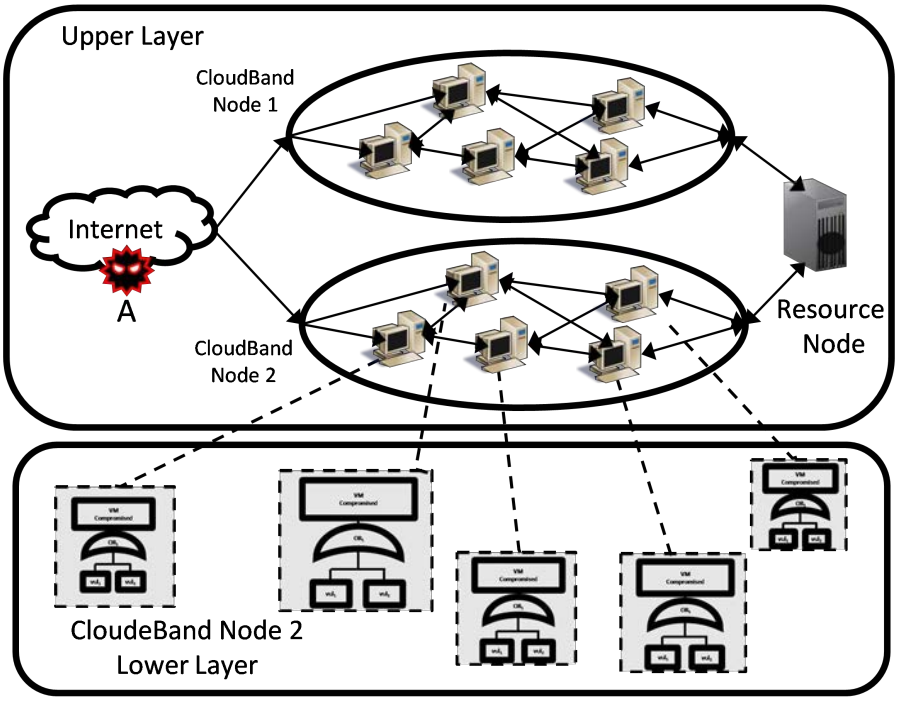


Figure 5.5: HARM of the CloudBand Model with five VMs on Each Node

even with a small proportion of important nodes (e.g., VMs).

5.3 Diversity

5.3.1 Incorporating only Diversity in the HARM

The reachability of the networked system (as well as dependencies between network components) is assumed to be unchanged with *Diversity* (i.e., only changes the vulnerability information), because it must provide the same functionalities as with its variants. Therefore, it only changes the lower layer of the HARM as defined in Section 2.3.2. For example, if an OS *Diversity* technique is deployed on VM_1, then the AT in the lower layer for VM_1 needs to be updated with different vulnerabilities from the new OS. An example of OS diversity is shown in Figure 5.8, where VM_1 is initially modelled with the lower layer AT with an attack goal to compromise W7 (see Figure 5.8(a)). A substitute OS, WV (see Figure 5.8(b)), can be used instead.

One may not observe any difference in security analysis depending on the se-

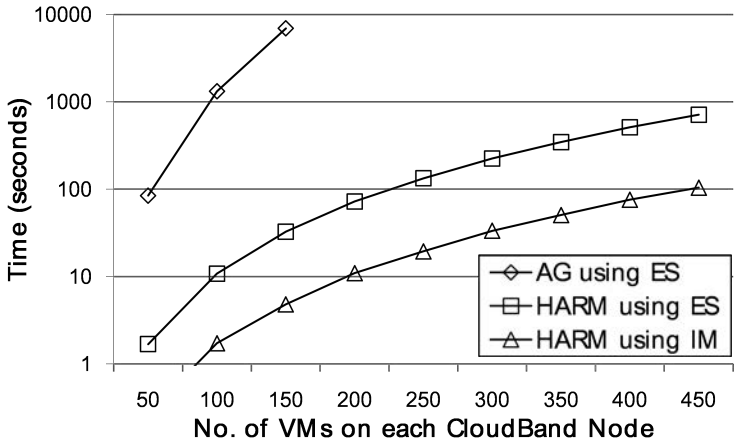


Figure 5.6: Performance comparison between AG and HARM when deploying a VM-LM

curity metrics used (e.g., risk values associated with a VM assigned with W7 and another VM assigned with WV are the same). However, one property of *Diversity* is that it changes the attack surface by forcing the attacker to use different exploits for different vulnerabilities (such as in [75, 147], with an assumption of non-overlapping vulnerabilities), and these effects may be captured using different security metrics, which will be shown in Section 5.3.2.

5.3.2 Assessing the Effectiveness of Diversity

An OS diversity technique is taken as an example (as shown in Section 5.3.1). The attack cost is taken into account with an assumption that if the attacker encounters an OS already exploited in the attack path, then the attack cost is reduced (e.g., reusing exploit tools, experience gained). Other security metrics that reflect different path information can be used as well (e.g., mean time to compromise). An assumption is that exploiting a VM with any OS has an attack cost of 1 unit dollar, and the difficulty of exploiting any OS is the same. If the attacker has previously exploited the OS successfully (i.e., already exploited same vulnerabilities before), then the attack cost is reduced to 0.5 unit dollars. Using the OS diversity technique on a single VM is considered, with the attack goal of compromising VM_5. More complex diversity assignment scenarios are shown later in this section.

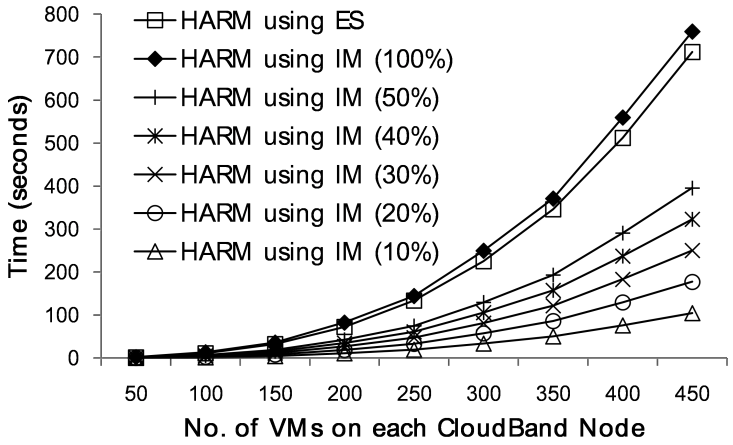


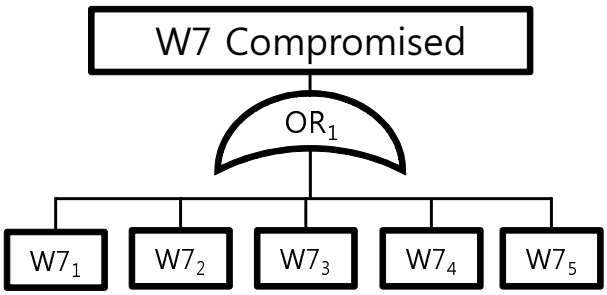
Figure 5.7: Comparison between the ES method and using the IMs when deploying a VM-LM

First, the ES method is used with the virtualised system in the initial state (as shown in Figure 5.1). The OS diversity technique can be deployed to each VM, and the associated attack costs are computed. For instance, the attack cost of the initial state can be computed as shown in equation (5.4). The attack cost of deploying the OS diversity technique to VM_{id} is denoted as $AC^{VM_{id}}$, and the total attack cost of an attack path is denoted as $AP_{i \in path}^{VM_{id}}$ (for $path$ being a list of all possible attack paths and $path^{VM_{id}}$ is a list of all possible attack paths associated with the OS diversity). The attack cost of deploying the OS diversity to VM_1 is shown in equation (5.5), and it shows that this increases the total attack cost by one unit dollar in comparison to the initial state.

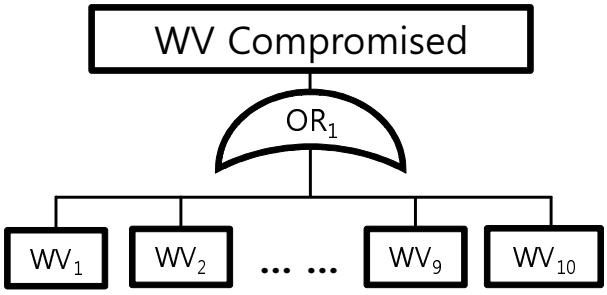
$$AC = \sum_{i \in path} AP_i = 8.5 \tag{5.4}$$

$$AC^{VM_1} = \sum_{i \in path^{VM_1}} AP_i = 9.5 \tag{5.5}$$

The summary of deploying the OS diversity to each VM is shown in Table 5.3, where the attack paths are represented by the sequence of VM ID. Assigning the



(a) Attack Goal to Compromise W7



(b) Attack Goal to Compromise WV

Figure 5.8: Possible Lower Layer ATs for VM_1 with OS Diversity

OS diversity on VM_4 maximises the attack cost.

Secondly, the IMs are used, which are already shown in Table 5.2. It showed that VM_4 is the most important (i.e., first rank), which yields an equivalent solution to the ES method. Of course, the order of important nodes may not be the same with the ES method in different network settings and attack scenarios. However, using the IMs can compute the equivalent solution to the ES method as demonstrated in [89, 90].

Assigning *Diversity* to multiple nodes is considered. Some of the constraints

Table 5.3: Total Attack Cost of deploying OS Diversity (in unit dollars)

Path	None	VM_1	VM_2	VM_3	VM_4	VM_5
1 3 4 5	3	3.5	3	3.5	3.5	3.5
1 4 5	2.5	3	2.5	2.5	3	2.5
2 3 4 5	3	3	3.5	3.5	3.5	3.5
Total Attack Cost	8.5	9.5	9.0	9.5	10.0	9.5

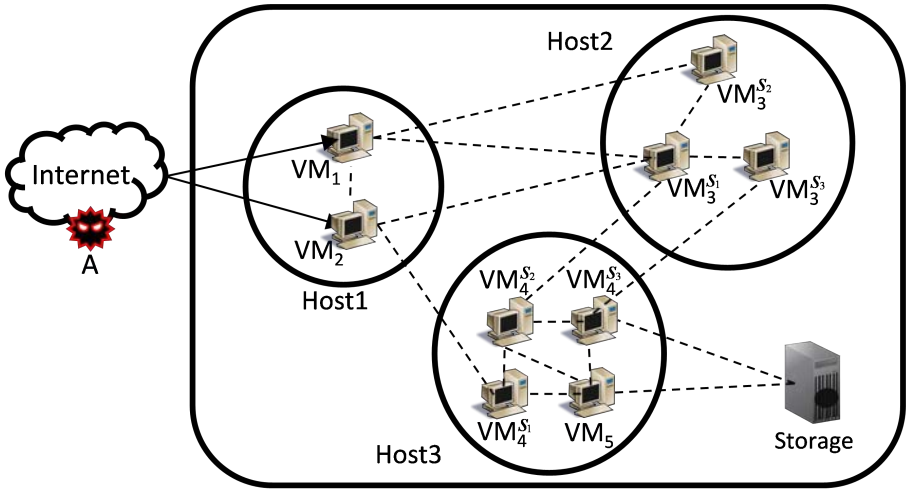
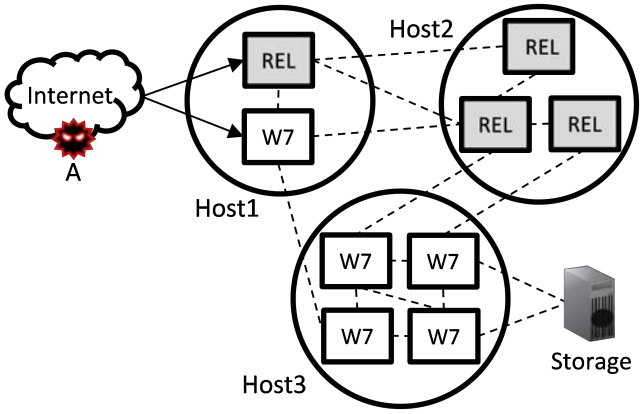


Figure 5.9: An Example Virtualised System for DAP

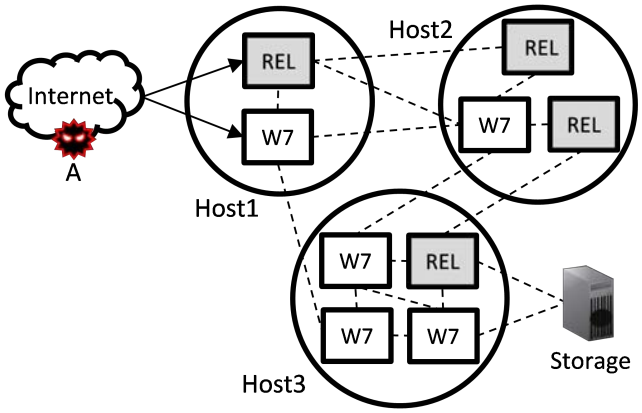
in Section 5.1.1 are relaxed to incorporate more VMs in each host as shown in Figure 5.9, where $VM_{id}^{S_j}$ represents the jth server of VM_{id}. Two OS variants W7 and REL are taken into account, with the probabilities of each being compromised as 0.1 and 0.15 respectively regardless of their vulnerabilities. Then, the expected client (host) connectivity (ECC) can be computed as described in [147] to solve the *Diversity* assignment problem (DAP) with respect to performance of the networked system. Figure 5.10 shows a comparison of three different cases of assigning the OS diversity (REL assigned VMs are shaded for readability). It shows that a randomly assigned *Diversity* is unlikely to provide an optimal solution for any metrics (Figure 5.10(a) show ECC of 0.838 with Risk 1054), because this particular random case ended up with both ECC and Risk worse off than the ECC-optimal case (shown in Figure 5.10(b)). Further experiments are carried out in Section 5.3.3. Also, there is a trade-off between ECC and Risk (as shown in Figure 5.10(b) and 5.10(c)), where ECC-optimal case does not hold Risk-optimal, and also Risk-optimal does not hold ECC-optimal. However, this multi-metrics optimisation problem is out of scope for this chapter.

5.3.3 Experiment 5B: Analysing Diversity

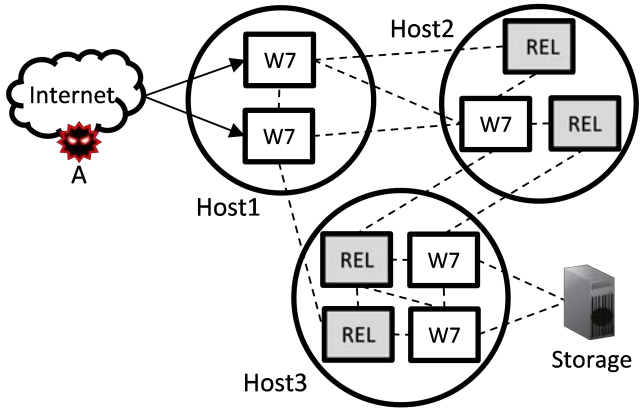
The changes in security with various *Diversity* assignments are analysed in this section. In [147], ECC was used to measure the connectivity of network clients



(a) Random OS diversity assignment with 0.838 ECC and 1054 Risk.



(b) ECC-optimal OS diversity assignment with 0.957 ECC and 1048 Risk.

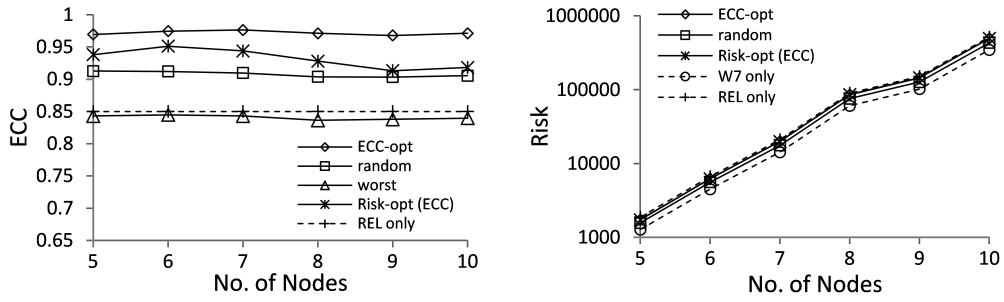


(c) Risk-optimal OS diversity assignment with 0.81 ECC and 1026 Risk.

Figure 5.10: OS diversity assignments for our example

when routing nodes are attacked. However, ECC is a performance measure and they did not consider how security may change when they optimise the ECC. Similarly, other related works on *Diversity* did not consider various security metrics, which are described in Section 5.6. Experiments are carried out to analyse the relationship between ECC (performance metric) and system risk (security metric). Similar assumptions and settings are used for the simulation as in [147], with a randomly generated networked system with a given density value that specifies the average number of connections a node has. There are no duplicated connections between nodes. For the example virtualised system, an attacker is assumed to be connected as a client, with an intension of compromising the back-end server (i.e., VM_5 in Figure 5.1) that is treated as a client. Other VMs are considered as routing nodes (e.g., intermediate server nodes).

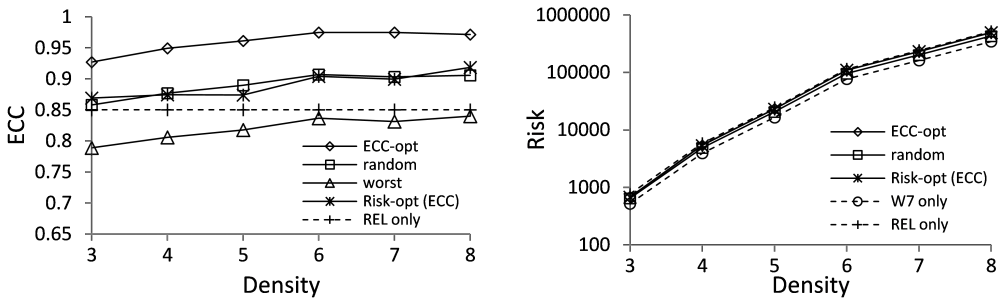
The OS diversity technique is considered for the experiment with the following assignment cases: (i) ECC-optimal (ECC-opt), (ii) random, (iii) worst, (iv) Risk-optimal (Risk-opt), (v) W7 only, and (vi) REL only. *ECC-opt* optimises the ECC. *random* and *worst* are random and worst OS diversity assignments respectively. *Risk-opt* optimises the risk using the same number of variants as *ECC-opt*. Cases (v) and (vi) describe the same OS used on all routing nodes in the CloudBand (i.e., a homogeneous network with a single OS variant). Three variables are taken into account: (a) a number of nodes (Figure 5.11(a) and 5.11(b)), (b) a density of the networked system (Figure 5.12(a) and 5.12(b)), and (c) a number of variants (i.e., available OSes) (Figure 5.13(a) and 5.13(b)). For scenario (c), newly added OS variants are assumed with a less probability of an attack value than existing OSes.



(a) ECC (b) Risk

Figure 5.11: ECC and Risk Changes with Respect to the Number of Nodes

Scenario (a): Figure 5.11 shows the analysis of ECC and Risk with respect to the number of nodes in the virtualised system. Figure 5.11(a) shows that there are no significant changes in the ECC value as the number of nodes increases. Moreover, *Risk-opt* provides relatively the same value of ECC with the case of randomly assigned OS diversity. A single variant (i.e., *REL only*) provides a constant ECC. Also, there are *Diversity* assignments providing a worse ECC value using more variants (this is also mentioned in [147], but not shown via an experiment). Taking into account the risk analysis shown in Figure 5.11(b), increasing the number of nodes also increases the system risk for all cases. Also, the difference between these cases are minimal. For example, *Risk-opt* has less risk than *ECC-opt* by an average of 0.4% (although we have to take into account that *Risk-opt* has the same number of nodes for each variant as the *ECC-opt*). The largest average risk difference was between *REL only* and *W7 only* with 48%.

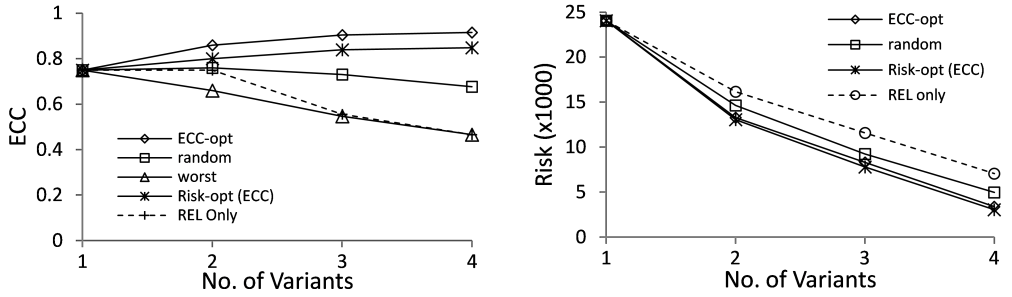


(a) (b)

Figure 5.12: ECC and Risk Changes with Respect to the Network Density

Scenario (b): Figure 5.12 shows the analysis of ECC and Risk with respect to the network density of the virtualised system. All scenarios with two variants gradually increase the value of ECC as the networked system becomes more dense, where an ECC value of a single variant remains the same. This is depicted in Figure 5.12(a). The crossovers between *Risk-opt* and *random* is due to the randomness of the networked system created, but they both increase gradually at a similar rate. Similarly for a risk analysis shown in Figure 5.11(b), the system risk increases due to an increase in the number of possible attack paths (i.e., higher density creates more attack paths). Similar to scenario (a), the values of the system risk between difference cases are insignificant, where the largest average risk

difference was between *REL only* and *W7 only* with 33% (two extreme cases optimal and worst). A similar result can be observed in [147], where increasing the density of the networked system improves the ECC.



(a) (b)

Figure 5.13: ECC and Risk Changes with Respect to the Number of Variants

Scenario (c): Figure 5.13 shows the analysis of ECC and Risk with respect to the number of variants in the virtualised system. If the number of variants is increased (i.e., there are more available OSes), then randomly assigning the OS diversity or using no OS diversity (i.e., *REL only*) reduces the value of ECC as the number of variants increases. Also, no OS diversity quickly converged to the worst-case in the experiment. In contrast, the value of ECC for *ECC-opt* and Risk-opt increases with more variants. This result indicates that from all possible *Diversity* assignments, only a small subset would improve the ECC. The result is also reflected in the risk analysis shown in Figure 5.13(b). Both *ECC-opt* and *Risk-opt* show a minimal value of the system risk compared to the other cases.

In conclusion, there is a trade-off between ECC-oriented and Risk-oriented *Diversity* assignments. However, both solutions are better than any other cases in terms of ECC and system risk (e.g., random or none). For the ECC analysis, the number of nodes does not affect the ECC for a constant density value, but increasing the density value gradually improves the ECC as well. Increasing the number of variants only improves the ECC for ECC-oriented and Risk-oriented assignments, whereas the other cases gradually decreased. For the risk analysis, there are minimal differences between the experiment cases, where using a single variant with the lowest risk is optimal. However, risk is minimised using Risk-oriented or ECC-oriented solutions when compared to other non-homogeneous

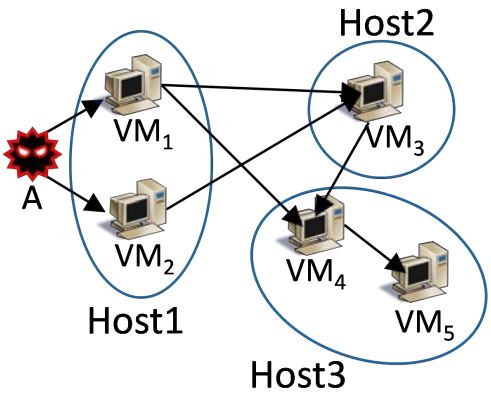
Diversity assignment cases (e.g., randomly assigned).

5.4 Redundancy

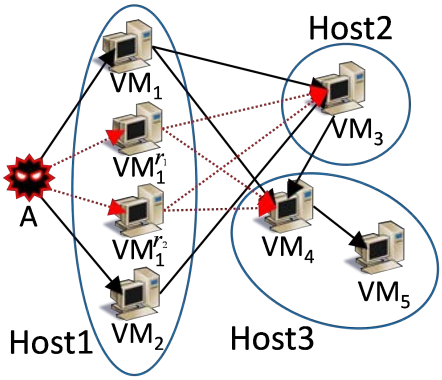
Incorporating only Redundancy in the HARM

Redundancy creates multiple replicas of network components (e.g., services, nodes or paths) to enhance the performance (e.g., reliability). For example, a front-end server may have multiple replicas in case of a DDoS attack [76, 95]. Therefore, modelling *Redundancy* creates new nodes in the AG of the upper layer HARM and duplicated ATs of the replicated upper layer AG nodes in the lower layer. The number of replicas is denoted as k-R for a k number of replications. If VMs in the example (shown in Figure 5.1) are assumed as servers (e.g., a web server), then Figure 5.14 depicts two server replicas applied to different VMs (by relaxing the assumption about a host holding only up to two VMs). Two different scenarios of analysing *Redundancy* are considered: (a) To analyse the change in the security, and (b) to analyse the performance. In the case of (a), attack scenarios with the attack goal of compromising some target (e.g., sequential attacks) is taken into account. Hence, the intension of the attacker is unlikely to compromise all replicas. On the other hand, case (b) considers attack scenarios with the attack goal to shut down functionalities (e.g., DDoS attacks to block services). Therefore, the attacker is likely to have an intension of compromising a certain number of replicated network components (if not all). In this case, the upper layer of the HARM can be substituted with a performance analysis model with a corresponding lower layer model if necessary (e.g., a reliability graph to measure the reliability).

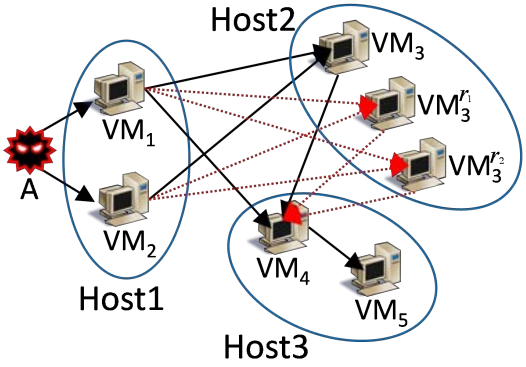
Both risk and reliability analyses are considered, because *Redundancy* does not necessarily decrease the attack surface, but the aim is to provide a better performance (e.g., to mitigate DDoS attacks). All VMs are assumed to be running the same OS (e.g., W7) with a probability of an attack of 0.1 and an impact value of 10. Moreover, the networked system is attacked at a rate of 0.1 attack per hour (i.e., the rate of a VM attacked is once every 10 hours). The *Redundancy* technique is deployed to each VM with 2-R. Equation (2.1) is used to compute the system risk, and a reliability analysis tool named *SHARPE* (Symbolic Hierarchical Automated Reliability and Performance Evaluator) is used to compute the



(a) No *Redundancy*



(b) *Redundancy* with 2-R on VM_1



(c) *Redundancy* with 2-R on VM_3

Figure 5.14: Deploying *Redundancy* Technique in the Virtualised System

reliability of the virtualised system (denoted as r) (details can be found in [174]).

5.4.1 Assessing the Effectiveness of Redundancy

First, the ES method is used to compute system risk and reliability of the virtualised system for all *Redundancy* deployment scenarios. Here, the working processes for computing the system risk is omitted, as it was already shown in Section 5.2.2. Reliability is measured at time $t = 10$ (i.e., system running time at 10 hours). Figure 5.14(c) shows the transformation of the upper layer HARM with *Redundancy* for VM_3 with 2-R. It shows that the increase in the number of attack paths is equal to the number of edges times the number of replicas (i.e., $degree(n_i) \times k$, for node n_i with k replicas, where $degree(n_i)$ is the number of

Table 5.4: System Risk, Reliability and Probability using *Redundancy*

Replicated VM	VM_1	VM_2	VM_3	VM_4	VM_5
System Risk	104748.9	99715.18	148314.30	153348. 05	153348. 05
Reliability	0.099	0.162	0.116	0.189	0.154

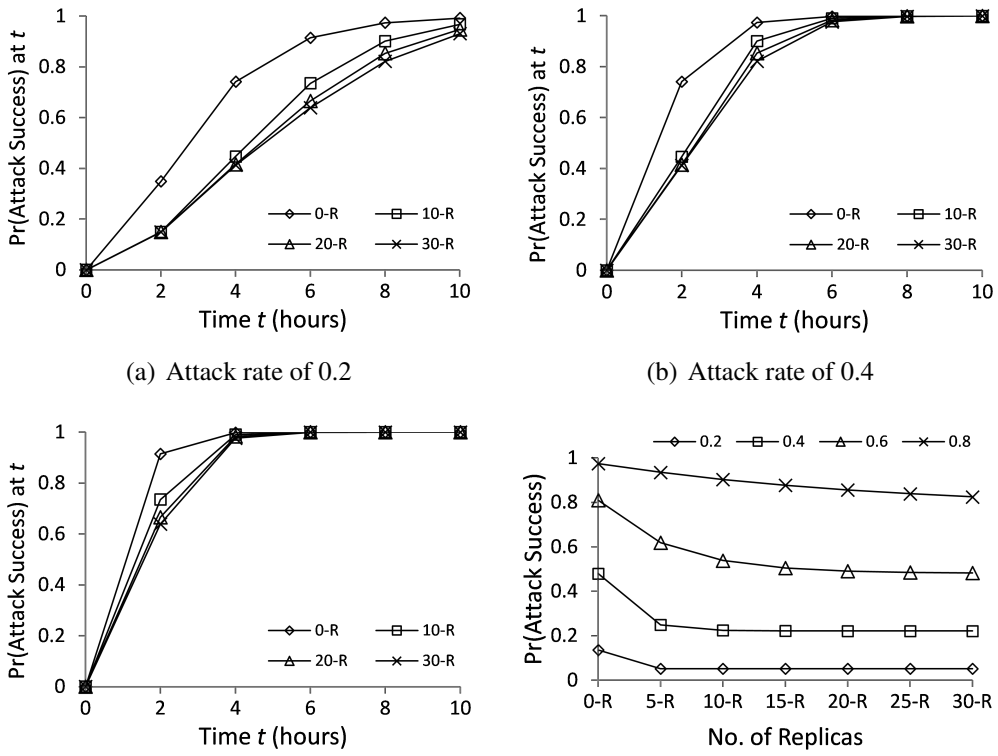
edges connected to node n_i). Table 5.4 shows the result of *Redundancy* using the ES method. *Reliability* is measured by the probability of an attack success at time $t = 10$.

Second, the IMs are used. Since servers are replicated (e.g., web servers hosts on VMs are captured in the upper layer HARM), NCMs shown in Table 5.2 are used to rank important nodes in the upper layer of the HARM. VM_4 has the highest rank, and as shown in Table 5.4, replicating VM_4 provides the highest reliability. However, the system risk is also the highest (i.e., ratio between risk and reliability). Various scenarios of *Redundancy* is analysed in Section 5.4.2.

5.4.2 Experiment 5C: Analysing Redundancy

Changes in the security and performance are investigated when deploying the *Redundancy* technique in the networked system. Additional replicas of network components may improve the performance (e.g., reliability, availability), but it also increases the security concern (e.g., system risk). The virtualised system shown in Figure 5.1 is used for the experiment, where VM_1 and VM_2 are front-end servers and VM_5 is the back-end server. The *Redundancy* technique is deployed onto a front-end server (e.g., VM_1) with all replicas being active. Three scenarios are analysed: (i) probability of the attack success (Figure 5.15(a) to 5.15(d)), (ii) mean-time-to-attack (MTTA) (Figure 5.16), and (iii) changes in the system risk and availability (Figure 5.17). SHARPE is used to compute the reliability and availability.

Scenario (i): Figures 5.15(a) – 5.15(c) shows the probability of the attack success (Pr(AS))over a period of time for varying attack rates. As the attack rate increases, the Pr(AS) increases rapidly. Also, increasing the number of replicas (i.e., k-R) reduces the Pr(AS). However, the difference of Pr(AS) between high numbers of replicas quickly diminishes (e.g., minimal difference between 20-R and 30-R compared to other k values). Figure 5.15(d) also shows that the Pr(AS)



(a) Attack rate of 0.2

(b) Attack rate of 0.4

(c) Attack rate of 0.6

(d) Changes in Probability of an Attack Success

Figure 5.15: Various Probability of the Attack Success with Respect to *Redundancy*

decreases as the number of replicas increases, but the difference becomes negligible for a large number of replicas (e.g., $k > 20$ for k-R in this example).

Scenario (ii): Figure 5.16 shows the mean-time-to-attack (MTTA) for varying values of attack rates. For low attack rates (e.g., 0.2 attacks per hour) the value of MTTA increases logarithmically, whereas for higher attack rates (e.g., 1 attack per hour) the value of MTTA is almost linear proportional to the lower attack rates. Also, the value of MTTA converges as the number of replicas increases, where it converges faster for higher attack rates.

Scenario (iii): Figure 5.17 shows the changes in the system risk and availability. The system risk increases linearly, as additional replicas provide a constant increase in the number of attack paths. However, availability increases logarithmically and quickly converges as the number of replicas increases.

In conclusion, there is a trade-off between security and performance when de-

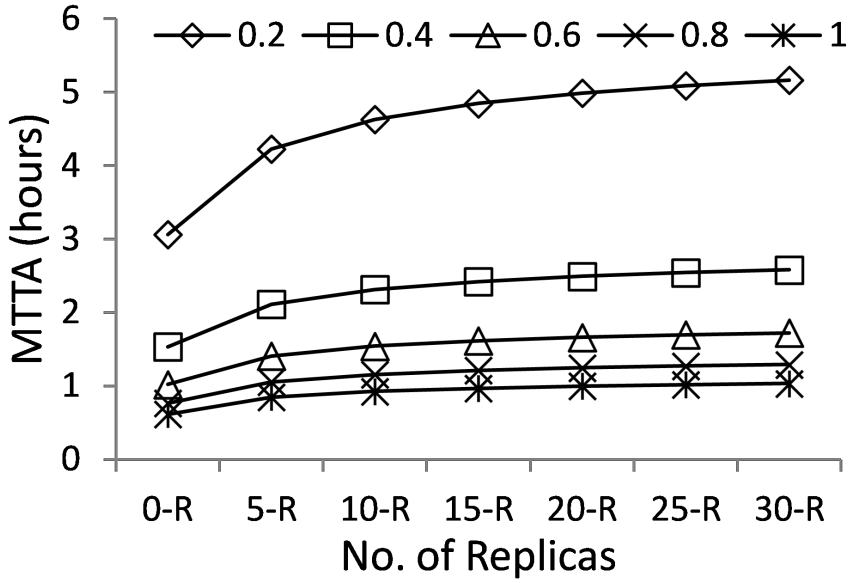


Figure 5.16: Mean-Time-To-Attack with Respect to *Redundancy*

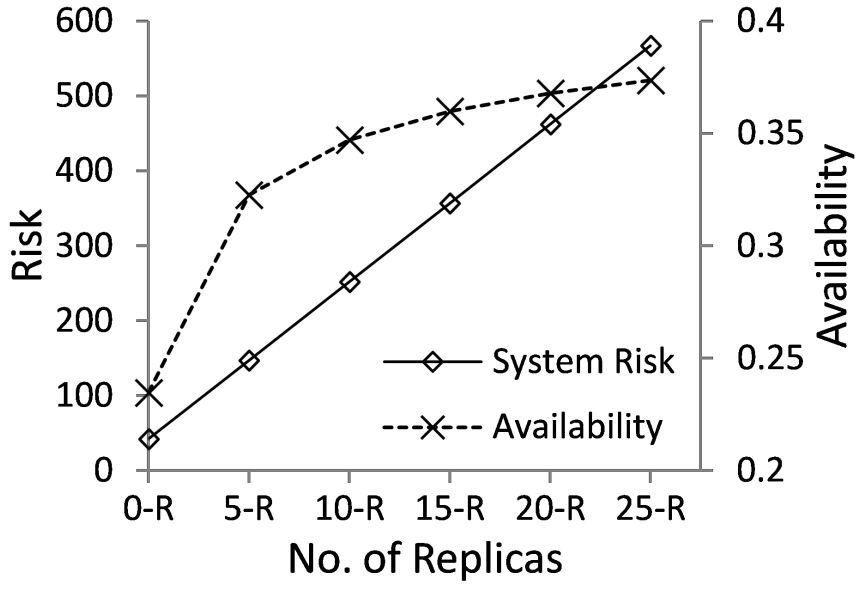


Figure 5.17: Risk and Availability with Respect to *Redundancy*

ploying *Redundancy*. The system risk increases linearly with respect to the number of replicas, but reliability and availability increase logarithmically. Hence, one can optimise the trade-off between system risk and reliability. However, optimising security and performance when deploying *Redundancy* is out of scope for this chapter.

5.5 Discussion

Adaptability of the HARM is studied in this chapter, taking into account changes in the networked system using MTD techniques. One application of adaptability is to deploy the most effective MTD technique in a large sized networked system, which is difficult without a comparative security analysis of various MTD techniques. Moreover, the effectiveness of using the IMs is described when deploying the *Shuffle* technique in comparison to the ES method. Changes in the performance (e.g., ECC) and security (e.g., system risk, attack cost) for the *Diversity* assignments are analysed, as well as changes in performance (e.g., reliability) and security (e.g., system risk) when deploying the *Redundancy* techniques. This section addresses some of the limitations of the HARM and its adaptability.

5.5.1 Validation using a Real System

Although the networked system is modelled as close to the real network settings, one of the major limitations is the lack of implementation in a real testbed for validation. There is a limited number of previous works that used a testbed or a practical network [60, 104, 147, 192].

5.5.2 Incorporating Various Vulnerabilities

Only the OS vulnerabilities are taken into account for simplicity, but various vulnerabilities can be incorporated and modelled in the HARM. For example, application vulnerabilities can be incorporated by creating another lower layer in the HARM, or combined with OS vulnerabilities. If vulnerabilities from different layers are related (e.g., an application layer is a precondition of an OS layer vulnerability), then their relationship can be captured in the HARM with post and preconditions.

Zero-day vulnerabilities (or also known as unknown vulnerabilities) are not taken into account for adaptability of the HARM with other known vulnerabilities. However, incorporating zero-day vulnerabilities is an additional function to the security models [14, 99, 198, 199]. Incorporating zero-day vulnerabilities in the HARM can be done by adding functions to analyse zero-day vulnerabilities (i.e., use security model in the layer capable of analysing zero-day vulnerabilities), which will be further described in chapter 8.

5.5.3 Optimisation of the IMs w.r.t the MTD techniques

Various network scenarios are not taken into account for the adaptability of the HARM (e.g., network topologies, the number of hosts and VMs, and the number of vulnerabilities) and how the effectiveness of the MTD techniques are affected when using the IMs. For security analysis, a nearly equivalent security solution to the ES method was computed for various network scenarios using the IMs [89], but it is difficult to validate without conducting experiments with various network scenarios.

5.5.4 Optimisation between performance and security metrics

Experiments were conducted using both performance (e.g., ECC, reliability) and security (e.c., system risk, attack cost) metrics. During the experiments, the degrading performance metrics in exchange for an increase in security metrics is observed. Conversely, improving the performance degrades the security. However, one can optimise both metrics to minimise the loss of both performance and security. For example, considering the experiment for *Diversity* assignments in Section 5.3.3, a random assignment provided a poor performance as well as poor security compared to both ECC-optimal and Risk-optimal solutions. Therefore, optimising both performance and security should be considered.

5.5.5 Combining multiple MTD techniques

Different MTD techniques can be combined to enhance the security of networked systems (e.g., *Shuffle* and *Redundancy* combined together to thwart DDoS attack [110], and *Diversity* and *Redundancy* combined together to enhance availability and reliability [76]). However, the combined MTD techniques are not con-

sidered for analysis, where the effect on security cannot be determined (as well as performance). However, modelling all possible scenarios of MTD deployment is an NP-Hard problem, and there is still a lack of modelling all possible attack scenarios in a scalable way.

5.6 Related Work on MTD Techniques

MTD Framework: One of the focus for adaptability of security models is to deploy the most effective MTD technique (i.e., the resulting state of the networked system after applying a MTD technique) based on security analysis using the HARM. Zhuang *et al*. [217] presented a MTD system that changes the configuration of a network proactively, and they compared a simple MTD system (random adaptations) with an intelligent MTD system (adaptation based on attack detection alerts) in [216]. However, their focus was to see how frequent the MTD technique should be applied, while the security of assigning the MTD technique is not assessed. Manadhata [133] introduced a two-player stochastic game model to determine an optimal MTD strategy with a method to quantify the shift in the attack surface, where the attack surface metrics are from [134]. Their focus is to reduce the attack surface based on the trade-off between security and usability, where the effectiveness of the MTD techniques is not considered. Crouse *et al*. [48] used a Genetic Algorithm to find temporally and spatially diverse configurations, which increased the spatial diversity and reduced their defined vulnerability scores. However, without a proper security assessment (e.g., using a security model), one cannot guarantee that the spatially diverse configurations are secure. Evans *et al*. [60] classified diversity MTD techniques with respect to four different types of attack strategies. However, they did not take into account any specific network and its configuration.

MTD Applications:A wide range of applications use MTD frameworks, including a Self-shielding Dynamic Network Architecture (SDNA) [209], security layer (diversity on cryptosystems) and physical layer (diversity on firmware) of networked systems [39], and Advanced Adaptive Applications environment [163].

MTD techniques: MTD techniques can be deployed in various layers of the networked system as shown in Table 5.1, and they can improve the MTD framework. However, these work did not consider the effectiveness of their schemes.

Some of the most recent MTD techniques are enlisted, where their effectiveness could be measured using the method described in this chapter.

Shuffle: System settings in various layers are rearranged when *Shuffle* is deployed. At the TCP/IP layer, Jafarian *et al*. [104] showed changing IP addresses in a software defined network (SDN), with their major goal of maximising the unpredictability and the mutation rate. Antonatos *et al*. [22] shuffled IP addresses, with a specified objective to harden networks against Hitlist Worms. At the infrastructure layer, Danev *et al*. [52] used A VM-LM in private clouds with focuses on integrity of the software prior to the VM-LM, and Zhang *et al*. [212] considered a VM-LM in clouds with focuses on practicability considering the availability and duration of the VM-LM. Okhravi *et al*. [155] showed the environment migration (e.g., migrating applications to a new host on a different operating system (OS)). At the application layer, Vikram *et al*. [192] randomised HTML elements to mitigate web bots. Jia *et al*. [109] showed the secure service access for clients by relocating secret proxies and shuffling client-to-proxy assignments.

Diversity: Equivalent functionalities are maintained, but the implementations vary in various layers when *Diversity* is deployed. At the topology layer, Rohrer *et al*. [170] formalised family of metrics for path diversity (e.g., reliability and resilience) and proposed path diversification selection algorithm. Newell *et al*. [147] assigned diversity to routing nodes in the networked system to increase the ECC with an optimal diversity assignment solution for medium-sized networks, and a greedy approximation solution that scales to large networks. Also, they solved the DAP by assigning an appropriate variant to new nodes in the networked system. At the application layer, Glynis *et al*. [75] diversified active software components to change their implementation versions and resources continuously. Jackson *et al*. [103] used a compiler based software diversity technique (two orthogonal compiler-based techniques) to automatically create multiple functionally equivalent, but internally different variants of a program. Huang *et al*. [96] automatically created a set of virtual servers with a unique software mix (e.g., web server program, web application programs, OS, and virtualisation layer). Nguyen *et al*. [148] used a data diversity to increase the difficulties for attackers to compromise the system, whereas Williams *et al*. [204] used it on VMs. Cox *et al*. [47] applied address space partitioning and instruction set tagging as a *Diversity* example, but can be extended with different methods in other layers of the system.

Redundancy: The redundancy technique produces multiple replications of network components to enhance the performance. At the topology layer, Kirrmann and Dzung [117] used *Redundancy* on any level of the communication stack (e.g., physical, link, or network layers). Al-Wakeel and Al-Swailemm [10] used path redundancy to mitigate sensor network attacks by using alternative routing paths for data transmission. At the application layer, Yuan *et al*. [211] used a software redundancy (examples used are web server and database redundancy) for the system architecture to adapt to the attack pattern in the event of an attack. Chang *et al*. [41] used data redundancy to replicate storage data to provide reliability and availability in case of disasters. Huang *et al*. [95] used server redundancy to increase service availability and dependability. Gorbenko *et al*. [76] used service redundancy to enhance the availability and reliability of the networked system.

Security modelling and analysis: Many studies attempted to show the effectiveness of using MTD techniques. However, existing studies did not use a formal security model to analyse the security, so different MTD techniques cannot be compared to determine the most effective solution. Often in a large sized networked system, not all network components can be secured due to constraints and limited resources. Hence, the MTD techniques are incorporated into the HARM for security modelling and analysis to measure their effectiveness, which are comparable using the same security (as well as performance) metrics. In addition, the IMs are used [89, 90] to deploy MTD techniques in an efficient way.

5.7 Conclusions

MTD is a new paradigm of network security that continuously changes the attack surface to prevent cyber crimes and thwart attacks. By doing so, potential socio-economic impact on enterprises and individuals can be minimised, as well as protecting important assets and critical infrastructures. However, a major problem of adopting MTD techniques is the inability to guarantee that the security is enhanced by changing the attack surface. Therefore, it is essential to assess the change in the security prior to deploying any MTD techniques. However, the effectiveness of deploying various MTD techniques cannot be compared to one another, because they did not consider using a formal security model to analyse them. Also, it is difficult to decide how to deploy the MTD techniques efficiently,

especially for a large sized networked system.

In this chapter, aforementioned problems are addressed by incorporating MTD techniques *Shuffle*, *Diversity* and *Redundancy* into the HARM to analyse the security change associated with deploying them. A formal security analysis of MTD techniques using various performance and security metrics is described, which are used to compare their effectiveness. Also, the IMs are used to select highly important network components (e.g., hosts and vulnerabilities) to deploy MTD techniques, and a significant improvement using the IMs (in terms of scalability) is shown in comparison to the ES method in the experiments. Moreover, the experimental results showed that the effectiveness of MTD techniques can be assessed and compared, as well as the changes in performance (e.g., ECC, reliability and availability) and security (e.g., system risk and attack cost) to evaluate the trade-offs between those metrics prior to deploying MTD techniques.

This chapter concludes the properties of the HARM and its adaptability with respect to MTD techniques, whereby existing security models significantly lack in complexity and performance analysis.

Part III

Importance Measure based Security Assessments

Chapter 6

Attacker Located Outside the Networked System

Although structural advances in security models improve the evaluation of all possible attack scenarios [88, 100, 158, 207], assessing the security without approximation for a very large networked systems are infeasible [87, 90, 93]. Existing approximation methods are based on specific metrics and models, which limits the scope of possible security analysis. To address this limitation, importance measure (IM) based security analysis methods are developed [90], where the HARM is generated based using only the important components in the networked system. IMs are based on the properties of the networked system, which does not depend on metrics or models. Also, they can be used by other security models to conduct equivalent security analysis. Further, the accuracy of IM-based security analysis is nearly equivalent to the security analysis based on all possible attack scenarios [86, 89, 90], while the performance of using the IMs is improved significantly (shown in Section 6.4).

In this chapter, methods to generate the HARM based on IMs are described [90]. The attacker is assumed to be outside of the networked system. IMs are further specified into important hosts and vulnerabilities in the networked system. The idea is to use k-importance measures to generate a 2-HARM, where k_1 is the number of important hosts based on network centrality measures (NCMs), and k_2 is the number of important vulnerabilities in the hosts based on security metrics. First, computing IMs is described in Section 6.1. Second, k_1 and k_2 are used to analyse the security of networked systems in Section 6.2, then combining the k_1 and k_2 measures to compute a prioritised set of vulnerabilities is described in Section 6.3. Lastly, simulation results in Section 6.4 show a significant improvement in the performance using the IMs, while the accuracy of the IMs with respect to analysing all possible attack scenarios is nearly equivalent to the ES method used in Chapters 4 and 5.

6.1 Computing IMs

The IMs are further categorised into (i) ranking important hosts in the networked system, and (ii) ranking important vulnerabilities in the hosts. Important hosts are ranked based on NCMs (e.g., degree, closeness, and betweenness centrality measures [38]), and important vulnerabilities are ranked based on their associated metrics (e.g., CVSS BS [70]).

6.1.1 Ranking Important Hosts

The reachability information of the networked system is used in conjunction with NCMs to rank important hosts. NCMs identify the characteristics of network components based on the structure. The structure of the networked system is important in a cyber attack, as some attacks (e.g., sequential attacks) progress based on how the network components are connected. As a result, NCMs can be used to distinguish attack paths that are most likely to be used in an event of an attack. Security metrics reflect characteristics of vulnerabilities in the hosts. Security metrics can be measured from real systems [181], cloud systems [215], emulations [143], and honey pots [11].

Among many NCMs, only the basic NCMs are used (e.g., degree, closeness and betweenness centrality measures) [38]. The degree centrality computes the popularity of a node (e.g., a host in a networked system) based on the number of direct connections with other nodes (e.g., single-hop neighbour hosts), with its computational complexity of O(n) where n is the number of nodes in the graph. Equation (6.1) shows the computation of the degree centrality, NC_{degree}, for a node n_i, where $deg(n_i)$ is the number of edges connected to n_i. The closeness centrality computes the distance of a node to all other nodes, with its computational complexity of O(n^3) using Floyd algorithm [64]. Equation (6.2) shows the computation of the closeness centrality, $NC_{closeness}$, for a node n_i, where g is the total number of nodes and $d(n_i, n_j)$ is the existence of an edge between node n_i and n_j. The betweenness centrality computes the significance of a node between all node pairs, with its computational complexity of O(n^3) using Floyd algorithm. Equation (6.3) shows the computation of the betweenness centrality, $NC_{betweenness}$, for a node n_i, where g_{jk} is the number of geodesics connecting jk, and $g_{jk}(n_i)$ is the

number the number of geodesics connecting jk that includes n_i.

$$NC_{degree}(n_i) = deg(n_i) \tag{6.1}$$

$$NC_{closeness}(n_i) = \left[\sum_{j=1}^{g} d(n_i, n_j)\right]^{-1} \tag{6.2}$$

$$NC_{betweenness}(n_i) = \sum_{j<k} \frac{g_{jk}(n_i)}{g_{jk}} \tag{6.3}$$

The normalised NCMs of the example networked system (Figure 2.3) is shown in Table 6.1, where high numeric values represent the higher importance. Each NCM ranks are equivalently weighed and combined to give the overall rankings of the hosts.

Table 6.1: NCMs of Hosts in the Example Networked System

NCM	U_1	U_2	U_3	U_4	U_5
Degree	0.67	0.5	0.67	0.67	0.333
Closeness	0.71	0.56	0.71	0.71	0.45
Betweenness	0.6	0.3	0.5	0.7	0.3
Sum	1.98	1.36	1.88	2.08	1.08
Rank	2	4	3	1	5

Another importance of using NCMs is to compute the proportion of important hosts (i.e., the value of k_1). The density of the network (i.e., the proportion of host interconnections in respect to the number of hosts) is one of the important factors, because the number of available attack paths is proportional to the network density (i.e., there are more available attack paths in a dense network (e.g., fully connected or mesh topologies) than a sparse network (e.g., star or tree topologies)). The closeness centrality is used to compute the value for k_1, because the closeness centrality directly measures the amount of host connections in the networked system. In a fully connected network, the sum of normalised degree centrality measure is equal to the number of hosts in the network (i.e., all network components are used in at least one attack path). The sum of degree centrality for the example networked system is three (out of five), hence $k_1 = 3$ for the example.

Table 6.2: Vulnerability Rankings of W7 Hosts

	$W7_1$	$W7_2$	$W7_3$	$W7_4$	$W7_5$
CVSS BS	7.5	7.5	5.8	9.3	4.3
Rank	2	2	4	1	5

Table 6.3: Vulnerability Rankings of REL Hosts

	REL_1	REL_2	REL_3	REL_4	REL_5	REL_6
CVSS BS	2.6	4.3	10.0	7.5	5.0	10.0
Rank	6	5	1	3	4	1

6.1.2 Ranking Important Vulnerabilities

Various security metrics evaluate different aspects of vulnerabilities. Values are assigned to security metrics (e.g., CVSS BS [70]) and these values are relative to each other. For example, CVSS BS is used to rank vulnerabilities. The rank based on CVSS BS is shown in Table 6.2 and Table 6.3 for W7 hosts (i.e., hosts with W7) and REL hosts (i.e., hosts with REL), respectively. The proportion of important vulnerabilities is chosen by their CVSS BSes. The average CVSS BS is calculated, and the vulnerabilities with the CVSS BS higher than the average are selected (i.e., the k_2 value is computed with the threshold value based on the average CVSS BSes). The average CVSS BSes are 6.88 and 6.57 for W7 hosts and REL hosts, respectively. Therefore, in this example, the value of $k_2^{W7} = k_2^{REL} = 3$, where k_2^{OS} denotes the number of important vulnerabilities for the hosts with *OS*.

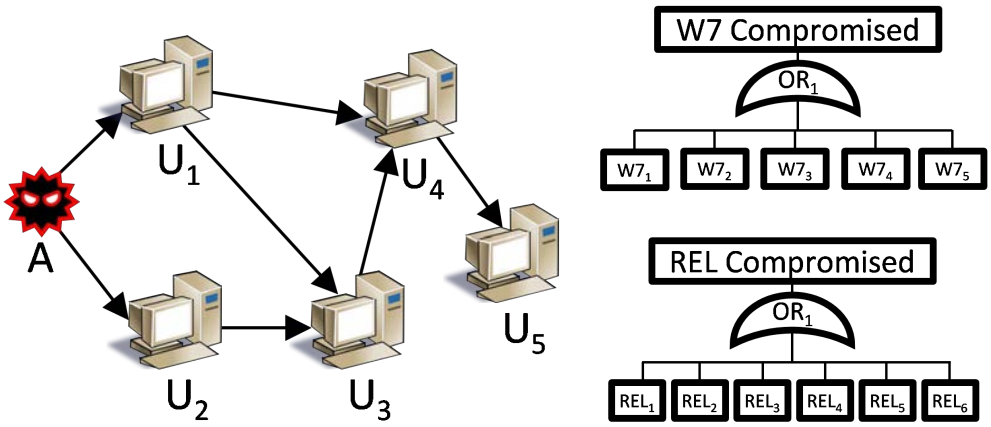
6.1.3 Generating the HARM

Generating the HARM Using All Network Information

First, the 2-HARM is generated where an AG is used in the upper layer, and an AT is used in the lower layer, as shown in Figure 6.1.

Generating a Reduced HARM using k Importance Measures

Second, a reduced HARM (denoted as *ReHARM*) is generated based on *k*-importance measures shown in Sections 6.1.1 and 6.1.2, as shown in Figure 6.2.



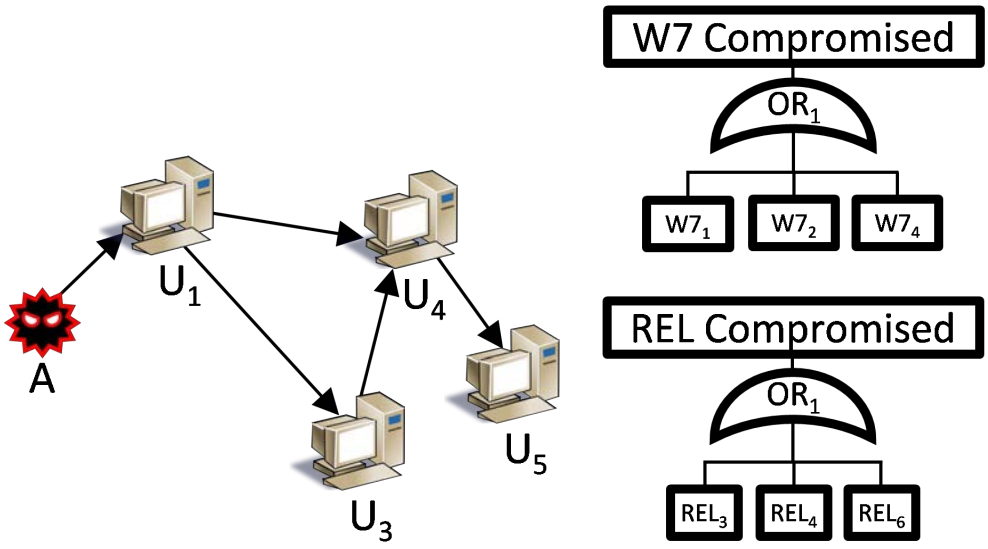
(a) Upper Layer with an AG (b) Lower Layer with an AT

Figure 6.1: The 2-HARM of the Example Networked System

The size of the ReHARM is significantly smaller than the HARM. The selected important hosts with $k_1 = 3$ are U_1, U_3, U_4. Since U_5 is the target, it is included in the upper layer. A full HARM can also be generated first, and then take into account important hosts and vulnerabilities to generate a ReHARM. This is useful if a full HARM is already generated, otherwise generating the ReHARM directly from the preprocessing phase is more efficient.

6.2 Security Analysis using IMs

There are various approximation methods using security models, such as model simplifications (e.g., graph aggregation [151], adjacency matrix clustering [152], graph simplification via collapsing similar nodes [100]) and heuristic methods (e.g., Particle Swarm [7], new heuristic algorithms [102]) are used to improve the scalability, but these methods require a security model generated specific to their evaluation method using all network information. On the other hand, k-importance measures are used to generate security models and evaluate the security of networked systems using only those selected hosts and vulnerabilities to improve the scalability.



(a) Upper Layer (b) Lower Layer

Figure 6.2: The ReHARM of the Example Networked System

6.2.1 Risk Analysis using IMs

Risk Analysis using the HARM: The risk associated with each attack path using the HARM is analysed with all details (e.g., a full HARM), denoted by R_{ap}. The computation of the risk is shown in equation (2.1). Note that it is possible to apply different security analysis by adopting different methods or even different security model in the lower level of the HARM.

The risk associated with W7 hosts is shown in equation (6.4), and the risk associated with REL hosts is shown in equation (6.5). The CVSS BS is taken into account as the risk value of each vulnerability, but other measurements can also populate risk values of these vulnerabilities.

$$R_{W7} = max(7.5, 7.5, 5.8, 9.3, 4.3) = 9.3 \tag{6.4}$$

$$R_{REL} = max(2.6, 4.3, 10, 7.5, 5.0, 10) = 10 \tag{6.5}$$

Then, all possible attack paths are computed in the upper layer of the HARM shown in Figure 6.1 based on the ES method. The list of all possible attack paths

Table 6.4: Risk Analysis of Attack Paths

Path Number	Attack path	R_{ap}
ap_1	$U_1U_4U_5$	28.6
ap_2	$U_1U_3U_4U_5$	38.6
ap_3	$U_2U_3U_4U_5$	38.6

Table 6.5: Risk analysis of attack paths using ReHARM

Path Number	Attack path	R^*_{ap}
ap^*_1	$U_1U_4U_5$	28.6
ap^*_2	$U_1U_3U_4U_5$	38.6

and the risk associated with each attack path (denoted as ap) are summarised in Table 6.4. Each attack path is presented with sequences of the hosts. The highest risk of an attack path is 38.6 (i.e., paths ap_2 and ap_3).

Risk Analysis using the ReHARM: ReHARM is used to evaluate the system risk. The risk analysis using ReHARM is denoted as R^*_{ap}. The risk associated with W7 hosts is shown in equation (6.6), and the risk associated with REL hosts is shown in equation (6.7). Similarly, the CVSS BS is taken into account as the risk value of each vulnerability.

$$R^*_{W7} = max(7.5, 7.5, 0.3) = 9.3 \tag{6.6}$$

$$R^*_{REL} = max(10, 7.5, 10) = 10 \tag{6.7}$$

Based on the ReHARM as shown in Figure 6.2, all possible attack paths are computed using the ES method. Table 6.5 shows the risk analysis based on the ReHARM. The highest risk value is 38.6 (from paths ap^*_2), which is the same result as the HARM shown in Table 6.4.

6.3 Combining the Importance Measures

Section 6.2 showed that the significant attack path can still be computed by identifying important hosts and vulnerabilities. One application for identifying

significant attack paths is to compute a prioritised set of vulnerabilities (PSV) to patch, which is one of the most important tasks to harden the networked system [34, 40]. This set determines the order of vulnerabilities to be patched, which is based on their damaging effects in a networked system. A study shows that up to a 95% of security breaches could have been prevented if the systems were up-to-date [72]. National Institute of Standards and Technology (NIST) also provided vulnerability management program for security managers and system administrators with a guidance to managing vulnerabilities and patching them [139, 193]. Hence, this section focuses on computing the PSV to patch as one of the network hardening solutions. A major drawback of existing security models is that security solutions are optimised for only the current state of the networked system [93]. As a result, the security model must reanalyse the security of the networked system every time a vulnerability has been patched, causing multiple iterations of the same task to obtain the PSV. In contrast, IMs rank network hosts and vulnerabilities, and the rank is used directly as the PSV, which only requires a single computation.

The PSV is defined as a set of vulnerabilities in which they are most important to enhance the security of the networked system (e.g., to minimise the system risk). One can use security models to analyse the security [88, 100, 158], as they can compute various security metrics (e.g., probability of an attack success, impact, risk, return on investment) [12, 173] that will result in various PSV. A naive approach is to compute all possible attack scenarios using the ES method. However, computing all possible attack scenarios has an exponential computational complexity due to exponentially growing number of attack paths with different node combinations [124]. Therefore, it becomes infeasible to compute the PSV in a large sized networked system (e.g., an enterprise system) using the ES method.

6.3.1 Security Analysis using the ES Method

To compare the result of using the combined IMs, the ES method is first used to find the PSV taking into account two security metrics, system risk (R) and attack cost (C). The algorithm using the ES method is shown in Figure 6.3. This algorithm is also used to compute the attack cost by replacing the risk variables with cost variables, and taking into account the minimum cost (i.e., line 11). De-

tails of these calculations can be found in [93].

```
1:  procedure EXHAUSTIVESEARCHRISK
2:      initialize system risk (R) → 0
3:      compute R
4:      solution set (S) → {}
5:      while R > 0 do
6:          current risk (CR) → 0
7:          solution → none
8:          for all vulnerability ∈ network do
9:              try patching vulnerability
10:             compute new R
11:             if new R > CR then
12:                 CR = R
13:                 solution → vulnerability
14:             end if
15:         end for
16:         R = CR
17:         append solution to S
18:     end while
19: end procedure
```

Figure 6.3: ES Method using the Risk Metric

Risk Analysis

Table 6.6 shows the PSV for the example networked system based on the risk metric using the algorithm shown in Figure 6.3.

Table 6.6: Risk-based PSV using the ES Method

ID	$W7_4$	REL_3	REL_6	REL_4	REL_5	REL_2
Rank	1	2	2	4	5	6
ID	REL_1	$W7_1$	$W7_2$	$W7_3$	$W7_5$	
Rank	7	8	8	10	11	

Cost Analysis

The attack cost, C, is defined as in equation (6.8), and the path cost PC as in equation (6.9). The host cost HC is calculated by evaluating the lower layer. The attack cost calculation of the example networked system is shown in equation (6.10), where MIN is the minimum PC_i. An assumption is that the CVSS BS is taken into account as the attack cost, and the maximum attack cost for any vulnerability is 10. However, other means of populating the attack cost values can be adopted. Using the method described, Table 6.7 shows the PSV based on the attack cost metric.

$$C = MIN(PC_i, i \in \{\text{all possible attack paths}\}) \tag{6.8}$$

$$PC_i = SUM(HC_j, j \in \{\text{hosts in attack path } i\}) \tag{6.9}$$

$$\begin{aligned}
C =& MIN(PC_i, i \in \{\text{All possible attack paths}\}) \\
C_{example} =& MIN(PC_{\{U_1U_4U_5\}}, PC_{\{U_1U_3U_4U_5\}}, PC_{\{U_2U_3U_4U_5\}}) \\
=& MIN(SUM(HC_{U_1}, HC_{U_4}, HC_{U_5}), \\
& \quad SUM(HC_{U_1}, HC_{U_3}, HC_{U_4}, HC_{U_5})) \\
& \quad SUM(HC_{U_2}, HC_{U_3}, HC_{U_4}, HC_{U_5})) \\
=& MIN(SUM(4.3, 4.3, 2.6), SUM(4.3, 2.6, 4.3, 2.6), \\
& \quad SUM(4.3, 2.6, 4.3, 2.6)) \\
=& 11.2
\end{aligned} \tag{6.10}$$

Table 6.7: Attack Cost based PSV using the ES Method

ID	$W7_5$	$W7_3$	REL_1	REL_2	REL_5	REL_4
Rank	1	2	3	4	5	6
ID	$W7_1$	$W7_2$	$W7_4$	REL_3	REL_6	
Rank	7	7	9	10	10	

6.3.2 *Security Analysis using the Combined IMs*

Top-Down (TD), Bottom-Up (BU) and Hybrid (HB) methods are taken into account using the combined IMs, which are compared with the ES method. IMs shown in Section 6.1 are used, where both risk and cost metrics are based on the CVSS BS for vulnerabilities.

PSV using the TD Method

The preference of the ranks is prioritised based on the host importance, followed by the vulnerability importance. First, the important hosts are sorted by their NCM values. For the example networked system, the order is U_4, U_1, U_3, U_2 and U_5. For each host, vulnerabilities are ranked based on the security metric used. Table 6.8 shows the top five vulnerabilities in the PSV based on the TD method. *ES Rank* represents the vulnerability rankings from using the ES method. The result shows that the PSV using the TD method does not match well with the ES method. Considering the top five vulnerabilities to patch, only one highly ranked vulnerability (based from the ES method) is in the risk-based PSV (20% included), and two in the cost based PSV (40% included). The top priority vulnerability is correctly ranked for the risk-based PSV, but not for the cost based PSV.

Table 6.8: A List of PSV using the TD Method

Risk Rank	ES Rank	ID	Cost Rank	ES Rank	ID
1	1	$W7_4$	1	9	$W7_4$
2	8	$W7_1$	2	7	$W7_1$
3	8	$W7_2$	3	7	$W7_2$
4	10	$W7_3$	4	2	$W7_3$
5	11	$W7_5$	5	1	$W7_5$

PSV using the BU Method

The preference of the ranks is prioritised based on the vulnerability importance, followed by the host importance. All vulnerabilities are ranked first, then

the list is reordered by host ranks when there are conflicting vulnerability values. The top five vulnerabilities in the PSV using the BU method is shown in Table 6.9. The result shows that the BU method matches reasonably well with the ES method. However, some low *ESRank* vulnerabilities are still ranked high, because the attack path information is not taken into account. Considering the top five vulnerabilities, three highly ranked vulnerabilities are the risk-based PSV (60% included) and all five are in the cost based PSV (100% included). The top priority vulnerability is not correctly ranked for both risk and cost based PSV.

Table 6.9: A List of PSV using the BU Method

Risk Rank	ES Rank	ID	Cost Rank	ES Rank	ID
1	2	REL_6	1	3	REL_1
2	2	REL_3	2	1	$W7_5$
3	1	$W7_4$	3	4	REL_2
4	8	$W7_1$	4	5	REL_5
5	8	$W7_2$	5	2	$W7_3$

PSV using the HB Method

The ranking is based on the values calculated from assigning weights to values of hosts and vulnerabilities and combining them. All IM values are normalised using the largest value as a factor. The combined IM values, CV_{vul}, are calculated for each vulnerability *vul* as in equation (6.11), where $0 \leq \alpha \leq 1$ is the weight value. NS_{vul} is the host value for the host containing *vul*, and VS_{vul} is the vulnerability value of *vul*. $\alpha = 0.5$ is selected as a default value in this example. The top five vulnerabilities in the PSV using the HB method is shown in Table 6.10. The result shows that the PSV computed using the HB method is similar with the PSV computed using the BU method. Considering the top five vulnerabilities, three highly ranked vulnerabilities are the risk-based PSV (60% included) and all five are in the cost based PSV (100% included). The top priority vulnerability is not

correctly ranked for both risk and cost based PSV.

$$CV_{vul} = \alpha NS_{vul} + (1-\alpha)VS_{vul} \tag{6.11}$$

Table 6.10: A List of PSV using the HB Method

Risk Rank	ES Rank	ID	Cost Rank	ES Rank	ID
1	2	REL_6	1	3	REL_1
2	2	REL_3	2	4	REL_2
3	1	$W7_4$	3	1	$W7_5$
4	8	$W7_1$	4	5	REL_5
5	8	$W7_2$	5	2	$W7_3$

6.4 Simulation Results

6.4.1 Experiment 6A: Effectiveness of IMs

Simulations are carried out to investigate the effectiveness of security analysis using the IMs. The example networked system shown in Figure 4.1 is used for the simulations. The networked system consisted of 1000 hosts. 500 hosts were assigned in the DMZ network, 500 hosts were assigned in the Internal network, and one target host was assigned in the Database network. The attack scenario is for an attacker located outside the network to compromise the target host. All hosts were assigned with 10 vulnerabilities, where a single vulnerability (v_{root}) granted the admin privilege when exploited, two vulnerabilities (v^1_{user} and v^2_{user}) granted the user privilege, and the rest does not change the privilege status. To exploit v_{root}, the attacker must exploit either v^1_{user} or v^2_{user}. There is no restriction to exploit all other vulnerabilities.

Security Analysis Based on the Importance of Hosts The System Risk of the networked system is analysed, where different vulnerabilities are assigned with different impact values chosen reasonably (v_{root} with 10, v^1_{user} and v^2_{user} with 5, and the rest with 1). The probability of an attack success is assumed to be one for all vulnerabilities. The security analysis based on the ES method and IMs is

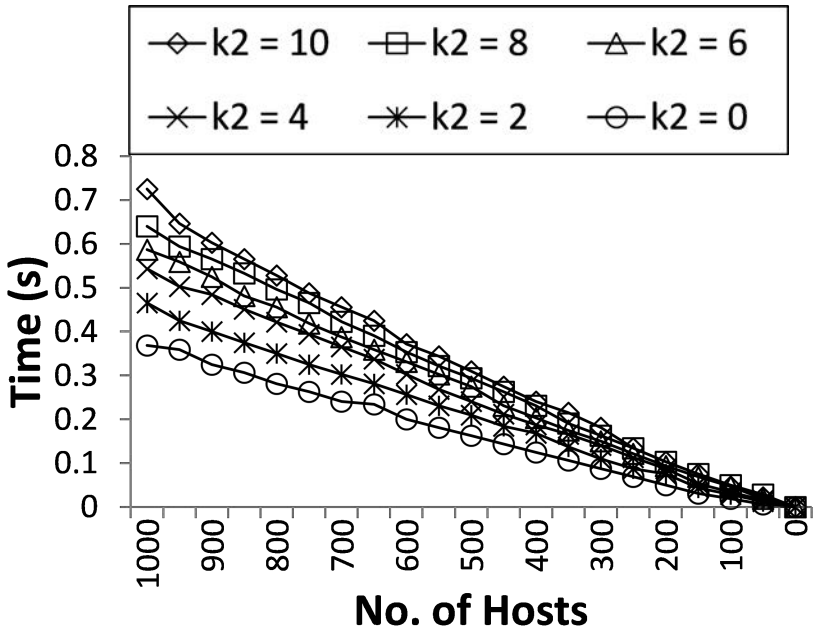
Table 6.11: Security Analysis using k_1 values ($k_2 = 10$)

No. of hosts	Generation (s)	Evaluation (s)	Risk value	No. of attack paths
1000	0.725	113.515	348	55942475
900	0.603	108.358	348	55942475
800	0.528	104.626	348	55942475
700	0.456	102.873	348	55942475
600	0.372	101.780	348	55942475
500	0.309	96.998	348	5699925
400	0.241	42.796	348	3274425
300	0.181	8.172	348	848925
200	0.103	0.250	0	0
100	0.047	0.172	0	0
0	0.0	0.0	0	0

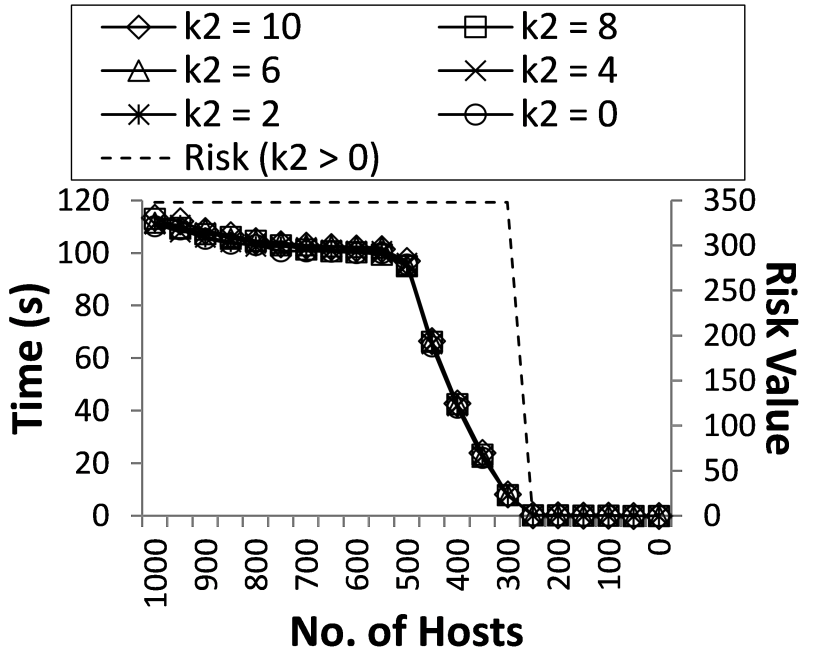
compared. Degree centrality was used to measure the importance of hosts (i.e., k_1). First, all network hosts are taken into account to compute the risk of the example networked system in the example. Then, the risk value is continuously computed by generating the ReHARM with varying values of k_1. As the number of hosts reduces, generation and evaluation times also reduce. The simulation result when $k_2 = 10$ is shown in Table 6.11. Generation and evaluation times are shown in Figure 6.4.

The density of simulation network is 0.006 (i.e., each host on average has a direct connection to six other hosts). The simulation result shows that the risk value is still equivalent when the network size has reduced by 70% (i.e., $k_1 = 300$). The generation time consistently reduces as the number of hosts decreases as shown in Figure 6.4(a). The evaluation time decreases steadily down to 50% of hosts modelled. When the number of attack paths reduced, the evaluation time decreases rapidly. When the number of hosts is reduced to 250, attack paths that were directly affecting the risk analysis have been removed, such that the risk output is misleading. This is shown in Figure 6.4(b). Also, changing number of vulnerabilities (i.e., k_2) does not affect the scalability of the evaluation phase. The optimal solution using degree centrality measures was found at $k_1 = 266$.

Security Analysis Based on Vulnerability Importance Vulnerabilities are



(a) Generation time using k_1 values



(b) Evaluation time using k_1 values

Figure 6.4: Performance of Security Analysis using k_1 values

Table 6.12: Security analysis using k_2 values ($k_1 = 1000$)

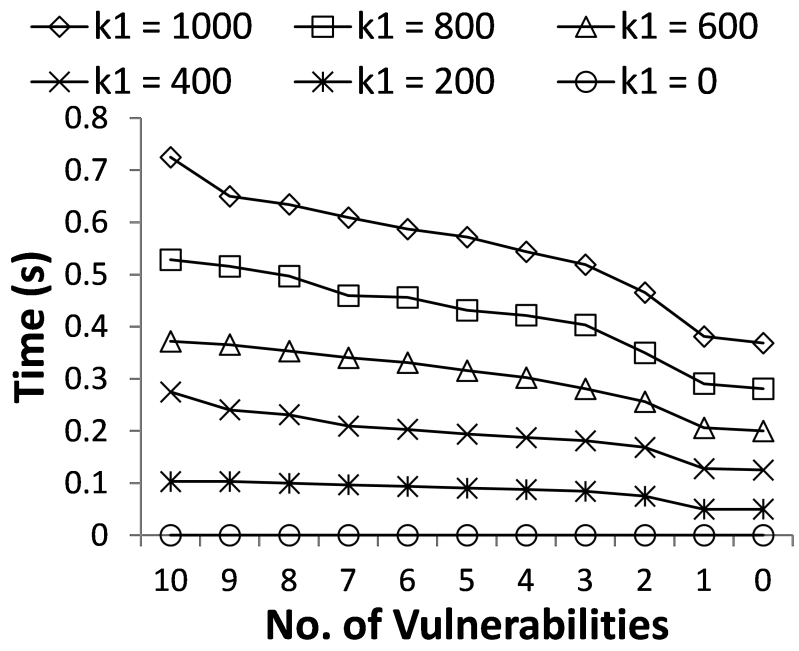
No. of Vulnerabilities	Generation (s)	Evaluation (s)	Risk value
10	0.725	113.515	348
9	0.650	110.796	348
8	0.634	110.576	348
7	0.609	111.343	348
6	0.587	111.608	348
5	0.572	111.858	348
4	0.544	111.108	348
3	0.519	111.920	348
2	0.466	111.218	348
1	0.381	110.264	3
0	0.369	110.233	3

ranked with given impact value information. All network hosts are modelled to investigate the performance of risk analysis when k_2 number of important vulnerabilities are considered in the risk analysis. The simulation result is shown in Table 6.12. Generation and evaluation times are shown in Figure 6.5.

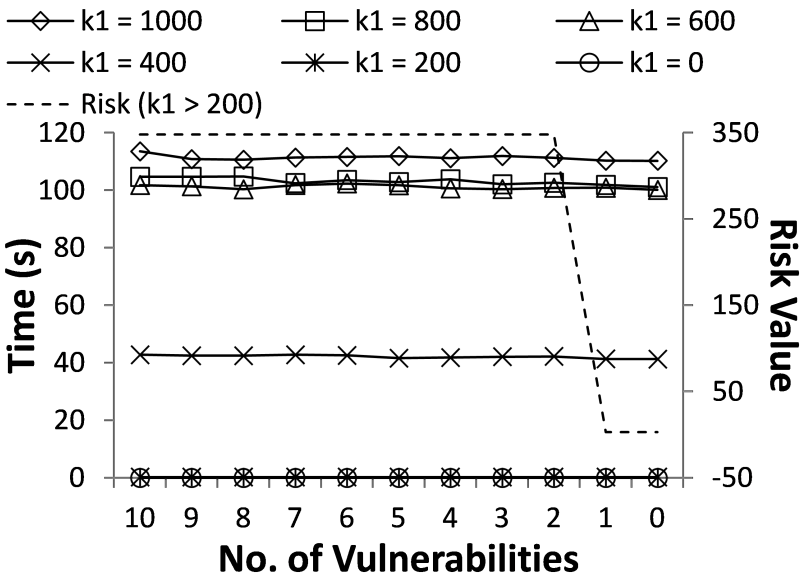
The generation time shows a constant improvement, because there are only a few number of components to generate in the lower layer. However, there are no improvements shown in the evaluation time for all k_1 values. It shows that k_2 values do not affect the performance of the evaluation phase for the HARM. If vulnerability models become more complex in the lower layer (i.e., multiple paths in exploiting vulnerabilities), the computational complexity of the lower layer may increases. The optimal solution is found when $k_2 = 2$. The naive solution compared to the optimal solution is shown in Table 6.13. The optimal solution shows approximately 87% generation time improvement and 99.5% evaluation time improvement respectively. However, the optimisation of IMs is out of scope in this chapter.

6.4.2 Experiment 6B: Effectiveness of Combined IMs

Further simulations are conducted to investigate the ranking of vulnerabilities in the PSV in different network scenarios using the risk analysis. The performance



(a) Generation time using k_2 values



(b) Evaluation time using k_2 values

Figure 6.5: Performance of Security Analysis using k_2 values

Table 6.13: Comparison of Naive and Optimal Solutions

	Generation (s)	Evaluation (s)	Risk value	No. of attack paths
Naive	0.725	113.515	348	5942475
Optimal	0.097	0.578	348	24255

and accuracy of computing the PSV are measured. The same example networked system in Section 6.4.1 is used for simulations (i.e., Figure 4.1), where the attack goal is to compromise a host in the database (DBs). The number of hosts ranges from 20 to 100, because the ES method timed out ($>$ 3 hours) when analysed using 200 hosts. To compare results, k-percentage of ranked vulnerabilities are selected for the PSV, for $k \in \{0.05, 0.25, 0.5\}$ (i.e., top 5%, 25% and 50%).

Scenario 1: Identical hosts and vulnerabilities

In this scenario, identical topology and vulnerability information are used (i.e., a homogeneous networked setting). The result showed that all combined IMs (TD, BU and HB) computed an equivalent solution to the ES method, but the performance is dramatically improved as shown in Table 6.14. Computational time for TD, BU and HB are equal because the algorithms used are the same.

Table 6.14: Time to Compute the PSV (in seconds)

No. of Hosts	20	40	60	80	100
ES	0.344	6.708	39.46	134.762	345.642
TD	0	0.031	0.073	0.112	0.204
BU	0	0.031	0.073	0.112	0.204
HB	0	0.031	0.073	0.112	0.204

Scenario 2: Varying number of hosts

In this scenario, the number of hosts in subnets are varied from Scenario 1, with the number of hosts in *Subnet*3 and *Subnet*4 are twice larger than in *Subnet*1 and *Subnet*2. The result showed an equivalent solution as in Scenario 1.

Scenario 3: Varying number of vulnerabilities

A different number of identical vulnerabilities are assigned to each host in an identical network topology in this scenario: (i) no vulnerabilities for *Subnet*1 hosts, (ii) two for *Subnet*2 hosts, (iii) four for *Subnet*3 hosts, (iv) six for *Subnet*4 hosts.

When $k \in \{0.05, 0.25\}$, all solutions were equivalent to the ES method. When $k = 0.5$, the solution did not match fully, but as the number of hosts increased, the set coverage (the percentage of vulnerabilities in the PSV) converged to 100% as shown in Table 6.15.

Table 6.15: Set Coverage in Scenario 3, $k = 0.5$

No. of Hosts	20	40	60	80	100
TD	97.14	80	73.68	100	100
BU	97.14	80	73.68	100	100
HB	97.14	80	73.68	100	100

Scenario 4: Varying vulnerability values

In this scenario, the network topology was the same with different risk values for different vulnerabilities. Two vulnerabilities are assigned for hosts in *Subnet*1 and *Subnet*2 with the risk value of five, and two vulnerabilities for hosts in *Subnet*3 and *Subnet*4 with the risk value of two. The result shows that the PSV of the TD method did not match well with the ES method, but others matched equivalent. However, as the number of hosts increased, the set coverage increased for the TD method (converging to 100%). Table 6.16 shows the result for $k = 0.25$, which showed the largest difference in the solution.

Table 6.16: Set Coverage in Scenario 4, $k = 0.25$

No. of Hosts	20	40	60	80	100
TD	45.45	42.86	41.94	41.18	59.15
BU	90.91	76.19	70.97	100	100
HB	90.91	76.19	70.97	100	100

Scenario 5: Varying the number of hosts, the number of vulnerabilities and vulnerability values

This scenario consists of mixed topologies, varying number of hosts and different vulnerabilities. Details are omitted due to space limitations. The result is shown in Table 6.17. The result showed that the PSV of the BU method did not match well for $k = 0.05$. The result shows that by increasing the size of the PSV by

20% for the TD and the HB methods, an equivalent solution to the ES method is observed for all k values. For a small sized PSV (e.g., $k = 0.05$), the size increase is not large (i.e., 20% of 5%, which is equivalent to $k = 0.06$, or 6%). Also, the different weight of the alpha value can give a better solution (e.g., $HB2$ at 40 hosts and $k = 0.25$, where $HB2$ is a HB method with $\alpha = 0.9$).

Table 6.17: Set Coverage in Scenario 5

	No. of Hosts	20	40	60	80	100
	TD	0	0	25	50	57.14
$k =$	BU	0	0	0	0	0
0.05	HB	0	0	0	0	0
	HB2	0	0	25	50	14.29
	TD	56.25	45.16	30.53	32.23	33.33
$k =$	BU	100	96.78	97.98	98.31	98.63
0.25	HB	100	96.78	97.98	98.31	98.63
	HB2	100	100	97.98	98.31	98.63
	TD	25	22.58	25	23.81	24.05
$k =$	BU	100	96.78	93.75	93.65	93.67
0.5	HB	100	96.78	93.75	93.65	93.67
	HB2	87.5	83.87	89.58	87.30	93.67

6.5 Discussion

IMs are used to generate a ReHARM, which is more efficient in both the generation and evaluation phases. A risk analysis showed that an equivalent security solution can be achieved compared to the ES method, while the size of the HARM is significantly reduced. Accuracy and performances of security analysis using the IMs are investigated through simulations with various network scenarios. The NCM effectively ranked important hosts based on the network topology, and nearly equivalent risk value is computed. Moreover, the time performance was also improved when generating and evaluating the ReHARM.

However, the security analysis of the example networked system showed that not all vulnerabilities have associated security metrics. Also, a single security

metric often does not capture various effects of vulnerabilities (e.g., a high CVSS BS, but a low structural importance), and other requirements to satisfy the success of an attack are not well defined (e.g., privilege requirements). Another point is that network topologies and attack goals determine which hosts in a network are important in an event of an attack. Also, the proportion of the network hosts unused in an attack also depends on the network density (e.g., a sparse network and a dense network), such that determining the value of k_1 is difficult. In addition, an attacker located inside the network reduces the coverage of the network, but NCMs cover all network hosts. Lastly, attack on less important hosts and vulnerabilities are not properly addressed. This section discusses some of these limitations presented in this chapter.

6.5.1 Vulnerabilities without security metrics

Using a vulnerability scanner *NESSUS* [28], about 60 vulnerabilities of a real host machine were reported. However, only 11 vulnerabilities had CVSS BS, which gives a set of security metrics associated with these vulnerabilities. There was a textual description of vulnerabilities (e.g., Vulnerability Synopsis and Vulnerability Description), but this is difficult to process automatically. Moreover, 10 vulnerabilities are scanned without any descriptions. Incomplete security data makes difficult to automate and analyse the network security. Other sources of vulnerability scanners and security metrics will be investigated in the future work.

6.5.2 Categorised vulnerability ranking

A single security metric cannot capture all aspects of vulnerabilities. The attack goal changes how vulnerabilities will be exploited by an attacker. For example, an attack goal of gaining the admin privilege defines a subset of vulnerabilities that must be exploited to achieve this, but an attack goal to hijack a communication of a host defines a different subset of vulnerabilities to achieve this goal. So, there needs a definition of vulnerability categories that satisfy different attack goals. Then vulnerabilities can be ranked within each subset. In addition, the ranking of vulnerabilities can be combined from vulnerability rankings based on various security metrics, such as CVSS BS rankings and structural importance rankings [173]. Improvements and effectiveness of categorising and combining

vulnerability rankings should be further investigated.

6.5.3 Network features for k_1 selection

The network topology defines possible attack scenarios. A dense network (e.g., fully connected or mesh network topologies) allows the attacker to take many different attack paths to reach the target, so that a large proportion of network hosts will be used in at least one attack path. In contrast, a sparse network (e.g., star or tree network topologies) limits the number of attack paths, and the number of unused hosts (i.e., hosts that does not benefit the attacker in any attack scenario) increases. Therefore, the value of k_1 will depend on the attack scenario and the network topology. The effect of different network topologies for determining the value of k_1 importance measures needs to be studied, and the relationship between the number of hosts and the value of k_1 for the same network topology needs to be taken into account.

6.5.4 Attack on less important hosts and vulnerabilities

By enforcing security only on important hosts and vulnerabilities allow attackers to exploit less important hosts and vulnerabilities, and security analysis based on important hosts and vulnerabilities cannot capture such attacks. One solution is that if all attack paths are covered with selected set of hosts and vulnerabilities, then any attack scenarios, regardless of using important or less important hosts and vulnerabilities, are covered. However, a naive approach to check the coverage of attack paths is computationally expensive (i.e., exponential number of attack paths need to be checked). More efficient method to cover all attack paths with a set of hosts and vulnerabilities will be studied in the future work.

6.5.5 Adjusting the weight between host and vulnerability IMs

The weight value of $\alpha = 0.5$ was used, which was reasonably selected. Results showed that using the combined IMs, a nearly equivalent solution to the ES method is computed. Further, the result showed different solution that can match more closely with the ES method using different weights. That is, assigning the weight value appropriately can improve the set coverage, as well as the order of

vulnerabilities in the PSV. However, optimising the value of the weight is out of scope in this thesis.

6.5.6 Order of vulnerabilities in the PSV

Although the set coverage was relatively high for various network scenarios, the actual order of vulnerabilities varies when combined IMs are used. To address this problem, one can perform a security analysis for the vulnerabilities in the PSV only, which does not consider all possible cases. The performance of this method is in fraction of the ES method.

6.6 Related Work on Scalable Security Analysis

Security analysis using all possible attack scenarios can cover all set of known attacks. Consequently, there are difficulties computing the PSV in practice due to the scalability problem. Various modelling techniques are proposed to improve the scalability of security models [87, 100, 158, 207], but computing all possible attack scenarios (e.g., full AG [182], attack response trees [218]) for a large sized network still suffers from a scalability problem [88, 124]. Model simplifications and heuristic methods are widely used to improve scalability in the evaluation phase, but not in the generation phase.

Attack scenarios are often used to generate security models [218]. Chen *et al*. [43] used *compact* AG, similar to a logical AG in [158], to find n-valid paths that has less than n steps to reach the target, where n denotes the number of stepping stone hosts in the network. Further, they defined a *weighted-greedy* algorithm to find the optimal security solution in the evaluation phase. Their experiment results clearly showed that covering a larger set of attack scenarios is computationally expensive. Mehta *et al*. [138] ranked AG components based on attack probabilities. A full AG is constructed (i.e., capturing all possible attack scenarios [182]), which requires computing exponential number of attack paths in a large sized network. AG components cannot be ranked until full AG is generated.

Other approaches consist of heuristic methods. Poolsappasit *et al*. [167] used Genetic Algorithm (GA) in Attack Graphs (AGs) to analyse the security. However, when considering a PSV, the changes in the networked system after patching a vulnerability is not captured. As a result, the process of security analysis

is repeated to find the next vulnerability to patch after updating the AG, which is similar to the ES method. Sawilla *et al*. [176, 177] used *AssetRank* to rank vulnerabilities using a dependency AG, which used partial cuts in the evaluation phase. Partial cuts divide network components by their importance, such that the relevance between a network component and the attack is decided based on the generated AG. However, the structure of the AG is heavily dependent on network reachability. That is, network structure affects how important network components are chosen in the AG. Various techniques (e.g., model simplification and heuristic methods) are proposed to improve scalability in the evaluation phase, but they did not consider reducing the computation complexity in the generation phase.

Importance measures are used in some application domains. Cadini *et al*. [38] used NCMs to capture strengths and weaknesses of network safety. Georgiadis *et al*. [71] described network security using NCMs, but only implications of NCMs are described. Gallon *et al*. [70] integrated CVSS framework with AGs to construct a CVSS AG, but the structure of the security model is the same with other AGs. Previous works using NCMs and security metrics to assess the performance network security were applied only in the evaluation phase of security models.

6.7 Conclusions

Security analysis allow system and security administrators to become aware of vulnerable network components and configurations, and security solutions can be enforced to enhance the security (e.g., computing the PSV). However, existing security models have a scalability problem when the size of the networked system becomes too large, such that computing the PSV to patch becomes a challenge. Generating a security model requires all the network information, but enforcing network hardening techniques may only change a subnet of the networked system. That is, not all network information is required for the security analysis, but only the important hosts and vulnerabilities. Existing security models analyse the security and provide solutions that are only for the current state of the networked system. To address these problems, IMs are used to improve the generation and evaluation of security analysis based on only the network information. $k_1$number of important hosts and k_2 number of important vulnerabilities are ranked and se-

lected to generate and evaluate the HARM. Moreover, k_1 and k_2 metric values are combined to give a single value that reflects both network and vulnerability information. The simulation results showed equivalent security solutions can be achieved using IMs in comparison to the ES method, while the performance has improved in both the generation and evaluation phases (in terms of time and computation requirements). Moreover, time and computation requirements can be optimised by selecting appropriate number of important hosts and vulnerabilities, which showed a significant improvement compared to the ES method.

Chapter 7

Attacker Located Inside the Networked System

Using the importance measures (IMs) in conjunction with the HARM can analyse the security of networked systems efficiently as described in Chapter 6. However, when evaluating different attack scenarios using the same security model (e.g., multiple attack host locations and target host locations), the importance of networked components varies in different attack scenarios. To address this problem, security analysis using the IMs is conducted in the evaluation phase, which adapts to different attack scenarios [89], which uses newly developed NCMs to identify important hosts when the locations of the attacker and target are specified.

As described in Chapter 6, it is infeasible to use existing security models to compute all possible attack scenarios to evaluate the security of networked systems. To overcome the scalability problem, especially for attack scenarios with known attacker and target locations, newly developed IMs are used, which are described in Section 6.2 [89]. In this chapter, two attack scenarios are evaluated and compared using these new IMs; (i) the attacker is outside of the networked system (as in Chapter 6), and (ii) the attacker inside the networked system with known location. In both cases, the attack goal is the same (i.e., to compromise the same target host assigned), but the attack scenarios are different. In the former case, the attacker must penetrate through all parts of the networked system, but in the later case, the specified locations of the attacker and the target limit all possible attack scenarios (i.e., only covers a subset of the networked system). In such case, existing NCMs may rank irrelevant hosts with respect to different attack scenarios. Hence, *location-based* IMs are developed to rank important hosts more accurately when modelling an attacker located inside the networked system.

7.1 Security Analysis with Existing IMs

The AG representation of hosts in the upper layer of the 2-HARM (e.g., Figure 2.4) can be seen as a reachability map of the networked system. Taking into account the overall security of the networked system (i.e., the source of an attack and the target is unknown), it is essential to rank hosts by their structural importance, because highly ranked hosts are more likely to be used in an attack. As a result, these hosts can be monitored and secured more intensively than others for higher detection and mitigation of attacks. But in case when locations of the attacker and the target are specified, NCMs could be misleading due to the attack scenario considering only a subset of the network. Thus, *location-based* IMs are developed to deal with location specified attack scenarios in section 6.2. Three NCMs are calculated; (i) degree, (ii) closeness, and (iii) betweenness centralities as described in Section 6.1, as well as the ranking of vulnerabilities based on security metrics (e.g., a CVSS BS).

7.1.1 Comparing against the ES method

ES method computes all possible attack scenarios, which are used to formulate the best countermeasures. Security solutions from ranking important hosts and vulnerabilities in terms of performance are compared against the ES method. First, all possible attack scenarios are computed using the host information only. A depth-first-search [187] can be used on the upper layer of the HARM to compute all possible attack scenarios. The sequence of an attack path is denoted logically (i.e., U_1 through U_4 to get U_5 is denoted as $U_1U_4U_5$), and all possible attack paths for the example networked system shown in Figure 2.3 are $\{U_1U_4U_5$, $U_1U_1U_3U_4U_5$, $U_2U_3U_4U_5\}$.

An assumption is that for a large sized networked system, frequently used hosts will greatly affect the security of the networked system. Table 7.1 shows the occurrence of hosts in all possible attack paths. The result shows the most frequently used hosts, and also the most important hosts, areU_5 (i.e., the target), as well as U_4. The target is not taken into account in the ranking. Since the attack scenario specifies the location of the attacker and the target, the IMs based on the NCMs cannot capture such details. As a result, the orders of important hosts are different.

Table 7.1: Occurrence of Hosts in All Possible Attack Paths

Hosts	U_1	U_2	U_3	U_4	U_5
Occurrence	2	1	2	3	3
rank	2	4	2	1	-
NCM rank	2	4	3	1	5

Secondly, the lower layer details are aggregated into the upper layer. Based on the lower layer information, the number of ways to exploit a host increases. Without combining the upper and the lower layer information, the rankings of vulnerabilities are solely dependent on their importance measures (e.g., CVSS score, probability of an attack success). A vulnerability of a host is denoted as U_i^{vul} for a vulnerability *vul* for a host U_i. When the lower layer information is aggregated to the upper layer, there are total 3750 different attack paths for the example networked system.

Similarly with hosts, vulnerabilities are assumed to have equal probabilities of an attack such that the most frequently used vulnerability has the highest threat. The overall occurrence of vulnerabilities is shown in Table 7.2. Since exploiting any of listed vulnerabilities has the same effect, they have the same ranking for each OS of the hosts. For the example, W7 has higher ranking than the REL. More complex example can be found in [89]. The ranking of vulnerabilities based on the occurrence can be sometimes misleading, because it did not consider any security metrics associated with them (e.g., risk, impact, return on investment (ROI), and return on assets (ROA) [173]). However, it gives an indication that a vulnerability with a high occurrence ranking can affect a large proportion of the networked system even if the impact value is low (i.e., a large number of attacks on a vulnerability with low impact value can accumulate to have a larger impact). Therefore, both the occurrence and security metrics should be used together to formulate more accurate rankings of vulnerabilities.

The ranking of vulnerabilities is significantly different to rankings shown in Section 6.1, because the locations of the attacker and the target are specified. Hence, there is a need for specialised IMs that can capture these details more precisely.

Table 7.2: Occurrence of Vulnerabilities in Attack Paths

Vulnerabilities	$W7_1$	$W7_2$	$W7_3$	$W7_4$	$W7_5$	
Occurrence	780	780	780	780	780	
Vulnerabilities	REL_1	REL_2	REL_3	REL_4	REL_5	REL_6
Occurrence	625	625	625	625	625	625

7.2 Location-based IMs

In the case where the locations of the attacker and the target are specified, using the existing NCMs could not effectively compute important hosts and vulnerabilities due to the attack scenario only covering a subset of the networked system. The degree centrality becomes ineffective, as hosts that are close to the attacker and the target will be used more frequently than the others. The closeness and betweenness centrality measures use all-pair shortest paths information. Based on this idea, an *Attacker to Victim Centrality* (AVC) measure is developed by scoping down the measurement specific to the host pairs near the attacker and the target. All hosts directly reachable from the attacker (i.e., no stepping stones) are selected and the shortest path to the target is computed. All hosts directly connected to the target host are selected and the shortest paths to the attacker are computed. The AVC values are assigned by the occurrence of hosts in the shortest paths. To compute AVC, Floyd algorithm is used for all pair shortest paths [64], which has a computational complexity of O(n^3). Table 7.3 shows the AVC of the example networked system. The target host is not taken into account, and the ranking of the AVC provides the equivalent solution to the result obtained in Table 7.1 (i.e., equivalent to the ES method).

Table 7.3: Ranking of Hosts based on AVC Measures

Hosts	U_1	U_2	U_3	U_4	U_5
Occurrence	1	0	1	2	0
rank	2	4	2	1	-

In addition, a modified version of the AVC is also developed to rank important hosts based on their *neighbour* information, which is denoted as an *Attacker*

to Victim Neighbour Centrality (AVNC) measure. When the most important host is identified, other hosts near the important host are also likely to have a high impact in an event of an attack. So these neighbour hosts are also given a high importance. The AVNC values of neighbour hosts are computed with respect to the value of the important host (i.e., hosts are further ranked based on neighbourhood information). For the example networked system, the highest ranked host using the AVC measures is U_4. Its neighbouring hosts are U_1 and U_3, but they both have the same ranks. Hence, there is no change for the example networked system using the AVNC measures. However, the effectiveness of the AVNC measures are demonstrated in simulations in Section 7.3, which is to investigate the performance and accuracy of using IMs against the ES method for the security analysis.

7.3 Simulation Results

Two major questions are aimed to be answered from simulation results; (i) Which NCMs is the most suitable method for ranking hosts and vulnerabilities (where locations of the attacker and the target can be unknown or specified) and is security solutions obtained using NCMs close to the ES method? (ii) What is the performance improvement with respect to time when using NCMs compared to the ES method?

7.3.1 Experiment 7A: Attacker Outside the Networked System

A networked system without any subnets is taken into account (i.e., no firewalls), with a mesh topology to connect hosts. Each host has two vulnerabilities, and it is connected to at least three other hosts. The attack is specified to be coming from the outside of the network, and a target host is also specified. The attacker has a single-hop connection to a third of hosts in the networked system, and a third of networked system hosts which are not directly reachable from the attacker (i.e., intermediate nodes), are connected to the target host (i.e., the attacker cannot directly reach the target host). The resulting networked system contained over 45,000 possible attack scenarios. The risk of the networked system is assessed by assigning impact values to all hosts. To distinguish between different NCMs, all hosts are assumed to have the same lower layer information (i.e., two

vulnerabilities are assigned to all hosts with the same configurations). In addition, the probability of a host being compromised when attacked (i.e., probability of an attack success) is equal to one (i.e., the vulnerability will be exploited successfully if an attack is repeated infinitely).

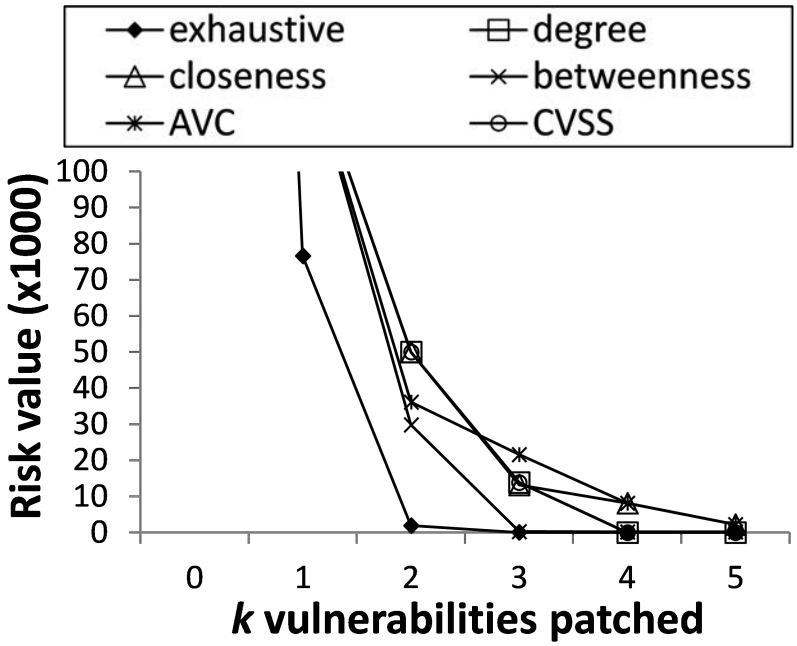


Figure 7.1: Performance of NCMs

The changing effects of the risk value are simulated when a k number of vulnerabilities are continuously patched. Using the ES method, the best security practice (e.g., patching vulnerabilities) with the given networked system state is computed (i.e., a greedy method). Various NCMs rank hosts by their importance based on the network topology, and vulnerabilities of each host are patched in the order of host ranks. Figure 7.1 shows the reduction in risk value of the networked system by patching k vulnerabilities. The result shows that all NCMs provide similar results to the ES method, and in particular, the betweenness centrality measure provided the closest trend. The crossovers between lines are due to the rankings of similar hosts in the networked system. For example, host H_x is fully patched (i.e., no vulnerabilities) and it cannot be used as a stepping stone. Consequently, other hosts that are attacked through H_x are no longer reachable from the attacker (i.e.,

redundant hosts in the event of an attack). But if their ranks are high, their vulnerabilities will be patched, which does not enhance the security of the networked system.

7.3.2 Experiment 7B: Attacker Inside the Networked System

The AVC showed the worst performance in the Experiment 7A (Section 7.3.1) when the attack scenario covered all networked system. In contrast, a networked system with an attack scenario that only covers a subset of the networked system is taken into account as shown in Figure 7.2 in this experiment. The networked system consists of the complex relationship between hosts, structured like a mesh similar to the Experiment 7A. An attack scenario with the attacker located at the internal network is considered. To reduce the coverage of the attack scenario in the network, 400 hosts were assigned in the DMZ, and 25 hosts inside the IN. All hosts in the DMZ and in IN were assigned with two vulnerabilities. For a DMZ host to be compromised, the attacker could exploit only one vulnerability, whereas to compromise the IN host, the attacker must exploit vulnerability v_1 and vulnerability v_2 in sequence (i.e., both vulnerabilities assigned to IN hosts). In the DB, 8 hosts were assigned with a single vulnerability each. Figure 7.3 shows the performance of different NCMs.

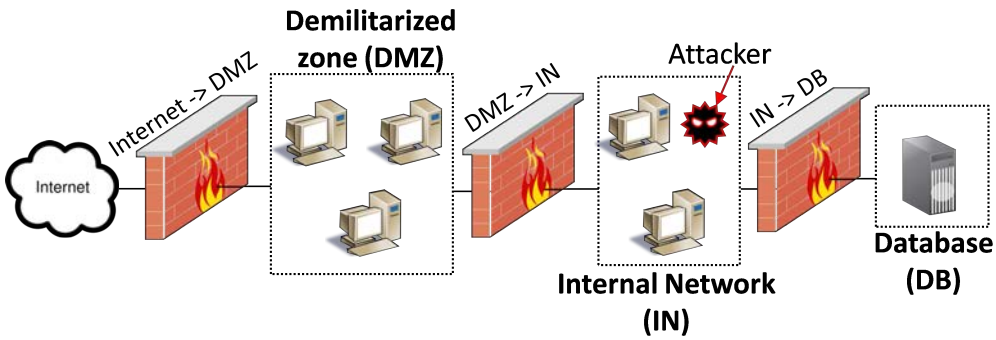


Figure 7.2: A Networked System for the Second Simulation

The AVC measure is very close to the ES method when the value of k is very small. However, the performance using the AVC measure degrades as the k value increases. This is because there are hosts with the same rank that no longer enhances the networked system security when their vulnerabilities are patched. This

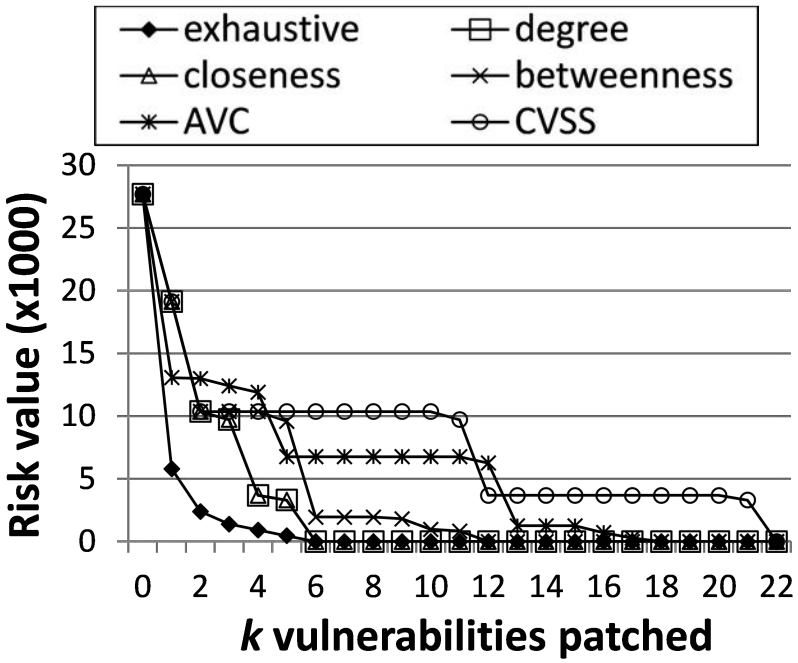


Figure 7.3: Performance of NCMs with an Inside Attacker

problem is described in Section 7.2, which is improvised by using the AVNC measures. Figure 7.4 shows the performance using NCMs, including the use of AVNC measures. The use of AVNC measures show a significant improvement when compared with the AVC measures, and the performance of AVNC measure comes closest to the ES method. This shows that when the attack scenario only covers a subset of the network, the AVNC measures are most effective to rank important hosts and vulnerabilities.

Next, an investigation was carried out to find out what happens when the attack scenario covers more hosts in the networked system. Using the same attack scenario, the number of hosts are proportionally distributed between the DMZ and IN subnets, so that the attack scenario covers half of the hosts in the networked system. There were 100 DMZ hosts and 100 IN hosts with two vulnerabilities for each host for this experiment. Figure 7.5 shows the performance using NCMs, AVC measures and AVNC measures. It shows that using the AVNC measure still performs most closely with the ES method. Also, the performance using the betweenness centrality measures have improved as the attack scenario covered more hosts in the networked system. But in conclusion, the results show that the

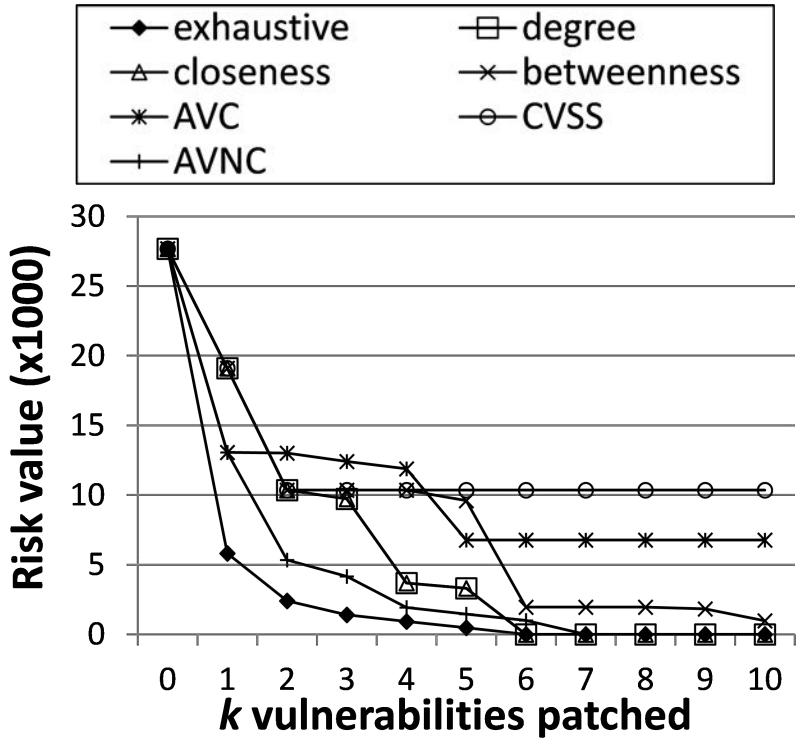


Figure 7.4: Performance using AVNC Measures

location-based centrality measures, particularly AVNC measures, performed better than any other NCMs even when the attack scenario covered as much as half of hosts in the networked system.

7.3.3 Experiment 7C: Scalability of IMs

The ES method has exponential time and space complexities [87,88,124]. The computation of security solution using NCMs showed that by ranking important hosts and vulnerabilities, the security solution can be computed nearly equivalent to the ES method [89, 90, 93]. Computing the NCMs has a polynomial time complexity to rank important hosts and vulnerabilities. In this experiment, the evaluation time of the ES method is compared with using NCMs for analysing the security of networked systems. The networked system has the same settings used in Experiment 7A (i.e., Section 7.3.1). The number of hosts was increased to compare the scalability of the ES method and NCMs. Figure 7.6 shows the time increase in security evaluation using NCMs. The ES method could not be

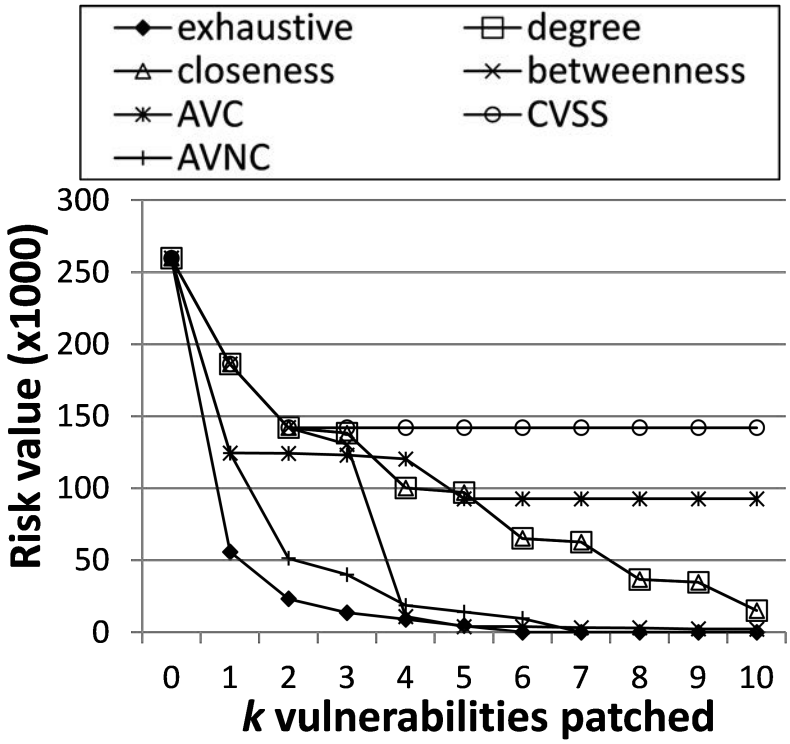


Figure 7.5: Attack Scenario Covering Half of Networked system Hosts

computed for a large number of hosts in a dense networked system topology (i.e., mesh). For 15 hosts, the ES method took just over 12 seconds, but for 17 hosts, it took over a minute. The scalability of degree centrality and CVSS score based rankings are almost negligible, because their computational complexity is O(n). Other NCMs (closeness, betweenness, AVC, and AVNC) show a polynomial complexity as defined by their underlying algorithm, which is given by O(n^3). The memory space requirement is also reduced using NCMs. The ES method has an exponential size complexity to store exponential number of attack scenarios, but NCMs require at most O(n^3) memory space to store all pair shortest paths. The trend for memory space requirement is similar to the trend shown in Figure 7.6.

7.4 Discussion

This chapter demonstrated that nearly equivalent security solutions can be computed using the location-based IMs with respect to the ES method when the

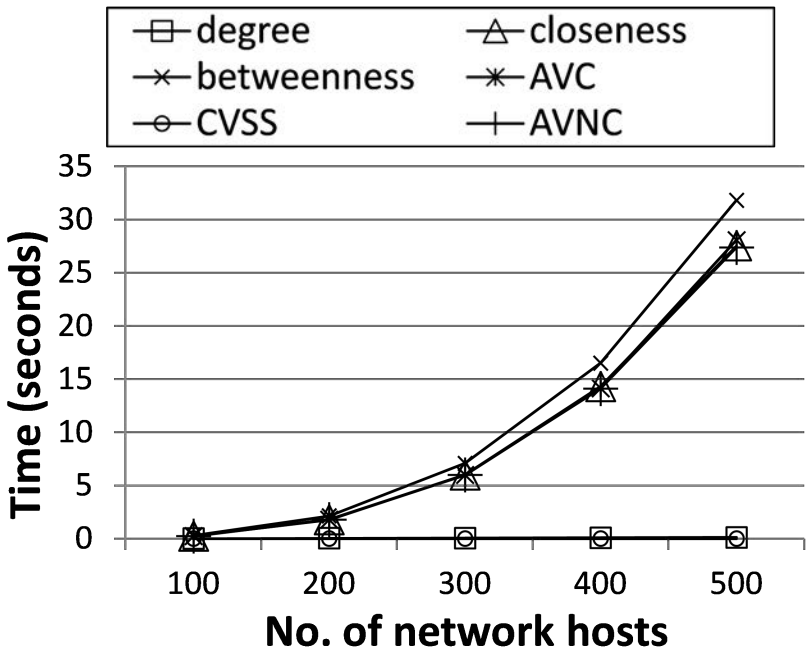


Figure 7.6: Evaluation Time for NCMs

attack scenario only covered a subset of the networked system. The performance and accuracy of various NCMs are compared (i.e., degree, closeness, and betweenness centrality measures), as well as two newly developed NCMs (i.e., AVC and AVNC measures). When considering a general security of networked systems, existing NCMs showed effective rankings of important hosts and vulnerabilities. But in case where the attack scenario covers only a subset of the network, the location-based NCMs performed better in terms of accuracy.

The performance of security analysis using NCMs may be improved multiple NCMs are combined together. Also, when a networked system consisting of multiple unique vulnerabilities is considered, the security analysis should take into account the combined effect of network configurations and vulnerabilities. However, this chapter only considered vulnerabilities having an equal probability of an attack success. Attack scenarios with an attacker located outside or inside of the networked system are taken in the experiments, but the target host was always fixed, limiting all possible attack scenarios. Moreover, the case of an attack launched to all hosts or subset of hosts has not been considered (e.g., Denial of Service attacks).

7.4.1 Combinations of NCMs

Many crossovers are observed in the simulation results caused by patching redundant vulnerabilities. To reduce crossovers, combining the rankings of different centrality measures may be considered, as shown in Section 6.1. By assigning weights to each centrality measures, they can be combined to give a single ranking. Depending on the attack scenario and the networked system topology, the proportion of these weights will affect the performance (e.g., high weight on location-based centrality measures should be applied when the attack scenario covers a small proportion of the network). The optimal proportionality calculation between weights and centrality measures is not taken into account.

7.4.2 Combining NCMs with Vulnerabilities

Two major factors in security analysis are how an attacker may reach the target (i.e., attack scenarios considering the networked system topology), which vulnerabilities are used, and what is the impact of the attack on the networked system (e.g., cost of an attack, probability of an attack success). In the simulation, only the network perspectives (top-down) or vulnerability perspectives (bottom-up) are considered. These two factors should be considered together, as they are correlated in an event of an attack (e.g., using a hybrid approach as in Section 6.3). The simulation results showed that in a complex networked system, vulnerability based rankings performed consistently worse than NCM based rankings, so no improvements could be made in this example. That is, the top-down approach has more impact than the bottom-up approach, although the vulnerability models used in the experiments were too simple. In more complex networked system with various vulnerabilities, the performance could be affected if only the top-down approach is considered.

7.4.3 Multiple Target Hosts and Locations

Results in this chapter are on the basis of a single target host, but the attack scenario could also compromise multiple target hosts in multiple locations (e.g., denial of service attack). NCMs considered the overall networked system security (i.e., implicitly covering an attack with multiple targets), but it is uncertain how close the security solution comes near to the optimal solution in such cases.

However, the number of attack scenarios increases in such cases, whereas computing IMs only changes the rankings of hosts and vulnerabilities without affecting the performance. Hence, a different location for the target host may result in new attack scenarios, but such cases do not compromise the performance of IMs. However, the accuracy of security solutions should be taken into account.

7.5 Related Work on Using Network Centrality Measures

Security analysis method based on NCMs is proposed in this chapter (e.g., degree, closeness, betweenness [38, 71]), which is to rank important hosts. NCMs are already used in various studies, including Chapter 6, which has been used in electrical networks [83]. Cadini *et al*. [38] used centrality measures on the power network to investigate the strength and weakness of the safety that are not captured from the topological basis. However, this does not take into account when the attacker is located within the networked system. Also, Georgiadis *et al*. [71] showed various NCMs and how they could be utilised in the network security analysis. However, they only introduced various NCMs that are aimed to be used with the whole networked system, and did not consider the effect of attack scenarios only affecting subsection of the networked system. Vulnerability based evaluation is widely used, such as in [176], where the security analysis depends on the vulnerability information. However, the importance of hosts should be taken into account as the topology of the networked system may affect the security analysis significantly.

7.6 Conclusions

Security analysis using the ES method considers all possible attack scenarios to encounter progressive attacks, but that solution suffers from a scalability problem. This chapter described more scalable security analysis method, which is to rank important hosts using NCMs, and vulnerabilities using security metrics in a given networked system (i.e, using the IMs). Further, the focus was to improve the accuracy of security analysis using the IMs when the attack scenario covered only a subset of the networked system. The experimental results show that security solutions using the IMs were nearly equivalent to the ES method, where

the betweenness centrality measure performed the best when the attack scenario covered all networked system (i.e., the attacker located outside the networked system). On the other hand, location-based NCMs, AVC and AVNC, are developed and performed the best when the attack scenario covered only a subset of the networked system (i.e., the attacker located inside the networked system). Security solutions using both existing NCMs (i.e., degree, closeness, and betweenness centrality measures) and newly developed NCMs (i.e., AVC and AVNC measures) computed nearly equivalent security solution with the ES method, but the performance (in terms of time) and the memory usage were significantly reduced.

Part IV

Effects of Unknown Attacks

Chapter 8

Security Assessment of Unknown Attacks

Previous chapters only dealt with known attacks and vulnerabilities in the networked system. Although the majority of cyber attacks are based on exploiting only the known vulnerabilities, there are incidents that used both known and unknown attacks that caused a severe socio-economic impact upon users. Stuxnet [63] and RSA SecurID breach [97] are two well-known incidents of cyber attacks that used a combination of both known and unknown attacks. Not only unknown attacks are difficult to detect and mitigate, their attack scenarios are also unpredictable. For example, the attack scenario of RSA SecurID breach began with phishing emails to exploit CVE-2011-0609 vulnerability, which was a zero-day vulnerability at the time (details of the vulnerability can be looked up at the National Vulnerability Database [146]). Then, a backdoor (a Poison Ivy variant) was installed, and the attacker penetrated through the networked system via privilege escalations. Although such attack scenarios consisting of unknown attacks are possible, they are often not taken into account when hardening the networked system because it is difficult to measure the security posture of them [30]. On the other hand, the occurrence of unknown attacks is growing rapidly [137]. Thus, there is an urgent need for assessing the combined effects of both known and unknown attacks.

Security models are widely used to assess the security of networked systems [12, 84, 99]. However, one of the shortcomings of security models is the difficulty of accurately assessing the implication of unknown attacks due to a lack of knowledge about their properties to quantify security metrics [30] (e.g., how to measure the impact of an unknown attack?). Previous work focused on assessing the effectiveness of unknown vulnerabilities only (i.e., zero-day vulnerabilities) [45, 99, 198, 199], but they did not take into account the combined effects of unknown attacks consisting of unknown vulnerabilities, unknown devices, and unknown attack paths.

This chapter aims to address the aforementioned problems by classifying unknown attacks, and incorporate them into the HARM. Also, the severity of unknown attacks are evaluated when they are introduced in networked systems. Lastly, two algorithms are proposed to determine mitigation strategies that minimise the effects of unknown attacks, as well as approximation algorithms to enhance the performances.

8.1 Security Modelling of unVIP

This section describes: (i) classification of unknown attacks in Section 8.1.1, (ii) introduction to a running example networked system in Section 8.1.2, and (iii) definition of a security model and incorporation of unknown attacks into the security model in Section 8.1.3.

8.1.1 Classification of Unknown Attacks

Unknown attacks consist of not only unknown vulnerabilities, but a combination of unknown vulnerabilities, devices, and attack paths. Hence, unknown attacks can be classified as (i) **U**nknown **V**ulnerabilities (UVs), (ii) **U**nknown dev**I**ces (UIs), and (iii) **U**nknown attack **P**aths (UPs), which are together denoted as *unVIP*.The classification of unknown attacks into three categories is as follows:

Unknown Vulnerability (UV): UVs are vulnerabilities that do not have known patches, in other words, zero-day vulnerabilities. They may exist in a networked system, which if an attacker can reach the host with an UV, then it can be exploited with an unpredictable outcome. Since it is impractical to assume any properties of the UVs, successfully exploiting any UVs is assumed to grants the attacker with a full control (e.g., a root privilege). To incorporate UVs into the security model, similar assumptions about UVs in the networked system can be taken into account as in [45, 99] and [198, 199] (e.g., hosts and/or applications can be assumed to have an UV).

Unknown Device (UI): UIs are new or existing devices with unspecified properties. They expand the attack surface of the networked system, which can be either (a) a new network component (e.g., new hosts, VPN hosts, Wi-Fi hotspots in a corporate network, BYOD), or (b) an existing network component with changes in its attack surface (e.g., USB sharing, phishing emails installing backdoors). UIs

are assumed to have any configurations, which may introduce both existing and new sets of known and unknown vulnerabilities into the networked system. In the case of (a), a new host is introduced as an UI host in the networked system and capture changes of the reachability and vulnerability information. In the case of (b), an existing host can be specified as an UI host and update changes of the reachability and vulnerability information. If the configuration of the UI host is given, then they can be imported into the security model. Otherwise, an UV is assumed.

Unknown attack Path (UP): UPs are unspecified attack paths that are created as a result of UVs, UIs or combinations of both appearing in the networked system. Appearances of UPs create new attack scenarios that were not anticipated previously (e.g., security analysis based on known vulnerabilities and exploits only cannot assess such attack scenarios). When UVs or UIs are assumed in the networked system, possible UPs are created and incorporated into the security model. Network components (e.g., hosts) with UVs are assumed to enable reachability from other network components without any restrictions (e.g., no right privilege is required), but the firewall rules are still applied (e.g., hosts in the DMZ subnet cannot reach hosts in the DB subnet with an UV directly).

UVs (also known as zero-day vulnerabilities) are vulnerabilities without knowledge of their properties and how to fix them [30]. UIs are devices with undefined properties that may change the attack surface of the networked system (e.g., Bring Your Own Device (BYOD) policies [142]), and there is a lack of methods to analyse the security of them [73]. UPs are new attack paths in the networked system that creates unforeseen attack scenarios (e.g., identifying unknown attack paths using system object dependencies [8, 51]). For example, CVE-2011-0609 vulnerability, an UV, is used in conjunction with a phishing email, which is an UI. The attack used unknown attacks in a combination (i.e., both UV and UI), which created UPs that caused an unforeseen attack scenario. However, it is difficult to assess where UPs may occur. As a result, there is a lack of methods to analyse the security of networked systems when unknown attacks are used in a combination. Therefore, failing to assess the combined effects of unVIP may result in security analysis without properly assessing all possible attack scenarios.

Table 8.1: Details of Hosts

Host	OS	Applications	Vulnerability	Priv. Esc.
U_1, U_2, U_3	MS Server 2003 SP1	MS SQL server	CVE-2002-0721	$g \rightarrow r$
U_4	Windows 7 SP1	Basic	CVE-2014-2781	$g \rightarrow r$
U_5	Redhat Enterprise Linux 6	MySQL server	CVE-2008-0086	$u \rightarrow r$
DS	MS Server 2008 SP2	Oracle server	CVE-2012-3220	$u \rightarrow r$
UR_{VPN}	MAC OS X	Basic	CVE-2014-0515	$g \rightarrow r$

8.1.2 An Example Networked System and Attack Scenarios

Description of the Example Networked System

Different network settings are assigned to provide a better insight of unVIP and their effect on the security. The same networked system is used as shown in Figure 2.3, but different properties of hosts are assigned as shown in Table 8.1, which also includes the information about a VPN user UR_{VPN} (i.e., an UI, assuming no restrictions on the technology used). Hosts are further divided into groups: (i) U_1, U_2 and U_3 are in a Web Server (WS) group, (ii) U_4 is in a User (UR) group, and (iii) U_5 is in a Application Server (AS) group. The DB server is denoted as DS. Only a single vulnerability for each host is assumed, which is to enable the attack scenario in the simplest form. However, multiple vulnerabilities can be modelled for a comprehensive security assessment as in previous chapters. Further, three levels of privileges are assumed: (i) guest (g), (ii) user (u), and (iii) root (r), which specifies the access level required to utilise the host application (e.g., at least a user privilege is required to access the U_5).

Attack Scenarios

Table 8.2 shows some of the possible attack scenarios with and without unVIP assumed in the example networked system. The table also includes the k-zero-

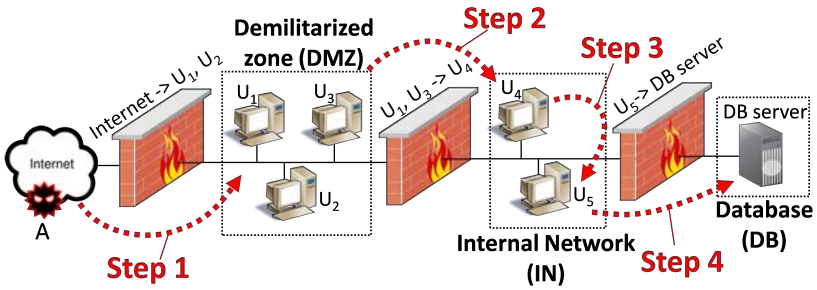
Table 8.2: Possible Attack Scenarios

Case	Assumption	Attack Scenario	Steps	*kzd*
1	No unVIP	$A \rightarrow WS \rightarrow UR \rightarrow AS \rightarrow DS$	4	3
2a	UV (AS_{UV})	$A \rightarrow WS \rightarrow AS_{UV} \rightarrow DS$	3	3
2b	UV (DS_{UV})	$A \rightarrow WS \rightarrow UR_{UV} \rightarrow DS$	3	
3	UI (UR_{VPN})	$A \rightarrow UR_{VPN} \rightarrow AS \rightarrow DS$	3	3
4	UP (UR_{VPN}, DS_{UV})	$A \rightarrow UR_{VPN} \rightarrow DS$	2	2

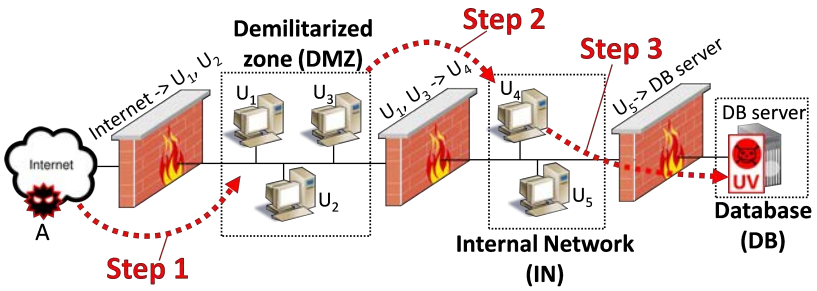
day (*kzd*) safety metric (as in [198, 199]) to show that it can be changed when unVIP are used in a combination. *kzd* safety measures the minimum number of UVs required for the attacker to compromise the target host for a given attack scenario. These attack scenarios are illustrated in Figure 8.1. Those examples show that without any unVIP (Case 1 shown in Figure 8.1(a)), the attacker must compromise WS, UR, AS, and DS (i.e., the minimum number of exploits required is four). When an UV or an UI is assumed, it reduces the number of minimum steps required by the attacker (Case 2 as shown in Figure 8.1(b) for an UV in DS, and Case 3 as shown in Figure 8.1(c) for an UI in the IN subnet). However, both UVs and UIs are assumed, the attacker can exploit the UI (e.g., the VPN host) and directly to the DS exploiting the UV (as Case 4 as shown in Figure 8.1(d)). Moreover, the combined effects of unVIP changes the *kzd* safety (i.e., from $k = 3$ to $k = 2$). Thus, these results show that (1) the combined effects of unVIP can reduce the number of minimum exploits required, and (2) different attack scenarios have different security effects depending on where unVIP are assumed.

8.1.3 Incorporate unVIP into the Security Model

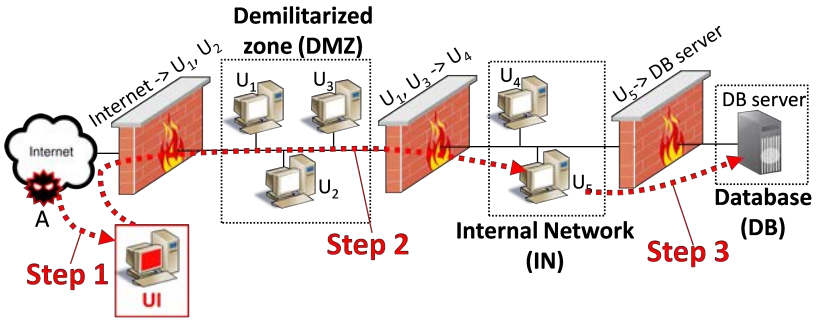
2-HARM with AGs in both layers is incorporated with unVIP, which is shown in Figure 8.2. The upper layer remains the same as in Figure 2.4, but the lower layer information has changed as shown. The square box in the lower layer represents the state (e.g., $WS : g$ represents the guest privilege of WS), and the oval represents the condition (e.g., CVE-2002-0721 is a vulnerability condition that can be exploited from $WS : g$ to gain the $WS : r$). For example, a u privilege is required to exploit CVE-2012-3220 in the DS to gain the r privilege.



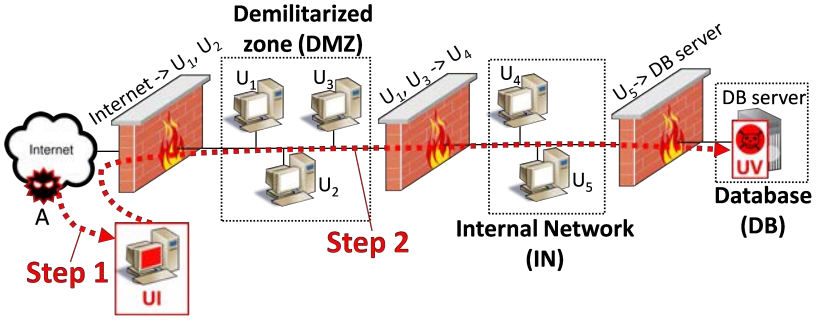
(a) Case 1



(b) Case 2

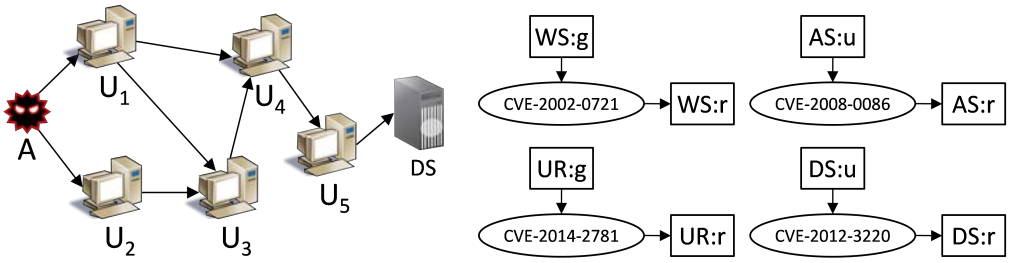


(c) Case 3



(d) Case 4

Figure 8.1: unVIP Attack Scenarios in the Example Networked System



(a) Upper Layer of the HARM (b) Lower Layer of the HARM

Figure 8.2: HARM of the Example Networked System

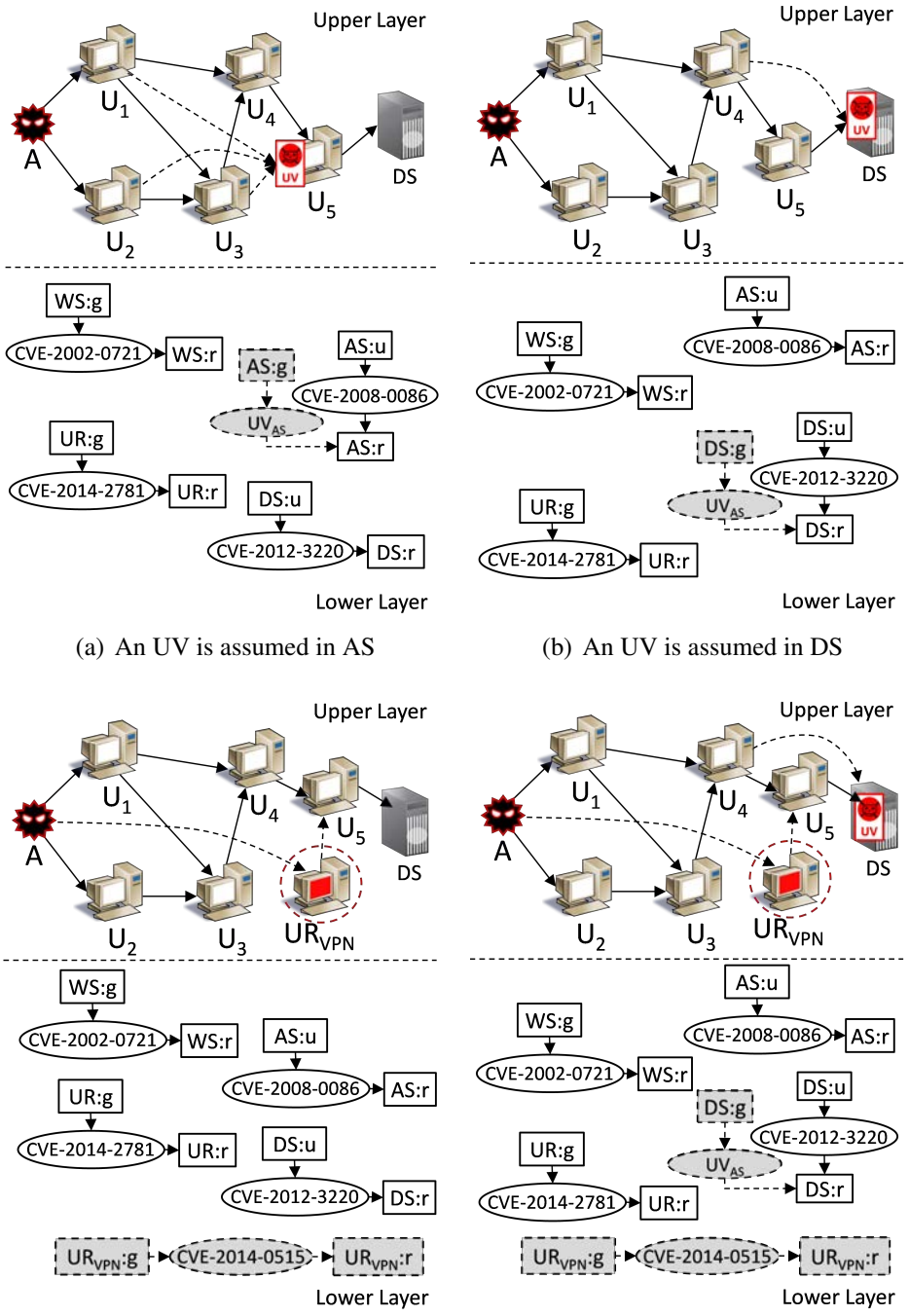
unVIP are incorporated into the HARM. The definition of incorporating UVs into the HARM is as follows:

Definition 10. Given a HARM of a 3-tuple $H = (h, M, C)$ where $h = 2$ (as shown in Section 2.3), an UV (uv_i) in a node (i.e., a host) $n \in M_1$ changes the lower layer HARM to $G_n^{2*} = (N_n^{2*}, E_n^{2*})$, where $N_n^{2*} = N_n^2 \cup uv_i$ is a finite set of vulnerabilities, host states and unknown vulnerabilities, and $E_n^{2*} = (N_n^2 \times N_n^2) \cup \{(s(g), uv_i), (uv_i, s(r))\}$ is a set of edges where $s(g)$ is a guest state of the host and $s(r)$ is the root state of the host (i.e., exploiting the UV directly grants the root privilege from a guest privilege).

Example 13. Shown in Figure 8.3(a):
An UV assumed in AS (uv_{AS} and $AS \in M_1$), a new lower layer HARM $G_{AS}^{2*} = (N_{AS}^{2*}, E_{AS}^{2*})$ is created, where $N_{AS}^{2*} = N_{AS}^2 \cup \{uv_{AS}\}$ and $E_{AS}^{2*} = E_{AS}^2 \cup \{(s(g), uv_{AS}), (uv_{AS}, s(r)\}$. UPs associated UV in AS ($up_{AS} = \{(WS, AS)\}$) (i.e., created new attack paths from WS to AS directly) creates a new upper layer $G_{AS}^{1*} = (N_{AS}^1, E_{AS}^{1*})$, where $E_{AS}^{1*} = (N^1 \times N^1) \cup \{up_{AS}\}$.

Example 14. Shown in Figure 8.3(b):
An UV assumed in DS (uv_{DS} and $DS \in U$), a new lower layer HARM $G_{DS}^{2*} = (N_{DS}^{2*}, E_{DS}^{2*})$ is created, where $N_{DS}^{2*} = V_{DS}^2 \cup \{uv_{DS}\}$ and $E_{DS}^{2*} = E_{DS}^2 \cup \{(s(g), uv_{DS}), (uv_{DS}, s(r)\}$. UPs associated UV in DS ($up_{AS} = \{(UR, DS)\}$) (i.e., created new attack paths from UR to DS directly) creates a new upper layer $G_{DS}^{1*} = (N_{DS}^1, E_{DS}^{1*})$, where $E_{DS}^{1*} = (N^1 \times N^1) \cup \{up_{DS}\}$.

The definition of incorporating UIs into the HARM is as follows:



(a) An UV is assumed in AS

(b) An UV is assumed in DS

(c) An UI is assumed in AS

(d) UPs are assumed as a result of an UV in DS and UI in the IN Subnet

Figure 8.3: Incorporating unVIP in the HARM

Definition 11. Given the upper layer of the HARM $G^1 = (N^1, E^1)$ with an attack graph representing the host reachability in the networked system, an UI (ui_i) creates a new upper layer $G^{1*} = (N^{1*}, E^{1*})$, where $N^{1*} = N^1 \cup \{ui_i\}$, and $E^{1*} = N^{1*} \times N^{1*}$. The corresponding lower layer, $ui_i \leftrightarrow G^2_{ui_i}$, is also created (either with the given information, or with an assumed UV).

Example 15. Shown in Figure 8.3(c):
Given the upper layer of the HARM $G^1 = (N^1, E^1)$, an UI in AS (ui_{VPN}) creates a new upper layer $G^{1*} = (N^{1*}, E^{1*})$, where $N^{1*} = N^1 \cup \{ui_{VPN}\}$, and $E^{1*} = (N^1 \times N^1) \cup \{(A, UR_{VPN}), (UR_{VPN}, AS)\}$. The corresponding lower layer, $ui_{VPN} \leftrightarrow G^2_{ui_{VPN}}$, is also created with the given information in Table 8.1.

Lastly, the definition of incorporating UPs into the HARM is as follows:

Definition 12. Given the upper layer of the HARM $G^1 = (N^1, E^1)$, an UP (a list of new attack paths $up_i = \{(n_j, n_k), \ldots (n_l, n_m)\} \mid j \neq k \neq l \neq m$) creates a new upper layer $G^{1*} = (N^1, E^{1*})$, where $E^{1*} = (N^1 \times N^1) \cup \{up_i\}$.

Example 16. Shown in Figure 8.3(d):
(a) An UV assumed in DS (uv_{DS} and $DS \in U$), a new lower layer HARM $G^{2*}_{DS} = (N^{2*}_{DS}, E^{2*}_{DS})$ is created, where $N^{2*}_{DS} = N^2_{DS} \cup \{uv_{DS}\}$ and $E^{2*}_{DS} = E^2_{DS} \cup \{(s(g), uv_{DS}), (uv_{DS}, s(r)\}$.
(b) An UI in AS (ui_{VPN}) creates a new upper layer $G^{1*} = (N^{1*}, E^{1*})$, where $N^{1*} = N^1 \cup \{ui_{VPN}\}$, and $E^{1*} = (N^1 \times N^1) \cup \{(A, UR_{VPN}), (UR_{VPN}, AS)\}$. The corresponding lower layer, $ui_{VPN} \leftrightarrow G^2_{ui_{VPN}}$, is also created with the given information in Table 8.1.
(c) UPs associated with UV in DS and UI in AS ($up_{DS} = \{(A, UR_{VPN}), (UR_{VPN}, AS), (UR, DS)\}$) creates a new upper layer $G^{1\#}_{DS} = (N^{1\#}, E^{1\#})$, where $E^{1\#} = E^{1*} \cup up_{DS}$.

Various attack scenarios are taken into account (as shown in Table 8.2), where these scenarios are modelled using the HARM as shown in Figure 8.3. Newly added components are represented by dotted lines (e.g., dotted lines in the upper layer representing new attack paths, dotted red circles representing a new device, and dotted ovals with shades in the lower layer representing new vulnerabilities). Figures 8.3(a) and 8.3(b) show that placement of UVs changes the attack scenario, respectively. Figure 8.3(c) shows that assuming UIs can also create new

attack paths. Figure 8.3(d) shows the most increased number of new attack paths (by nine), which implies that combined effects of unVIP can be more severe in comparison to individual unVIP assumed in the example networked system.

8.2 Security Analysis of Systems with unVIP

This section demonstrates security analysis with and without unVIP in the networked system. First, the security analysis of the networked system without any unVIP is shown in Section 8.2.1, and then with unVIP in Section 8.2.2.

8.2.1 Security Analysis without unVIP

System Risk is taken into account as the security analysis (as in previous chapters), using the HARM generated in Section 8.1. The probability of an attack is assumed to be equiprobable across all hosts in the networked system. Further, because unknown vulnerabilities do not have the CVSS BS, the impact values are assumed to be dependent on the type of the privilege escalation (e.g., $g \rightarrow r$ with an impact value of 10, and $u \rightarrow r$ with an impact value of 5). Using these assumptions, system risk, R_{system} of the example networked system is calculated as shown in equation (8.1) (details see Section 2.5).

$$\begin{aligned} R_{system} &= \sum_{i \in path} PR_i \\ &= \sum (PR_{\{U_1,U_4,U_5,DS\}}, PR_{\{U_1,U_3,U_4,U_5,DS\}}, PR_{\{U_2,U_3,U_4,U_5,DS\}}) \\ &= 110 \end{aligned} \tag{8.1}$$

8.2.2 Security Analysis with unVIP

Due to an exponential number of possible attack scenarios in respect to the number of unVIP, security analysis of the networked system with unVIP is conducted for only up to two UVs without any UIs as an example. This section is to demonstrate the analysis procedure, and more comprehensive security analyses taking into account various attack scenarios with unVIP assumed will be shown in Section 8.4.

The assumed UV is incorporated in the HARM as shown in Section 8.1.3, and evaluation of the system risk is shown in Section 8.2.1. The risk associated with each combination of hosts is evaluated. The same probability of an attack with known vulnerabilities for UVs is assumed, but the impact value of an UV is assumed as 20 (i.e., greater than any existing impact values), under the assumption that exploits using an UV is unlikely to be mitigated in comparison to exploiting known vulnerabilities (or much more difficult). These values can be replaced based on empirical studies such as in [137]. There are total 15 possible host combinations with up to two assumed UVs in the networked system, but only the host combinations with different groups are taken into account (e.g., UVs assumed in U_1 and U_2 has the same effect as UVs assumed in U_2 and U_3). Therefore, there are total 10 distinctive attack scenarios. Table 8.3 shows the system risk of the networked system for these attack scenarios with up to two UVs assumed. It shows that in some cases, having one UV can have higher severity than two UVs (e.g., an UV in AS has higher system risk than UVs in WS and UR). Also, the number of attack paths does not necessarily increase the system risk proportionally (e.g., the system risk differences between UVs in WS and AS compared to UVs in UR and AS).

Table 8.3: System Risk with Two UVs

UVs in	No. of Attack Paths	System Risk
None	3	110
WS only	3	160
UR only	4	180
AS only	6	260
DS only	6	295
WS and UR	4	240
WS and AS	6	340
WS and DS	6	395
UR and AS	8	400
UR and DS	8	460
AS and DS	9	510

```
1: procedure Analyse_unVIP(N)
2:     for l = 1...|N| do
3:         for d ∈ S|S ⊆ N do
4:             Evaluate Security
5:         end for
6:     end for
7: end procedure
```

Figure 8.4: Pseudocode to Analyse all unVIP Scenarios

8.3 unVIP Mitigation Strategies

Incorporating unVIP in the HARM and the security analysis of the networked system are shown in Sections 8.1 and 8.2 respectively. These sections showed that depending on where unVIP are assumed, the effects of them vary (e.g., system risks as shown in Table 8.3). Therefore, it is important to identify the most significant network components (e.g., hosts), and harden them to minimise the effects of unVIP. In this Section, methods to mitigate unVIP are described by means of: (i) identifying and hardening significant hosts to minimise the effects of unVIP (shown in Section 8.3.2), and (ii) identifying and patching significant vulnerabilities to minimise the system risk when unVIP are assumed (shown in Section 8.3.3).

8.3.1 All Possible Attack Scenarios

Analysis of all possible attack scenarios is taken into account with unVIP assumed in the networked system, which is shown by a pseudo-code in Figure 8.4. The pseudo-code is used to compute significant hosts and vulnerabilities later in Sections 8.3.2 and 8.3.3. l represents the assumed number of UVs, d represents the assumed UI in a subnet S, and N represents the networked system. Combinations of hosts are computed to consider all possible attack scenarios with a given value l (e.g., with $|UV| = 2$, UVs are assumed for every pair of hosts).

8.3.2 Identification of Significant Hosts

An algorithm is developed to identify significant hosts by evaluating all possible attack scenarios and ranking the occurrence of each host in all possible attack

1: **procedure** $Identify\_Significant\_Hosts(f, l, d, N)$
2: $N \leftarrow N \cup d$
3: $UV = \{uv_1, uv_2, \ldots uv_l\}$
4: **for** $UV^* \in N \mid UV^* \subseteq UV$ **do**
5: `Analyse` N^{VU^*}
6: `Compute host occurrence` $score(h)$
7: **end for**
8: Return $\{h_{max} \in N \cap f \mid score(h_max) > score(h_i), i = 0 \ldots |N \cap f|, i \neq max\}$
9: **end procedure**

Figure 8.5: Identifying Significant Hosts Algorithm

paths. There are various mitigation techniques available to minimise the effects of unVIP when significant hosts are identified, such as increasing the diversity of services, strengthening isolation techniques, enforcing more strict access control policies, and patching known vulnerabilities (as described in [198]). the initial security analysis of unVIP in Section 8.2 showed various effects of unVIP in the networked system. To harden the networked system accordingly, it is important to identify and deploy mitigation techniques on significant hosts in respect to attack scenarios developed from unVIP assumed.

Naive Algorithm

The algorithm, $Identify\_Significant\_Hosts$ (ISH), to identify significant hosts is shown in Figure 8.5. f is a filter expression that allows users to input hosts to be filtered from the result, l is the number of UVs, d is the subnet containing the UI, and N is the networked system. This algorithm is processed in line 4 in the pseudo-code shown in Figure 8.4 (i.e., a function call at line 4). Line 4 in *ISH* specifies all possible host combinations of UVs assumed in the networked system, and the security of a given state is evaluated along with the occurrence of hosts. $score(h)$ computes the number of occurrences a host appears in all possible attack paths for the given state of the networked system. The returned result (i.e., line 8 in *ISH*) is the host with the most occurrence in all attack scenarios (i.e., the most utilised host in any attack scenarios with unVIP in the networked system).

1: **procedure** $Identify\_Significant\_Hosts\_Fast(f,l,d,N)$
2: $N \leftarrow N \cup d$
3: $UV = \{uv_1, uv_2, \ldots uv_l\}$
4: $h_j \in N^* \mid N^* \subseteq N, |N^*| = |UV|$
5: **while** $|N^*| > 1$ **do**
6: `remove` $h_{min} \in N^* \mid c(h_{min}) < c(h_j)$
7: **end while**
8: Return $\{h_{max} \in N \cap f \mid c(h_{max}) > c(h_i), 1 \geq i \geq |N \cap f|, i \neq max\}$
9: **end procedure**

Figure 8.6: Identifying Significant Hosts Approximation Algorithm

Approximation Algorithm

A major problem with *ISH* is the scalability problem, where the number of host combinations increases exponentially as the number of hosts grows. To address this problem, an approximation solution is developed using dynamic programming and greedy algorithm. Figure 8.6 shows the approximation algorithm , $Identify\_Significant\_Hosts\_Fast$ (ISHF). f,l,d,N are as described in 8.3.2, h_j represents hosts assumed with UVs, and $c(h_{min})$ is a function that computes the occurrence of hosts using dynamic programming. Equation (8.2) shows the computation of $c(h_i)$, where $w(h_i)$ is there weight of host h_j (initially assigned to zero, and different weights can be assigned to distinguish importance of hosts, such as asset values or probabilities), $in(h_i)$ is a list of hosts connected to h_i, and $out(h_i)$ is a list of hosts connected from h_i. The greedy algorithm (i.e., step 6 of $ISHF$) removes the least occurring host in a given set N^*, and this process is repeated until all hosts are removed. Then, it returns the host with the maximum occurrence in all possible attack paths (i.e., step 8 of $ISHF$).

$$c(h_i) = \left(w(h_i) + \sum_{h_j \in in(h_i)} c(h_j) \right) \times \left(w(h_i) + \sum_{h_k \in out(h_i)} c(h_k) \right) \tag{8.2}$$

Identification of Significant Hosts in the Example Networked System

ISH and *ISHF* are used to compute the significant hosts in the example networked system. The significance of hosts are taken into account (i.e., from the most significant to least significant), and an UI in the networked system (i.e.,

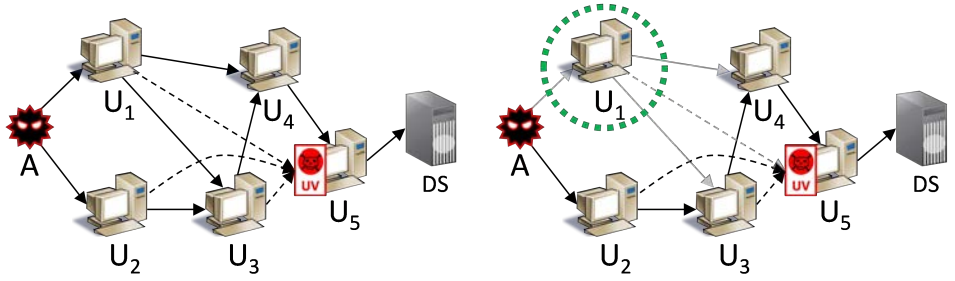
Table 8.4: Significant Hosts of the Example Networked System

		Significance (1=Most, 6=Least)					
UI		1	2	3	4	5	6
None	*ISH*	U_5	*DS*	U_3	U_4	U_1	U_2
	ISHF	***DS***	***U_5***	U_3	U_4	***U_2***	***U_1***
DMZ	*ISH*	U_5	*DS*	U_3	U_4	U_2	U_1
	ISHF	***DS***	***U_5***	***U_4***	***U_3***	U_2	U_1
IN	*ISH*	U_5	U_3	*DS*	U_4	U_1	U_2
	ISHF	***DS***	***U_5***	***U_3***	***U_1***	***U_4***	U_2
DB	*ISH*	U_5	U_3	*DS*	U_4	U_2	U_1
	ISHF	***DS***	***U_5***	***U_4***	***U_3***	***U_2***	U_1

no UI, UI in DMZ, UI in IN, and UI in DB). Table 8.4 shows the significant hosts computed using *ISH* and *ISHF*, where differences are highlighted by bold coloured texts (e.g., U_5 in *ISHF* without any UI at a significance of 2). Although the solution sets between *ISH* and *ISHF* are different, the individual significance of hosts are nearly the same. The accuracy and performance between these algorithms are further evaluated in Section 8.4.

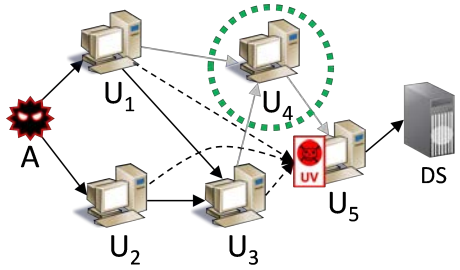
Effectiveness of Identifying Significant Hosts

Figure 8.7 shows the effectiveness of identifying significant hosts in comparison to randomly selecting hosts to harden. An UV is assumed in AS group (i.e., an UV in U_5), and hardening a host disables the attack path through the hardened host (which are highlighted by green dotted circles). The most significant host to harden can be computed using the *ISH*, which in this example case is U_5 followed by *DS* as shown in Table 8.4, but hardening either one of them would disable all possible attack paths. Hence, the next important host is hardened to show the difference of security analysis, which is host U_3. When U_3 is hardened (shown in Figure 8.7(d)), the system risk is reduced from 260 down to 105, which is more than the half of the system risk without hardening any hosts. The same effect is observed when other randomly chosen hosts are hardened (as shown in Figures 8.7(b) and 8.7(c)). However, both of these resulted in higher system risk in comparison to hardening the significant host U_3.

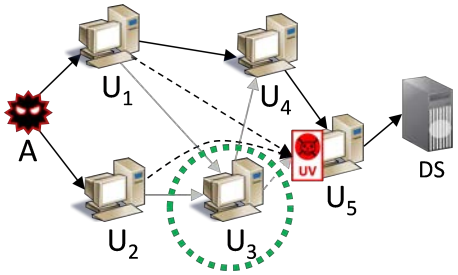


(a) No Hardening. The System Risk is 260.

(b) Hardening U_1. The System Risk is 135.



(c) Hardening U_4. The System Risk is 160.



(d) Hardening U_3. The System Risk is 105.

Figure 8.7: Security Analysis After Hardening a Host in the Networked System

8.3.3 Identification of Significant Vulnerabilities

This section shows the computation of identifying significant vulnerabilities known in the networked system when unVIP are assumed. Similarly as in Section 8.3.2, an algorithm is developed to compute all possible attack scenarios to analyse the significance of patching vulnerabilities, and evaluate the effectiveness of it. For the security assessment, changes in the system risk are taken into account when patching known vulnerabilities. However, other security metrics can also be used (e.g., probability of an attack and mitigation costs).

Naive Algorithm

Figure 8.8 shows the algorithm, $Identify\_Significant\_Vuls$ (ISV), to identify significant vulnerabilities, where it is used in conjunction with the pseudo-code (in line 4) shown in Figure 8.4. The variables have the same implications as in Section 8.3.2 (apart from the filter expression, f, where it filters vulnerabilities specified by users), and $v_i \in N \cap f$ is a set of all known vulnerabilities v_i in the networked system N. Steps from 6 to 8 in *ISV* patches the known vulnerability and calculate

1: **procedure** $Identify\_Significant\_Vuls(f, l, d, N)$
2: $N \leftarrow N \cup d$
3: $UV = \{uv_1, uv_2, \ldots uv_l\}$
4: **for** $UV^* \in N \mid UV^* \subseteq UV$ **do**
5: **for** $v_i \in N \cap f$ **do**
6: `Patch` v_i
7: $score(v_i) \leftarrow score(v_i) + \Delta R_{system}$
8: `Unpatch` v_i
9: **end for**
10: **end for**
11: Return $\{v_{max} \in N \cap f \mid score(v_max) > score(v_i), i = 0 \ldots |N \cap f|, i \neq max\}$
12: **end procedure**

Figure 8.8: Identifying Significant Vulnerabilities Algorithm

the differences in the system risk. Then, step 11 returns the vulnerability with the most reduction in system risk. *ISV* computes all possible host combinations to take into account all possible attack scenarios. Thus, the algorithm has an exponential computational complexity.

Approximation Algorithm

An approximation algorithm, $Identify\_Significant\_Vuls\_Fast$ (ISVF), to identify significant vulnerabilities are shown in Figure 8.9, which is to improve the scalability of the naive algorithm shown in Figure 8.8. The greedy algorithm computes the unVIP scenario that maximises the host occurrence using the dynamic programming method given in equation (8.2) (i.e., step 5 in $ISVF$). Given an attack scenario computed from the greedy algorithm, each known vulnerability is patched to compute the change in system risk (i.e., steps from 7 to 9). $ISVF$ returns the set of known vulnerabilities that minimises the system risk in case of unVIP assumed.

Identification of Significant Vulnerabilities in the Example Networked System

Since only a single vulnerability is assumed to each host in the example networked system (as described in Section 8.1.2), patching any one of them will disable all attacks to be successful in the security model. So, more vulnerabili-

1: **procedure** $Identify\_Significant\_Vuls\_Fast(f, l, d, N)$
2: $N \leftarrow N \cup d$
3: $UV = \{uv_1, uv_2, \ldots uv_l\}$
4: $V = \{v_1, v_2, \ldots v_m\}$
5: $h_j \in N^* \mid N^* \subseteq N, |N^*| = |UV|, max(\sum c(h_j))$
6: **for** $v_i \in N \cap f$ **do**
7: `Patch` v_i
8: $score(v_i) \leftarrow score(v_i) + \Delta R_{system}$
9: `Unpatch` v_i
10: **end for**
11: Return $\{max(score(v_{min})) \in N \cap f \mid score(v_{min}) > score(v_i), 1 \geq i \geq |N \cap f|, i \neq max\}$
12: **end procedure**

Figure 8.9: Identifying Known Vulnerabilities Approximation Algorithm

Table 8.5: Arbitrarily Assumed Vulnerabilities

Vulnerabilities	Impact Values	Assigned Hosts
V_A	4	WS, UR
V_B	3	WS
V_C	6	WS, AS
V_D	5	UR, DS
V_E	7	UR
V_F	8	AS
V_G	2	AS
V_H	10	DS

ties are assumed in the example networked system as shown in Table 8.5, with an assumption that vulnerability V_H cannot be fixed due to system constraints (e.g., loss of quality of service), which can be filtered in the solution using the filter expression provided by aforementioned algorithms. The significant vulnerabilities can be computed as shown in Table 8.6. It shows that ISV and $ISVF$ solutions are nearly equivalent for any number of known vulnerabilities to be patched. Further accuracy and performance analyses in Section 8.4.

Table 8.6: Significant Vulnerabilities of the Example Networked System

		Significance (1=Most, 6=Least)					
UI		1	2	3	4	5	6
None	ISV	V_C	V_F	V_E	V_D	V_B	V_A
	$ISVF$	V_C	V_F	V_E	V_D	V_B	V_A
DMZ	ISV	V_C	V_F	V_E	V_D	V_B	V_A
	$ISVF$	V_C	V_F	V_E	V_D	V_B	V_A
IN	ISV	V_C	V_F	V_E	V_D	V_B	V_A
	$ISVF$	V_C	V_F	V_E	V_D	V_B	V_A
DB	ISV	V_C	$\boldsymbol{V_E}$	$\boldsymbol{V_F}$	V_D	V_B	V_A
	$ISVF$	V_C	$\boldsymbol{V_F}$	$\boldsymbol{V_E}$	V_D	V_B	V_A

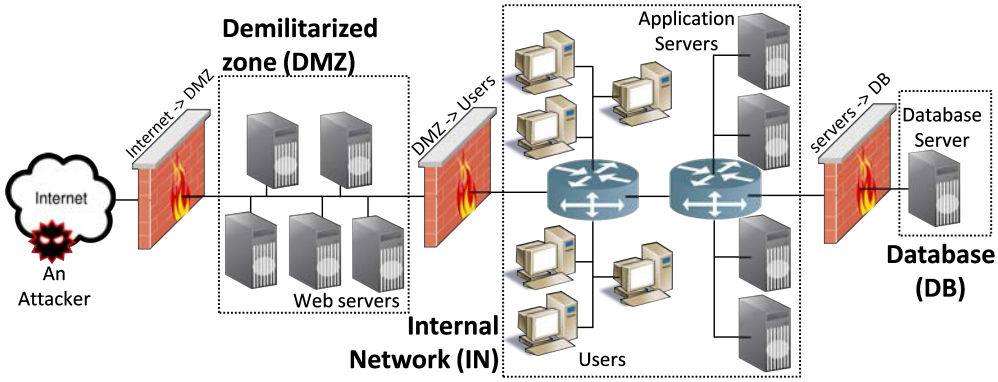


Figure 8.10: The Networked System for Simulations

8.4 Experimental Results

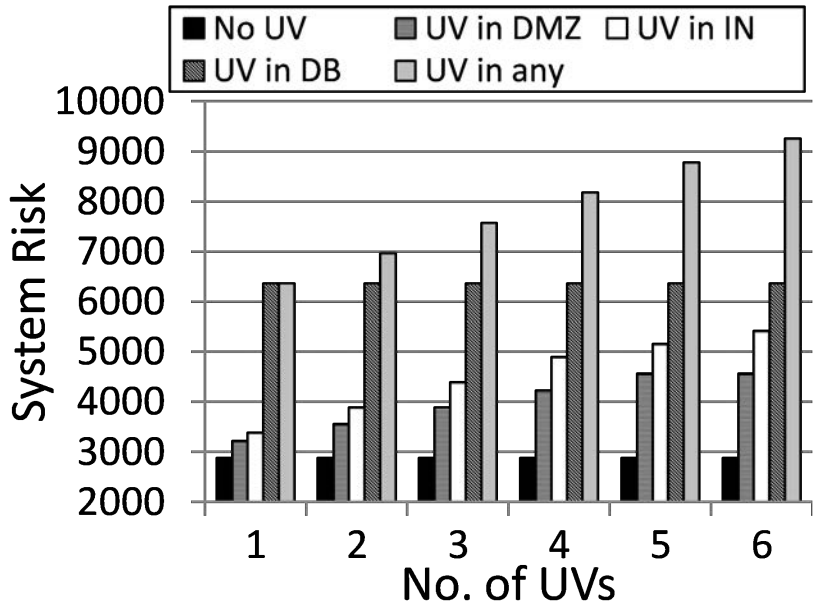
In this section, comprehensive security analyses are conducted, which shows how security of the networked system changes with respect to various attack scenarios assuming unVIP in Section 8.4.1. Further, the accuracy and performance between mitigation strategy algorithms are compared (i.e., naive algorithms ISH and ISV, and approximation algorithms $ISHF$ and $ISVF$) in Section 8.4.2. The networked system for simulations (as shown in Figure 8.10) is assumed with vulnerabilities in Table 8.5, and the same assumptions as in Section 8.1.2.

8.4.1 Security Analysis

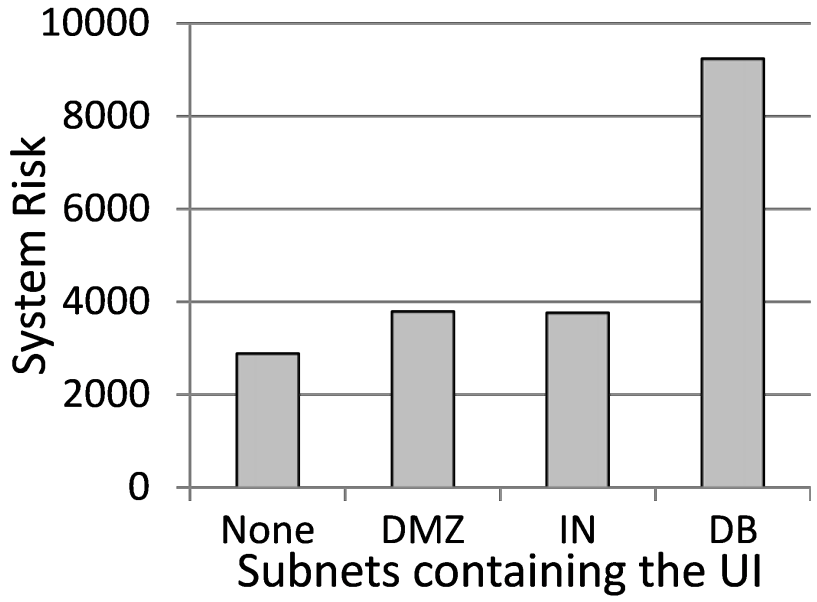
Only UVs and Only UIs in the Networked System: Figure 8.11 shows the security analysis of the networked system with only UVs or only UIs assumed in the networked system. Figure 8.11(a) shows only UVs assumed in the networked system. UPs created as a result of UVs are also taken into account. It shows that the system risk varies based on where UVs are assumed in the networked system (e.g., an UV in the DB subnet resulted in a higher system risk than an UV in the DMZ subnet). An UV in the DB increased the system risk the most (which is the subnet that contained the target host), where the system risk is increased more than double the amount when no UVs are assumed. If subnets not containing the target host (i.e., DS) are taken into account, UVs in the IN subnet has higher system risk when compared to UVs in the DMZ subnet. As the number of UVs grows, system risks for all attack scenarios have increased linearly. In the case of UVs in the DB subnet, increasing the number of UVs did not affect the system risk since there was only a single host.

Figure 8.11(b) shows only an UI assumed in the networked system (with UPs created as a result of an UI), where the VPN host with a single UV is taken into account as the UI. It shows that assuming an UI in the DB subnet increases the system risk the most, followed by an UI in the DMZ and IN subnets. Also, the number of attack paths has increased from 120 to 270 when an UI is assumed in the DB subnet compared to no UIs, which resulted in more than triple the amount of system risk compared to no UIs. The number of attack paths when an UI is assumed in the DMZ and IN subnets are 144 and 145 respectively, but the increase in system risk was higher with an UI in the DMZ subnet (i.e., less number of attack paths with higher system risk). Therefore, the increase in system risk is not linearly proportional to the number of attack paths.

unVIP in only the IN Subnet: Different effects of unVIP assumed only in the IN subnet are shown in Figure 8.12. It shows that an attack scenario without unVIP has a constant system risk with respect to the number of UVs, as well as the attack scenario with only an UI assumed (i.e., these two attack scenarios are not affected by the number of UVs). In the case of UVs assumed, increasing the number of UVs almost linearly increases the system risk. The combined effects of UVs and an UI in the IN subnet showed the highest system risk.



(a)



(b)

Figure 8.11: Security Analysis of the Networked System with only UV/UI

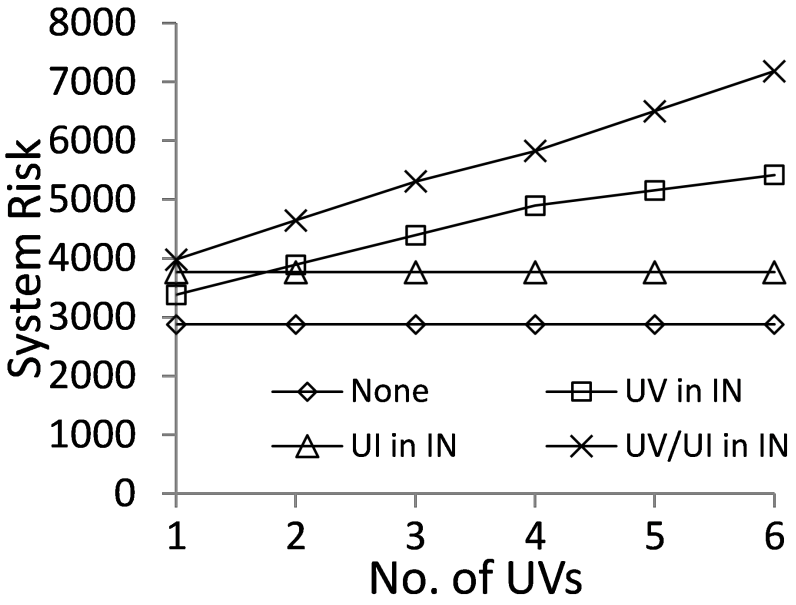
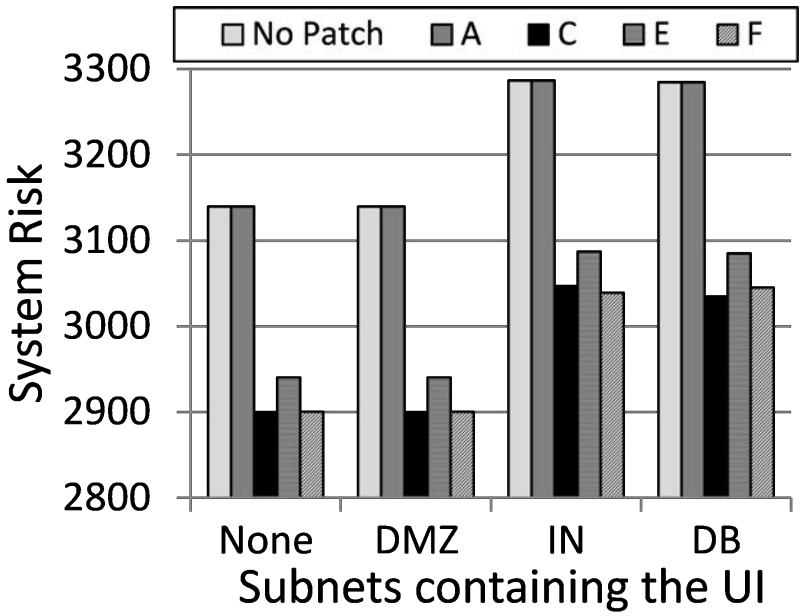


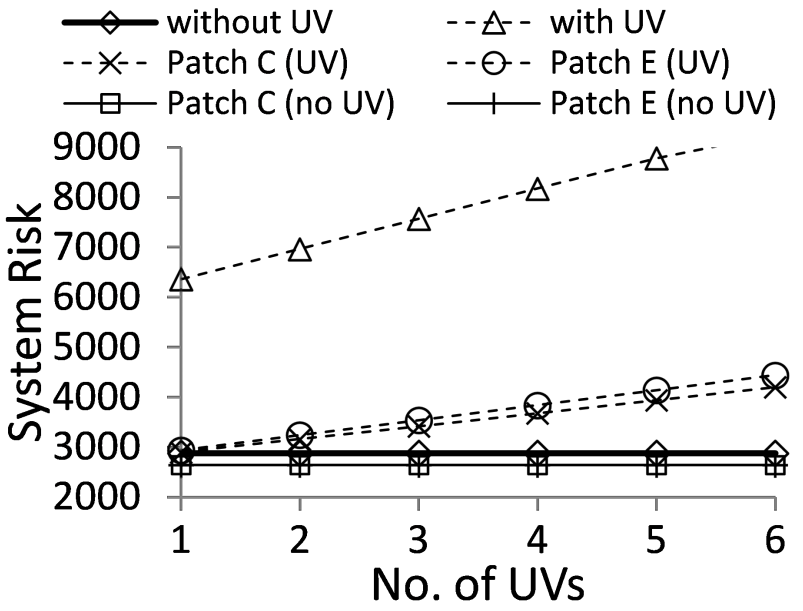
Figure 8.12: unVIP assumed only in the IN Subnet

Patching Different Vulnerabilities in the Networked System: Figure 8.13 shows the effect of patching vulnerabilities in the networked system with unVIP assumed in the networked system. Figure 8.13(a) shows the effect of patching different vulnerabilities. One UV in the DB subnet is assumed with an UI in various subnets. Patching some vulnerabilities does not affect the system risk (e.g., when patching vulnerability V_A, no changes in system risk is observed). Also, patching different vulnerabilities have different changes in the system risk. For example, patching vulnerabilities V_C or V_F minimises the system risk. However, patching vulnerability V_E can also reduce the system risk, but it does not decrease the system risk as much as patching vulnerabilities V_C or V_F.

Figure 8.13(b) shows the changes in system risk when the number of UVs is increased, and the effectiveness of patching a significant vulnerability, which in this attack scenario is vulnerability V_C. When there are no UVs, patching vulnerabilities V_C or V_E does not have a significant effect decreasing the system risk. On the other hand, assuming UVs increased the system risk rapidly (e.g., more than triple for $|UV| = 6$), but patching vulnerabilities V_C or V_E reduces the system risk significantly. Also, patching vulnerability V_C is more effective (e.g., minimises the system risk) compared to patching vulnerability V_E when UVs are assumed in



(a)



(b)

Figure 8.13: Effect of Patching Vulnerabilities in the Networked System

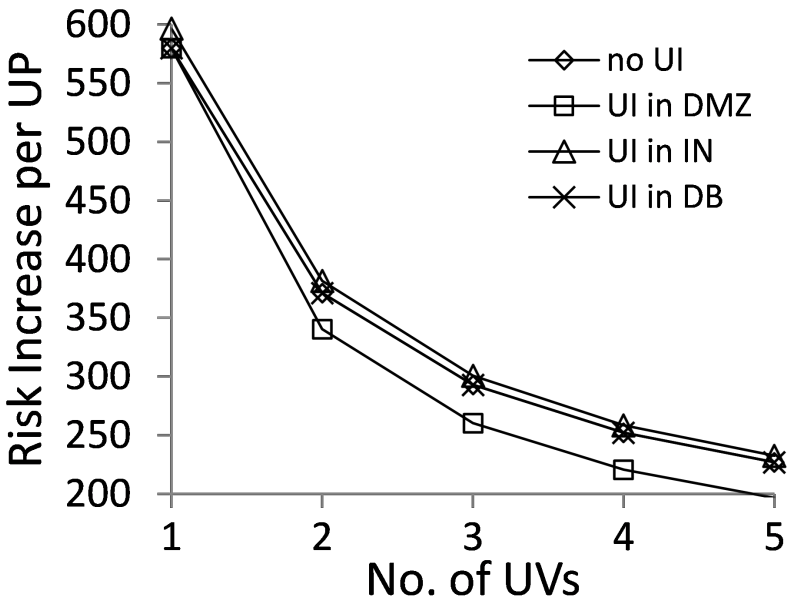


Figure 8.14: UVs in Respect to the System Risk

the networked system. The simulation result shows that it is important to identify significant vulnerabilities to minimise the system risk (e.g., as shown in Section 8.3.3).

Changing Effects of unVIP with respect to the System Risk and the Number of UPs: As the number of unVIP increases (i.e., UVs and UIs in any combinations), the number of UPs are also increased. Figure 8.14 shows the effects of increasing the number of UVs in respect to the system risk. The result represents the increase in system risks for every UP created as a result of UVs and UIs assumed in the networked system. When an UI is assumed in the DMZ subnet, the system risk increased for each UP are less than any other attack scenarios (i.e., UPs created as a result of an UI in the DMZ subnet has the least impact on the system risk). It also shows that each UP created in the IN subnet increases the system risk the most. This shows that appearance of UPs in different subnets have different effects on the system risk (i.e., UPs in some subnets have higher security impact).

These results clearly show that unVIP in certain subnets maximises the system risk, such as in the DB subnet (Figure 8.11(a) and 8.11(b)). Also, the combined ef-

fects of unVIP significantly affect the system risk in comparison to individual unVIP assumed (Figure 8.12), and patching significant vulnerabilities can minimise the effects of these attack scenarios (Figure 8.13(a) and 8.13(b)). Lastly, existence of unVIP in different subnets increased the system risk at different amount, such as UPs in the IN subnet (Figure 8.14).

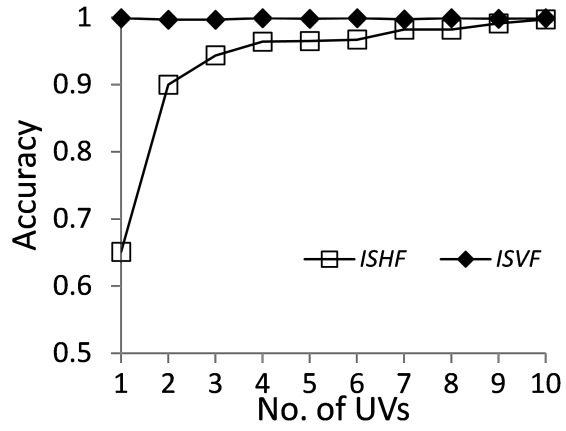
8.4.2 Performance Analysis

unVIP mitigation strategy algorithms are proposed in Section 8.3, and the accuracy and performances of them are evaluated with the networked system shown in Figure 8.10, where the connections between hosts are randomly assigned and the number of vulnerabilities is randomly assigned to each host. The networked system was randomised every iteration of the simulation to get the mean accuracies and performances for various network settings.

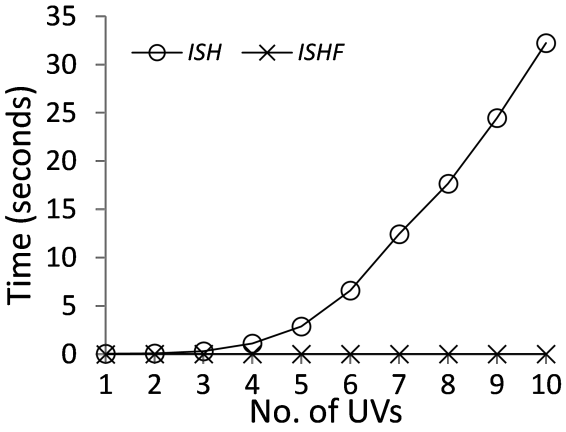
Fixed number of Known Vulnerabilities ($= 10$) **and a Varying Number of UVs:** Figure 8.15 shows the effect of varying number of UVs in the networked system. Varying the number of UVs with a fixed number of known vulnerabilities has affected the accuracy of $ISHF$ significantly, as shown in Figure 8.15(a). When the number of UVs is small (e.g., less than 3), the accuracy of $ISHF$ is less than 0.9. However, the accuracy increases as the number of UVs increases. On the other hand, the accuracy of $ISVF$ is also increased as the number of UVs are increased, but the differences are negligible in respect to $ISHF$. The result shows that the accuracy of identifying significant hosts and vulnerabilities using $ISHF$ and $ISVF$ are considerably low for a small number of UVs. However, increasing the number of UVs increases the accuracy for both $ISHF$ and $ISVF$.

Figure 8.15(b) shows the performance of identifying significant hosts. While the performance of ISH increases exponentially, the performance of $ISHF$ is almost constant. Since increasing the number of UVs exponentially increases the number of possible host combinations, the time taken to compute all possible attack scenarios are exponentially increased as well. In contrast, $ISHF$ has a polynomial time complexity based on dynamic programming and greedy algorithm, so it is much more scalable. As a result, the performance of $ISHF$ has a linear trend and almost negligible compared to the ISH.

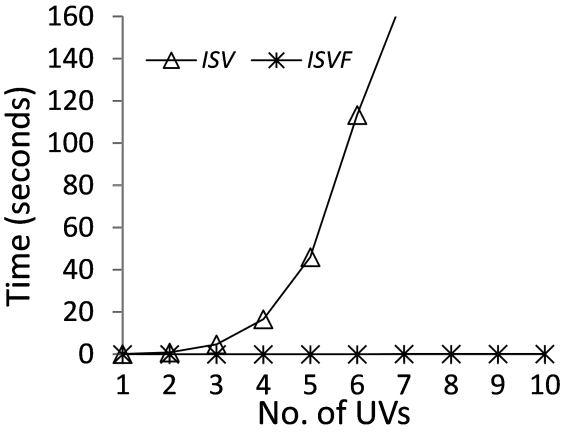
Figure 8.15(c) shows the performance of identifying significant vulnerabili-



(a)



(b)



(c)

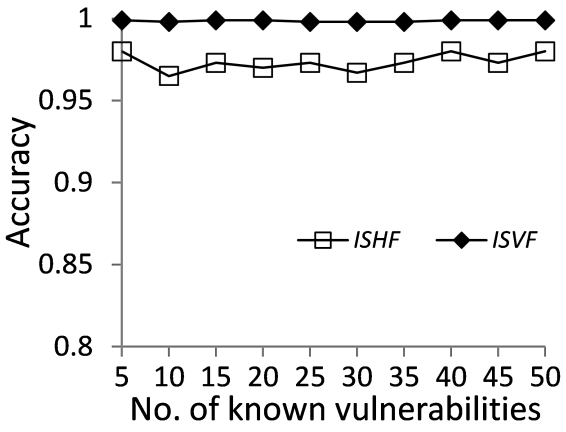
Figure 8.15: Effect of Varying Number of UVs

ties. The performance of ISV increases exponentially even with a small number of UVs. In contrast, the performance of $ISVF$ is almost constant with respect to ISV. It also shows that ISV is significantly slower than ISH (e.g., he time taken for ISH at UV$=5$ is 2.9 seconds and for ISV is 46.0 seconds). Even for a small sized networked system (i.e., 16 hosts), it becomes quickly impractical to use ISV (e.g., time taken is greater than 160 seconds for UV$=7$).

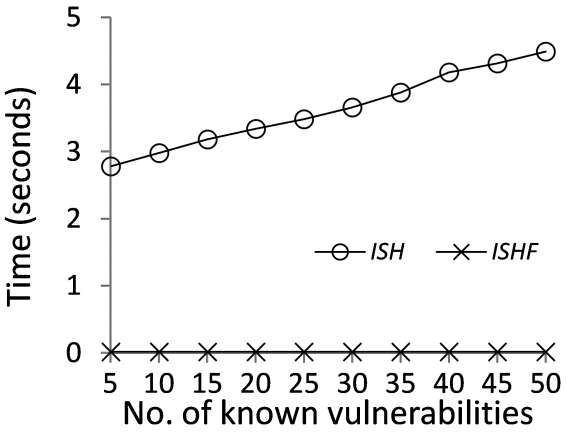
Fixed number of UVs ($=5$) and a Varying Number of Known Vulnerabilities: Figure 8.16 shows the effect of varying the number of known vulnerabilities in the networked system. The accuracies of $ISHF$ and $ISVF$ are shown in Figure 8.16(a). It shows that varying the number of known vulnerabilities has a minimal effect to the accuracies of $ISHF$ and $ISVF$, where they are highly accurate (i.e., greater than 96%). As seen in Table 8.6, the results of $ISVF$ were almost equivalent with them of ISV, and the experimental result shows that the algorithm is very accurate in random network scenarios as well. Although the accuracy of $ISHF$ is not as accurate as $ISVF$, it still achieved the accuracy of 96% and higher for randomly generated network scenarios.

The performances of identifying significant hosts and vulnerabilities are shown in Figure 8.16(b) and 8.16(c) respectively. As the number of known vulnerabilities increases, the time taken for ISH increases linearly. On the other hand, the performance of $ISHF$ is almost constant in respect to the performance of ISH. The time taken to compute significant hosts is linearly increased with respect to the number of known vulnerabilities, because increasing the number of known vulnerabilities does not increases the number of host combinations. Also, the time taken for ISV to compute significant vulnerabilities increases linearly while $ISVF$ is almost constant in respect to ISV. Moreover, the time taken to compute significant vulnerabilities is worse than the time taken to compute significant hosts.

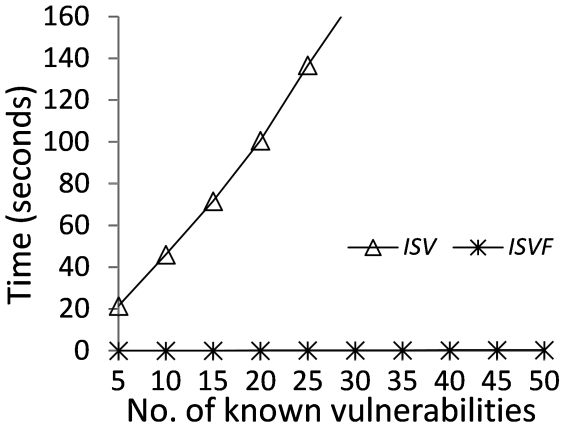
In conclusion, both $ISHF$ and $ISVF$ are much more scalable than ISH and ISV, because they have a polynomial computational complexity using dynamic programming and greedy algorithm. Moreover, their accuracies based on random networked systems show that they can compute nearly equivalent solutions with ISH and ISV. Therefore, it is practical to use $ISHF$ and $ISVF$ to formulate unVIP mitigation strategies, especially for large sized networked systems.



(a)



(b)



(c)

Figure 8.16: Effect of Varying Number of Known Vulnerabilities

8.5 Discussion

unVIP are incorporated into the HARM, and the security of the networked system was analysed taking into account various attack scenarios when unVIP are assumed. Further, new algorithms are developed to determine mitigation strategies that minimise the effects of unVIP based on identifying significant hosts and vulnerabilities in the networked system. The experimental results showed that the combined effects of unVIP vary depending on the attack scenario. Moreover, the approximation algorithms developed (i.e., $ISHF$ and $ISVF$) can compute nearly equivalent solutions and significantly improve the performances compared to the naive algorithms (i.e., ISH and ISV). However, there are limitations of solutions proposed in this chapter, which are discussed in this section.

8.5.1 unVIP Mitigation Strategies and Effectiveness

Although various mitigation strategies exist (e.g., as described in [198]), only two unVIP mitigation strategies are considered (i.e., based on identifying significant hosts and vulnerabilities). However, it is believed that identifying significant hosts and vulnerabilities are important intermediate steps for further network hardening plans, because different mitigation strategies have different effects in security (as demonstrated in the experiments in Section 8.4). There are also various network hardening techniques (NHTs) to enhance the security (e.g., Moving Target Defenses [92, 147, 217]). However, the effectiveness of deploying NHTs are not compared when unVIP are introduced in the networked system. The effect of patching vulnerabilities are taken into account in Section 8.3.3, but other NHTs are not considered. As a result, it is difficult to determine which NHTs should be used as an unVIP mitigation strategy. Moreover, the combined effects of NHTs to minimise the effects of unVIP have not been studied previously. The security model does not take into account multiple NHTs, therefore it is uncertain how the attack surface of the networked system may be changed. Therefore, other NHTs should be incorporated into the security model and analyse the changes in the attack surface when multiple NHTs are deployed.

8.5.2 Implementation in a Real Testbed

Previous work on Analysing the security of unknown vulnerabilities used simulations [45,99], or on a real systems with an assumption of unknown vulnerabilities [198,199]. However, the nature of unVIP uncertainties are difficult to estimate and analyse [30], and evaluating all possible attack scenarios are impractical for a large sized networked system. One of the major limitations of our research is the lack of implementation in a real testbed, where the example networked system was modelled as close to real systems. However, real testbed experiments of unknown attacks remain a difficult task. To address this problem, a real testbed is to be used with an assumed unVIP in the future work, as well as using existing vulnerability databases (e.g., NVD [146]) to conduct experiments such as in [137] (which only dealt with unknown vulnerabilities), along with unknown devices and unknown attack paths. In addition, online unknown attack detection mechanisms can also be used (i.e., as shown in [8,51]) to analyse the security of networked systems with unVIP in real time. By incorporating unVIP detection mechanisms and evaluating possible attack scenarios of an ongoing attack (e.g., advanced persistent threats), one can formulate effective mitigation strategies to prevent unknown attacks penetrating through the networked system.

8.5.3 Security Metrics for Assessing Unknown Attacks

System risk metric was used to analyse the security of the networked system, and also unVIP mitigation strategy algorithms are based on the results of risk analysis. However, assessing the effectiveness of unVIP with an assumed risk value can be misleading, as one cannot estimate the nature of unknown attacks precisely [30]. Therefore, it is necessary to develop various security metrics in respect to unVIP to understand the impact of them from various aspects of security (such as *kzd* safety used in [199], or update existing metrics to measure them [172]).

8.6 Related Work on Security Assessment of Unknown Attacks

Existing work did not take into account the combined effects of unVIP. As a result, not all possible attack scenarios are evaluated in their security analysis.

Unknown Vulnerabilities: Ingols *et al*. [99] assumed an existence of UVs in the networked system to evaluate the security. This approach is later incorporated into a security assessment tool named *NAVIGATOR* [45], where it provided additional functionalities that allow users to either assume a single zero-day vulnerability on an open port of a host, or on all open ports of all hosts in the networked system (i.e., zero-day vulnerabilities assigned to applications in the networked system). The security assessment is based on *what-if* analysis, and therefore it is possible to evaluate the combined effects of known and unknown vulnerabilities. Also, UPs can be identified as a result of introducing UVs in the networked system. However, they did not take into account the combined effects of unVIP (i.e., they did not consider Analysing the security of UIs), and the efficiency of their security model (e.g., adaptability) to analyse various attack scenarios of unVIP is undefined. Moreover, mitigation strategies are not provided (e.g., which host or vulnerabilities with zero-day vulnerabilities maximises the system risk?). Wang *et al*. [198, 199] proposed a new security metric, namely *k*-zero-day (*kzd*) safety metric, to evaluate the effects of unknown vulnerabilities. The *kzd* safety metric describes the safety of the networked system (i.e., a minimum number of zero-day vulnerabilities required from the attacker to the target host). By increasing the value *k* (i.e., more zero-day vulnerabilities are required to compromise the networked system), the difficulty of worst case attack scenario (i.e., only using zero-day vulnerabilities) increases. However, this metric does not consider the combined effects of known and unknown vulnerabilities (i.e., only consider taking into account known vulnerabilities when they can reduce the length of an attack), such as in real life incidents where both known and unknown vulnerabilities are utilised (e.g., Stuxnet [63] and RSA SecurID breach [97], regardless of minimising the length of the attack). Furthermore, combined effects of unVIP are not considered in their analysis.

Unknown Devices: There are concerns with the security of UIs (e.g., BYOD, VPN, USB sharing, Wi-Fi hotspots, phishing emails) [142], but there is a lack of methods to incorporate UIs into the security assessment [73]. UIs introduce new components with undefined properties into the networked system (e.g., new hosts and vulnerabilities). Previous work mainly focused on protection strategies based on trusts. Amoroso [20] proposed three zones with different security protocols enforced depending on the trust level. However, it is difficult to determine the

precise security without Analysing the security of the networked system using a formal security model. Therefore, it is of paramount importance that UIs are incorporated into the security model that is capable of adapting to various attack scenarios (e.g., scalable and adaptable) to cope with these changes.

Unknown Attack Paths: When UVs or UIs are introduced to the networked system, unforeseen UPs may be created. Dai *et al*. [51] proposed an identification method to discover new attack paths created by zero-day vulnerabilities based on system object dependencies. However, these can only be identified at the operating system layer (e.g., zero-day vulnerabilities in other layers may bypass the detection mechanism), and it is based on detection (i.e., after the attack) with possible false-positive alerts. Similarly, Ahmadinejad *et al*. [8] used a hybrid model to correlate alerts of unknown attacks to update an AG. This approach is based on unknown attack detection, which analyses the effects of them after the attack has been launched. In contrast, the aim of this chapter is to explore all possible UPs in the networked system to analyse the security, prior to the attack and enforce mitigation strategies to minimise the effects of unVIP.

8.7 Conclusions

The volume of cyber attacks are increasing, where attackers penetrate through by finding and exploiting vulnerabilities in the networked systems. Many of the security assessments utilise only the known attacks to address flaws in the networked system, but recent cyber incidents (e.g., Stuxnet and RSA SecurID breach) showed that attackers utilise not only known attacks, but a combination of both known and unknown attacks. Only the unknown vulnerabilities are incorporated into security models in previous studies, which are analysed with existing and/or newly developed security metrics (e.g., impact, risk, k-zero-day safety). However, these studies did not take into account the combined effects of unknown attacks.

This chapter addresses the problem of assessing the effects of combined unknown attacks in the networked system. First, unknown attacks are classified into unknown vulnerabilities, devices, and attack paths, which are incorporated into the HARM. Second, the security of networked systems with and without unVIP (taking into account both singular and combined effects) are analysed, as well as developing two unVIP mitigation strategy algorithms that identify significant

hosts and vulnerabilities to harden the networked system. Lastly, experiments were conducted, where the results showed that different unVIP attack scenarios had varying effects on the system risk. These results also showed that deploying the unVIP mitigation strategies can effectively minimise the system risk. Hence, the security effects of unVIP can be analysed using the HARM, which can also provide effective unVIP mitigation strategies.

Part V

General Discussion and Conclusions

Chapter 9

Discussion

The fast-paced growth of modern networked systems (e.g., size, protocols, services) gave many aspects of our lives with versatility, efficiency, and productivity. Through networked systems, various technologies (e.g., mobile devices, sensor nodes) can communicate, share information, and process data in a matter of seconds. However, the diverse and complex nature of modern technologies and networked systems resulted in a lack of fully comprehending flaws that are potentially critical to many aspects of our lives (e.g., loss of cyber/physical assets, information, and communication). Consequently, security is a growing concern for enterprises and individuals, where cyber attacks are becoming vigorous, targeted, and complex.

The work presented in this thesis is intended to inform various communities and individuals of the performance benefits with new methods to assess the security and harden the networked system. The HARM (Chapter 2) is developed on the basis of hierarchical structure to enhance the scalability and adaptability of security assessments. In addition, scalable security evaluation methods based on Importance Measures (IMs) are developed (Chapters 6 and 7) to overcome the exponential complexity of using the Exhaustive Search (ES) method. Lastly, a novel method of assessing and mitigating unknown attacks are developed (Chapter 8), which provides an initial study of analysing the effectiveness of known and unknown attacks combined.

However, there are numerous assumptions and limitations that can be further improved to fully understand security flaws and to strengthen networked systems. This chapter addresses some of these limitations, including directions for extending the work presented in this thesis, as well as ideas for improving methods and techniques of security modelling and analysis in general. The following sections describe the how this thesis addresses research objectives, discuss the limitations of the results presented in the thesis, and provide open problems for future work.

9.1 Research Objectives

The general goal of this thesis was to advance security analysis of large sized and dynamic networked systems and establish efficient and effective security assessment methods. As outlined in Chapter 1, Four specific research objectives were derived from the goal.

1. Develop security modelling and analysis methods to improve scalability and adaptability.

2. Investigate the scalability and adaptability of security modelling and analysis for modern networked systems.

3. Propose efficient and effective security assessment methods to enhance capabilities of formulating countermeasures.

4. Incorporate unknown attacks and analyse the combined effects of them.

Objective 1 required a development of scalable and adaptable security modelling and analysis method, which was addressed in Chapter 2 by presenting a hierarchy based security model, the HARM. A formal definition of the HARM is given, and security assessment methods are described. Moreover, various tools and generation methods are described to generate the HARM, as well as existing security models.

To investigate the scalability and adaptability of security models, Objective 2 required complexity and performance analyses. This objective was addressed in three parts by Chapters 3, 4 and 5, with Chapter 3 addressing the complexity analysis of the HARM and existing security models, and Chapters 4 and 5 looking at performance of security models with respect to their scalability and adaptability, respectively. In addition, new methods of security assessment methods presented in Chapters 6, 7 and 8 are all taken into account with their performance in respect to the scalability and adaptability.

Objective 3 covered the development of new security assessment methods that enhances the scalability of security models without metric or model limitations. Chapters 6 and 7 presented Importance Measures (IMs) to address this objective,

which showed a significant improvement over the ES method in both attack scenarios of the attacker located inside and outside of the networked system. Additionally, Chapter 8 presented mitigation strategies to reduce the effect of unknown attacks with improved algorithms for scalability.

Objective 4 required estimating the unknown behaviours arising from unknown attacks. Chapter 8 addresses this objective by specifying three categories of unknown attacks and incorporating them into the HARM, which enabled the security assessment of unknown attacks in combination with known attacks. Two mitigation strategies are proposed, with each strategy computed based on newly developed naive and approximation algorithms.

In summary, the work presented in this thesis addresses all four of the stated goals: it developed a new security model that is scalable and adaptable; it investigated the performance of existing security models with respect to their scalability and adaptability; it developed scalable security assessment methods based on network properties without limitations of metrics and models; and it incorporated unknown attacks into the HARM and analysed the combined effects of known and unknown attacks.

9.2 Limitations and Future Work

In this thesis, various assumptions are made in the experiments that limited the scope of the research. The sections below discuss some of the limitations in this thesis and identify opportunities for future work.

9.2.1 Properties of the HARM

Practical Issues of Security Modelling and Analysis

There are a few practical issues of modelling security. First, the scalability is a major problem when modelling a large sized networked system [87, 88, 100, 124, 151, 158]. As presented in Section 3.6, not all phases of security models are scalable, especially in the evaluation and modification phases. Second, security models lack in reusability. Various security metrics and their analysis methods are proposed, but they cannot be transferred between security models due to their structural difference. As a result, users must choose their security model concisely

for their need. Third, there is a lack of tools availability. Section 2.4 shows that only seven security models have tools available and only two security models (i.e., AG and AT) have commercial tools. The lack of tools availability significantly reduces the scope of security analysis for users. Currently, the HARM also does not have an available tool, which is to be completed in future work.

Usability of Security Models

Security models are widely used to analyse the security of networked systems in various domains, and they can evaluate many different aspects of security (as shown in Section 3.1). However, some security models can analyse the security more efficiently in different contexts, showing strengths and weaknesses. For example, tree-based security models can efficiently perform security analysis (assuming the tree structure is optimised), but it cannot capture the sequential information without the use of specialised features (e.g., using sequential-*AND* gates). However, the generation of graph-based security models is scalable with a polynomial time complexity, but the generation of tree-based security models is impractical with an automated generation method taken into account. That is, the user must make a decision on which type of security models will be best suited for the security analysis, considering various features of security models and how suitable the security model is for the intention of the user. This problem can be resolved using the HARM, where different security models can be used in various layers in the HARM, and it is also scalable and adaptable.

Changes in Network Systems

Changes in the networked system, such as dynamic network reconfigurations, energy conservation techniques and user populations, are not well captured in existing security models (i.e., adaptability of security models), where existing security models do not consider changes in the networked system [12, 33, 43, 44, 55, 153, 173, 200]. As described in Section 3.5, the security model can be rebuilt every time when there is a change in the networked system. However, the generation of a security model again is costly and time consuming due to many unchanged components in the security model that do not require changes.

Incorporating Various Vulnerabilities

Only a few vulnerabilities (e.g., only the OS vulnerabilities) are taken into account in the examples and experiments for simplicity, but various vulnerabilities can be incorporated and modelled in the HARM. For example, application vulnerabilities can be incorporated by creating another lower layer in the HARM. If vulnerabilities from different layers are related (e.g., an application layer is a pre-condition of an OS layer vulnerability), then their relationship can be captured in the HARM with post and pre conditions. However, experimental results showed that regardless of varying number of vulnerabilities, the HARM outperforms existing security models (as shown in Chapters 4 and 5).

Enhancing Generation Methods for Security Models

The logic reduction algorithms can be used to improve the generation of tree-based security models [94]. However, there are limitations of the methods used to automatically generate tree-based security models. First, equivalent attack paths are assumed to be grouped, but this must be pre-processed (i.e., similar nodes to be grouped). A naive approach (e.g., brute force) will result in an exponential computational complexity. Second, the memory space to keep track of the current attack path is required to avoid any loops during the recursion using the logic reduction technique, but the memory space required has a size complexity of O($n.n!$).

The Cloud Computing resources can be used to enhance the generation of security models [86]. However, there is a computational overhead, because this method depends on how the security model is subdivided onto sub-processors. A naive partition method (i.e., one node is partitioned at a time) was used, which is relatively simple to implement. As a result, the overhead observed when the number of nodes in the AG was negligible. If more complex partitioning algorithms are used (e.g., as in [115]), the computational complexity of the overhead may grow. Also, there is a big graph mining technique that can analyse a very large sized graph efficiently [113]. Also, there is MapReduce/Hadoop framework and its methods for processing large data sets (e.g., a very large sized AG) [54]. However, the use of big graph mining techniques and MapReduce/Hadoop framework has not been taken into account. Lastly, there is a cost associated with using

the Cloud (e.g., running costs, implementation and update costs). Although the price of using Cloud computing is becoming more flexible (i.e., affordable [17]), the sum of all costs may be significant depending on the amount of hardware resources required. The relationship between the security models and their requirements should be investigated to optimally distribute hardware resources to minimise the cost in future work.

Security Evaluation and Overheads

Various security metrics can be computed using the HARM (e.g., system risk, benefit, ROI), which are equivalent to existing security models such as an AG. However, the complexity analysis and experimental results show a significant scalability improvement for the HARM over traditional security models (e.g., an AG). Using the hierarchy improves the performance because multiple computations can be grouped in the HARM, whereas a single layered security model does not have such function. In contrast, if the outputs of all possible attack scenarios are expected, then there exists an overhead in the HARM to compute the correlation between the upper and lower layer components. However, such output was not taken into account nor analysing this overhead, which should be addressed in future work.

9.2.2 Scalability and Adaptability

Validation using a Real System

The networked system was modelled as close as to the real network settings, but one of the limitations is the lack of implementation in a real testbed for validation. There are a limited number of previous work that used a testbed or a practical network [60, 104, 147, 192]. On the other hand, assessing the security of unknown attacks (unVIP) remains a difficult task as it is difficult to populate properties of them [30]. To address this problem, a real testbed with an assumed unVIP in the future work should be taken into account, as well as using existing vulnerability databases (e.g., NVD [146]) to conduct experiments such as in [137] (which only dealt with unknown vulnerabilities), along with unknown devices and unknown attack paths. To extend, one can incorporate online unknown attack detection

mechanisms (i.e., as shown in [8, 51]) to analyse the security of networked systems with unVIP in real time. By incorporating unVIP detection mechanisms and evaluating possible attack scenarios of an ongoing attack (e.g., advanced persistent threats), we can formulate effective mitigation strategies to prevent unknown attacks penetrating through the networked system.

Scalability of Security Model Phases

The AG suffered the scalability problem due to independent connections between the model components (Chapters 3 and 4). The representation of the AG had more edges compared to the HARM. The number of nodes was the same, but the number of edges was greater in the AG. However, the construction time shows that there is only a little difference between the HARM and the AG. Also, only a few existing security models compared the performance against the AG in the construction and the evaluation phases. None of the security models considered an update event in the networked system, and how their models are updated. The similarity between the AG, LAG, and the MPG is that they are represented as a single layer in a security model. Those security models suffer from a scalability problem in the representation, and also the modification may affect all nodes in the security model in the worst case. However, hierarchical models, such as the HARM and the TLAG, have fewer structural changes as they have less relationship between nodes and security models that are represented in a single layer. In addition, the HARM has fewer components to update, because the TLAG has higher number of lower layer models.

Network Structure and Attack Scenarios

In a real life networked systems, network topologies are complex with many combinations of simple topologies (e.g., combinations of star, tree, ring and mesh topologies). The worst case complexity defined in Chapter 3 by analysing a fully connected topology gives the upper bound performance. Although a few complex network topologies are created in the experiment, it is difficult to estimate the scalability of security models when deployed in real systems. However, the complexity analysis and experimental results are a good estimator of using security models in real life. Moreover, such estimation can be used to design networked

systems that are secure as well as efficient to analyse the security to mitigate attacks.

MTD techniques with Various Security Metrics

A single security metric cannot capture all effects of MTD techniques. Moreover, not all MTD techniques can be analysed using the HARM. For in-depth security analysis of MTD techniques, various security metrics should be used. Hence, a wide range of security metrics should be taken into account in future work, and how security metrics are affected by different MTD techniques, as well as how and which layer MTD techniques should be deployed.

9.2.3 Security Analysis using the HARM

Vulnerabilities without Security Metrics

Using a vulnerability scanner *NESSUS* [28], about 60 vulnerabilities of a real host machine were reported (the host based on a Windows XP SP1). However, only 11 vulnerabilities had CVSS BS, which gives a set of security metrics associated with these vulnerabilities. There was a textual description of vulnerabilities (e.g., Vulnerability Synopsis and Vulnerability Description), but this is difficult to process automatically. Moreover, 10 vulnerabilities are scanned without any descriptions. Incomplete security data makes difficult to automate and analyse the network security. Other sources of vulnerability scanners and security metrics should be investigated in future work.

Optimal Network Hardening

Existing optimal network hardening methods use heuristic algorithms to find the likely attacks, such as min-paths, and analyse the security based on these calculations [12, 33, 43, 44, 55, 153, 173, 200]. Heuristic methods provide reasonable security solutions, but not all security analyses can be covered by heuristic methods, such as analysing the overall network security from attacks that have intentions to compromise most or all network hosts. Moreover, security metrics are considered as a static value, that the network hardening may not be an optimal

solution if the security metric values change. To address this problem, network-centric security analysis method using the network centrality and security metric measures are developed (Chapters 6 and 7). However, the accuracies of these methods cannot be guaranteed. Hence, a further investigation is required to find the optimal network hardening methods that are not model and metric oriented in future work.

Changes in Security Metric Values

Changes of security metric values (e.g., the impact of an attack, the probability of an attack success and the risk of an attack) affect the network hardening decisions. The network hardening methods are dependent on the values of security metrics, and these are considered as static values (i.e., constant). The effect on the network hardening decision should be changing over a period of time as security metric values change. As a result, the network hardening decision may not be an optimal solution, and the networked system may requires additional network hardenings deployed to satisfy the security constraints, which will result in higher cost and time spent. Therefore, it is required to take into account changes of security metric values and how they can affect the security of the networked system.

Optimisation of the IMs w.r.t MTD techniques

The proportion of important nodes in the HARM is randomly but reasonably selected. As discussed in [90], there exists an optimal number of important nodes for an optimal performance of security analysis. Results in Chapters 6 and 7 showed that using the ES method is infeasible in practice for a large sized networked system, whereas using the IMs is more scalable and practical. However, it is difficult to decide what proportion of nodes to select, as different networks have different optimal proportions based on the topology and settings.

Also, various network scenarios are not taken into account (e.g., network topologies, the number of hosts and VMs, and the number of vulnerabilities) and how the effectiveness of MTD techniques are affected. For security analysis, a nearly equivalent security solution is computed for various network scenarios [89]. This study should be extended in future work to investigate various

network scenarios using the IMs to compute the effectiveness of MTD techniques.

Multiple Target Hosts and Locations

Examples and experiments only considered a single target host, but the attack scenario could also compromise multiple hosts in multiple locations (e.g., a denial of service attack). NCMs are used to consider the overall security of the networked system (i.e., implicitly covering an attack with multiple targets), but it is unknown how close the security solution comes near to the optimal solution. However, the number of attack scenarios may increase in such cases, but computing IMs only change the rankings of hosts and vulnerabilities without affecting the performance. A different location for the target brings a new attack scenario, and such cases should be considered in future work.

Adjusting the Weight of Combined IMs

The combined IMs are measured with a weight value of $\alpha = 0.5$ in Chapter 6, which was reasonably selected. Results showed that using the combined IMs, a nearly equivalent solution to the ES method is computed. Further, the result showed that it is possible to compute different solution that can match more closely with the ES method using different weights. That is, assigning the weight value appropriately can improve the set coverage, as well as the order of vulnerabilities in the PSV.

Order of Vulnerabilities in the PSV

Although the set coverage was relatively high for various network scenarios, the actual order of vulnerabilities varies when combined IMs are used. To address this problem, one can perform a security analysis for the vulnerabilities in the PSV only, which does not consider all possible cases. The performance of this method is in fraction of the ES method.

Attacks on Less Important Hosts

The drawback of IMs based security analysis is when an attacker attempts to penetrate through the networked system by compromising less important hosts

(and/or less significant vulnerabilities). In such case, one cannot capture threats efficiently using the IMs. However, it is possible that by covering certain amount of important hosts and vulnerabilities, there will be a minimum set that covers all possible attack routes (i.e., set of hosts that at least one of selected host is in any given attack scenario). Such method should be analysed to investigate how many important hosts and vulnerabilities should be considered (e.g., proportion of hosts and vulnerabilities) to cover all possible attack scenarios.

9.2.4 Assessing the Effectiveness of Unknown Attacks

9.2.5 Unknown Attack Modelling

Unknown attacks exploit a vulnerability that exists in a system that has no countermeasure yet. Although zero-day attacks are rare among all cyber attacks, they are discovered and used by attackers [61, 77]. Finding zero-day vulnerabilities is a difficult task in a complex system, for both developers and attackers. There are means to detect zero-day attacks [168, 194, 195], but measuring security metrics of zero-day attacks are difficult in general because their characteristics are diverse and not well known [30]. One can model zero-day vulnerabilities in a security model by hypothesising each application in the networked system with a zero-day vulnerability [45, 99]. Another method is to measure the length of the zero-day attack path [198, 199]. However, current zero-day expressions lack in functionalities in the security models due to their limited descriptions, and security solutions do not reflect practical requirements because they are not considered in combinations with known attacks. To address these problems, Chapter 8 proposed methods to incorporate unknown attacks into the HARM and analyse the combined effects of known and unknown attacks. However, there is still a need for validation using a real testbed to confirm experimental results.

Effectiveness of Mitigation Strategies for Unknown Attacks

Although various mitigation strategies exist (e.g., as described in [198]), only two unVIP mitigation strategies are considered (i.e., based on identifying significant hosts and vulnerabilities). But identifying significant hosts and vulnerabilities are one of the most important intermediate steps for further network hardening plans, because different mitigation strategies have different effects in security.

Nevertheless, other unVIP mitigation strategies should be incorporated and evaluated for their effectiveness to further enhance the security of networked systems.

Security Metrics for Assessing Unknown Attacks

System risk metric was used to analyse the security of the networked system, and also unVIP mitigation strategy algorithms are based on the results of the risk analysis. However, assessing the effectiveness of unVIP with an assumed risk value can be misleading, as one cannot estimate the nature of unknown attacks precisely [30]. Therefore, it is necessary to develop various security metrics in respect to unVIP to understand the impact of them from various aspects of security (such as *kzd* safety used in [199], or update existing metrics to measure them [172]).

Open Problems

There is still a lack of features of security models that aid users to decide the optimal security decision. Most security models do not incorporate how countermeasure selection may affect the system performance. There are studies considering both security and performance, but these are not captured in security models. As a result, users may know the effectiveness of countermeasures, but the influence to the performance in their system may change unexpectedly as a result of deploying the countermeasure.

Also, different security models have different features to enhance the security analysis. However, these features are specifically made for each security models, such that different capabilities of different security models cannot be shared. This becomes cumbersome from the user's point of view. For example, if the user have used an AG to evaluate the risk of the system but now wants to evaluate the risk of the network based on different reachability groups, the user must reconstruct the AG for each reachability group. However, this could have been done without reconstruction if the user used a MPAG [99], which has a feature to capture different reachability groups.

Chapter 10

Conclusions

Security analysis of the networked system in an efficient and effective manner is an important task for enterprises and individuals to protect cyber/physical assets from cyber criminals. This thesis has identified two problems of existing security models and their analysis methods that significantly affect the performance of security analysis. First, the scalability problem of security models resulted in an inability to evaluate the security of large sized networked systems, and second, the adaptability of existing security models are either inefficient or unknown to analyse the security of networked systems with frequent changes. To address these problems, the HARM is developed, and scalable security analysis methods are developed and analysed. Moreover, unknown attacks are specified and incorporated into the HARM framework to assess the effect of known and unknown attack combined.

This thesis has made four distinctive contributions to knowledge in the domain of network security and network hardening. These are

1. A development of the HARM that improves the scalability and adaptability of security modelling and analysis. The HARM is formally defined where it is possible to incorporate various security models onto its layers for functionalities. Further, efficient automated generation method for tree-based security models is developed based on logic reduction techniques, and the use of parallel computing techniques to enhance the performance of security evaluation is described.

2. A comparative analysis of security models is presented with respect to complexity and performance. The lifecycle of security models is described, where each five phases in the lifecycle can be analysed and compared for various security models. This analysis provides a basis for emerging security models to evaluate and compare their scalability and adaptability, which

sets a guideline for developing new security modelling and analysis methods.

3. Scalable security assessment methods based on importance measures are developed to efficiently compute important hosts and vulnerabilities, which are based on only the properties of networked systems. Attack scenarios with an attacker located in either inside or outside of the networked system are taken into account, where existing network centrality measures are used for the attacker located outside the networked system, and newly developed network centrality measures are used for the attacker located inside the networked system. The performance of security assessment using the importance measures has improved significantly compared to the exhaustive search method, while maintaining a high accuracy of security solutions.

4. A comprehensive security analysis of unknown attacks is conducted by incorporating them into the HARM and analysing the effects of both known and unknown attacks combined. Unknown attacks are specified into three different categories, and new mitigation strategies to minimise the effects of unknown attacks are developed, which are to identify significant hosts and vulnerabilities when unknown attacks are assumed in the networked system. The location of unknown attacks assumed significantly affected the security assessment, The effective mitigation strategies are computed efficiently using the approximation algorithms, where the accuracies of them are nearly equivalent to the naive method with poor performance.

Together, this research provides a foundation for further work, both in developing scalable and adaptable security modelling and analysis methods, and in formulating cost-effective and time-efficient optimal network hardening solutions. The security modelling and analysis results presented in this thesis may be used to inform security analysts and security researchers as they advance in securing the cyberspace.

References

[1] 15408-1:2005, I. Common criteria for information technology security evaluation - part 1: Introduction and general model, 2005.

[2] 17799-1:2005, I. Information technology - security techniques - code of practice for information security management, 2005.

[3] 800-30, N. S. P. Risk management guide for information technology systems, July 2002.

[4] 800-55, N. S. P. Performance measurement guide for information security, Sept. 2007.

[5] 800-55, N. S. P. Security metrics guide for information technology systems, Sept. 2007.

[6] 800-80, N. S. P. Guide for developing performance metrics for information security, May 2006.

[7] ABADI, M., AND JALILI, S. A Particle Swarm Optimization Algorithm for Minimization Analysis of Cost-Sensitive Attack Graphs. *The ISC International Journal of Information Security 2*, 1 (2010), 13–32.

[8] AHMADINEJAD, S., JALILI, S., AND ABADI, M. A Hybrid Model for Correlating Alerts of Known and Unknown Attack Scenarios and Updating Attack Graphs. *Computer Networks 55*, 9 (2011), 2221 – 2240.

[9] AHN, Y., HAN, S., KWAK, H., MOON, S., AND JEONG, H. Analysis of Topological Characteristics of Huge Online Social Networking Services. In *Proc. of the 16th international conference on World Wide Web (WWW 2007)* (2007), ACM, pp. 835–844.

[10] AL-WAKEEL, S., AND S., A.-S. PRSA: A Path Redundancy Based Security Algorithm for Wireless Sensor Networks. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC 2007)* (March 2007), pp. 4156–4160.

[11] ALATA, E., NICOMETTE, V., KAANICHE, M., DACIER, M., AND HERRB, M. Lessons learned from the deployment of a high-interaction honeypot. In *Proc. of Sixth European Dependable Computing Conference (EDCC 2006)* (Oct. 2006), pp. 39 –46.

[12] ALBANESE, M., JAJODIA, S., AND NOEL, S. Time-efficient and Cost-effective Network Hardening using Attack Graphs. In *Proc. of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)* (Los Alamitos, CA, USA, 2012), IEEE Computer Society.

[13] ALBANESE, M., JAJODIA, S., PUGLIESE, A., AND SUBRAHMANIAN, V. Scalable Analysis of Attack Scenarios. In *Proc. of the 16th European Symposium on Research in Computer Security (ESORICS 2011)*, V. Atluri and C. Diaz, Eds., vol. 6879 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011, pp. 416–433.

[14] ALBANESE, M., JAJODIA, S., SINGHAL, A., AND WANG, L. An Efficient Approach to Assessing the Risk of Zero-day Vulnerabilities. In *Proc. of the 10th International Conference on Security and Cryptography (SECRYPT 2013)* (2013).

[15] ALCATEL-LUCENT. Alcatel-Lucent CloudBand.

[16] ALMOHRI, H. *Security Risk Prioritization for Logical Attack Graphs*. PhD thesis, Kansas State University, 2008.

[17] AMAZON. Amazon EC2.

[18] AMMANN, P., PAMULA, J., RITCHEY, R., AND STREET, J. A Host-based Approach to Network Attack Chaining Analysis. In *Proc. of the 21st*

Annual Computer Security Applications Conference (ACSAC 2005) (2005), pp. 10 pp.–84.

[19] AMMANN, P., WIJESEKERA, D., AND KAUSHIK, S. Scalable, graph-based network vulnerability analysis. In *Proc. of the 9th ACM conference on Computer and communications security (CCS 2002)* (New York, NY, USA, 2002), ACM, pp. 217–224.

[20] AMOROSO, E. From the Enterprise Perimeter to a Mobility-Enabled Secure Cloud. *IEEE Security & Privacy 11*, 1 (Jan 2013), 23–31.

[21] AMOROSO, E. G. *Fundamentals of Computer Security Technology*. Prentice-Hall, Inc., 1994.

[22] ANTONATOS, S., AKRITIDIS, P., MARKATOS, E., AND ANAGNOSTAKIS, K. Defending Against Hitlist Worms Using Network Address Space Randomization. In *Proc. of the 2005 ACM Workshop on Rapid Malcode (WORM 2005)* (New York, NY, USA, 2005), ACM, pp. 30–40.

[23] ARDI, S., BYERS, D., AND SHAHMEHRI, N. Towards a Structured Unified Process for Software Security. In *Proc. of International Workshop on Software Engineering for Secure Systems (SESS 2006)* (New York, NY, USA, 2006), SESS '06, ACM, pp. 3–10.

[24] ARTZ, M. NetSPA : a Network Security Planning Architecture. In *Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2002*. Massachusetts Institute of Technology, 2002. Thesis (M.Eng.)–Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2002.

[25] BACA, D., AND PETERSEN, K. Prioritizing Countermeasures Through the Countermeasure Method for Software Security (CM-Sec). In *Product-Focused Software Process Improvement*, M. A. Babar, M. Vierimaa, and M. Oivo, Eds., no. 6156 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2010, pp. 176–190.

[26] BAGNATO, A., KORDY, B., MELAND, P., AND SCHWEITZER, P. Attribute Decoration of Attack-Defense Trees. *Int. J. Secur. Softw. Eng. 3*, 2 (Apr. 2012), 1–35.

[27] BALZAROTTI, D., MONGA, M., AND SICARI, S. Assessing the Risk of Using Vulnerable Components. In *Quality of Protection*, D. Gollmann, F. Massacci, and A. Yautsiukhin, Eds., vol. 23 of *Advances in Information Security*. Springer US, 2006, pp. 65–77.

[28] BEALE, J., DERAISON, R., MEER, H., TEMMINGH, R., AND WALT, C. The NESSUS project. *Syngress Publishing* (2002).

[29] BHATTACHARYA, P., AND GHOSH, S. Analytical Framework for Measuring Network Security using Exploit Dependency Graph. *Information Security, IET 6*, 4 (Dec 2012), 264–270.

[30] BILGE, L., AND DUMITRAS, T. Before We Knew It: An Empirical Study of Zero-day Attacks in the Real World. In *Proc. of the 19th ACM Conference on Computer and Communications Security (CCS 2012)* (New York, NY, USA, 2012), CCS '12, ACM, pp. 833–844.

[31] BISTARELLI, S., DALL'AGLIO, M., AND PERETTI, P. Strategic Games on Defense Trees. In *Formal Aspects in Security and Trust*, T. Dimitrakos, F. Martinelli, P. Ryan, and S. Schneider, Eds., vol. 4691 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 1–15.

[32] BISTARELLI, S., FIORAVANTI, F., AND PERETTI, P. Defense trees for economic evaluation of security investments. In *Proc. of the first International Conference on Availability, Reliability and Security (ARES 2006)* (2006).

[33] BOYD, S., AND VANDENBERGH, L. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

[34] BRYKCZYNSKI, B., AND SMALL, R. Reducing Internet-based Intrusions: Effective Security Patch Management. *Software, IEEE 20*, 1 (Jan 2003), 50–57.

[35] BULDAS, A., LAUD, P., PRIISALU, J., SAAREPERA, M., AND WILLEMSON, J. Rational Choice of Security Measures via Multi-parameter Attack Trees. In *Critical Information Infrastructures Security*. Springer, 2006, pp. 235–248.

[36] BULDAS, A., AND MAGI, T. Practical Security Analysis of E-Voting Systems. In *Advances in Information and Computer Security*, A. Miyaji, H. Kikuchi, and K. Rannenberg, Eds., vol. 4752 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 320–335.

[37] BYERS, D., ARDI, S., SHAHMEHRI, N., AND DUMA, C. Modeling Software VulnerabilitiesWith Vulnerability Cause Graphs. In *Proc. of the 22nd IEEE International Conference on Software Maintenance (ICSM 2006)* (2006), pp. 411–422.

[38] CADINI, F., ZIO, E., AND PETRESCU, C. Using centrality measures to rank the importance of the components of a complex network infrastructure. In *Critical Information Infrastructure Security*, R. Setola and S. Geretshuber, Eds., vol. 5508 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 155–167.

[39] CASOLA, V., DE BENEDICTIS, A., AND ALBANESE, M. A Moving Target Defense Approach for Protecting Resource-constrained Distributed Devices. In *Proc. of the IEEE 14th International Conference on Information Reuse and Integration (IRI 2013)* (2013), pp. 22–29.

[40] CAVUSOGLU, H., CAVUSOGLU, H., AND ZHANG, J. Economics of Security Patch Management. In *Proc. of the Fifth Workshop on the Economics of Information Security (WEIS 2006)* (2006).

[41] CHANG, F., JI, M., LEUNG, S., MACCORMICK, J., PERL, S., AND ZHANG, L. Myriad: Cost-Effective Disaster Tolerance. In *Proc. of*

USENIX Conference on File and Storage Technologies (FAST 2002) (2002), vol. 2, p. 8.

[42] CHEN, B., KALBARCZYK, Z., NICOL, D., SANDERS, W., TAN, R., TEMPLE, W., TIPPENHAUER, N., VU, A., AND YAU, D. Go with the Flow: Toward Workflow-oriented Security Assessment. In *Proc. of the 2013 Workshop on New Security Paradigms Workshop (NSPW 2013)* (New York, NY, USA, 2013), NSPW '13, ACM, pp. 65–76.

[43] CHEN, F., LIU, D., ZHANG, Y., AND SU, J. A Scalable Approach to Analyzing Network Security using Compact Attack Graphs. *Journal of Networks 5*, 5 (2010), 543–550.

[44] CHEN, F., WANG, L., AND SU, J. An efficient approach to minimum-cost network hardening using attack graphs. In *Proc. of 4th International Conference on Information Assurance and Security (ISIAS 2008)* (Sept. 2008), pp. 209 –212.

[45] CHU, M., INGOLS, K., LIPPMANN, R., WEBSTER, S., AND BOYER, S. Visualizing Attack Graphs, Reachability, and Trust Relationships with NAVIGATOR. In *Proc. of the 7th International Workshop on Visualization for Cyber Security (VizSec 2010)* (New York, NY, USA, 2010), ACM, pp. 22–33.

[46] CHUNG, C., KHATKAR, P., XING, T., LEE, J., AND HUANG, D. NICE: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems. *IEEE Transactions on Dependable and Secure Computing 10*, 4 (July 2013), 198–211.

[47] COX, B., EVANS, D., FILIPI, A., ROWANHILL, J., HU, W., DAVIDSON, J., KNIGHT, J., NGUYEN-TUONG, A., AND HISER, J. N-Variant Systems: A Secretless Framework for Security through Diversity. In *Proc. of the 15th USENIX Security Symposium (USENIX Security 2006)* (2006), pp. 105–120.

[48] CROUSE, M., AND FULP, E. A Moving Target Environment for Computer Configurations using Genetic Algorithms. In *Proc. of the 4th Symposium on Configuration Analytics and Automation (SAFECONFIG 2011)* (2011), pp. 1–7.

[49] DACIER, M., AND DESWARTE, Y. Privilege Graph: An Extension to the Typed Access Matrix Model. In *Proc. of European Symposium on Research in Computer Security (ESORICS 1994)*, D. Gollmann, Ed., vol. 875 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1994, pp. 319–334.

[50] DACIER, M., DESWARTE, Y., AND KAANICHE, M. Models and Tools for Quantitative Assessment of Operational Security. Tech. Rep. LAAS Reserach Report 96493, LAAS-CNRS, 1996.

[51] DAI, J., SUN, X., AND LIU, P. Patrol: Revealing Zero-Day Attack Paths through Network-Wide System Object Dependencies. In *Proc. of the 18th European Symposium on Research in Computer Security (ESORICS 2013)*, J. Crampton, S. Jajodia, and K. Mayes, Eds., vol. 8134 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 536–555.

[52] DANEV, B., MASTI, R., KARAME, G., AND CAPKUN, S. Enabling Secure VM-vTPM Migration in Private Clouds. In *Proc. of the 27th Annual Computer Security Applications Conference (ACSAC 2011)* (New York, NY, USA, 2011), ACM, pp. 187–196.

[53] DAWKINS, J., AND HALE, J. A Systematic Approach to Multi-stage Network Attack Analysis. In *Proc. of Second IEEE International Information Assurance Workshop (IWIA 2004)* (2004), pp. 48–56.

[54] DEAN, J., AND GHEMAWAT, S. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM 51*, 1 (2008), 107–113.

[55] DEWRI, R., POOLSAPPASIT, N., RAY, I., AND WHITLEY, D. Optimal Security Hardening using Multi-objective Optimization on Attack Tree Models of Networks. In *Proc. of ACM conference on Computer and communi-*

cations security (CCS 2007) (New York, NY, USA, 2007), ACM, pp. 204–213.

[56] DEWRI, R., RAY, I., POOLSAPPASIT, N., AND WHITLEY, D. Optimal Security Hardening on Attack Tree Models of Networks: a Cost-benefit Analysis. *International Journal of Information Security 11*, 3 (June 2012), 167–188.

[57] EDGE, K. *A Framework for Analyzing and Mitigating the Vulnerabilities of Complex Systems via Attack and Protection Trees*. PhD thesis, Air Force Institute of Technology, 2007.

[58] EDGE, K., RAINES, R., GRIMAILA, M., BALDWIN, R., BENNINGTON, R., AND REUTER, C. The Use of Attack and Protection Trees to Analyze Security for an Online Banking System. In *Proc. of the 40th Annual Hawaii International Conference on System Sciences (HICSS 2007)* (Jan 2007), pp. 144–151.

[59] ERDEM SARIYÜCE, A., SAULE, E., KAYA, K., AND ÇATALYÜREK, Ü. Shattering and Compressing Networks for Centrality Analysis. *ArXiv e-prints* (Sep 2012).

[60] EVANS, D., NGUYEN-TUONG, A., AND KNIGHT, J. Effectiveness of Moving Target Defenses. In *Moving Target Defense*, S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Eds., vol. 54 of *Advances in Information Security*. Springer New York, 2011, pp. 29–48.

[61] EVANS, N., AND YUAN, X. Observation of recent Microsoft Zero-Day vulnerabilities. In *Proc. of the 49th Annual Southeast Regional Conference (ACMSE 2011)* (New York, NY, USA, 2011), ACM, pp. 328–329.

[62] EWING, G., PAWLIKOWSKI, K., AND MCNICKLE, D. Akaroa-2: Exploiting network computing by distributing stochastic simulation. In *Proc. European Simulation Multiconference (ISCS 1999)* (1999), pp. 175–181.

[63] FALLIERE, N., MURCHU, L., AND CHIEN, E. W32. Stuxnet Dossier. *White paper, Symantec Corp., Security Response* (2011).

[64] FLOYD, R. Algorithm 97: Shortest path. *Commun. ACM 5*, 6 (1962), 345.

[65] FOO, B., WU, Y.-S., MAO, Y.-C., BAGCHI, S., AND SPAFFORD, E. ADEPTS: Adaptive Intrusion Response using Attack Graphs in an E-commerce Environment. In *Proc. of International Conference on Dependable Systems and Networks (DSN 2005)* (2005), pp. 508–517.

[66] FOR INTERNET SECURITY, T. C. Cis consensus information security metrics, Nov. 2010.

[67] FOVINO, I., AND MASERA, M. Through the Description of Attacks: A Multidimensional View. *LNCS 4166* (2006), 15.

[68] FRIGAULT, M., AND WANG, L. Measuring Network Security Using Bayesian Network-Based Attack Graphs. In *Proc. of the 32nd Annual IEEE International Computer Software and Applications (COMPSAC 2008)* (July 2008), pp. 698–703.

[69] FRIGAULT, M., WANG, L., SINGHAL, A., AND JAJODIA, S. Measuring Network Security Using Dynamic Bayesian Network. In *Proc. of the 4th ACM Workshop on Quality of Protection (QoP 2008)* (New York, NY, USA, 2008), QoP '08, ACM, pp. 23–30.

[70] GALLON, L., AND BASCOU, J. Using CVSS in Attack Graphs. In *Proc. of the 6th International Conference on Availability, Reliability and Security (ARES 2011)* (2011), pp. 59–66.

[71] GEORGIADIS, G., AND KIROUSIS, L. Lightweight centrality measures in networks under attack. *Complexus 3*, 1 (2006), 147–157.

[72] GERACE, T., AND CAVUSOGLU, H. The Critical Elements of the Patch Management Process. *Communnications of the ACM 52*, 8 (Aug. 2009), 117–121.

[73] GHOSH, A., GAJAR, P., AND RAI, S. Bring Your Own Device (BYOD): Security Risks and Mitigating Strategies. *Journal of Global Research in Computer Science 4*, 4 (2013), 62–70.

[74] GHOSH, N., AND GHOSH, S. An Approach for Security Assessment of Network Configurations Using Attack Graph. In *Proc. of the 1st International Conference on Networks and Communications, (NETCOM 2009)* (2009), pp. 283–288.

[75] GLYNIS, D., SALIM, H., YOUSSIF, A., AND GABRIEL, R. Resilient Dynamic Data Driven Application Systems (rDDDAS). *Procedia Computer Science 18*, 0 (2013), 1929 – 1938. 2013 International Conference on Computational Science.

[76] GORBENKO, A., KHARCHENKO, V., AND ROMANOVSKY, A. Using Inherent Service Redundancy and Diversity to Ensure Web Services Dependability. In *Methods, Models and Tools for Fault Tolerance*, M. Butler, C. Jones, A. Romanovsky, and E. Troubitsyna, Eds., vol. 5454 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 324–341.

[77] GRACE, M., ZHOU, Y., ZHANG, Q., ZOU, S., AND JIANG, X. RiskRanker: Scalable and Accurate Zero-day Android Malware Detection. In *Proc. of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys 2012)* (New York, NY, USA, 2012), ACM, pp. 281–294.

[78] GROCHOCKI, D., HUH, J., BERTHIER, R., BOBBA, R., SANDERS, W., CARDENAS, A., AND JETCHEVA, J. AMI Threats, Intrusion Detection Requirements and Deployment Recommendations. In *Proc. of the 3rd International Conference on Smart Grid Communications (SmartGridComm 2012)* (Nov 2012), pp. 395–400.

[79] GUPTA, S., AND WINSTEAD, J. Using Attack Graphs to Design Systems. *IEEE Security & Privacy 5*, 4 (2007), 80–83.

[80] GUTTMAN, B., AND ROBACK, E. *An Introduction to Computer Security: The NIST Handbook*. DIANE Publishing, 1995.

[81] HIGUERO, M., UNZILLA, J., JACOB, E., SAIZ, P., AGUADO, M., AND LUENGO, D. Application of 'attack trees' technique to copyright protection protocols using watermarking and definition of a new transactions protocol secdp (secure distribution protocol). *LNCS 3311* (2004), 264–275.

[82] HIGUERO, M., UNZILLA, J., JACOB, E., SAIZ, P., AGUADO, M., AND LUENGO, D. Application of'attack trees' in security analysis of digital contents e-commerce protocols with copyright protection. In *Proc. of International Carnahan Conference Security Technology (CCST 2005)* (2005), pp. 57–60.

[83] HINES, P., AND BLUMSACK, S. A centrality measure for electrical networks. In *Proc. of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)* (2008), pp. 185–185.

[84] HOMER, J., VARIKUTI, A., OU, X., AND MCQUEEN, M. Improving Attack Graph Visualization through Data Reduction and Attack Grouping. In *Visualization for Computer Security*, J. Goodall, G. Conti, and K. Ma, Eds., vol. 5210 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 68–79.

[85] HONG, J., CHUNG, J., HUANG, D., AND KIM, D. Survey on Graph-based and Tree-based Security Models. *to be submitted to the IEEE Journal of Surveys* (2014).

[86] HONG, J., EOM, T., PARK, J., AND KIM, D. Scalable Security Analysis using a Partition and Merge Approach in an Infrastructure as a Service Cloud. In *Proc. of the the 11th International Conference on Ubiquitous Intelligence & Computing (UIC 2014)* (Dec 2014), pp. 50–57.

[87] HONG, J., AND KIM, D. HARMs: Hierarchical Attack Representation Models for Network Security Analysis. In *Proc. of the 10th Australian In-*

formation Security Management Conference on SECAU Security Congress (SECAU 2012) (2012), pp. 1–8.

[88] HONG, J., AND KIM, D. Performance Analysis of Scalable Attack Representation Models. In *Security and Privacy Protection in Information Processing Systems (SEC 2013)*, L. Janczewski, H. Wolfe, and S. Shenoi, Eds., vol. 405 of *IFIP Advances in Information and Communication Technology*. Springer Berlin Heidelberg, 2013, pp. 330–343.

[89] HONG, J., AND KIM, D. Scalable Security Analysis in Hierarchical Attack Representation Model using Centrality Measures. In *Proc. of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSNW 2013)* (2013), pp. 1–8.

[90] HONG, J., AND KIM, D. Scalable Security Model Generation and Analysis Using k-importance Measures. In *Security and Privacy in Communication Networks (SecureComm 2013)*, T. Zia, A. Zomaya, V. Varadharajan, and M. Mao, Eds., vol. 127 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer International Publishing, 2013, pp. 270–287.

[91] HONG, J., AND KIM, D. Performance Analysis of Graph-based Security Models. *submitted to the Elsevier Computer & Security in Nov 2014* (2014).

[92] HONG, J., AND KIM, D. Scalable Security Models for Assessing Effectiveness of Moving Target Defenses. In *Proc. of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014)* (Jun 2014), pp. 515–526.

[93] HONG, J., KIM, D., AND HAQIQ, A. What Vulnerability Do We Need to Patch First? In *Proc. of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSNW 2014)* (2014), pp. 684–689.

[94] HONG, J., KIM, D., AND TAKAOKA, T. Scalable Attack Representation Model Using Logic Reduction Techniques. In *Proc. of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2013)* (2013), pp. 404–411.

[95] HUANG, Y., ARSENAULT, D., AND SOOD, A. Closing Cluster Attack Windows Through Server Redundancy and Rotations. In *Proc. of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2006)* (May 2006), vol. 2, pp. 12 pp.–21.

[96] HUANG, Y., AND GHOSH, A. Introducing Diversity and Uncertainty to Create Moving Attack Surfaces for Web Services. In *Moving Target Defense*, S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Eds., vol. 54 of *Advances in Information Security*. Springer New York, 2011, pp. 131–151.

[97] HYPPONEN, M. How We Found the File That Was Used to Hack RSA.

[98] INC, S. S. *Skybox*, June 2014.

[99] INGOLS, K., CHU, M., LIPPMANN, R., WEBSTER, S., AND BOYER, S. Modeling Modern Network Attacks and Countermeasures Using Attack Graphs. In *Proc. of the 25th Annual Computer Security Applications Conference (ACSAC 2009)* (2009), pp. 117–126.

[100] INGOLS, K., LIPPMANN, R., AND PIWOWARSKI, K. Practical Attack Graph Generation for Network Defense. In *Proc. of the 22nd Annual Computer Security Applications Conference (ACSAC 2006)* (2006), pp. 121 – 130.

[101] INTERNATIONAL TELECOMMUNICATION UNION, T. S. S. Security architecture for systems providing end-to-end communications. Tech. rep., ITU-T Rec. X.805, Oct. 2003.

[102] ISLAM, T., AND WANG, L. A Heuristic Approach to Minimum-Cost Network Hardening Using Attack Graph. In *Proc. of New Technologies, Mobility and Security (NTMS 2008)* (2008), pp. 1–5.

[103] JACKSON, T., SALAMAT, B., HOMESCU, A., MANIVANNAN, K., WAGNER, G., GAL, A., BRUNTHALER, S., WIMMER, C., AND FRANZ, M. Compiler-Generated Software Diversity. In *Moving Target Defense*, S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Eds., vol. 54 of *Advances in Information Security*. Springer New York, 2011, pp. 77–98.

[104] JAFARIAN, J., AL-SHAER, E., AND DUAN, Q. Openflow Random Host Mutation: Transparent Moving Target Defense Using Software Defined Networking. In *Proc. of the 1st Workshop on Hot Topics in Software Defined Networks (HotSDN 2012)* (New York, NY, USA, 2012), ACM, pp. 127–132.

[105] JAJODIA, S., NOEL, S., AND OBERRY, B. Topological Analysis of Network Attack Vulnerability. In *Managing Cyber Threats*, V. Kumar, J. Srivastava, and A. Lazarevic, Eds., vol. 5 of *Massive Computing*. Springer US, 2005, pp. 247–266.

[106] JAQUITH, A. Security metrics blog. http://www.securitymetrics.org.

[107] JHA, S., SHEYNER, O., AND WING, J. Minimization and Reliability Analyses of Attack Graphs. Tech. Rep. CMU-CS-02-109, School of Computer Science, Carnegie Mellon University, Feb. 2002.

[108] JHA, S., SHEYNER, O., AND WING, J. Two Formal Analyses of Attack Graphs. In *Proc. of the 15th IEEE Computer Security Foundations Workshop (CSFW 2002)* (2002), pp. 49 – 63.

[109] JIA, Q., SUN, K., AND STAVROU, A. MOTAG: Moving Target Defense against Internet Denial of Service Attacks. In *Proc. of the 22nd International Conference on Computer Communications and Networks (ICCCN 2013)* (2013), pp. 1–9.

[110] JIA, Q., WANG, H., FLECK, D., LI, F., STAVROU, A., AND POWELL, W. Catch Me if You Can: A Cloud-Enabled DDoS Defense. In *Proc. of the the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014)* (Jun 2014).

[111] JIN, C., WANG, X., AND TAN, H. Dynamic Attack Tree and Its Applications on Trojan Horse Detection. In *Proc. of the Second International Conference on Multimedia and Information Technology (MMIT 2010)* (2010), vol. 1, pp. 56–59.

[112] JÜRGENSON, A., AND WILLEMSON, J. Processing Multi-parameter Attacktrees with Estimated Parameter Values. In *Advances in Information and Computer Security*. Springer, 2007, pp. 308–319.

[113] KANG, U., TSOURAKAKIS, C., AND FALOUTSOS, C. Pegasus: A Peta-scale Graph Mining System Implementation and Observations. In *Proc. of 9th IEEE International Conference on Data Mining (ICDM 2009)* (2009), IEEE, pp. 229–238.

[114] KARGL, F., KLENK, A., SCHLOTT, S., AND WEBER, M. Advanced Detection of Selfish or Malicious Nodes in Ad Hoc Networks. In *Security in Ad-hoc and Sensor Networks*, C. Castelluccia, H. Hartenstein, C. Paar, and D. Westhoff, Eds., vol. 3313 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 152–165.

[115] KARYPIS, G., AND KUMAR, V. Parallel Multilevel k-way Partitioning Scheme for Irregular Graphs. In *Proc. of ACM/IEEE Conference on Supercomputing (ICS 1996)* (1996), vol. 42, pp. 278–300.

[116] KING, S. Science of cybersecurity, 2010. http://www.fas.org/irp/agency/dod/jason/cyber.pdf.

[117] KIRRMANN, H., AND DZUNG, D. Selecting a Standard Redundancy Method for Highly Available Industrial Networks. In *Proc. of the 3rd IEEE International Workshop on Factory Communication Systems (WFCS 2006)* (2006), pp. 386–390.

[118] KORDY, B., MAUW, S., RADOMIROVIĆ, S., AND SCHWEITZER, P. Foundations of Attack–Defense Trees. In *Formal Aspects of Security and Trust*, P. Degano, S. Etalle, and J. Guttman, Eds., vol. 6561 of *Lecture Notes in Computer Science*. Springer, 2011, pp. 80–95.

[119] KORDY, B., MAUW, S., RADOMIROVIC, S., AND SCHWEITZER, P. AttackDefense Trees. *Journal of Logic and Computation* (2012).

[120] KORDY, B., PIETRE-CAMBACEDES, L., AND SCHWEITZER, P. DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. *CoRR abs/1303.7397* (2013).

[121] LANTZ, B., HELLER, B., AND MCKEOWN, N. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *Proc. of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (New York, NY, USA, 2010), Hotnets-IX, ACM, pp. 19:1–19:6.

[122] LEMAY, E., FORD, M., KEEFE, K., SANDERS, W., AND MUEHRCKE, C. Model-based Security Metrics Using ADversary VIew Security Evaluation (ADVISE). In *Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on* (Sept 2011), pp. 191–200.

[123] LEMAY, E., UNKENHOLZ, W., PARKS, D., MUEHRCKE, C., KEEFE, K., AND SANDERS, W. H. Adversary-driven State-based System Security Evaluation. In *Proc. of the 6th International Workshop on Security Measurements and Metrics (MetriSec 2010)* (New York, NY, USA, 2010), MetriSec '10, ACM, pp. 5:1–5:9.

[124] LIPPMANN, R., AND INGOLS, K. An Annotated Review of Past Papers on Attack Graphs. *ESC-TR-2005-054* (2005).

[125] LIPPMANN, R., INGOLS, K., SCOTT, C., PIWOWARSKI, K., KRATKIEWICZ, K., ARTZ, M., AND CUNNINGHAM, R. Evaluating and Strengthening Enterprise Network Security Using Attack Graphs. Tech. rep., MIT Lincoln Lab, Oct. 2005.

[126] LIPPMANN, R., INGOLS, K., SCOTT, C., PIWOWARSKI, K., KRATKIEWICZ, K., ARTZ, M., AND CUNNINGHAM, R. Validating and Restoring Defense in Depth Using Attack Graphs. In *Proc. of IEEE Military Communications Conference (MILCOM 2006)* (2006), pp. 1–10.

[127] LIU, Y., AND MAN, H. Network vulnerability assessment using Bayesian networks. In *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2005* (Mar. 2005), B. V. Dasarathy, Ed., vol. 5812 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 61–71.

[128] LOUTHAN, G. *Hybrid Attack Graphs for Modeling Cyber-physical Systems*. MS thesis, University of Tulsa, 2011.

[129] LOUTHAN, G., HARDWICKE, P., HAWRYLAK, P., AND HALE, J. Toward Hybrid Attack Dependency Graphs. In *Proc. of the 7th Annual Workshop on Cyber Security and Information Intelligence Research (CSIIRW 2011)* (New York, NY, USA, 2011), CSIIRW '11, ACM.

[130] LTD., A. T. *SecurITree*, January 2013.

[131] LTD., A. T. *RedSeal Networks*, June 2014.

[132] LTD, I. *AttackTree+*, August 2011.

[133] MANADHATA, P. Game Theoretic Approaches to Attack Surface Shifting. In *Moving Target Defense II*, S. Jajodia, A. K. Ghosh, V. Subrahmanian, V. Swarup, C. Wang, and X. S. Wang, Eds., vol. 100 of *Advances in Information Security*. Springer New York, 2013, pp. 1–13.

[134] MANADHATA, P., AND WING, J. An Attack Surface Metric. *IEEE Transactions on Software Engineering 37*, 3 (2011), 371–386.

[135] MAUW, S., AND OOSTDIJK, M. Foundations of Attack Trees. In *Information Security and Cryptology - ICISC 2005*, D. Won and S. Kim, Eds., vol. 3935 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 186–198.

[136] MCQUEEN, M., BOYER, W., FLYNN, M., AND BEITEL, G. Quantitative Cyber Risk Reduction Estimation Methodology for a Small SCADA Control System. In *Proc. of the 39th Annual Hawaii International Conference on System Science (HICSS 2006)* (Jan 2006), vol. 9, pp. 226–226.

[137] MCQUEEN, M., MCQUEEN, T., BOYER, W., AND CHAFFIN, M. Empirical Estimates and Observations of 0Day Vulnerabilities. In *Proc. of the 42nd Hawaii International Conference on System Sciences (HICSS 2009)* (Jan 2009), pp. 1–12.

[138] MEHTA, V., BARTZIS, C., ZHU, H., CLARKE, E., AND WING, J. Ranking Attack Graphs. In *Proc. of the 9th international conference on Recent Advances in Intrusion Detection (RAID 2006)* (Berlin, Heidelberg, 2006), Springer-Verlag, pp. 127–144.

[139] MELL, P., BERGERON, T., AND HENNING, D. Creating a Patch and Vulnerability Management Program. *NIST Special Publication 800* (2005), 40.

[140] MELL, P., AND GRANCE, T. SP 800-145. The NIST Definition of Cloud Computing. Tech. rep., NIST, Gaithersburg, MD, United States, 2011.

[141] MESHKAT, L., XING, L., DONOHUE, S., AND OU, Y. An Overview of the Phase-modular Fault Tree Approach to Phased Mission System Analysis. *Space* (2003).

[142] MILLER, K., VOAS, J., AND HURLBURT, G. BYOD: Security and Privacy Considerations. *IT Professional 14*, 5 (Sept 2012), 53–55.

[143] MIRKOVIC, J., BENZEL, T., FABER, T., BRADEN, R., WROCLAWSKI, J., AND SCHWAB, S. The DETER project: Advancing the science of cyber security experimentation and test. In *Proc. of IEEE International Conference on Technologies for Homeland Security (HST 2010)* (2010), pp. 1 –7.

[144] MODELO-HOWARD, G., BAGCHI, S., AND LEBANON, G. Determining Placement of Intrusion Detectors for a Distributed Application through

Bayesian Network Modeling. In *Recent Advances in Intrusion Detection*, R. Lippmann, E. Kirda, and A. Trachtenberg, Eds., no. 5230 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2008, pp. 271–290.

[145] MOORE, A., ELLISON, R., AND LINGER, R. Attack Modeling for Information Security and Survivability. *CMU/SEI-2001-TN-001* (2001).

[146] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. National Vulnerability Database.

[147] NEWELL, A., OBENSHAIN, D., TANTILLO, T., NITA-ROTARU, C., AND AMIR, Y. Increasing Network Resiliency by Optimally Assigning Diverse Variants to Routing Nodes. In *Proc. of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2013)* (2013).

[148] NGUYEN-TUONG, A., EVANS, D., KNIGHT, J., COX, B., AND DAVIDSON, J. Security Through Redundant Data Diversity. In *Proc. of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2008)* (2008), pp. 187–196.

[149] NING, P., CUI, Y., AND REEVES, D. Constructing Attack Scenarios Through Correlation of Intrusion Alerts. In *Proc. of the 9th ACM conference on Computer and Communications Security (CCS 2002)* (New York, NY, USA, 2002), CCS '02, ACM, pp. 245–254.

[150] NING, P., AND XU, X. Learning Attack Strategies from Intrusion Alerts. In *Proc. of the 10th ACM conference on Computer and Communications Security (CCS 2003)* (New York, NY, USA, 2003), CCS '03, ACM, pp. 200–209.

[151] NOEL, S., AND JAJODIA, S. Managing Attack Graph Complexity through Visual Hierarchical Aggregation. In *Proc. of the 2004 ACM workshop on Visualization and data mining for computer security (VizSec 2004)* (New York, NY, USA, 2004), ACM, pp. 109–118.

[152] NOEL, S., AND JAJODIA, S. Understanding Complex Network Attack Graphs through Clustered Adjacency Matrices. In *Proc. of the 21st Annual Computer Security Applications Conference (ACSAC 2005)* (2005), pp. 10–169.

[153] NOEL, S., JAJODIA, S., O'BERRY, B., AND JACOBS, M. Efficient minimum-cost network hardening via exploit dependency graphs. In *Proc. of the 19th Annual Computer Security Applications Conference (ACSAC 2003)* (2003), pp. 86 – 95.

[154] NOEL, S., JAJODIA, S., WANG, L., AND SINGHAL, A. Measuring Security Risk of Networks using Attack Graphs. *International Journal of Next-Generation Computing 1*, 1 (2010), 135–147.

[155] OKHRAVI, H., COMELLA, A., ROBINSON, E., YANNALFO, S., MICHALEAS, P., AND HAINES, J. Creating a Cyber Moving Target for Critical Infrastructure Applications. In *Critical Infrastructure Protection V*, J. Butts and S. Shenoi, Eds., vol. 367 of *IFIP Advances in Information and Communication Technology*. Springer Berlin Heidelberg, 2011, pp. 107–123.

[156] ONGSAKORN, P., TURNEY, K., THORNTON, M., NAIR, S., SZYGENDA, S., AND MANIKAS, T. Cyber Threat Trees for Large System Threat Cataloging and Analysis. In *Proc. of 4th Annual IEEE Systems Conference 2010* (2010), IEEE, pp. 610–615.

[157] ORTALO, R., DESWARTE, Y., AND KAANICHE, M. Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security. *IEEE Transactions on Software Engineering 25*, 5 (1999), 633–650.

[158] OU, X., BOYER, W., AND MCQUEEN, M. A scalable Approach to Attack Graph Generation. In *Proc. of the 13th ACM conference on Computer and communications security (CCS 2006)* (2006), ACM, pp. 336–345.

[159] OU, X., AND GOVINDAVAJHALA, S. MulVAL: A Logic-based Network Security Analyzer. In *Proc. of the 14th USENIX Security Symposium (USENIX Security 2005)* (2005), pp. 113–128.

[160] OU, X., AND SINGHAL, A. Security Risk Analysis of Enterprise Networks Using Attack Graphs. In *Quantitative Security Risk Assessment of Enterprise Networks*, SpringerBriefs in Computer Science. Springer New York, 2011, pp. 13–23.

[161] PATEL, S., GRAHAM, J., AND RALSTON, P. Quantitatively Assessing the Vulnerability of Critical Information Systems: A New Method for Evaluating Security Enhancements. *International Journal of Information Management 28*, 6 (2008), 483–491.

[162] PATSOS, D., MITROPOULOS, S., AND DOULIGERIS, C. Expanding Topological Vulnerability Analysis to Intrusion Detection through the Incident Response Intelligence System. *Information Management & Computer Security 18*, 4 (2010), 291–309.

[163] PAULOS, A., PAL, P., SCHANTZ, R., AND BENYO, B. Moving Target Defense (MTD) in an Adaptive Execution Environment. In *Proc. of the 8th Annual Cyber Security and Information Intelligence Research Workshop (CSIIRW 2013)* (New York, NY, USA, 2013), ACM, pp. 62:1–62:4.

[164] PAWLIKOWSKI, K., JEONG, H., AND LEE, J. On credibility of simulation studies of telecommunication networks. *Communications Magazine, IEEE 40*, 1 (2002), 132 –139.

[165] PHILLIPS, C., AND SWILER, L. A Graph-based System for Network-vulnerability Analysis. In *Proc. of the 1998 Workshop on New Security Paradigms (NSPW 1998)* (New York, NY, USA, 1998), ACM, pp. 71–79.

[166] POOLSAPASSIT, N., AND RAY, I. Investigating Computer Attacks Using Attack Trees. In *Advances in Digital Forensics III*, P. Craiger and S. Shenoi, Eds., vol. 242 of *IFIP The International Federation for Information Processing*. Springer New York, 2007, pp. 331–343.

[167] POOLSAPPASIT, N., DEWRI, R., AND RAY, I. Dynamic Security Risk Management Using Bayesian Attack Graphs. *IEEE Transactions on Dependable and Secure Computing 9*, 1 (2012), 61–74.

[168] PORTOKALIDIS, G., SLOWINSKA, A., AND BOS, H. Argos: An Emulator for Fingerprinting Zero-day Attacks for Advertised Honeypots with Automatic Signature Generation. In *Proc. of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys 2006)* (New York, NY, USA, 2006), ACM, pp. 15–27.

[169] RAY, I., AND POOLSAPASSIT, N. Using Attack Trees to Identify Malicious Attacks from Authorized Insiders. In *Proc. of the 10th European Symposium on Research in Computer Security (ESORICS 2005)*, S. Vimercati, P. Syverson, and D. Gollmann, Eds., vol. 3679 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 231–246.

[170] ROHRER, J., JABBAR, A., AND STERBENZ, J. Path Diversification for Future Internet End-to-End Resilience and Survivability. *Telecommunication Systems* (2013), 1–19.

[171] ROY, A., KIM, D., AND TRIVEDI, K. Cyber security analysis using Attack Countermeasure Trees. In *Proc. of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research (CSIIRW 2010)* (New York, NY, USA, 2010), CSIIRW '10, ACM, pp. 28:1–28:4.

[172] ROY, A., KIM, D., AND TRIVEDI, K. Attack Countermeasure Trees (ACT): Towards Unifying the Constructs of Attack and Defense Trees. *Security and Communication Networks 5*, 8 (2012), 929–943.

[173] ROY, A., KIM, D., AND TRIVEDI, K. Scalable Optimal Countermeasure Selection using Implicit Enumeration on Attack Countermeasure Trees. In *Proc. of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)* (Los Alamitos, CA, USA, 2012), IEEE Computer Society, pp. 1–12.

[174] SAHNER, R. A., TRIVEDI, K., AND PULIAFITO, A. *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package*. Springer Publishing Company, Incorporated, 2012.

[175] SAINI, V., DUAN, Q., AND PARUCHURI, V. Threat Modeling using Attack Trees. *Journal of Computer Science in Colleges 23*, 4 (2008), 124–131.

[176] SAWILLA, R., AND OU, X. Identifying Critical Attack Assets in Dependency Attack Graphs. In *Proc. of the 13th European Symposium on Research in Computer Security (ESORICS 2008)* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 18–34.

[177] SAWILLA, R., AND SKILLICORN, D. Partial Cuts in Attack Graphs for Cost Effective Network Defence. In *Proc. of IEEE Conference on Technologies for Homeland Security (HST 2012)* (2012), pp. 291–297.

[178] SCHABERREITER, T., WIESER, C., SANCHEZ, I., RIEKKI, J., AND RONING, J. An Enumeration of RFID Related Threats. In *Proc. of the 2nd International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2008)* (2008), pp. 381–389.

[179] SCHNEIER, B. Modeling security threats. *Dr. Dobb's journal 24*, 12 (1999).

[180] SCHNEIER, B. *Secrets and Lies: Digital Security in a Networked World*. John Wiley and Sons Inc., 2000.

[181] SHARMA, A., KALBARCZYK, Z., BARLOW, J., AND IYER, R. Analysis of security data from a large computing organization. In *Proc. of the 41st Annual IEEE/IFIP International Conference on Dependable Systems Networks (DSN 2011)* (2011), pp. 506 –517.

[182] SHEYNER, O., HAINES, J., JHA, S., LIPPMANN, R., AND WING, J. Automated Generation and Analysis of Attack Graphs. Tech. rep., CMU, 2002.

[183] SHEYNER, O., AND WING, J. Tools for Generating and Analyzing Attack Graphs. In *Formal Methods for Components and Objects*, M. M. Bonsangue, S. Graf, and W.-P. d. Roever, Eds., no. 3188 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2004, pp. 344–371.

[184] SINTEF. *SeaMonster Security Modelling Software*, August 2010.

[185] STEFFAN, J., AND SCHUMACHER, M. Collaborative Attack Modeling. In *Proc. of the 2002 ACM Symposium on Applied Computing* (2002), ACM, pp. 253–259.

[186] SWILER, L., PHILLIPS, C., ELLIS, D., AND CHAKERIAN, S. Computer-attack Graph Generation Tool. In *Proc. of DARPA Information Survivability Conference Exposition II, (DISCEX 2001)* (2001), vol. 2, pp. 307–321 vol.2.

[187] TARJAN, R. Depth-first search and linear graph algorithms. *SIAM journal on computing 1*, 2 (1972), 146–160.

[188] TEN, C., LIU, C., AND GOVINDARASU, M. Vulnerability Assessment of Cybersecurity for SCADA Systems Using Attack Trees. In *Proc. of IEEE Power Engineering Society General Meeting* (June 2007), pp. 1–8.

[189] TIPPENHAUER, N., TEMPLE, W., HOA VU, A., CHEN, B., NICOL, D., KALBARCZYK, Z., AND SANDERS, W. Automatic Generation of Security Argument Graphs. *ArXiv e-prints* (May 2014).

[190] VIDALIS, S., AND JONES, A. Using Vulnerability Trees for Decision Making in Threat Assessment. In *Proc. of the 2nd European Conference on Information Warfare and Security (ECIW 2003)* (2003), Academic Conferences Limited, pp. 329–342.

[191] VIGO, R., NIELSON, F., AND NIELSON, H. Automated Generation of Attack Trees. In *Proc. of the 27th IEEE Computer Security Foundations Symposium (CSF 2014)* (July 2014), pp. 337–350.

[192] VIKRAM, S., YANG, C., AND GU, G. NOMAD: Towards Non-Intrusive Moving-Target Defense against Web Bots. In *Proc. of the 1st IEEE Conference on Communications and Network Security (CNS 2013)* (2013), pp. 1–9.

[193] WALKER, M. NIST: Patch management has inherent challenges.

[194] WANG, K., CRETU, G., AND STOLFO, S. Anomalous Payload-Based Worm Detection and Signature Generation. In *Recent Advances in Intrusion Detection*, A. Valdes and D. Zamboni, Eds., vol. 3858 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 227–246.

[195] WANG, K., PAREKH, J., AND STOLFO, S. Anagram: A Content Anomaly Detector Resistant to Mimicry Attack. In *Recent Advances in Intrusion Detection*, D. Zamboni and C. Kruegel, Eds., vol. 4219 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 226–248.

[196] WANG, L., ISLAM, T., LONG, T., SINGHAL, A., AND JAJODIA, S. An Attack Graph-Based Probabilistic Security Metric. In *Data and Applications Security XXII*, V. Atluri, Ed., vol. 5094 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, pp. 283–296.

[197] WANG, L., AND JAJODIA, S. An Approach to Preventing, Correlating, and Predic. In *Intrusion Detection Systems*, no. 38 in Advances in Information Security. Springer US, Jan. 2008, pp. 93–128.

[198] WANG, L., JAJODIA, S., SINGHAL, A., AND NOEL, S. k-Zero Day Safety: Measuring the Security Risk of Networks against Unknown Attacks. In *Proc. of the 15th European Symposium on Research in Computer Security (ESORICS 2010)*, D. Gritzalis, B. Preneel, and M. Theoharidou, Eds., vol. 6345 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 573–587.

[199] WANG, L., JAJODIA, S., SINGHAL, A., P., C., AND NOEL, S. k-Zero Day Safety: A Network Security Metric for Measuring the Risk of Unknown

Vulnerabilities. *IEEE Transactions on Dependable and Secure Computing 11*, 1 (Jan 2014), 30–44.

[200] WANG, L., NOEL, S., AND JAJODIA, S. Minimum-cost Network Hardening using Attack Graphs. *Computer Communications 29*, 18 (2006), 3812 – 3824.

[201] WANG, L., SINGHAL, A., AND JAJODIA, S. Measuring the Overall Security of Network Configurations Using Attack Graphs. In *Data and Applications Security XXI*, S. Barker and G.-J. Ahn, Eds., no. 4602 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2007, pp. 98–112.

[202] WANG, L., SINGHAL, A., AND JAJODIA, S. Toward Measuring Network Security using Attack Graphs. In *Proc. of the 2007 ACM workshop on Quality of protection (QoP 2007)* (New York, NY, USA, 2007), QoP '07, ACM, pp. 49–54.

[203] WEISS, J. A System Security Engineering Process. In *Proc. of the 14th National Computer Security Conference* (1991), vol. 249.

[204] WILLIAMS, D., HU, W., DAVIDSON, J., HISER, J., KNIGHT, J., AND NGUYEN-TUONG, A. Security Through Diversity: Leveraging Virtual Machine Technology. *IEEE Security & Privacy 7*, 1 (Jan. 2009), 26–33.

[205] WILLIAMS, L., LIPPMANN, R., AND INGOLS, K. GARNET: A Graphical Attack Graph and Reachability Network Evaluation Tool. In *Proc. of the 5th International Workshop on Visualization for Computer Security (VizSec 2008)* (2008), pp. 44–59.

[206] WU, Y., FOO, B., MAO, Y., BAGCHI, S., AND SPAFFORD, E. Automated Adaptive Intrusion Containment in Systems of Interacting Services. *Computer Networks 51*, 5 (Apr. 2007), 1334–1360.

[207] XIE, A., CAI, Z., TANG, C., HU, J., AND CHEN, Z. Evaluating Network Security With Two-Layer Attack Graphs. In *Proc. of the 25th Annual*

Computer Security Applications Conference (ACSAC 2009) (2009), pp. 127 –136.

[208] XIE, P., LI, J., OU, X., LIU, P., AND LEVY, R. Using Bayesian Networks for Cyber Security Analysis. In *Proc. of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010)* (June 2010), pp. 211–220.

[209] YACKOSKI, J., LI, J., DELOACH, S., AND OU, X. Mission-oriented Moving Target Defense Based on Cryptographically Strong Network Dynamics. In *Proc. of the 8th Annual Cyber Security and Information Intelligence Research Workshop (CSIIRW 2013)* (New York, NY, USA, 2013), ACM, pp. 57:1–57:4.

[210] YAGER, R. OWA Trees and Their Role in Security Modeling using Attack Trees. *Information Sciences 176*, 20 (2006), 2933–2959.

[211] YUAN, E., MALEK, S., SCHMERL, B., GARLAN, D., AND GENNARI, J. Architecture-Based Self-Protecting Software Systems. In *Proc. of the 9th International ACM Sigsoft Conference on the Quality of Software Architectures (QoSA 2013)* (2013), pp. 33–42.

[212] ZHANG, Y., LI, M., BAI, K., YU, M., AND ZANG, W. Incentive Compatible Moving Target Defense against VM-Colocation Attacks in Clouds. In *Information Security and Privacy Research*, D. Gritzalis, S. Furnell, and M. Theoharidou, Eds., vol. 376 of *IFIP Advances in Information and Communication Technology*. Springer Berlin Heidelberg, 2012, pp. 388–399.

[213] ZHANG, Z., AND WANG, S. Boosting Logical Attack Graph for Efficient Security Control. In *Proc. of the 7th International Conference on Availability, Reliability and Security (ARES 2012)* (Aug 2012), pp. 218–223.

[214] ZHANG, Z., WANG, S., AND KADOBAYASHI, Y. Exploring attack graph for cost-benefit security hardening: A probabilistic approach. *Computers & Security* (2012).

[215] ZHU, Y., HU, H., AHN, G., HUANG, D., AND WANG, S. Towards temporal access control in cloud computing. In *Proc. of Annual IEEE International Conference on Computer Communications (INFOCOM 2012)* (2012), pp. 2576–2580.

[216] ZHUANG, R., ZHANG, S., BARDAS, A., DELOACH, S., OU, X., AND SINGHAL, A. Investigating the Application of Moving Target Defenses to Network Security. In *Proc. of the 6th International Symposium on Resilient Control Systems (ISRCS 2013)* (2013), pp. 162–169.

[217] ZHUANG, R., ZHANG, S., DELOACH, S., OU, X., AND SINGHAL, A. Simulation-based Approaches to Studying Effectiveness of Moving-Target Network Defense. In *Proc. of National Symposium on Moving Target Research* (2012).

[218] ZONOUZ, S., KHURANA, H., SANDERS, W., AND YARDLEY, T. RRE: A Game-theoretic Intrusion Response and Recovery Engine. In *Proc. of the 39th IEEE/IFIP International Conference on Dependable Systems Networks (DSN 2009)* (2009), pp. 439–448.

Appendices

Publications removed due to copyright restrictions

Hong, J. and Kim, D.; “Security Assessment and Mitigation of Unknown Attacks” submitted to the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2015), Dec 2014 (Appendix A)

Hong, J. and Kim, D.; “Performance Analysis of Graph-based Security Models” under revision for the Elsevier Special Issue of the Computers and Security Journal, Oct 2014 (Appendix B).

Hong, J. and Kim, D.; “Assessing the Effectiveness of Moving Target Defense using Security Models” under revision for the IEEE Special Issue on Transactions of Dependable and Secure Computing (TDSC) Journal, Oct 2014 (Appendix C).

Hong, J. and Chung, C. and Huang, D. and Kim, D.; “Survey of Graph-based and Tree-based Security Models” to be submitted to the IEEE Communications Surveys & Tutorials, Oct 2014 (Appendix D)