

論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	A Formal Approach for Secured Risk Analysis towards Disciplined Alignment
著者(和文)	銭谷謙吾
Author(English)	Kengo Zenitani
出典(和文)	学位:博士(理学), 学位授与機関:東京工業大学, 報告番号:甲第12331号, 授与年月日:2023年3月26日, 学位の種別:課程博士, 審査員:田中 圭介,伊東 利哉,尾形 わかは,松浦 知史,安永 憲司,脇田 建
Citation(English)	Degree:Doctor (Science), Conferring organization: Tokyo Institute of Technology, Report number:甲第12331号, Conferred date:2023/3/26, Degree Type:Course doctor, Examiner:,,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Doctoral Thesis

A Formal Approach for Secured Risk Analysis towards Disciplined Alignment

Kengo Zenitani

Graduate Major in Mathematical and Computing Science
School of Computing
Tokyo Institute of Technology

Supervisor: Keisuke Tanaka

January, 2023

Preface

This dissertation’s implicit but central question is how we can ensure our managerial practices as reproducible rational activities. It is not uncommon to see that any of us unconsciously believes we are acting rationally, but the whole picture of a project composed of those “rational” people is disrupting. Information security management is such a managerial practice. There is no controversy about the importance of information security these days. However, business people tend to avoid security controls while security officials try to enforce them. If we know and understand the appropriate form of information security in each enterprise, there should be no such conflicts. Unfortunately, the reality is not. Even though we have plenty of guidance that recommends assessing risks and communicating the risks and agreed security controls, there is a stockpile of grave security incidents. Why?

Prior to this study, I spent five years on the research of information security management. I had been a researcher in a team studying the reality and desirable approaches in information security management. My colleagues and I found there were two critical factors in information security management practices. Firstly, the lack of understanding and support in information security management from business executives can easily nullify information security. Secondly, there is no objective risk assessment method that can tell those executives the nature of risks.

This study is an attempt to provide a mathematical framework for secured risk analysis that is objective and robust against intentional risk underestimation provoked under pressure from uncooperative business executives. It is also an answer to the second annoying factor mentioned above.

[Chapter 1](#) elaborates on the need for secured risk analysis by examining the actual large-scale incidents. [Chapter 2](#) explains the detail of attack graph analysis, a model-based security risk analysis method. The method mechanically generates a mathematical graph called an attack graph, which depicts the set of sequences of logically possible attacks. Our approach in this study is based on attack graph analysis and overcomes limitations in past studies. [Chapter 3](#) introduces a network security metric applicable to cyclic attack graphs. [Chapter 4](#) discusses a multi-objective optimization algorithm for security control selection under a given attack graph. [Chapter 5](#) gives yet another interpretation of an attack graph. It helps us to extend the use of attack graph analysis. Accordingly, [Chapter 6](#) proposes a formal approach for secured risk analysis where less investment in risk assessment always gives a larger set of incidents; thus, it urges business executives to consider the appropriate balance in the information security investment.

This dissertation is a revised and updated compilation of my works [184, 186, 187, 182, 183, 185] reproduced with permission from each publisher.

Acknowledgements

I take this opportunity to thank Dr. Keisuke Tanaka and Dr. Junichi Iijima for their support, advice, guidance, and encouragement. Before my admission to the doctoral course, Dr. Junichi Iijima gave me many valuable insights. Then, as my doctoral supervisor, Dr. Keisuke Tanaka spent his precious time and effort on the course of this dissertation. I owe every result in this dissertation to both professors.

I gratefully acknowledge the time and commitment of the other members of my committee, Dr. Ken Wakita, Dr. Kenji Yasunaga, Dr. Satoshi Matsuura, Dr. Toshiya Itoh, and Dr. Wakaha Ogata. I also want to thank the Department of Mathematical and Computing Science for supporting me throughout my graduate studies.

I also would like to express my gratitude to Dr. Shuichi Sakai, a professor at the Graduate School of Information Science and Technology at the University of Tokyo. As previously noted in the preface, my research is a sequel of the past study, and the study was a project under him. His direction and support during the project and his ongoing encouragement and patience helped me in many ways.

Additionally, I want to express my personal but deep appreciation to the three people below.

First, Hirokazu Okumura is the person who gave me the opportunity to work as a member of the project under Dr. Shuichi Sakai. I worked as a researcher with him on the team. He is the person who introduced me to Dr. Junichi Iijima, too. Even more, he asked me to conduct several surveys years before, and those jobs afforded the economic opportunity that enabled me to return to the university. The rich experience in those surveys is another foundation for this dissertation. Without those days with him, nothing stands as it is now. I owe a quarter of my life to him.

Second, Dr. Hisaya Fujioka was an associate professor at Kyoto University during my time as a master's course student. He was not my supervisor. However, when I attended a conference in Okinawa with him in March 2011, we exchanged words in a short conversation, which decided my life after that. At that time, I told him that I had been wondering if it was morally correct, in the time of the catastrophic Tohoku disaster, to study as a student even when I had practical skills to contribute to society. Because I was a skilled programmer but temporarily quit my job to return to the university. He answered that there was always something people could do at their place, which also applied to life as a student. That word and several days spent in Fukushima in the summer of the year determined me to keep studying and going forward. Unfortunately, I cannot show my appreciation to him. He had been gone in the fall of the year all of a sudden. He was a graduate of Tokyo Tech, too.

Third, Dr. Yahiko Kambayashi was a professor at Kyoto University during my time as an undergraduate student. He was not my supervisor but a project manager who recognized my potential as a promising young programmer for the Japanese government's Mitou project. After that project,

I encountered economic difficulties that were almost something that should be called trials and tribulations. I was in the dark, and the struggles were helpless then. When I decided to leave the university for at least several years, he called on me and gave his time to listen to me so carefully. After an hour of pessimistic explanation of my reality, he put his hands on mine and told me strongly that I must return to the university someday. It took about seven years since then. When I finally returned to the university, it was four years after his passing. I missed the opportunity forever to express my heartfelt gratitude to him.

It is now almost ten years since the inception of the information security management study. It has been a long journey and will be far longer towards the frontier yet unknown to me. It is a fabric of support and encouragement from every people I met and worked with. I am deeply grateful to them, again and again.

Last and most, I wish to thank my wife and my one-year-old little boy. The time I spent on this research otherwise should have been shared with them. Their patience and encouragement were the essential foundation. I dedicate this dissertation to them. And I hope this research can be a piece of contribution to the world better than today, where I, my family, and everyone live.

Contents

1	Motivation	1
1.1	Introduction	1
1.2	Issues in information security management framework	2
1.2.1	Information security management framework	3
1.2.2	Insecurity in risk analysis	3
1.2.3	Distance from business risk analysis	5
1.2.4	Alienation from corporate governance	6
1.3	Gaps between researches and practices	8
1.3.1	Lack of tools integration	8
1.3.2	Lack of installation assistance	9
1.3.3	Lack of performance validation	9
1.4	A proposal of a research program	9
1.4.1	Make employable integrations	10
1.4.2	Clarify the verification and validation criteria	10
1.4.3	Towards a disciplined alignment	11
1.5	Conclusion	12
2	Attack graph analysis	13
2.1	Introduction	13
2.2	Foundation	14
2.2.1	Conceptual framework	14
2.2.2	Attack graph representation	15
2.2.3	Attack template representation	17
2.2.4	Attack graph generation	19
2.2.5	Attack paths extraction	20
2.2.6	Discussions	20
2.2.6.1	Computational complexity	22
2.2.6.2	Cyclic structures	22
2.2.6.3	Attack template preparation	23
2.2.6.4	Reliability of inference	23
2.2.6.5	Attack expressiveness	23
2.2.6.6	Other forms of attack graphs	24
2.3	Visualization	25
2.3.1	Information reduction	25

2.3.2	Aspect oriented view	26
2.3.3	Visual syntax	27
2.4	Network security metrics	27
2.4.1	Non-probabilistic metrics for cyclic graphs	28
2.4.2	Non-probabilistic metrics for acyclic graphs	30
2.4.2.1	Combination of major metrics and assistive metrics	30
2.4.2.2	Composition of metrics for acyclic subgraphs	31
2.4.3	Probabilistic metrics for cyclic graphs	32
2.4.3.1	State rank for state-enumeration graph	32
2.4.3.2	Asset rank for exploit-dependency graph	33
2.4.4	Probabilistic metrics for acyclic graphs	34
2.4.5	Discussions	35
2.4.5.1	Computational complexity	35
2.4.5.2	Cycle handling	35
2.4.5.3	Reliability of metrics	36
2.4.5.4	Further readings	36
2.5	Network hardening	37
2.5.1	Minimal-set search	37
2.5.2	Least-cost-set search	39
2.5.3	Allowable-set search	41
2.5.4	Approximated-set search	41
2.5.5	Discussions	43
2.5.5.1	Computational complexity	43
2.5.5.2	Cycle handling	43
2.5.5.3	Multi-objective network hardening	43
2.5.5.4	Game theoretic approach	44
2.6	Bayesian attack graph	44
2.6.1	Brief introduction to Bayesian network	44
2.6.2	Bayesian attack graph	45
2.6.3	Advanced exploit-dependency modeling	46
2.6.4	Loopy belief propagation and Noisy OR/AND	47
2.6.5	Discussions	48
2.6.5.1	CPT and Canonical models	48
2.6.5.2	Reliability of inference	48
2.6.5.3	Network hardening	48
2.6.5.4	Uncertain graph	49
2.7	Conclusion	49
3	Network security metric	50
3.1	Introduction	50
3.2	Related work	51
3.3	Problem Settings	52
3.3.1	Attack Graph	52
3.3.2	Network reachability analysis	53

3.4	Approach	53
3.4.1	Bayesian attack graph	53
3.4.2	Reachability graph	55
3.4.3	The interpretation of reachabilities	59
3.5	Experiment	62
3.5.1	Performance comparison	62
3.5.2	Applicability to large-scale, cyclic networks	66
3.6	Discussions	68
3.6.1	Computational complexity	68
3.6.2	Variations in filter functions	70
3.6.3	Monotonicity of reachability-graph analysis	73
3.6.4	Other related works	73
3.6.5	Limitations	74
3.7	Conclusion	74
4	Network hardening	76
4.1	Introduction	76
4.2	Related work	78
4.2.1	Algorithmic frameworks for risk evaluation	78
4.2.2	Minimum-cost set cover problem	79
4.2.3	Satisfiability modulo theory	79
4.2.4	Markov decision process	80
4.2.5	Integer linear programming	80
4.2.6	Classical planning	81
4.2.7	Evolutionary computation	82
4.2.8	Towards the multi-objective network hardening in practice	82
4.3	Problem settings	83
4.3.1	Multi-objective network hardening	83
4.4	Approach	84
4.4.1	Monotonic gradient descent search	84
4.4.2	Iterative improvement	88
4.5	Experiments	90
4.5.1	Common settings	90
4.5.1.1	Network structure	90
4.5.1.2	Attack graph	92
4.5.1.3	Assignment of attack success probabilities	92
4.5.1.4	Risk functions and cost assignments	93
4.5.2	Comparison between the one-pass algorithm and NSGA-II	94
4.5.3	Iterative improvements	95
4.5.4	Comparison between the iterative algorithm and NSGA-II	95
4.5.5	Processing time comparison	98
4.6	Discussions	98
4.6.1	Computational complexity	98
4.6.2	Applicability to other problems	99

4.6.3	Parameter assignments and CVSS	99
4.6.4	Limitations	100
4.7	Conclusion	100
5	Attack function analysis	101
5.1	Introduction	101
5.2	Two forms of attack graphs	102
5.2.1	State enumeration graph	102
5.2.2	Exploit dependency graph	103
5.2.3	Comparison between the two forms	104
5.3	Attack functions for attack graph generation	105
5.3.1	Attack function	105
5.3.2	Proofs for Theorem 14	106
5.4	Induced automata of exploit dependency graphs	107
5.4.1	Induced automata	108
5.4.2	Example and implication	109
5.4.3	Proofs for Theorem 19	111
5.5	Example	113
5.6	Discussions	115
5.6.1	Mixture of different forms of attack graphs	115
5.6.2	Coarse grained analysis	116
5.6.3	Lifting and its application	117
5.7	Conclusion	117
6	Secured risk analysis	119
6.1	Introduction	119
6.2	Related work	120
6.3	Approach	122
6.3.1	Monotonicity, aggregation and segregation	122
6.3.2	Secured business risk assessment	124
6.3.3	Proofs for Theorem 26	124
6.3.3.1	About attack functions	124
6.3.3.2	About Datalog program segregations	126
6.4	Example	129
6.5	Discussions	130
6.5.1	Importance of vocabulary development	132
6.5.2	Need for granularity free quantitative metrics	132
6.5.3	Integration with information security risk treatment	132
6.5.4	Testing in the field	132
6.6	Conclusion	133
7	Summary	134
7.1	Contributions	134
7.2	Future works	135

List of Figures

2.2.1 The process of attack graph analysis	15
2.2.2 A minimal example of attack graph	16
2.2.3 Another small example of attack graph	16
2.2.4 An example of minimal attack graph with Datalog	18
2.2.5 The extracted attack paths	21
2.3.1 Non-aggregated attack graph (quoted from Noel and Jajodia [110])	25
2.3.2 Aggregation to machines and exploit sets (quoted from Noel and Jajodia [110])	26
2.3.3 Attack graph aggregated to protection domains (quoted from Noel and Jajodia [110])	27
2.4.1 Attack graph metrics families (quoted from Noel and Jajodia [112])	28
2.4.2 An example of non-independent scenarios	34
2.5.1 Another example of an attack graph	38
2.6.1 AND and OR combinations by CPTs	45
2.6.2 An example of dependent exploits	46
3.3.1 A small example of an attack graph	53
3.4.1 An example of the reachability-graph undecidability	56
3.4.2 Typical minimal parts in reachability graphs	59
3.4.3 An input attack-graph for reachability calculation	60
3.4.4 Another attack graph for reachability calculation	62
3.4.5 The overlapping volume of convergent events	62
3.5.1 General structure of the experimental network	63
3.5.2 An example partial attack graph within a node	64
3.5.3 Processing times of Bayesian attack graphs	65
3.5.4 Processing times of reachability graphs	65
3.5.5 Comparison of processing times	66
3.5.6 Processing times of large-scale reachability graphs	67
3.5.7 Replacement of a gateway with a hub	67
3.5.8 Processing times of cyclic reachability-graphs	68
3.6.1 Update counts per node in acyclic attack graphs	69
3.6.2 Processing times per node in acyclic attack graphs	70
3.6.3 Evaluation counts per node in acyclic attack graphs	71
3.6.4 Processing timers per node in cyclic attack graphs	71
3.6.5 Update counts per node in cyclic attack graphs	72
3.6.6 Evaluation counts per node in cyclic attack graphs	72

4.4.1 An example of the Pareto front	85
4.4.2 An example with approximated solutions overlaying	85
4.4.3 An example of three dimensional dominant solutions	86
4.4.4 Interrelationships among P^- , P , and P^+	88
4.4.5 An example of an approximated Pareto front	88
4.4.6 Interrelationships among neighbor patchsets	89
4.5.1 General structure of the experimental network	91
4.5.2 General structure of the attack graph per node	92
4.5.3 Comparison between NSGA-II and Algorithm 4.1	94
4.5.4 Performance improvements enabled by the proposed technique	95
4.5.5 Security score evaluation counts in each case	96
4.5.6 Comparison between NSGA-II and Algorithm 4.3 with 10 iterations	96
4.5.7 Comparison between NSGA-II and Algorithm 4.3 with 20 iterations	97
4.5.8 Comparison between NSGA-II and Algorithm 4.3 with 30 iterations	97
5.2.1 A minimal example of attack graph	103
5.2.2 Another small example of attack graph	103
5.4.1 A mini network	109
5.4.2 An example of an attack automaton	110
5.4.3 Another corresponding exploit dependency graph	110
5.5.1 An example network	113
5.5.2 Exploit dependency graph	115
5.5.3 Induced automaton	116
6.4.1 Initial coarse-grained program diagram	130
6.4.2 Segregated form	131
6.4.3 Fine-grained program diagram	131

List of Tables

2.1	Three major forms of attack graphs	24
2.2	An example CPT with varying exploitability	47
3.1	Network size parameters for the experiment	63
3.2	Network size parameters for the large-scale experiment	66
4.1	Network size parameters for the experiment	91
4.2	Average processing times in seconds of each algorithm	98

Chapter 1

Motivation

1.1 Introduction

Information security and its management is one of the vital issues in today's world. Many best practices and research are discussing this topic. These tools enable us to protect ourselves from malicious attacks, in theory. In reality, it is not uncommon to see large-scale security breaches happening in the industry and the public sector. To understand the situation, let us see the case in the Office of Personnel Management [17].

The Office of Personnel Management (OPM) is a federal government agency in the United States. The agency implements federal human resources policy, oversight, and support for federal government employees, retirees, and their dependents. On July 4, 2015, OPM disclosed that it was found to be attacked by an unknown attacker, and over twenty million personnel records, including the background investigation results of counter-terrorism officials, were breached. Michael Hayden, the former director of the CIA, said that “[OPM data] remains a treasure trove of information that is available to the Chinese until the people represented by the information age off. There's no fixing it.”¹ The OPM data breach is one of the most severe incidents in history. In contrast to Mr. Hayden's argument, the report cannot show clear evidence of the culprit. It only enumerates circumstances, as is often the case in this sort of attack. This anonymity of the attack even emphasizes the shock of this unprecedented incident and its aftermath.

The OPM officials were not without tools to prevent the incident. For example, it was pointed out later that if the IC card-based authentication, which was standardized and promoted in advance, had been introduced earlier, the intrusion could have been prevented. Also, the US federal government has published many guidelines called the SP-800 series [102, 105, 101, 103, 104, 100] for federal information security management to comply with the Federal Information Security Management Act (PL 107-347). That is, they had an enforcing law, detailed implementation guidelines, and a technical solution to establish their security.

The critical factor behind this catastrophic result is the strong pressure on their business. After the 911 terror attacks, the US federal government, vigorously promoting counter-terrorism policy, dramatically increased the number of relevant personnel. “That staffing surge caused a backlog in processing background investigations. The backlog was at least 188,000 by 2004. The Intelligence Reform and Terrorism Prevention Act (PL 108-458) required that 90 percent of clearance applica-

¹iii, [17]

tions had to be resolved within 60 days by 2009, a reduction of 84 percent from the then 375 day average wait time.”² Additionally, it was decided around the end of 2004 to consolidate to OPM the relevant investigation tasks previously held in other departments and agencies. As a result, “it is not uncommon for existing policies to be circumvented in order to achieve business functions while exposing the entire agency to unnecessary risk.”³

The OPM data breach is not a singular case. On June 1, 2015, the Japan Pension Service (JPS) announced that about 1.25 million records of privacy information had leaked out by advanced persistent threats [61, 162, 99]. This incident was also technically preventable; the breached information was assumed to be kept in an environment isolated from the Internet. However, it was duplicated onto a shared file server accessible from the Internet to ensure business efficiency. Japan’s government had plenty of guidelines too, and JPS had appointed several officers as the information security authorities. It is interesting to note that JPS was established for the swift and proper management of the enormous amount of pension records. It was rooted in a political event known as the pension record problem, where a significant amount of records were lost due to inappropriate operation, and it was very natural to be under strong pressure like OPM.

Both cases indicate serious issues in information security management. First, poor management, not technical issues, caused both problems. Secondly, management guidelines and strong regulations on information security were in place. Third, despite these circumstances, business operations took precedence over information security. This is nothing but a failure of information security management. If people are indifferent to implementing technical measures and codified best practices, then serious researchers’ dedicated studies never make a difference.

This work extends the analysis in the author’s prior study [135]. In the prior study, the OPM case was closely examined, but the discussion was centered on the responsibility of decision-makers and the business risk analysis. This time, the focus is on the issues in managerial mechanism. This chapter points out several issues in the current information security management framework, mainly referring to the OPM case. It presents a hypothesis about why effective countermeasures and best practices are ignored and considers the existing research efforts challenging those issues. The focal point of this chapter is to suggest that the misalignment of stakeholders’ risk conception is the critical factor in catastrophic failures. Then, we propose a draft research program to establish and continuously improve information security management. The program is intended to tackle the issues in the managerial mechanism from the perspective of the misaligned risk conception, and the concept named *disciplined alignment* is suggested as a whole.

1.2 Issues in information security management framework

Several well-known standards, such as ISMS standards [63, 64, 65] and the SP-800 series, codify the information security management framework. They are based on a common conceptual model. This section shows the contrast between the common framework and the practices.

²p.38, [17]

³p.76, [17]

1.2.1 Information security management framework

Information security management is generally regarded as a type of risk management. The ISMS standards define the process consists of the following four steps:

1. Information security risk identification: Identify risks relevant to the business.
2. Information security risk analysis: Examine individual risks' detailed behavior and properties.
3. Information security risk evaluation: Determine whether the level of risk is acceptable.
4. Information security risk treatment: Take measures against unacceptable risks.

The iteration of this series of steps is the common model in information security management.

The implicit assumption behind this model is that we can understand the risks. They are supposed to be identifiable, inspectable, and comparable with the predefined acceptable level of risks. On the other hand, by definition, risks involve some uncertainty. Therefore, the model cannot be viable without somehow managing the uncertainty of risks. If that uncertainty is not handled appropriately, risk identification, analysis, and evaluation results become unreliable, and the entire information security management process will be groundless.

One more thing to note is that information security management is an aid for the smooth execution of the business, not the purpose. Meanwhile, the business requires many resources to run. General business activities and information security management compete for finite resources. If the business case for information security management is flimsy, other business units will argue that the relevant resources should be spent on their activities.

In information security management practices, the handling of uncertainty is not only related to the appropriate handling of risks but is a matter of the existence of the information security management initiative.

1.2.2 Insecurity in risk analysis

Issues

The spread of incompliance with security controls indicates that OPM downplayed the information security risk. While the detail of risk analysis in OPM is unclear in the report, it is clear that it could not stop circumventing the security controls. In the face of conflicts with business activities, we need a mechanism to protect information security management from incomplete risk analysis that cannot necessarily convince business activities of the risk landscape. Although it is not an information security incident, the following case in seismic risk analysis that preceded the Fukushima nuclear accident demonstrates an important aspect of this issue and its key factor.

The Japan National Diet commission's investigation report [40] tells that the commissioners "believe that the tsunami risk was underestimated Meanwhile, with regard to severe accident countermeasures, the probabilistic safety assessment of tsunamis was deemed uncertain, so the consideration itself was pushed back and measures were not taken. As such, Tokyo Electric Power Co. used probabilistic evaluations in ways that suited them in order to avoid clarifying the tsunami risk."

Risk is generally defined as the product of probability and impact. Underestimated probability can make the risk looks ignorable. Moreover, supporting statistical information is usually unavailable

for information security risks. Thus, it could be easy to manipulate the degree of risk in practice. In other words, incomplete risk analysis is also vulnerable to motivated underestimating risks.

The reliability of information security management depends on how well the uncertainty is handled. Past studies of uncertainty relevant to risk analysis have mainly focused on elaborating on or reducing uncertainty. However, from the viewpoint of making information security management reliable, it is not necessarily the only solution. The focus is on the effective and steady information security management process, and uncertainty refinement is only one of the methods to ensure it. Rather, since we cannot eliminate every unknown factor, it is necessary to prevent the information security management process from being challenged based on that remaining uncertainty.

In short, if the analyzed risks can be legitimately and irresponsibly rephrased within a range of uncertainty, that freedom becomes a source of abuse. A possible way to prevent this problem is not only to reduce uncertainty but also to require sufficient accountability in such paraphrasing about risk.

Approaches

The ISMS standards and SP-800 series are organized as a type of risk management but do not include risk analysis methods as part of them. IEC/ISO 31010 [57], the reference standard for risk management in ISO standards, introduces 31 risk analysis methods. However, almost all of these methods rely on the experience and expertise of security analysts, and it is very difficult for ordinary staff within an enterprise to obtain objective results from them. These standards are fundamentally vulnerable in the sense that they could be challenged easily.

There are many studies on information security risk metrics, e.g., [84, 128, 122]. Risk metrics enable us to compare the level of risks easily. However, we should not confuse the ease of comparison with the validity of the comparison results. Verendel [161] points out the immaturity of the said quantitative metrics. Reliable quantitative risk metrics require that the input data are correct and that the metrics are well-tested, but both requirements are difficult to satisfy. While these two issues should be resolved in the long run, we need careful consideration in using these metrics at this time.

Attack graph analysis is an approach worth elaborating on among past studies; a model-based approach for network security analysis generates and analyzes a directed graph called an attack graph. In one definition, it is a generalized and computerized variation of bottom-up fault tree analysis. We can obtain an attack graph from network topology, its configuration, and the distribution of vulnerabilities. An attack graph is a consolidated depiction of logically possible attacks. The strength of this method is that (1) it is an algorithmic method based on objective data and can be implemented by non-experts, and (2) it is a qualitative analysis, i.e., it enables security analyses while separating the problem of quantitative metrics.

Phillips and Swiler [124] proposed the first comprehensive framework of attack graph analysis. Since then, many forms of attack graphs have been examined. For example, state enumeration graphs [73] are a kind of finite automata. Exploit dependency graphs [71, 119] are cyclic AND/OR graphs interpretable as a set of merged fault trees. Bayesian attack graphs [36] are the recast forms of exploit dependency graphs as Bayesian networks and enable the probability calculation of malicious events. We can choose the best suitable definition depending on the scope of available information and can intentionally avoid quantitative evaluation in the absence of information by using state enumeration or exploit dependency graphs. When using qualitative attack graphs, we only know whether an

attack is logically possible, and the probability of that attack is not included anywhere. As a result, risk comparison is separated from risk analysis, so involving decision-makers in the risk assessment process with this method provides a mechanism to clarify their responsibilities in weighing the risks.

1.2.3 Distance from business risk analysis

Issues

Decision-makers are not interested in the risk that a particular server will stop working but in its consequential risk to the business. The knowledge of information security experts is mainly helpful for the former analysis. The business impact mentioned in the latter cannot be clarified without the knowledge of the business practitioners in the concerned enterprise. Therefore, the risk assessment process requires joint efforts between the two parties, but such collaboration between people with different expertise is often problematic.

There was a considerable gap in perceptions between those involved in information security and those involved in business practices in the OPM case. For several years before the incidents occurred, OPM's CISO and internal audit department repeatedly pointed out "material weaknesses" in information security management. In contrast, as quoted in the introduction, the OPM's operational departments were uncooperative in implementing information security measures hindering their work. Furthermore, it is noteworthy that the U.S. Congress, which was supposed to receive the audit report and supervise and guide OPM, was also indifferent to information security. The OPM hearings in 2009 were concerned only with processing personnel cases, and no questions were raised about information security.⁴

The U.S. federal government had much guidance on risk assessment. The law and the guidance were carefully examined, and information security management practices were mandated under them. Thus, the lack of such a framework was not the problem.

The problem lies in the immaturity of business risk analysis methods. As discussed in the previous subsection, no objective and reliable risk analysis method is widely used. It is even more true for business risk analysis, which is broader in scope. In the OPM case, the "material weaknesses" pointed out by the audit reports were mainly about non-conformance to the security regulation and were not the direct warning about clear and recent danger. It is unclear if the nature of business risk then was shared within OPM. At least, the audit reports to Congress did not show such a description. The insufficient business risk description could explain the malfunction of the information security management in the OPM case, in addition to the uncertainty in risk analysis.

We need a trustworthy method for business risk estimation. The method presents incontrovertible facts under a vocabulary for both sides as the basis for further discussion. Without such a method, how can security professionals convince business practitioners of the need for cumbersome security controls? Cooperative dialogue between the two parties will be deadlocked easily without its aid.

Approaches

Enterprise Architecture (EA) is a method for understanding the complex relationship between IT infrastructure and business activities. Several well-known EA frameworks, such as DoDAF [160] and Zachman Framework, have been proposed, and describing the entire picture of an enterprise

⁴p.39, [17]

following these frameworks will help us understand the dependencies among various components, including business processes, digital services, and infrastructure.

Enterprise Information Security Architecture (EISA) extends EA to provide a comprehensive view of security measures and their contributions. O-ESA [164] is an example of EISA, which primarily focuses on automating security monitoring and configuration management. It is now under the ownership of The Open Group but has not been updated for years. The Open Group is also the owner of TOGAF [157] and ArchiMate [158]; both are the de facto standards for EA frameworks. These frameworks are not EISA, but the integration with SABSA, an EISA proposed by Sherwood [140] independently, has been explored [156]. These efforts give us a vocabulary to depict a whole picture of an enterprise, including security measures and its relevant business activities. Though it could be a practical means of describing the risk analysis results, it is not a risk analysis method.

In the literature, several risk analysis methods are proposed based on business model descriptions parallel to EA. Innerhofer-Oberperfler and Breu [59] suggested a conceptual enterprise architecture framework and proposed a top-down dependency analysis method to evaluate the impact and probability of the risk. Breu et al. [10] extended this study to elaborate on an algorithm for calculating the expected loss. Zambon et al. proposed a business availability analysis method [179, 180]. Johnson et al. [74] have combined enterprise architecture and attack tree (fault tree) analysis. In this approach, an attack tree is represented as an extended influence diagram; thus, it can calculate the occurrence probability of incidents. Later on, Sommestad et al. proposed leveraging a metamodel designed with the predefined dependencies between attributes to produce the diagram [144, 145]. Johnson et al. [75] incorporated the uncertainty in the enterprise model as the probabilistic perturbation in the Monte Carlo simulation. Although these studies can be regarded as EA-based risk analysis methods, they are vulnerable as the fore-mentioned risk analysis methods because they adopt an approach to calculate numerical risk metrics without sufficient consideration.

Some research deals with business risk analysis as an extension of attack graph analysis. Froh and Henderson [38] extended the inference rules used in attack graph analysis to accommodate the dependency between services and infrastructure. Johnson et al. [76] proposed a threat modeling approach named “pwnpr3d,” integrating the idea of enterprise architecture, attack graph analysis, and the Time To Compromise metric. The theoretical basis of exploit dependency graphs, the adopted form of attack graphs in these studies, is logic programming based on monotonic logic. Datalog [12] is a restricted logic programming language that matches this class. We can use Datalog to develop libraries for risk analysis with enjoying its powerful expressiveness [118]. The works by Froh and Henderson [38] and Johnson et al. [76] are in line with this intuition, and in fact, Johnson et al. [77] have proposed a domain-specific language called Meta Attack Language (MAL); this is theoretically a variation of Datalog with additional attributes.

The prior studies mentioned above indicate that, although they could be vulnerable too, and there is still significant room for development, a theoretical foundation for risk analysis supported by powerful expressiveness can enable various risk analyses, including business risks.

1.2.4 Alienation from corporate governance

Issues

The content discussed in the subsections above is a prerequisite for risk analysis to be meaningful to business and is just a necessary condition for risk management to function. Even if the risks are

clear and stakeholders agree on them, they are still competing for a finite amount of resources. It is unreasonable to give preference only to information security in this situation. It is legitimate for decision-makers to limit risk treatments based on rational consideration. However, the distinction between a reasonable restriction on risk treatments and an inappropriate and intentional restriction is ambiguous.

In the OPM case, the situation was as follows: “The Inspector General (IG) also found that the agency’s Office of CIO (OCIO) lacked the authority it needed to manage security matters effectively..., and most (of the designated security officers, DSO) are only managing their security responsibilities as a secondary duty to their primary job function. ... the OCIO has no authority to enforce security requirements.”⁵

It is easy to say that the OCIO should have been given sufficient authority since the incident occurred as a result. Nevertheless, it should not have been easy to determine what level of authority had been appropriate before the incident occurred. Rather, we should focus on the fact that the DSO managed its security responsibilities as secondary.

In principle, since information security is one of the requirements for smooth business execution, DSO should have treated it with the same importance as other business requirements. The DSO should have accurately recognized the danger of operational disruption when information security is compromised. The DSOs’ misconception about information security is the root cause of their unaligned comprehension. The strong authority of OCIO is just a means to adjust the misalignment by enforcing compliance.

Unfortunately, public awareness of the nature of information security is inadequate. In addition, humans tend to disregard what they do not understand well and alienate the tasks associated with those unknowns. It also applies to the members of Congress who did not raise any information security concerns during the OPM hearings. While educational efforts to improve public awareness are essential, we should keep in mind that the investment of time and money for education is also subject to competition for finite resources.

What is desirable is to have a mechanism that naturally identifies and corrects the deviation arising when businesses erroneously alienate information security efforts in need.

Approaches

Deficiencies in information security management are generally perceived as governance failures, and reinforcement of governance often applies as the corrective measure. However, information security and business are competing resources, and the optimal allocation of the resources is constantly changing. Immediate empowerment may be a short-term solution, but it could also cause problems in the long run.

In contrast, management frameworks such as BMIS [62] and the recently updated ISMS standards [69, 67, 68] seem to be aware of the importance of establishing a holistic approach. The shared recognition that an improved level of information security will enhance corporate value keeps information security management from alienation. This is a sound approach. The concern is that even though it is a good philosophy, measuring the level of information security and the resulting enhancement in corporate value is difficult.

⁵pp.40-41, [17]

The challenge of integrating quantitative assessment and management does not apply only to information security. It is one of the universal challenges in management; therefore, it might be possible to leverage the wisdom of management best practices.

Benefits realization management and portfolio management [159]⁶ are well-organized tools for problems alike. These management methods identify the elements relevant to the enterprise and their linkage and raise the level of stakeholders' understanding of the situation. This process functions as an embedded educational system about the complex nature of the enterprise. After that, quantitative metrics and evaluation criteria are introduced, and decision-making is performed consistently. The focal point of these approaches is the rational consensus-building process to agree on the desirable resource allocation; metrics are merely tools.

In risk management, where uncertainty is inevitable, it is impossible to reach a completely accurate conclusion. At the same time, claiming that we cannot decide anything unless we obtain a perfect conclusion is irrational. Evaluating the decision's reasonability by the result completeness is often nonsense as we do not know the complete answer. More often, we determine it by the exhaustiveness and the balance of considerations during the decision-making process. While we cannot assert that it is easier than measuring risk in information security, there is potential in the approach to improve the quality of the process, especially the process of consensus-building as the basis of decision-making.

1.3 Gaps between researches and practices

In the previous section, we discussed the issues in information security management and referred to several relevant studies and efforts that can help or ease the situation. Though these approaches have the potential, they are still insufficient to establish sound information security management because they lack the prerequisites for their use in practice. This section elaborates on three remaining gaps that should be filled to advance information security management research and practices.

1.3.1 Lack of tools integration

Research results in information security are not only diverse but also enormous. The ones mentioned in the previous section only reflect the authors' limited knowledge. There are many other advanced studies. Few researchers understand the whole picture. It is even more so for practitioners.

Practices need to combine various technical and managerial solutions, from risk analysis to risk treatment. Even if the individual research results are excellent, it does not guarantee their reliable performance under some combination. Exploring the best combination of those research results is a challenging research topic that requires careful consideration, e.g., [16, 46, 109]. It is illogical to think practitioners should identify this combination independently of researchers. Researchers should address this issue since it requires examining the wide variety of studies discussed in academic vocabulary. Unless employable integrations are provided, practitioners are unlikely to use those research results in their own decisions.

⁶These management practices are now incorporated into the well-known project management standards owned by the Project Management Institute (PMI) and AXELOS. Thorp's book is one of the origins of these concepts.

1.3.2 Lack of installation assistance

While the community of researchers consists only of experts, a wide variety of stakeholders, who are not necessarily familiar with information security, comprises the field where the information security practitioners work. Even when the desirable integration above is given, it will not be effective if it is not understandable to the people in the field. Therefore, it is essential to develop the instruments to explain research results in an easy-to-understand fashion, including various guidance materials for the general audience and teaching materials to support training. These materials should be developed through the joint effort of researchers and practitioners.

On the other hand, we should remember that OPM failed to implement information security management even though it had plenty of guidance. The ensured introduction of the methods well-considered and organized in academic research requires yet another step beyond the preparation of guidance.

Roughly speaking, most obstacles to installing rational information security management are relevant to the integrity of management and governance, not technologies or theories. At this point, practitioners seem to have made meaningful progress, such as benefits realization management and portfolio management, which clarify benefits and contributions to the business and support consensus-building. However, those best practices are more like an art than a science. Researchers can contribute by elaborating on the factors that govern the successful installation of those best practices.

1.3.3 Lack of performance validation

The information security study is not a natural science that seeks eternal truth but a practical science that tries to find solutions in the changing society. Actual practices must validate the value of studies in this field. However, it is unrealistic for most enterprises to share the details of incidents, practices, and even policies for information security management due to its sensitive nature. As a result, most security studies lack practice-based validation and depend on conceptual elaboration and numerical experiments. The restrictive communication between researchers and practitioners inhibits the further advancements of information security studies and detaches practices from advanced research results.

From practitioners' viewpoint, the lack of validation of their practices brings uncertainty in performance. When they feel that their practice does not provide enough security, the cause could be rooted in the way of implementing the standardized best practices or the design of best practices replicated in that implementation. Distinguishing these two causes is critical since the latter case urges some tailoring. It is easier said than done because comprehending the details of best practices requires a high level of expertise that not every security practitioner in the field has. The DSOs in the OPM case were a typical example of this; they were chosen among the ordinary staff in the business sections. Validation benchmarks that incorporate researchers' findings also have real value for practitioners.

1.4 A proposal of a research program

The hand-in-hand relationship between researchers and practitioners is the basis of sound information security management research and practice. Here, we try to clarify what researchers can do to build

mutual trust. It is a research program devoted to establishing a concrete and robust foundation for fulfilling the aforementioned three gaps.

1.4.1 Make employable integrations

The first thing to do is define and elaborate on integrating state-of-the-art studies in this field. This integration must be presented as an employable form by practitioners.

As a springboard for further discussion, this chapter proposes to structure this integration into the following four tiers:

1. **Foundational theory of secured risk analysis:** The first tier should provide the mathematical foundation to categorize the risk description based on its reliability to secure the risk analysis results from intentional underevaluation. The fore-mentioned separation of qualitative analysis and quantitative evaluation is an option, and the attack graph analysis can be a candidate for further development.
2. **Library and techniques for business risk analysis:** The second tier is built on top of the first tier to enable business risk analysis while enjoying the guaranteed robustness. Business risks are related to a broader vocabulary than simple security incidents analysis; thus, they require a library of clearly defined relevant elements. Also, business risk analysis algorithms based on the library should be incorporated into this tier. The MAL [77] and its application [76] are examples in line with this requirement.
3. **Concrete process model focusing on consensus-building with explicitly stated grounds:** The business risk analysis mentioned above enables discussions among stakeholders in a manner more relevant to the actual value of their business, at least in theory. However, we cannot eliminate the uncertainty and thus need to focus on consensus-building rather than the systematic but superficial procedure. The consensus-building should record the premises and the referenced information used in the judgment for later managerial forensics. This tier provides the management process model leveraging the robust business risk analysis method defined in the base tiers while focusing on preserving input for the process.
4. **Library of imaginary cases:** Even though real cases are not obtainable, we can construct imaginary enterprises and elaborate on their security landscapes and practices. Such imaginary cases work as the showcase and the living textbook of the proposed integrated methodology for practitioners. Moreover, they could be reference testbeds for other researchers.

Researchers can start building various integrated methodologies composed of the four tiers above independently from practitioners. The resultant deliverable is a suite of carefully chosen techniques and showcased examples with the full range of information security management. It would be the basis for further communication between the two parties.

1.4.2 Clarify the verification and validation criteria

The proposed integration is just a hypothesis, and we need to test it repeatedly and refine it gradually. This iterative improvement is a joint effort of researchers and practitioners. Both parties have their

responsibility and role in accomplishing the contribution. Therefore, it is of critical importance to have criteria that verify and validate the implementation of each party's effort.

The verification criteria examine the implementation by practitioners of the integrated methodology. It corresponds to the gap in [Section 1.3.2](#) and should focus on the factors affecting the successful performance, including, for instance, governance structure, internal compliance in the handling of risk register, accuracy and exhaustiveness of configuration management, and level of general risk awareness among an enterprise.

The validation criteria, filling the gap in [Section 1.3.3](#), provide multi-faceted viewpoints for the definition of successful implementation of information security management. The key question behind the evaluation should be the alignment with the ideal concept and the business value—the former alignment tests practitioners' level of understanding of the articulated methodology. In contrast, the latter alignment implies that any information security management practice is justified when and only when it contributes to the business value.

Although clarifying these criteria requires substantial research effort, existing studies and best practices could be the first draft. These are not novel concepts. It is legitimate to find a similarity with the Verification and Validation model. However, within the proposed research program, the criteria have a specific role in supporting the communication between the two parties; researchers and practitioners. In other words, we expect these criteria to work as a protocol to build mutual trust by reducing miscommunication and not requiring them to be dogmatic umpires. Therefore, they should be formulated as a set of suggestions, not as constraints.

1.4.3 Towards a disciplined alignment

Before proceeding to the conclusion, let us review what we have discussed up to here. Firstly, we have put our attention on the OPM case. It depicted the catastrophic failure of information security management. Subsequently, we have enumerated the issues in information security management and the researchers' efforts to tackle those issues. The main concern was the misaligned conception of information security between security practitioners and business staff. Then, we have examined further the gaps between information security researchers and practitioners. The gaps impede positive collaboration in continuously advancing research and practice. Lastly, we have proposed a draft of a research program to reconcile those misalignments.

The focus is on the misalignment between security practitioners and business staff and that between researchers and practitioners. At the very beginning, we must accept that we cannot eliminate all the uncertainties. We can only behave, at best, rationally to cope with the inherent uncertainty in the world. At the same time, every person struggles to thrive and has the right to choose one's action based on their standpoint, which is rational behavior. The failure in management is often a by-product of conflicts among those different rationalities. Integrating each stakeholder's perspective in a way overcoming the conflicts shall contribute to the organizational mission at its best performance. The alignment in stakeholders' conceptions and behaviors is the foundation of the best risk management.

This alignment should be scientifically disciplined if we want to replicate it everywhere in need. Such an attempt requires the definition of alignment and evaluation criteria for the level of alignment, the process model for consensus-building, and the method for explicit and objective descriptions of decision-making materials. The combination of making employable integration of existing studies

and defining verification and validation criteria is a starting point of this research and is an enabler of further collaboration between researchers and practitioners towards a *disciplined alignment*.

1.5 Conclusion

This chapter refers to the OPM data breach to discuss the issues in the information security management concept. We pointed out three issues; vulnerability to intentional sophism in information security risk analysis, immaturity in business risk analysis methods, and the tendency of information security decoupling from corporate governance. These are the reflection of the misalignment between security and business practitioners' conceptions. Although information security researchers have made several approaches to these challenges, there is plenty of room for further development. There are also gaps between researchers and practitioners that inhibit the productive collaboration of the two parties. Those gaps are the lack of employable integration of existing studies, the lack of clearly defined installation guidance and verification criteria for best practice implementation, and the lack of validation and communication process of actual cases. A draft research program is proposed to overcome these issues and build mutual trust among researchers and practitioners. It urges to make employable integration of existing studies and clarifies the verification and validation criteria for practices. All these efforts shall contribute to the broader collaboration towards the disciplined alignment of managerial activities.

This dissertation is devoted to the theory of secured risk analysis, the foundational tier for the disciplined alignment mentioned above. In the following chapters, we focus on and examine attack graph analysis as our starting point. Several issues of attack graph analysis will be identified. Then, we propose approaches to overcome them. Based on those results, a formal approach for secured risk analysis will be proposed lastly.

Chapter 2

Attack graph analysis

2.1 Introduction

Attack graph analysis is a model-based approach for network-security analysis. It analyzes a directed graph called an attack graph. Usually, each node in an attack graph corresponds to a malicious event caused by attackers, and the edges correspond to the causal relations between events. We can obtain an attack graph from the network topology, its configuration, and the distribution of vulnerabilities. It is a merged set of attack paths, each of which is a sequence of steps an attacker could take to compromise the concerned network. Various researchers have proposed network security metrics based on attack graphs. Also, several algorithms exist searching within them for the optimal combination of security controls. This broad applicability spanning from risk analysis to risk control is a unique strength of attack graph analysis.

Phillips and Swiler [124] proposed the first comprehensive framework of attack graph analysis. Much progress has been made since then. It includes the compact representation of attack graphs and monotonicity assumption, the two types of node combinations, hierarchical visualization technique, a mapping to propositional logic, approximated optimization algorithm for security controls, and more. The research history emphasizes several issues, too, such as scalability, the difficulty in metrics design, and handling of cyclic structures. This field still requires further effort.

Behind this progress, there have been various definitions of attack graphs. At an early age, it was defined as finite automata, called *state enumeration graphs* [124, 154, 73]. Later, the definition as cyclic directed AND/OR graphs, called *exploit dependency graphs* [6, 71, 119], was explored. Now, several researchers generalize it as a special case of Bayesian networks called *Bayesian attack graphs* [36, 95, 96]. One can find even more different definitions. Each of them is still in use by many researchers, and this situation makes it difficult for people new to this field to comprehend past achievements.

Many surveys focus on attack graphs. To exemplify a few, Hong et al. [52] provide a broad survey of graphical security models, including various forms of attack graphs. The survey by Kaynar [78] focuses on the attack graph generation process and representative definitions. Zeng et al. [181] mainly summarize the application studies of existing well-established theories, such as Markov models, to attack graph analysis. Unfortunately, none provides an in-depth description of the foundational concepts commonly used in attack graph analysis and is suitable for beginners.

This chapter aims to summarize the foundational concepts in attack graph analysis. We will focus on foundational studies and capture their major contributions. Most of those selected studies satisfy

at least one of the following criteria: (1) relatively well-cited and (2) focusing on exploit dependency graphs since it is the most well-studied form in this field. We also discuss Bayesian attack graphs because they generalize exploit dependency graphs with substantial progress in their expressiveness. The crucial concepts are elaborated, and we will discuss the dependencies among them. It is not our scope to encompass the past and recent studies as much as possible. The emphasis is on the brief, concrete understanding of the essential ideas. This approach provides a sound basis for further research.

This review summarizes five topics; foundation, visualization, network security metrics, network hardening, and Bayesian attack-graph. [Section 2.2](#) introduces the foundational concepts of attack graph analysis. It also provides the definition of relevant terms and an efficient, concise algorithm for attack-graph generation. [Section 2.3](#) refers to several visualization techniques for an attack graph. In general, an attack graph contains too many nodes and edges and is difficult to comprehend, and visualization techniques help. We will see attack-graph based network security metrics in [Section 2.4](#). It is an indicator summarizing the level of risk implied by an attack graph. Theoretical issues in practical metrics design will be pointed out. [Section 2.5](#) is dedicated to network hardening, i.e., combinatorial optimization of security controls applicable to the concerned network. While combinatorial optimization is known to be computationally expensive, an approximated optimization algorithm computationally efficient and sufficiently close to optimal is known. [Section 2.6](#) examines the Bayesian attack graph. It can leverage Bayesian networks' powerful expressiveness to handle the complex dependencies among exploits and observed facts such as detected intrusions. Finally, we will sum up and clarify the remaining issues in this field.

2.2 Foundation

In this section, we summarize the concept of attack graph analysis and the definition of attack graphs. This part provides the foundational understanding of attack graph analysis and its key components.

2.2.1 Conceptual framework

Phillips and Swiler [124] proposed the first comprehensive framework of attack graph analysis. Their framework enumerates the following inputs for attack graph generation:

- Network configuration: A list of hosts and the connection structure among them. Software configuration on each host is also included.
- Vulnerability distribution: A list of pairs of a vulnerability and a host. Each pair denotes that the host has the paired vulnerability.
- Threat: Entity, such as cybercriminals and malicious insiders, exploits vulnerabilities distributed over the network.
- Attack template: Descriptions of the way the threat exploits vulnerabilities to accomplish adversarial results.

This set of information enables us to simulate the threats' behavior. For instance, we can imagine that an attacker (Threat) connects to a gateway server with a known vulnerability (Network configuration and Vulnerability distribution) and then exploits the vulnerability to penetrate further

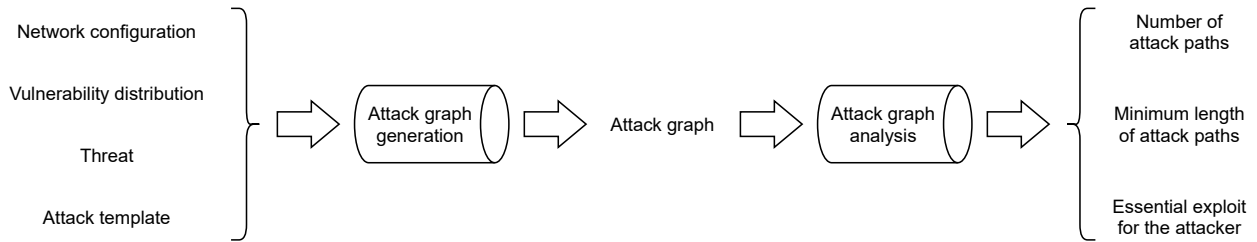


Figure 2.2.1: The process of attack graph analysis

(Attack template). Similarly, if we know that the gateway connects to another vulnerable server, it is expected for the attacker to compromise the server as a next step. Appropriately formalized input enables algorithms to automate these inferences. The repetition of step-by-step automated inference forms a chain of exploits. An attack graph is a graph structure given by merging those chains of exploits. It means that we can extract each chain of exploits from an attack graph. We call a chain of exploits extracted from an attack graph an attack path. We will define these terms more precisely later.

Phillips and Swiler [124] referred to fault-tree analysis as a comparison. A fault tree also gives us the logically inducible attack patterns. However, they pointed out that “fault trees don’t model cycles (such as an attacker starting at one machine, hopping to two others, returning to the original host, and starting in another direction at a higher privilege level).” That is, attack graphs are intended to allow such cyclic structures.

Once an attack graph is generated, it tells a lot about the security of the concerned network (See Figure 2.2.1). For example, the number of attack paths on the graph shows the availability of attack options for the attacker. If the minimum length of attack paths increases, it also raises the bottom line of the attacker’s required effort. An exploit being shared between many attack paths might correspond to an effective target for security patching. An attack graph is a rich source of these practical implications. As a whole, the generation and analysis of an attack graph have the potential for computerized assistance to cybersecurity management.

2.2.2 Attack graph representation

The attack graph generation algorithm needs to be computationally efficient. In its early history, attack graph generation algorithms got computationally exploded easily. The leading cause resided in the form of attack graph representation.

Suppose we have N computers on the concerned network, each computer has M vulnerabilities, and each vulnerability has a state expressing whether it is exploited or not. Under this setting, the number of possible states of the whole network is 2^{MN} . *State enumeration graph* is a representation form of attack graphs in which each node corresponds to a possible state in that 2^{MN} options, and each edge represents an exploit, i.e., an attacker’s action to cause a state transition. The size of a state enumeration graph is intractable even for a small-sized network with tens of machines. The earliest attempts for attack graph analysis, such as Swiler et al. [154] and Ritchey and Ammann [129], adopted this form and got stacked up in the scalability problem.

The scalability problem was later solved by *exploit-dependency graph*, another representation form of attack graphs. An exploit-dependency graph is a directed graph composed of two types of nodes; *exploit nodes* and *condition nodes*. Edges run from condition nodes to exploits nodes or

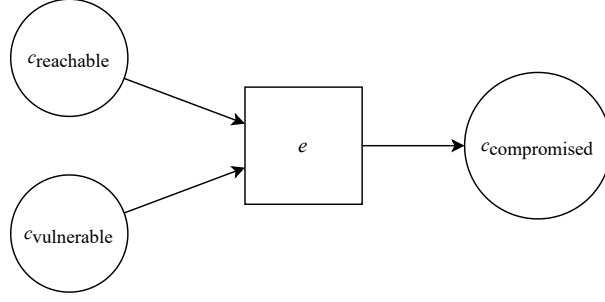


Figure 2.2.2: A minimal example of attack graph

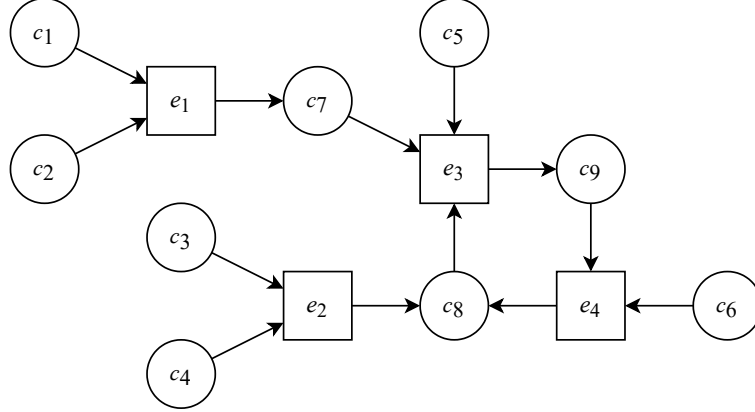


Figure 2.2.3: Another small example of attack graph

from exploit nodes to condition nodes. There are no edges directly connecting exploits to exploits or conditions to conditions, i.e., an exploit-dependency graph is a bipartite graph. Semantically, a condition node denotes a fact such as a specific host is reachable from the Internet, the host has a vulnerability, and an attacker compromises the host. Here, let us name each condition node as $c_{\text{reachable}}$, $c_{\text{vulnerable}}$, and $c_{\text{compromised}}$, respectively. An exploit node represents the causal relationship among these conditions. Suppose we have an exploit node e , and three edges $c_{\text{reachable}} \rightarrow e$, $c_{\text{vulnerable}} \rightarrow e$, and $e \rightarrow c_{\text{compromised}}$. Then, the small subgraph in Figure 2.2.2 implies that the two conditions $c_{\text{reachable}}$ and $c_{\text{vulnerable}}$ are the required *preconditions* for the exploit e to realize the *postcondition* $c_{\text{compromised}}$. A postcondition could be a precondition for another exploit. Figure 2.2.3 exemplifies a usual form of attack graphs; it is a merged set of minimal attack graphs like Figure 2.2.2. We can interpret an exploit node as a description of AND combination of its preconditions, and a condition node corresponds to an OR combination of its parent exploits. In this way, an exploit-dependency graph represents the dependency relationships among conditions relevant to the concerned network's security.

The concept of exploit dependency is introduced by Ammann et al. [6], and Jajodia et al. [71] organized it as the use of two types of nodes; condition and exploit nodes. The following definition is an adapted version of Wang et al. [166] and is a widely accepted form of attack graphs by various researchers:

Definition 1 (Attack graph). Given a set of *exploits* E , a set of *security conditions* C where $E \cap C = \emptyset$, a *require-relation* $R_r \subseteq C \times E$, and an *imply-relation* $R_i \subseteq E \times C$, an *attack graph* G is the connected, directed graph $G = (E \cup C, R_r \cup R_i)$, where $E \cup C$ is the node and $R_r \cup R_i$ is the edge set. For each attack graph $G = (E \cup C, R_r \cup R_i)$, *initial conditions* refer to the subset $C_i \subseteq C$ composed

of every condition $c \in C$ satisfying $\nexists e \in E$ such that $(e, c) \in R_i$, i.e., the set of all the leaves in the graph, whereas *intermediate conditions* refer to the subset $C \setminus C_i$.

Take a look at [Figure 2.2.3](#). In this attack graph, condition nodes are depicted by circles, and exploit nodes are in boxes. The conditions $\{c_7, c_8, c_9\}$ are the intermediate conditions, and others are the initial conditions. Also, the graph contains cyclic structure comprises e_3, c_9, e_4 , and c_8 . This composition shows the resemblance to and the difference from fault trees.

The major difference between exploit-dependency graphs and state-enumeration graphs is the meaning of nodes. In state-enumeration graphs, each node represents a network state, which is a combination of states of all the hosts in the network. In contrast, a node in exploit-dependency graphs holds far more limited information, such as one host connects another and whether a vulnerability is exploited or not. This difference makes exploit-dependency graphs substantially smaller than state-enumeration graphs. In general, the number of nodes in an exploit-dependency graph is $O(N^2 + NM)$ compared to $O(2^{MN})$ in a state-enumeration graph, where N is the number of machines, and M is the number of vulnerabilities each machine has.

2.2.3 Attack template representation

The definition of attack graphs lacks the detail of how we can describe the conditions and exploits in a practical manner. Ou et al. [118] proposed to use Datalog [12] for this purpose, which is a restricted but simple logic programming language.

A Datalog program is a set of *facts* and *rules*. Both facts and rules are composed of *terms*. A term is a *predicate* symbol followed by a list of *arguments*. An argument is either a *constant* or a *variable*. In this section, constants are denoted by lower-cased names and variables by capitalized names. For example, “`connectsTo(gw, H)`” is a term with predicate “`connectsTo`” with a constant “`gw`” and a variable “`H`”. This term represents a situation that a host named “`gw`” connects to some host referenced by “`H`”.

A *fact* is a term without variables. For example, “`connectsTo(gw, db)`” is a fact. On the other hand, “`connectsTo(gw, H)`” is not a fact since it contains a variable “`H`”. The term without arguments, such as “`attackerExists()`”, is also a fact. We can use facts as the representation of conditions in an attack graph. See the following examples:

```
connectsTo(inet, gw).
hasVulnerability(gw, v1).
compromised(gw).
```

Each fact corresponds to $c_{\text{reachable}}$, $c_{\text{vulnerable}}$, and $c_{\text{compromised}}$ in [Figure 2.2.2](#), respectively.

A *rule* is a pair of *head* and *body*. A *head* is a term. A *body* is a list of terms. There is a constraint that every variable in the head must also appear in the body. For example,

```
compromised(H):- compromised(X), connectsTo(X, H), hasVulnerability(H, V).
```

is a rule with the head “`compromised(H)`” and the body “`compromised(X), connectsTo(X, H), hasVulnerability(H, V)`”. A rule is a conjunctive function of facts, i.e., the fact implied by the head

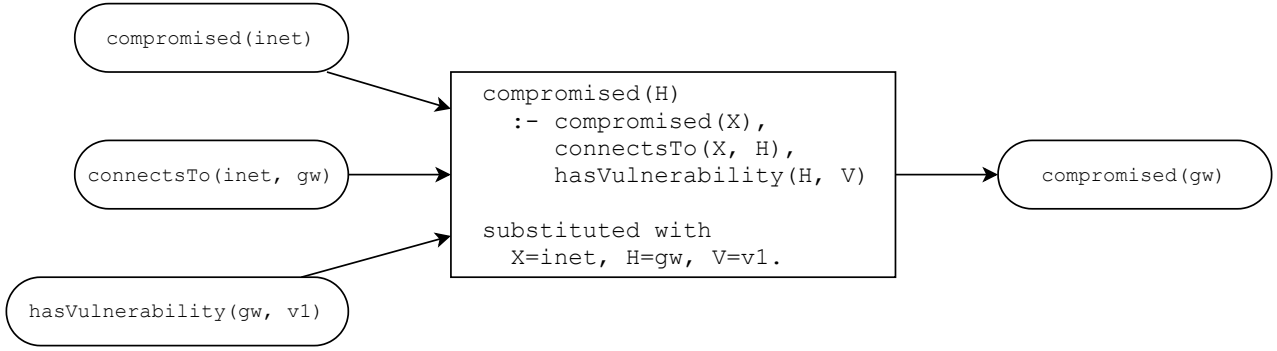


Figure 2.2.4: An example of minimal attack graph with Datalog

is inferred when every term in the body is presumed or inferred by the program. For the exemplified rule above, it means that we can say that “H” is compromised when all the three conditions, “X” is compromised, “X” connects to “H”, and “H” has some vulnerability “V”, are satisfied. In other words, the body implies a requirement for a constants substitution for variables “X”, “H”, and “V” which matches the facts defined in or inferred by the program. To understand what it means, suppose we have a Datalog program containing the following facts:

```

    compromised(inet).
    connectsTo(inet, gw).
    connectsTo(inet, www).
    hasVulnerability(gw, v1).

```

For these facts, the substitution, $X = \text{inet}$, $H = \text{gw}$, and $V = \text{v1}$ induces the body “`compromised(inet), connectsTo(inet, gw), hasVulnerable(gw, v1)`”. Then, the corresponding head “`compromised(gw)`” is inferred successfully. Similarly, another substitution, $X = \text{inet}$, $H = \text{www}$, and $V = \text{v2}$, induces the body “`compromised(inet), connectsTo(inet, www), hasVulnerability(www, v2)`”. However, the term “`hasVulnerability(www, v2)`” is not in the presumed facts. Therefore, this substitution fails, and the corresponding head “`compromised(www)`” cannot be inferred. We call a rule supplied with a successful substitution an *instance*. We can represent attack templates as rules, and exploits as instances in Datalog. Figure 2.2.4 depicts a Datalog leveraged form of the minimal attack graph in Figure 2.2.2.

As Figure 2.2.4 shows, Datalog provides rich expressiveness in the attack graph nodes and the attack template representation. Ou et al. [118] discuss various attack templates written in Datalog. The example in Figure 2.2.4 also explains why the size of an exploit-dependency graph has the complexity of $O(N^2 + NM)$. In the Datalog representation, network connectivities appear as the conditions describing point-to-point connection, such as “`connectsTo(inet, www)`”. The number of these conditions has the $O(N^2)$ complexity at worst. Similarly, the vulnerability distribution appears as conditions like “`hasVulnerability(gw, v1)`”. This sort of condition has the $O(NM)$ complexity. As the result, the spatial complexity is assured to be polynomial.

Meanwhile, Ou et al. [119] give the following definition of attack graphs, which is also popular among the researchers in this field:

Definition 2 (Logical attack graph (adapted from Ou et al. [119])). $((C_r \cup C_p \cup C_d), R)$ is a log-

ical attack graph, where C_r , C_p and C_d are three sets of disjoint nodes in the graph, and $R \subset (C_r \times (C_p \cup C_d)) \cup (C_d \times C_r)$. C_r , C_p , C_d are the sets of derivation nodes, primitive fact nodes, and derived fact nodes, respectively.

This definition is essentially equivalent to [Definition 1](#). The set of conditions C in [Definition 1](#) corresponds to $C_p \cup C_d$. The set of exploits E is to C_r , $C_d \times C_r$ is to R_r , and $C_r \times (C_p \cup C_d)$ is to R_i , respectively. From the viewpoint of Datalog, C_p is the set of predefined facts in a Datalog program, C_r is the set of instances, and C_d is the inferred facts appearing in the heads of instances in C_r .

2.2.4 Attack graph generation

Ou et al. [119] developed a software named MulVAL, a Datalog execution runtime specially designed for attack graph generation. It is freely available on the Internet and used by many researchers. Here, however, let us see a more simplified algorithm.

The process to generate legitimate instances from a rule and the set of facts is called *unification*. In the context of attack graph generation, the execution of a Datalog program is the repetition of unification. Successful unifications generate new instances, and the heads of them contain new facts. Therefore, repeated unifications grow the set of facts. This process terminates at some point when the facts set get saturated, i.e., it will stop when all the logically deducible facts are identified. After the saturation, the set of predefined facts and the set of instances and facts generated by unification compose an attack graph. [Algorithm 2.1](#) implements this flow.

Algorithm 2.1 An algorithm for attack graph generation

Input:

R : the set of rules, equivalent to the attack templates.

F : the set of facts, equivalent to the initial conditions.

Output:

C : the set of facts, equivalent to the inferred conditions.

E : the set of instances, equivalent to the inferred exploits.

R_r : the set of require relations.

R_i : the set of imply relations.

```

1:  $C' \leftarrow F$ 
2:  $E' \leftarrow \emptyset, R_r \leftarrow \emptyset, R_i \leftarrow \emptyset$ 
3: repeat
4:    $C \leftarrow C'$ 
5:    $E \leftarrow E'$ 
6:   for  $r \in R$  do
7:     for  $I \in \text{unification}(r, C)$  do
8:        $C' \leftarrow C' \cup \{\text{head}(I)\}$ 
9:        $E' \leftarrow E' \cup \{I\}$ 
10:       $R_r \leftarrow R_r \cup \{(c, I) \mid c \in \text{body}(I)\}$ 
11:       $R_i \leftarrow R_i \cup \{(I, \text{head}(I))\}$ 
12:     end for
13:   end for
14: until  $E = E'$ 

```

[Algorithm 2.1](#) has significant redundancy. In theory, the unification process examines every pair in $r \times 2^C$ although just a few of them are legitimate. However, new instances to be generated always

depends on the facts generated just before it. There is a Datalog execution strategy called *semi-naive evaluation*, which is based on this observation and is computationally efficient. The semi-naive evaluation-based algorithm scales almost linearly in the resultant attack graph's size if we can assume that each unification cost is $O(1)$.

[Algorithm 2.1](#) has an important implication. It grows the set of facts C monotonically, and the set never decreases its size. It means that any condition once achieved by the attacker keeps forever. For example, if the attacker has compromised the gateway, the attack-graph-based analysis supposes the gateway is kept compromised then and after. This assumption of *monotonicity* is introduced by Ammann et al. [6].

One more thing to note is that Datalog programs cannot accept negation in general. We cannot write a rule such that “if the attacker NOT yet compromised the gateway”. Because the rules containing negation make the inference result depend on the order of unification. There are several proposed approaches for the Datalog with negation (See Ceri et al. [12] for detail).

2.2.5 Attack paths extraction

An *attack path* is a connected subgraph of an attack graph satisfying the following conditions:

1. It ends at one condition node (called a *goal condition*).
2. Every condition node in it has only one incoming edge at most.
3. Every exploit node in it has all the incoming edges shown in the original attack graph.

An attack path represents a sequence of steps that an attacker must take to achieve the goal condition. As the second condition above implies, the sequence shown by an attack path is without freedom of choice. Therefore, it corresponds to a definite sequence of attack steps.

The extraction of attack paths gives us useful information. The number of attack paths in an attack graph indicates the degree of freedom available to the attacker. The minimum length of attack paths implies the least required effort for the attacker to compromise the network. An attack graph is a compact, combined form of these attack paths. [Algorithm 2.2](#) enumerates all the attack paths in the given attack graph.

[Figure 2.2.5](#) shows two attack paths leading to c_9 extracted from the attack graph in [Figure 2.2.3](#).

As this example shows, an attack path is not necessarily a chain of nodes. It is a directed acyclic graph. Besides, in general, the number of attack paths in an attack graph could be extremely larger than the size of the attack graph. Thus, [Algorithm 2.2](#) is not a generally feasible tool to evaluate the security of the concerned network. For network security evaluation, we will discuss more sophisticated ways in [Section 2.4](#).

2.2.6 Discussions

The explanation above simplifies various aspects to focus on the key concepts. However, the omitted details contain various issues relevant to the practical application of attack graphs. Also, there are different forms of attack graphs other than exploit dependency graphs suitable for specific application scenarios. The following subsections discuss them.

Algorithm 2.2 An algorithm for attack path extraction**Input:**

- C_g : the set of goal conditions.
- R_i : the set of imply relations.
- R_r : the set of require relations.

Output:

the set of attack paths, each path in which is represented as a set of exploits.

```

1: function ENUMERATE-ATTACK-PATHS( $C_g, R_i, R_r$ )
2:    $\mathcal{E} \leftarrow \{\{e \mid (e, c) \in R_i, c = g\} \mid g \in C_g\} \setminus \{\emptyset\}$ 
3:   if  $\mathcal{E} = \emptyset$  then
4:     return  $\{\emptyset\}$ 
5:   else
6:      $R'_i \leftarrow \{(e, c) \mid (e, c) \in R_i, c \notin C_g\}$ 
7:      $\mathcal{E}' \leftarrow \emptyset$ 
8:     for  $(e_1, e_2, \dots) \in \prod_{E \in \mathcal{E}} E$  do
9:        $E' \leftarrow \{e_1, e_2, \dots\}$ 
10:       $C'_g \leftarrow \{c \mid (c, e) \in R_r, e \in E'\}$ 
11:      for  $E'' \in \text{ENUMERATE-ATTACK-PATHS}(C'_g, R'_i, R_r)$  do
12:         $\mathcal{E}' \leftarrow \mathcal{E}' \cup \{E' \cup E''\}$ 
13:      end for
14:    end for
15:    return  $\mathcal{E}'$ 
16:  end if
17: end function

```

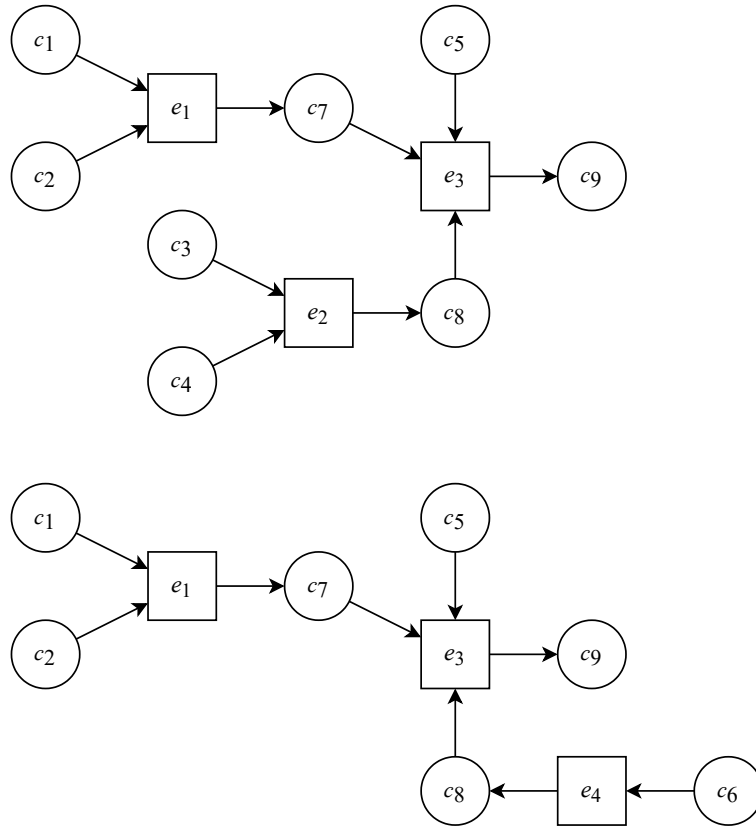


Figure 2.2.5: The extracted attack paths

2.2.6.1 Computational complexity

In general, the computational complexity of the state-of-the-art algorithm for attack graph generation is linear in the resultant attack graph's size. MulVAL is one of the most efficient tools which accomplish that performance. However, an attack graph's size depends on various factors, including the number of hosts, network topology, and attack templates. It is difficult to estimate the size of an attack graph precisely, thus the computational complexity too. For example, if the network has a ring structure, then the number of hosts and the edges are equal. In contrast, if the network forms a complete graph, then the number edges are approximately one-half of the square of the number of hosts. Such a difference in the network size also affects the size of the attack graph to be generated. Nevertheless, in the literature, the computational complexity is usually shown as a function of the number of hosts on a network.

Let us exemplify a few. In [6], the authors evaluate the computational complexity of the algorithm to be $O(A^2E)$, where A is the number of attributes (facts) and E is the number of exploits that might be performed. However, this complexity is evaluated to be $O(N^6)$ in Ou et al. [119], where N is the number of hosts on a network. Because the attribute in Ammann's algorithm is supposed to express the connectivity among hosts, and if we assume that the network topology is a complete graph, then the total number of attributes is $O(N^2)$. Additionally, in a complete graph, each node can be compromised from every other node. Thus the number of exploits is $O(N^2)$. In short, this evaluation is based on a worst-case scenario. Compared to that, the worst-case computational complexity of MulVAL is evaluated to be $O(N^2)$ in Ou et al. [119]. This estimation comes from the unification characteristics and the maximum number of variables appearing in the rules. Furthermore, the form of attack templates is different from Ammann's. In another more recent study, Kaynar and Sivrikaya [79] propose a distributed algorithm for attack graph generation. Their algorithm's computational complexity is evaluated to be $O(N^2/\log(N))$, where N is the number of hosts. It is not the worst-case complexity, and the algorithm processes an intentionally limited form of attack templates. It is inappropriate to compare these different estimations directly. As this paragraph illustrates, we need to clarify each complexity estimation's premises when comparing the algorithms.

Broad surveys in the field of attack graph analysis are given in [90, 142, 52]. The survey in [90] refers to studies before 2005, while important progress was made since then. The description in Singhal and Ou [142], Hong et al. [52] refers to more recent efforts. While these surveys are insufficient in considering computational complexities, they are a valuable source of information to examine the progress of attack graph analysis studies, including its performance improvement.

2.2.6.2 Cyclic structures

The attack graph in [Figure 2.2.3](#) contains a cyclic structure—the loop composed of e_3 , c_9 , e_4 , and c_8 . This does not mean that the attacker infinitely realizes that cycle. It is a natural consequence of merging overlapping attack paths. Cycles are inherent in attack graphs and could arise even in a network with non-cyclic topology. However, those cycles cause difficulty in network security evaluation and optimization. We will discuss this topic in [Section 2.4](#) and [Section 2.5](#).

2.2.6.3 Attack template preparation

The attack graph generation process depends on the set of attack templates defined as inference rules. Each attack template represents the knowledge about how an attacker exploits vulnerabilities. Therefore, we have to maintain the set of those attack templates up-to-date to make our analysis realistic.

The primary source of vulnerability information is National Vulnerability Database (NVD). Roschke et al. [130] use a rule-based approach to extract information from the textual description in NVD. Later, Aksu et al. [2] examine machine-learning-based approaches to generate attack templates. They report that the machine-learning method outperforms the simple rule-based approach. Although this research field is still immature, it seems promising to leverage the recent epochal progress in artificial intelligence research.

2.2.6.4 Reliability of inference

Sommestad and Sandström [143] have verified the attacks inferred by MulVAL. They have performed penetration tests by experts on the real network containing 199 machines and compared the result with generated attack graphs. The prediction accuracy of the attack graph analysis is reported to be poor. “Of the 119 machines MulVAL predicted as impossible to compromise, 34 (29 per cent) were successfully compromised as root. In 3 of these 34 cases, MulVAL reported the machine as reachable but failed to describe a path that included the exploited vulnerability.” The reasons for this mis-analysis are identified as (1) the lack of attack templates for several vulnerabilities and (2) the weakness in the default rules given in [116]. As this report tells, the reliability or credibility of the attacks inferred by attack graphs depends on the facts and rules we prepare.

2.2.6.5 Attack expressiveness

As mentioned just before, the coverage of inferable attacks depends on the variety of attack templates in use. That is the expressible variety of attacks by attack templates matters. In Datalog, rules represent attack templates, and predicates are their skeleton. Carefully designed predicates enable not only complex attack inference but also impact analysis.

Bacic et al. [8] expand the default rules given in [116] to model the network infrastructure, including subnets and routers. Furthermore, their effort includes the modeling of risk aspects such as confidentiality, integrity, and availability. Froh and Henderson [38] advance that effort to accommodate the dependency between services and their infrastructure. Stan et al. [149] propose a suite of Datalog predicates and rules based on the OSI reference model, i.e., their suite can infer the attack happens in the lower layer, such as the data-link layer. Their study also contains a detailed evaluation report of their rules on a physically constructed testbed. Inokuchi et al. [60] discuss a process model for designing practical sets of predicates, rules and facts.

From the theoretical point of view, Saha [134] points out that, in some cases, access control modeling requires negation, and it is processible with stratified Datalog. The study contains an efficient incremental algorithm for attack graph update, too. Johnson et al. [77] propose a domain-specific language named Meta Attack Language (MAL). At a glance, MAL is very different from Datalog. For example, it can define class hierarchies, the negation of initial conditions, and relation types among nodes. However, most of it can be interpreted as a syntax sugar to stratified Datalog.

Type	Space complexity	Feature
State-enumeration graph	$O(2^{MN})$	- A kind of finite automata. - Strong relevance to Markov chains. - Poor scalability.
Exploit-dependency graph	$O(N^2 + MN)$	- A kind of AND/OR graphs. - Strong relevance to Datalog. - Good scalability.
Host-based attack graph	$O(N)$	- A kind of network model. - Detail is specific to each study. - In general, highly scalable.

Table 2.1: Three major forms of attack graphs

All the above indicates the powerful expressiveness of Datalog and the importance of the exploit-dependency graph since its AND/OR combinations of nodes are the reflection of the description structure in Datalog.

2.2.6.6 Other forms of attack graphs

While state enumeration graphs are intractable due to their space complexity, there are many useful theoretical tools for them. They are a kind of finite automata. If we assign probabilities to edges, it becomes Markov chains. In more complex scenarios, we can use various theories like Hidden Markov Model (HMM), Markov Decision Process (MDP), Partially Observable Markov Decision Process (POMDP), and so on. Jha et al. [73] have clarified the fundamental properties of state enumeration graphs. Historically, this form is also called full attack graph or complete attack graph.

Another popular definition is host-based attack graph. In this form, usually, a node corresponds to a host, a graph describes the network topology, and a node is regarded as a set of its vulnerabilities. Its generation scales well, but the expressiveness is limited; it ignores the dependencies among exploits and vulnerabilities within a host. The studies by Ammann et al. [7] and Ingols et al. [58] are the well-cited ones based on this form. Serra et al. [139] propose a Mixed Integer Linear Programming based algorithm for multi-objective optimization for its security controls. Khouzani et al. [80] also examine a min-max multi-objective optimization algorithm with probabilities assignment.

The three variations, state-enumeration graph, exploit-dependency graph, and host-based attack graph, comprise major categories of attack graph forms. Table 2.1 summarizes the characteristics of these major forms of attack graphs, where N denotes the number of machines in the network and M is the number of vulnerabilities per machine.

One can even find a hybrid of these forms. For example, the Hierarchical Attack Representation Model (HARM) proposed by Hong and Kim [51], Ge et al. [41] is a hybrid of host-based attack graph and exploit dependency graph. In short, HARM models the inside of each host as an attack tree and treats the network topology as a general directed graph. This mixture enables scalability in applications and moderately limited but still rich expressiveness.

Each definition has its strengths and weaknesses. The prominent characteristics of exploit dependency graphs are (1) the powerful expressiveness explained just before and (2) the practical scalability compared with state enumeration graphs. However, as we will see in the following sections, it has several obstacles in handling security metrics and optimization. The optimal choice of definitions depends on its use case. This review sticks to the exploit-dependency graph because it is the most

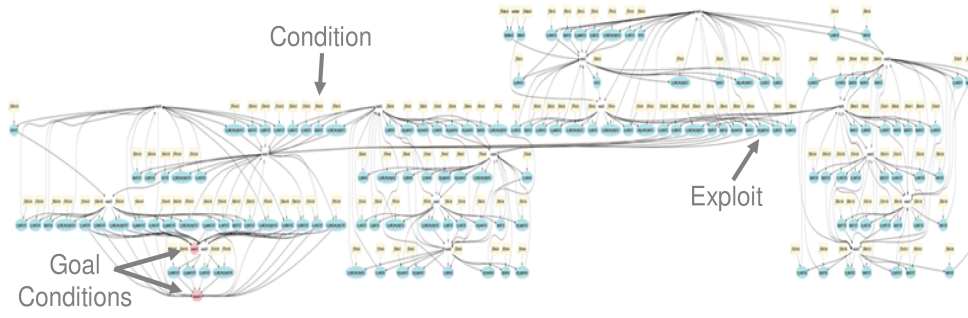


Figure 2.3.1: Non-aggregated attack graph (quoted from Noel and Jajodia [110])

well-studied form in the literature.

2.3 Visualization

Figure 2.3.1 shows an example of “small” attack graph. How can we grasp the implication about the security risk from Figure 2.3.1? It is not uncommon in practice to see an attack graph composed of thousands of nodes. It is impractical for humans to extract the risk implication directly from an attack graph in raw representation.

Several researchers proposed specially tailored visualization techniques for attack graphs. They provide a compact form of attack graph representation and help us comprehend. We take a look at them in this section.

2.3.1 Information reduction

Wei Li and Vaughn [175] examine a condition by which we can aggregate a subgraph in an attack graph into a node and restore the original subgraph from an aggregated node. For example, if two exploits share the same preconditions and postconditions, we can merge them into one exploit with new labeling indicating the original two exploits without losing information. This approach makes the detailed structure of an attack graph compact. However, the original attack graph is assumed to be represented in a form called E-graph in which condition nodes are omitted and only exploit nodes are shown.

Noel and Jajodia [110] propose an intuitive, useful hierarchical aggregation technique. Their approach defines several rules to aggregate condition and exploit nodes in an attack graph. The following are the major aggregation rules:

- Machine: Sets of conditions are aggregated into a machine abstraction if they are all conditions on that machine.
- Exploit set: Sets of exploits are aggregated into those with the same pair of (possibly identical) attacker/victim machines. That is, only exploits between the same pair of machines are aggregated.
- Protection domain: The protection domain abstraction represents a set of machines that have unrestricted access to one another’s vulnerabilities, so that the attack graph is fully connected among them (if one treats exploit sets between a pair of machines as graph edges).

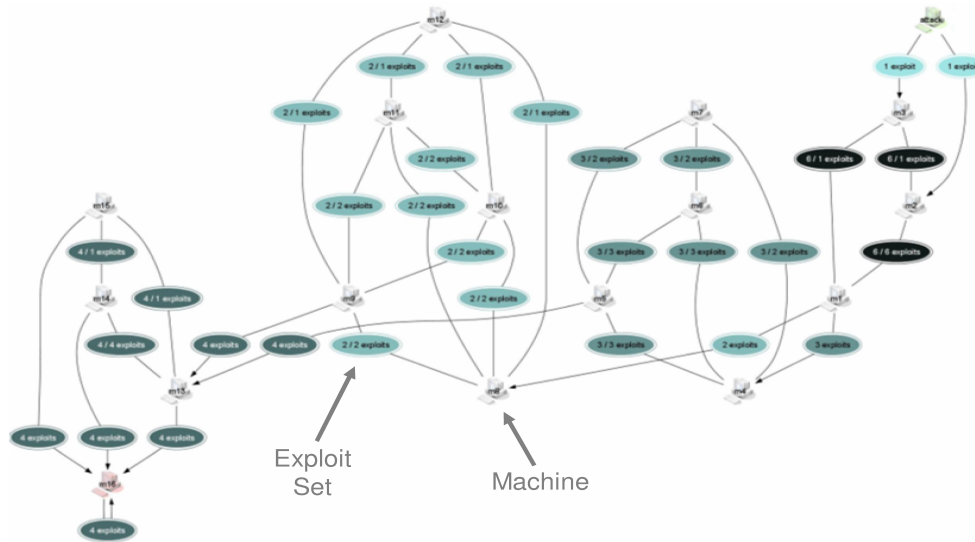


Figure 2.3.2: Aggregation to machines and exploit sets (quoted from Noel and Jajodia [110])

Figure 2.3.2 and Figure 2.3.3 show the example of aggregated forms of Figure 2.3.1. The technique dramatically reduces the number of nodes in an attack graph and helps us understand the whole picture.

The aggregation in Figure 2.3.2 and Figure 2.3.3 leverages that every exploit is bound to some machine or subnet in the concerned network. Thus, we can merge the set of exploits bound to the same region into one node in the aggregated attack graph. We can point to the most concerning part of an attack graph and dig into its detail with this aggregated form.

Similarly, Homer et al. [48] propose to depict the graph as a network of connected machines and not to show the condition/exploit nodes. The edges in the graph represent abstracted attack paths between machines. Furthermore, “useless” attack paths are omitted that do not provide the privilege escalation valuable to the attacker. Lastly, exploit nodes shared among multiple attack paths are explicitly shown in the graph. Because such exploit nodes are good candidates for applying countermeasures.

2.3.2 Aspect oriented view

Not every piece of information provided by attack graphs is relevant to the administrator’s concern. Irrelevant information could be noisy for comprehension. Visualization methods focusing on specific aspects of attack graphs are useful in such cases.

Noel and Jajodia [111] propose to use the adjacency matrix of an attack graph to compute the reachability among vertices. The key idea is that the adjacency matrix’s multiplication produces another matrix in which the non-zero numbers imply the reachability between vertices on the corresponding graph. Their algorithm computes the transitive closure of the matrix and generates a bitmap image to visualize the connectivity among vertices.

Chu et al. [15] use treemaps to show the targeted area and the depth of attacks where attack graph nodes are categorized into predefined areas corresponding to subnets. The size and color of rectangles in the treemap is leveraged to express the compromised privilege and the value of compromised assets. This visualization emphasizes the severity of the presumed attack on essential assets.

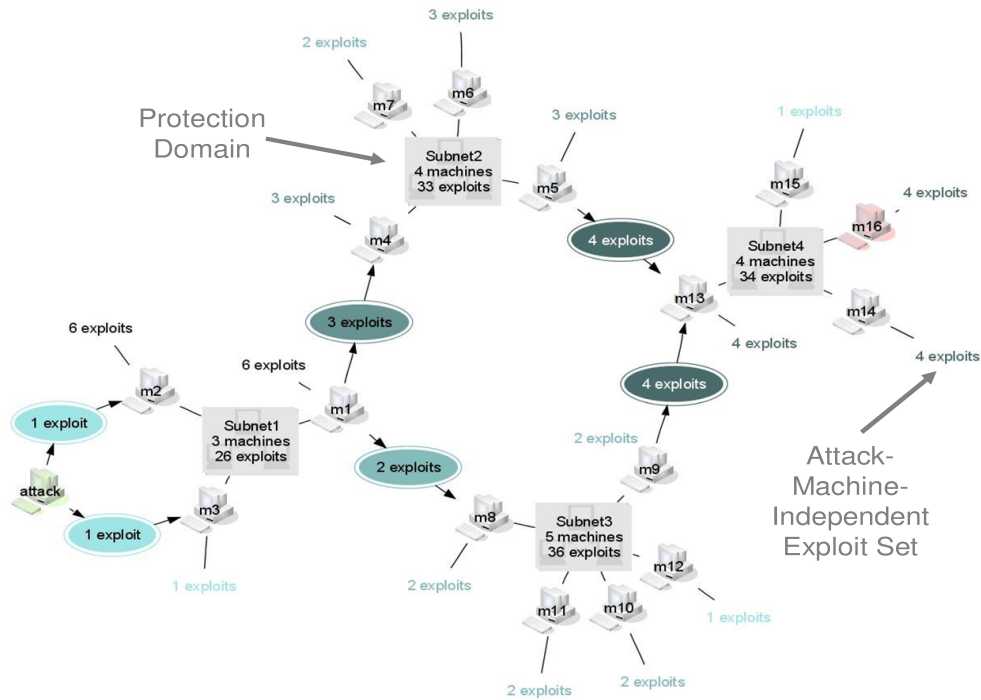


Figure 2.3.3: Attack graph aggregated to protection domains (quoted from Noel and Jajodia [110])

2.3.3 Visual syntax

There is no formal standardization for the attack graph representation. Should we depict exploit nodes by squares or by circles? Does the difference in depiction affect the perception by network administrators? Lallie et al. [86] focus on this topic and clarify the undesirable breadth of variety in visual syntax for attack graph representation. The need for visual syntax standardization and further research for it is discussed. This survey is also an exhaustive analysis of the past attack graph studies to date. Their previous study [85] examines the perceptual difference between attack graphs and attack trees (fault trees).

2.4 Network security metrics

An attack graph is too detailed for decision-makers who need to know the whole picture of their IT infrastructure's risk. We can use network security metrics instead. Attack-graph-based network security metrics are concise, numerical indicators expressing the various aspects of the risk landscape.

Here we categorize and explain the attack-graph-based network security metrics. Firstly, we distinguish whether the metric is a probabilistic indicator or not. Secondly, the applicability of the metrics to cyclic attack graphs is concerned. The product of these two categorizations leads to the following four subsections.

Note that the metrics explained below are chosen to provide a comprehensive view of attack-graph-based security metrics design issues and approaches. Even though some of them are specific to a particular form of attack graphs, the idea behind them applies to other cases.

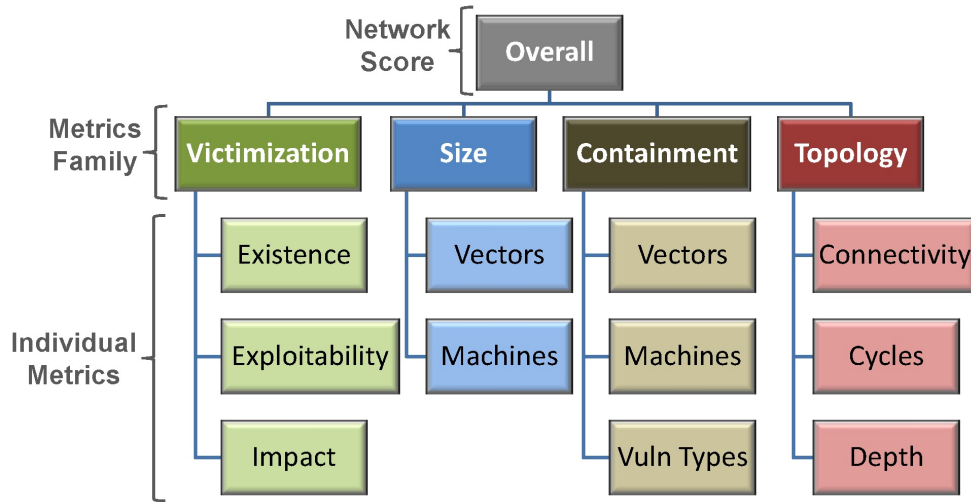


Figure 2.4.1: Attack graph metrics families (quoted from Noel and Jajodia [112])

2.4.1 Non-probabilistic metrics for cyclic graphs

In the most general settings, attack graphs are supposed to be cyclic, and there is no statistical data. In such cases, we cannot calculate probabilities. The metrics reveal qualitative aspects of network security.

The *Network Compromise Percentage (NCP)* in [89] indicates the percentage of network assets an attacker can compromise. This metric shows the range of possible compromise. The *Weakest Adversary (WA)* metric by Pamula et al. [120] gauges the minimum set of initial conditions required for the attacker to compromise the network. This metric has been introduced as a metric for state-enumeration graphs. We omit the detail here.

Gonda et al. [44] focus on graph centrality, a measure indicating the relative importance of a node in a graph. For example, the number of incoming and outgoing edges of a node is called a degree centrality. If the centrality gets greater, then the dependence of attack paths going through the node increases. Their study examines several types of graph centralities and compares their relevance to shortest attack paths. Also, their study proposes to convert an exploit dependency graph into a planning graph, which is assured to be acyclic.

Noel and Jajodia [112] propose a suite of metrics applicable to general attack graphs. Their metrics are categorized into four families composed of eleven metrics in Figure 2.4.1. We pick up some of the metrics and dig into the detail.

Let us clarify the implicit assumption for these metrics. All metrics are designed to be within the scale of $[0, 10]$. The larger value implies less security. Also, the network is supposed to be the connected graph of strongly connected components that every host can connect to each other within it. We call each strongly connected component a protection domain (See Section 2.3.1). A protection domain usually corresponds to a subnet in the network.

Victimization family metrics are the simple aggregation of the count and CVSS scores of the vulnerabilities in an attack graph. For example, the *Exploitability* metric is the average value of the CVSS Exploitability score (the relative ease of exploitation), averaged over all vulnerabilities over all

hosts. The metric $m_{\text{exploitability}}$ is defined as

$$m_{\text{exploitability}} = \frac{\sum_i^{|U|} \text{CVSS}_{\text{Exploitability}}(u_i)}{|U|},$$

where $\text{CVSS}_{\text{Exploitability}}(u_i)$ denotes the CVSS Exploitability score for vulnerability u_i , and U denotes the set of all vulnerabilities in the attack graph.

The *Size* family metrics measure risk in terms of the attack graph size since the larger graph implies the more ways available to the attacker. The *Attack Vectors* metric in this family is the number of exploits relative to the total possibilities for the network. The exploits are categorized into two groups: implicit and explicit. An explicit exploit is a step of attack performed between different protection domains, and an implicit exploit is a step performed within a protection domain. The metric $m_{\text{attackVectors}}$ is defined as follows:

$$m_{\text{attackVectors}} = 10 \sqrt{\frac{v_a}{v_p}},$$

where v_a and v_p are defined as follows:

$$v_a = \sum_i^d (m_i - 1) \sum_j^{m_i} v_{\text{implicit}}^{i,j} + \sum_{i,j}^d v_{\text{explicit}}^{i,j},$$

$$v_p = (m - 1) \sum_i^m s_i.$$

In this definition,

- d : the number of protection domains,
- m : the number of machines in the network,
- m_i : the number of machines contained in the i -th protection domain,
- $v_{\text{implicit}}^{i,j}$: the number of implicit exploits in the j -th machine on the i -th protection domain,
- $v_{\text{explicit}}^{i,j}$: the number of explicit exploits across i -th and j -th protection domains,
- s_i : the number of exploits in the i -th machine in the network.

The value v_a means the number of attack vectors, and v_p means the potential number of attack vectors. Note that this definition is based on the assumption that each implicit exploit can be invoked with every other machine in the same protection domain. This assumption introduces the term “ $(m_i - 1)$ ” in the definition. Similarly, in the worst case, every exploit can be invoked by access from every other machine. This is the reason why the term “ $(m - 1)$ ” appears on the right-hand side of v_p .

The *Containment* family measures the degree to which the attack graph contains attacks across network protection domains. The *Vulnerability Types* metric is the number of unique vulnerability types referred to by explicit exploits, relative to the total number of vulnerability types across the entire attack graph. This metric implies that multiple instances of the same vulnerability type are

less costly to mitigate. The metric $m_{\text{vulnerabilityTypes}}$ is defined as follows:

$$m_{\text{vulnerabilityTypes}} = 10 \frac{t_a}{t_a + t_w},$$

where t_a denotes the number of unique vulnerability types referred by explicit exploits, and t_w denotes the number of unique vulnerability types referred by implicit exploits.

The last family of metrics is the *Topology* family. It captures the graph-theoretic property of an attack graph reduced to the protection domain level. The key intuition behind this is that the network services should be kept separate as possible. Suppose we have two vulnerable services on a network. Even in such a situation, the whole network can be secure if there is no access route between the two. The *Cycles* metric in this family is the number of strongly connected components in the domain-level attack graph, relative to the best (most secure) cases possible. In the Cycles metric, strongly connected components are connected by directed edges and might have one or more directed routes in between. The metric m_{cycles} is defined as follows:

$$m_{\text{cycles}} = 10 \left(1 - \frac{w_{\text{strong}} - 1}{d - 1} \right),$$

where w_{strong} denotes the number of strongly connected components. Note that the number of domains d corresponds to the best-case, for it corresponds to the situation where no strongly connected domains.

The use of these metrics requires careful consideration. Each metric captures a specific aspect of network security. The improvement in the metrics does not necessarily mean the improvement of security. For example, software patching can reduce the Exploitability metric, while it does not guarantee to eliminate attack paths compromising some critical assets.

2.4.2 Non-probabilistic metrics for acyclic graphs

The metrics in the previous subsection ignore the attack detail. Suppose we have two attack graphs with the same graph structure. The difference exists only in the CVSS scores of the vulnerabilities contained in them. Then, most of the metrics show the same scores. One possible way to overcome this weakness is to enumerate attack paths and then examine those details. It means that we will see an attack graph as a combined set of attack paths, i.e., acyclic subgraphs.

2.4.2.1 Combination of major metrics and assistive metrics

Idika and Bhargava [56] propose to use multiple metrics derived from attack paths. They define two categories of metrics: Decision metrics and Assistive metrics. Most of the metrics are calculated from the extracted attack paths. Let us see the detail of their approach.

As preparation, let G be the attack graph concerned, n be the number of attack paths in G , each p_k for $k = 1, 2, \dots, n$ be the attack path extracted from G , and the function $l : P \rightarrow \mathbb{N}$ be the length of the given attack path.

The *Decision* metrics are composed of five metrics: Shortest Path, Number of Paths, Normalized Mean of Path Lengths, Network Compromise Percentage, and Weakest Adversary metric.

The *Shortest Path (SP)*, *Number of Paths (NP)*, *Normalized Mean of Path Lengths (NMPL)*

metrics are defined as follows:

$$\begin{aligned} SP(G) &= \min \{l(p_1), l(p_2), \dots, l(p_n)\}, \\ NP(G) &= n, \text{ and} \\ NMPL(G) &= \frac{\sum_i l(p_i)}{NP(G)^2}. \end{aligned}$$

The definition of NMPL comes from the observation that a simple averaged length of paths fails to consider the number of paths. Suppose we have two attack graphs, each of which has the same averaged length of paths while the number of paths is considerably different. It is inappropriate to think that the risk of both is the same. The additional division by $NP(G)$ metric is intended to reflect the difference.

The calculation and comparison of Decision metrics tell us some aspects of the security of the network. But, how should we determine which of G_1 and G_2 is more secure when both of $SP(G_1) \leq SP(G_2)$ and $NP(G_1) \leq NP(G_2)$ hold? In that case, Assistive metrics solve those conflicts.

The *Assistive* metrics are composed of four metrics: *Mean of Path Lengths (MPL)*, *Standard Deviation of Path Lengths (SDPL)*, *Mode of Path Lengths (MoPL)*, and *Median of Path Lengths (MePL)* metric. These are the standard set of well-known statistics of the path lengths.

In conjunction with the Decision and the Assistive metrics, the algorithm by Idika and Bhargava [56] defines a decision flow to determine which of G_1 and G_2 is better or worse. The final comparison result is either “all are equal”, “strictly dominated”, “majority dominated”, or “incomparable”. Their study reports that 99.9% of the experimental cases are concluded to be other than “incomparable”. Within the result, 99% of the cases are concluded only with the comparison of Decision metrics. Only 0.9% of cases resort to Assistive metrics. This approach implies that the insufficiency of a suite of metrics can be complemented with the composite use of additional metrics such as Assistive metrics.

2.4.2.2 Composition of metrics for acyclic subgraphs

An attack path can also be a subgraph of another attack path. In general, an attack path is a composition of its sub-paths. This fact brings a question of how we can define a consistent metric for the composite attack path if we have metrics defined for its sub-paths.

Wang et al. [167, 168] propose the following three principles for security metrics:

1. A longer path leading to the attack goal means better security.
2. A metric should never yield a value that is greater than the smallest attacking effort required for reaching the attack goal.
3. Multiple attack paths together are less secure than any of the paths alone.

Starting from the natural, convincing principles above, let us define a metric named *attack resistance* of node x in an attack graph $G = (E \cup C, R_r \cup R_i)$; remind that E denotes the set of exploit nodes, and C denotes the set of condition nodes. An attack resistance represents the difficulty or effort for the attacker to realize the condition/exploit of the given node. Thus, the larger the resistance, the more secure the network is. Let us introduce several functions as preparation. The “parent” function

is the mapping from a node in an attack graph to the set of parent nodes and is defined as follows:

$$\text{parent}(x) = \begin{cases} \{c \mid (c, x) \in R_r\} & (x \in E) \\ \{e \mid (e, x) \in R_i\} & (x \in C) \end{cases}.$$

Let $\mathbb{D}$ be a *domain* with total ordering, such as $\mathbb{N}$ and $\mathbb{R}$. The *choice operator* $\oplus$ is an arbitrary function $\oplus : \mathbb{D} \times \mathbb{D} \rightarrow \mathbb{D}$ satisfying that $r_1 \oplus r_2 \leq \min\{r_1, r_2\}$ for any $r_1, r_2 \in \mathbb{D}$, which denotes the composition of paths at a condition node. Its characteristic also corresponds to the second principle above. The *bind operator* $\otimes$ is an arbitrary function $\otimes : \mathbb{D} \times \mathbb{D} \rightarrow \mathbb{D}$ satisfying that $\max\{r_1, r_2\} \leq r_1 \otimes r_2$, which denotes the composition of paths at an exploit node. It corresponds to the third principle. The *individual resistance* of node x is a function $\rho : (E \cup C) \times \mathbb{D} \rightarrow \mathbb{D}$ satisfying that $r \leq \rho(x, r)$ for any combination of $x \in E \cup C$ and $r \in \mathbb{D}$, corresponding to the first principle. With these sub-functions, now we define the attack resistance $R : (E \cup C) \rightarrow \mathbb{D}$ as follows:

$$R(x) = \begin{cases} \rho\left(x, \bigoplus_{e \in \text{parent}(x)} R(e)\right) = \rho\left(x, R(e_1) \oplus \cdots \oplus R(e_{|\text{parent}(x)|})\right) & (x \in C) \\ \rho\left(x, \bigotimes_{c \in \text{parent}(x)} R(c)\right) = \rho\left(x, R(c_1) \otimes \cdots \otimes R(c_{|\text{parent}(x)|})\right) & (x \in E) \end{cases}.$$

(This definition is adapted from that in Wang et al. [167, 168]).

The above definition of attack resistance has some freedom in defining sub-functions, such as choice and bind operator and the individual resistance. It suffices for each sub-function to follow the constraint introduced above. As long as the constraints are observed, the derived attack resistance is guaranteed to follow the three principles. For example, the two operators, $r_1 \oplus r_2 = \min\{r_1, r_2\}$ and $r_1 \otimes r_2 = \max\{r_1, r_2\}$, are compliant with the constraints. In another formulation, $r_1 \oplus r_2 = \frac{r_1 r_2}{r_1 + r_2}$ and $r_1 \otimes r_2 = r_1 + r_2$ are also legitimate definition, and is analogous to the resistance in electric circuits. The definition discussed above is rather a framework for designing the specific metric compliant with the natural principles.

2.4.3 Probabilistic metrics for cyclic graphs

We can apply the probabilistic graphical model, such as Markov chain and Bayesian network, to attack graphs and calculate the various probability assigned to the nodes. The Markov chain model applies to cyclic graphs, while Bayesian network cannot. However, the existence of AND combination of preconditions inhibits the simple application of Markov chain model. In this subsection, we discuss this issue and a limited solution for it.

2.4.3.1 State rank for state-enumeration graph

Mehta et al. [92] propose a metric inspired by the PageRank algorithm by Google. Their metric defines the rank of states in a state enumeration graph. A state enumeration graph G is a 3-tuple (S, τ, s_0) , where S is the set of states, $\tau \subseteq S \times S$ is the set of transitions, and $s_0 \in S$ is the initial state. A probabilistic attack graph G' is a 3-tuple $(S, P, \mathbf{s})$, where $|S| = n$ for some $n \in \mathbb{N}$, P is a state transition matrix of size $n \times n$ in Markov chain, and $\mathbf{s} = (s_1, s_2, \dots, s_n)^\top \in [0, 1]^n$ is the initial state vector. Note that the content of P and $\mathbf{s}$ reflects that of τ and s_0 . For the probabilistic attack graph, we can calculate the vector $\mathbf{r}^{(m)}$ as follows, which denotes the set of probabilities of each state

to be activated after the m -steps of transitions:

$$\mathbf{r}^{(m)} = \sum_{n=0}^m P^n \mathbf{s}.$$

This equation derives from the property of Markov chain. In addition, we assume that the number of steps that the attacker takes to perform attacks follows a geometric distribution with the “failure” probability η on each trial. With these settings, we get the following formula which computes the reachability probability vector $\mathbf{r} = (r_1, \dots, r_n)^\top \in [0, 1]^n$ as follows:

$$\mathbf{r} = \frac{1 - \eta}{\eta} \sum_{m=1}^{\infty} \eta^m \sum_{n=0}^m P^n \mathbf{s}.$$

Each number in $\mathbf{r}$ denotes the probability, i.e., the rank, of the corresponding state to be reached by the attacker.

The formulation above is straightforward, and the Markov chain model applies without problems. However, state enumeration graphs are intractable for its state explosion. We need another method applicable to exploit-dependency based attack graphs.

2.4.3.2 Asset rank for exploit-dependency graph

In exploit dependency graphs, an exploit requires a set of preconditions to be realized simultaneously. The Markov chain cannot handle such a simultaneous activation of states. We need a twist to handle those AND combinations of preconditions.

Sawilla and Ou [137] propose a probability-like metric named AssetRank, which can handle exploit dependency graphs. AssetRank is a vector in $[0, 1]^n$. Each of the values represents the preference for an attack graph node by the attacker. The higher value means that the attacker is eager to achieve the corresponding exploit or condition. In a broad sense, we can regard each value in an AssetRank as a proxy indicator of the reachability probability by the attacker to the corresponding node.

The key intuition behind AssetRank calculation is the use of multi-agent simulation. Suppose we have an attack graph with a goal condition, and the attacker tries to get that condition. We put an agent on this goal initially. This agent can clone and distribute itself to the adjacent nodes. The cloned, distributed agents have the same ability. We assume agents’ distribution propagate from postconditions to its source exploits and from exploits to its preconditions. Thus, the agents spread over the graph as time goes on. Along with the process, some of the agents die out at each node at each step. We call the death rate of agents per step as damping factors. We use the damping factors assigned to nodes to express the difference between AND and OR combinations of conditions. Also, at each step, we feed a fixed number of agents continuously at the goal condition. After the passage of a sufficiently long time, each node is supposed to hold a stable number of agents. When the count of agents at a node is high, it implies that the node is a better place for the agents to stay, i.e., preferable. The numbers of agents at each node normalized to be within $[0, 1]$ compose the AssetRank. See [137] to confirm the detail of the definition.

AssetRank is similar to the state rank defined in the previous subsection in the sense that both approaches regard an attack graph as a kind of state transition model. The computation of AssetRank is based on the concept of multi-agent simulation, while the latter uses the Markov chain model.

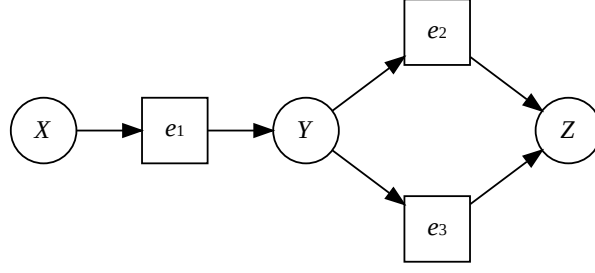


Figure 2.4.2: An example of non-independent scenarios

Nevertheless, the resultant ranks have similar semantics. The higher the rank, the attacker is likely to achieve the corresponding condition or exploit. Though we cannot use AssetRank as the actual probabilities, we can still compare the probabilistic degree of nodes to be compromised.

2.4.4 Probabilistic metrics for acyclic graphs

If the concerned attack graph is acyclic, we can apply a more straightforward approach to calculate the probability assigned to each node.

Let G be an attack graph $G = (E \cup C, R_r \cup R_i)$, and “parent” be the function defined in [Section 2.4.2.2](#). Besides, let us define a function $p : (E \cup C) \rightarrow [0, 1]$, which assign individual scores to each node in G . With these functions, we define the probability P of the event implied by an exploit e or a condition c in G as follows (Be cautious in distinguishing the use of P and p , and $P(e)$ means $P(\text{event } e \text{ occurs})$):

$$P(e) = p(e) \prod_{c \in \text{parent}(e)} P(c), \text{ and}$$

$$P(c) = \begin{cases} p(c) & (\text{parent}(c) = \emptyset) \\ p(c) \left\{ 1 - \prod_{e \in \text{parent}(c)} (1 - P(e)) \right\} & (\text{parent}(c) \neq \emptyset) \end{cases}.$$

This definition follows Wang et al. [169] and is intuitive and natural. Essentially, this form of probability follows the framework in Wang et al. [168].

Unfortunately, the probability calculations based on the above definition do not apply to cyclic attack graphs, as the cyclic structure leads to an infinite loop. Another important caveat is that it is implicitly assumed that the events represented by nodes are independent of each other.

In [Figure 2.4.2](#), suppose every individual score, such as $p(X)$ and $p(e_2)$, equals to 0.5. Then, the probability $P(Z)$ is calculated as follows:

$$P(Y) = p(Y) P(e_1) = p(Y) p(e_1) P(X) = 0.5^3,$$

$$P(e_2) = p(e_2) P(Y) = 0.5^4,$$

$$P(e_3) = p(e_3) P(Y) = 0.5^4,$$

$$P(Z) = p(Z) \{1 - (1 - P(e_2))(1 - P(e_3))\} \\ \simeq 0.0605.$$

However, two attack paths reach Z via e_2 and e_3 , and they are identical until halfway through. When one of the two occurs, the other is also likely to occur. The correct probability for Z , considering that they are not independent, is as follows:

$$\begin{aligned} P(Z) &= p(Z) \{1 - (1 - p(e_2))(1 - p(e_3))\} P(Y) \\ &\simeq 0.0469. \end{aligned}$$

As can be seen from this example, the event probabilities will be different depending on whether we assume each event's independence. The assumption of event independence implies that an attacker will only try one attack path at a time. In practice, the attacker will try numerous attack paths. We would say that the assumption of independence is an oversimplification.

Ou and Singhal [117], Homer et al. [50] propose an algorithm that correctly computes the probabilities of the events, assuming that they are not independent. This algorithm is very similar to that of Bayesian networks. We do not get into the details of their approach. Instead, we will discuss approaches based on Bayesian networks in [Section 2.6](#).

2.4.5 Discussions

The following subsections discuss several topics important in practice and further applications not mentioned above.

2.4.5.1 Computational complexity

For the non-probabilistic metrics, the metric by Idika and Bhargava [56] is intuitive but potentially very costly. This is because their metrics enumerate attack paths implicitly. The enumeration of attack paths can cause a computational explosion. In comparison, Noel and Jajodia [112] design the metrics carefully to keep the computational complexity within a minor range. However, it is not a simple task to use the metrics to understand the true nature of risks.

At first glance, the computational complexity of probabilistic metrics in this chapter seems relatively tractable; the computational complexity of StateRank and AssetRank is dominated by the sum of products computation performed for matrices, which is $O(N^3)$ for the size $N \times N$ of the matrices. Note that the size N is equal to the number of nodes in the attack graph. The metric proposed by Wang et al. [169] can be calculated even more efficiently by dynamic programming. The required computation scales linearly. Though, this metric assumes independence between events, which is an oversimplification of reality. At this point, the metric by Homer et al. [48, 50] helps. Unfortunately, it requires an exponential amount of computation in the worst case. After all, calculating accurate probabilistic metrics requires considerable computational effort.

2.4.5.2 Cycle handling

Wang et al. [169] apply to acyclic attack graphs. Although we did omit the detail, their method includes an algorithm to handle cycles in an attack graph. Similarly, several other studies, such as Noel et al. [115] and Homer et al. [48, 50], have proposed approaches that deal with cycles.

Noel et al. [115] propose to “resolve any attack graph cycles through their distance from the initial conditions”. More precisely, according to Noel et al. [113], “preference is given to dependency

edges that are closer to the initial conditions.” The implicit assumption behind this approach is that an attacker will only use the shortest attack paths. It could be problematic since shorter paths do not necessarily correspond to the paths with the least effort; it ignores the difficulty in performing exploits. At the very least, reducing the number of attack paths affects metrics such as Idika and Bhargava [56].

Wang et al. [169] propose an algorithm to eliminate cycles. When calculating the probability of reaching an exploit node e , consider an attack graph that removes the implicit edges going out from the exploit node e . We denote this subgraph as $A(G, e)$ where G is the whole attack graph before the edge removal. The subgraph $A(G, e)$ contains no cycles going through the exploit node e . We apply a probability calculation algorithm to this graph. The problem with this method is that $A(G, e)$ may still contain cycles that do not pass through the exploit node e . Therefore, we need to apply the cycle removal algorithm recursively. Consequently, depending on the order of these eliminations, the probabilities change because the obtained cycle-free graphs can vary. The algorithm by Wang et al. [169] uses breadth-first order, but its justification is not mentioned.

In the method proposed by Homer et al. [48, 50], roughly speaking, when a cycle is encountered during the probability calculation, the algorithm enumerates partial attack paths overlapping with the cycle. The individual paths enumerated are cycle-free, so probability calculations apply. The problem is that the number of extractable attack paths can explode.

Although the handling of cycles is not often discussed these days, it is one of the essential issues in attack graph analysis. In particular, the probabilistic interpretation of an attack graph requires decomposing cycles into attack paths up to some degree, which leads to a computational explosion.

2.4.5.3 Reliability of metrics

Non-probabilistic metrics are primarily qualitative; we can compare two metrics and argue which one is better, but we cannot say it is twice as safe just because it has twice the value. This means that we can choose the best of several options while it is impossible to calculate their cost-benefits. If the metrics are defined on a partially-ordered set, we cannot even choose the best option.

The probability value is a quantitative measure, thus, in theory, a preferable metric. The problem with using probabilistic metrics is that statistical data for input is generally unavailable. A probability derived from unreliable inputs cannot be reliable. However, it is possible to understand the risk trend even though the exact probability is not known. For example, Noel et al. [115] propose a risk assessment method using Monte-Carlo simulation with varying parameters.

Problems with the reliability of risk assessments are not limited to attack graphs and derived metrics; Verendel [161] has conducted a critical survey of security risk assessment studies that advocate quantitative assessments. The survey points out that the quantitative assessment of security risk is immature. While research efforts to overcome this challenge are indispensable, it is first and foremost important to understand the limitations of theories when we use them.

2.4.5.4 Further readings

There are many noteworthy security metrics based on attack graph analysis that are not covered in detail in this chapter. Let us refer to some of them here.

A common weakness of attack graph analysis is that it requires every vulnerability to be known. The k -zero day safety proposed by Wang et al. [170] compensates for this weakness. In this approach,

we calculate how many unknown vulnerabilities are required to make the attack successful if it fails with known vulnerabilities only. When the minimum number of vulnerabilities that need to be added is k , we say that the network is k -zero day safe. The required computation for k will be NP-hard in the worst case. However, it is possible to determine in polynomial time whether k is greater than an arbitrarily selected k' . Also, Wang et al. [171] propose an improved algorithm with significantly reduced computational complexity under some assumptions, such as that all the assumed unknown vulnerabilities are unique.

Wang et al. [172] focus on the network diversity metrics indicating the breadth of varieties of resources in a network. The idea behind this is that the more divergent network is more difficult to compromise. Later, Zhang et al. [189] have refined that idea to incorporate the dependency among exploits to handle the case where one successful exploit makes other exploits easier.

Alert correlation is a method to identify the occurrence of anomalous events by analyzing various network logs. We can regard it as a kind of dynamic security metric. Noel et al. [114] propose a method to infer ongoing attacks as a chain of exploits from the logs by applying the information contained in the attack graph. This method is further improved by Wang et al. [165] to overcome several weaknesses, such as the intolerance of slow attacks. Roschke et al. [131] propose applying attack-graph-based alert correlation to network security forensics. Husák et al. [55] give a survey about alert correlation, including attack graph applied methods.

Cao et al. [11] propose to expand the scope of security measurement to cover the business impact analysis. Their study leverages the expressiveness of MulVAL/Datalog to describe the dependency between services and its network infrastructure, i.e., the goal conditions represent the business impacts. They discuss pruning the generated attack graphs to simplify the relevant risk metrics calculation.

Most of the metrics referenced in this section are described in detail in the book by Wang et al. [173]. While not necessarily limited to attack graphs, see Kordy et al. [84], Ramos et al. [128], and Pendleton et al. [122] for a broader survey of security metrics, including recent research developments.

2.5 Network hardening

Network hardening is the problem of searching for the optimal set of security controls. Attack-graph-based network hardening primarily searches for a subset of the initial conditions that correspond to a minimal set of prerequisites for the chosen goal conditions. In an attack graph, the initial conditions represent vulnerabilities and configuration defects. The security controls correspond to the removal of initial conditions. In turn, the minimal set of initial conditions required for the goals implies the minimal set of sufficient security controls. In this section, we will see various approaches for network hardening with attack graphs.

2.5.1 Minimal-set search

We are interested in a subset of the initial conditions composed of only those essential conditions to the attack concerned. While, ideally, the minimum set should be present, it is not always met. Suppose we have an attack graph depicted in [Figure 2.5.1](#). The logical proposition corresponding to this graph is as follows:

$$c_g = (c_1 \wedge c_2 \wedge c_3) \vee (c_4 \wedge (c_{10} \vee c_{11}) \wedge c_6) \vee (c_7 \wedge c_8 \wedge c_9).$$

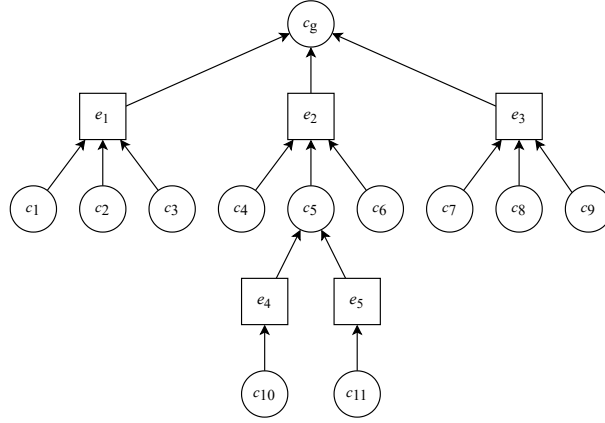


Figure 2.5.1: Another example of an attack graph

To make the goal condition c_g false, we need to all the exploits e_1 , e_2 , and e_3 to be false. Therefore, the number of possible combinations of initial conditions which are set to be false to make c_g false is $(2^3 - 1)^3 = 343$. For example, any of $\{c_1, c_{10}, c_{11}, c_8\}$, $\{c_1, c_2, c_6, c_8\}$, and $\{c_1, c_6, c_8\}$ can make c_g false. As this example shows, one combination can be a subset of another combination. In general, the solution set of a network hardening problem is a partially ordered set of sets. Thus, there is not necessarily a minimum element. Let us call these legitimate combinations of initial conditions *solutions* for the network hardening problem. A solution is said to be *minimal* when no solution is a subset of that solution. The minimum solution is always minimal, but the reverse is not true, e.g., $\{c_1, c_{10}, c_{11}, c_8\}$ is minimal but not minimum. It is the set of minimal solutions that we focus on as the solution set for network hardening problems.

Given an attack graph and a goal condition, Ammann et al. [6] propose an algorithm to find a minimal solution that prevents the goal condition from being fulfilled. Algorithm 2.3 shows the detail.

Algorithm 2.3 An algorithm to find a minimal solution (adapted from Ammann et al. [6])

Input:

C_g : the non-empty set of goal conditions.

R_i : the set of imply relations.

R_r : the set of require relations.

Output:

the set of initial conditions required for C_g .

```

1: function FIND-MINIMAL-SOLUTION( $C_g, R_i, R_r$ )
2:    $C'_g \leftarrow \emptyset$ 
3:   for  $e \in \{e \mid (e, c) \in R_i, c \in C_g\}$  do
4:      $C \leftarrow \{c \mid (c, e') \in R_r, e' = e\}$ 
5:      $C'_g \leftarrow C'_g \cup \{\text{Take an arbitrary element from } C\}$ 
6:   end for
7:   if  $C'_g = \emptyset$  then
8:     return  $C_g$ 
9:   else
10:     $C_i \leftarrow \{c \mid c \in C_g, (e, c) \notin R_i\}$ 
11:    return  $C_i \cup \text{FIND-MINIMAL-SOLUTION}(C'_g, R_i, R_r)$ 
12:   end if
13: end function

```

Note that this algorithm is only applicable for acyclic attack graphs. Besides, the algorithm's output is not uniquely determined since it chooses any one of the condition nodes in line 5. There are usually multiple minimal solutions, and this algorithm chooses one of them. An algorithm enumerating all the minimal solutions does not work. It can easily lead to a computational explosion.

2.5.2 Least-cost-set search

While every minimal solution can prevent the goal condition from being achieved, their desirabilities are different since the actual cost assigned to each security control is different. It would be more beneficial to find the cost-minimum minimal solution. We call the minimal solution with the lowest implementation cost as the *least-cost solution*.

Let us clearly define the problem to find the least-cost solutions. Suppose we have an attack graph $G = (E \cup C, R_r \cup R_i)$ and its initial conditions $C_i \subseteq C$. For this attack graph, let w be a function $C_i \rightarrow \mathbb{R}_+$, which maps an initial condition to its invalidation cost. For any set of initial conditions S , let us define the total cost of S as follows:

$$\text{cost}(S) = \sum_{c \in S} w(c).$$

Besides, let $c_g \in C$ be the goal condition, and $\mathcal{S}(c_g)$ be the set of all solutions, each in which invalidates c_g . With these terms, we can define the least-cost solution L as follows:

$$L = \arg \min_{S \in \mathcal{S}(c_g)} \text{cost}(S).$$

Note that L cannot necessarily be unique under this definition.

Noel et al. [113] propose an algorithm for finding a least-cost solution. The algorithm transforms an acyclic attack graph into a corresponding logical proposition and converts the expression into Conjunctive Normal Form (CNF). Each maxterm contained in it is the solution to the network hardening problem. A maxterm is a disjunction of initial conditions. After calculating the cost of each maxterm, we can choose the maxterm with the least cost. The problem is that the process of converting an attack graph to a CNF requires exponential computational complexity in the worst case since the number of maxterms can grow exponentially.

Wang et al. [166] propose an improved version of the above approach. This improved approach can handle multiple goal conditions and can also apply to cyclic attack graphs. [Algorithm 2.4](#) shows the detail of this approach.

The points of this algorithm are as follows. First, the derived logical proposition L is expressed in Disjunctive Normal Form (DNF) rather than CNF. This difference comes from the conjunction of the negation of goal conditions used as the starting point. The initial conditions contained in each conjunction in the obtained DNF are the solution to the network hardening problem. We can choose the least-cost solution by calculating and comparing each cost. Secondly, the “procedure avoids running into cycles by only expanding the search towards those vertices not reachable from the current vertex (line 13), and it also avoids introducing unsatisfiable logic loops into the final result (line 10). The procedure handles exploits in a similar way (line 18 through line 29).” (Wang et al. [166], the line numbers are recapped). This treatment makes this algorithm applicable to cyclic attack graphs. Still, the algorithm also has the potential computational explosion; the number of

Algorithm 2.4 A procedure for deriving solutions to the network hardening problem. (adapted from Wang et al. [166])

Input: $(E \cup C, R_r \cup R_i)$: An attack graph. $C_g \subseteq C$: The goal conditions.**Output:**A solution L to the goal $\bigwedge_{c \in C_g} \neg c$.

```

1:  $L \leftarrow \bigwedge_{c \in C_g} \neg c$  ▷ The initial goal
2: let  $Q$  be a queue initialized with  $C_g$  ▷ A queue used for searching in the graph
3: for  $e \in E$  do  $Pre(e) \leftarrow \{e\}$  ▷ The predecessor list
4: for  $c \in C$  do  $Pre(c) \leftarrow \{c\}$  ▷ The predecessor list

5: while  $Q \neq \emptyset$  do

6:   for each condition  $c$  dequeued from  $Q$  do
7:      $S_e \leftarrow \{e_k \mid (e_k, c) \in R_i\}$  ▷  $|S_e| = n$ 
8:      $T \leftarrow (e_1 \vee e_2 \vee \dots \vee e_n)$  ▷ Temporary variable
9:     for  $e_i \in S_e \cap Pre(c)$  do
10:      replace  $e_i$  with FALSE in  $T$ 
11:    end for
12:    replace  $c$  with  $T$  in  $L$ 
13:    for  $e_i \in S_e \setminus Pre(c)$  do
14:      enqueue  $e_i$  in  $Q$ 
15:       $Pre(e_i) \leftarrow Pre(e_i) \cup Pre(c)$ 
16:    end for
17:  end for

18:  for each exploit  $e$  dequeued from  $Q$  do
19:     $S_c \leftarrow \{c_k \mid (c_k, e) \in R_r\}$  ▷  $|S_c| = m$ 
20:     $T \leftarrow (c_1 \wedge c_2 \wedge \dots \wedge c_m)$  ▷ Temporary variable
21:    for  $c_i \in S_c \cap Pre(e)$  do
22:      replace  $c_i$  with FALSE in  $T$ 
23:    end for
24:    replace  $e$  with  $T$  in  $L$ 
25:    for  $c_i \in S_c \setminus Pre(e)$  do
26:      enqueue  $c_i$  in  $Q$ 
27:       $Pre(c_i) \leftarrow Pre(c_i) \cup Pre(e)$ 
28:    end for
29:  end for

30: end while

31: transform  $L$  into DNF

```

conjunctions in the DNF can increase exponentially for the number of initial conditions.

2.5.3 Allowable-set search

Suppose we have two initial conditions denoting (1) a web server has a vulnerability available to the attacker, and (2) the web server is running. We cannot invalidate the latter condition while keeping the first condition. That is, some initial conditions cannot be invalidated independently and can only be invalidated in combination with some other initial conditions. A generalization of this idea is the *allowable set* described below.

An *allowable action* is any subset of initial conditions of an attack graph such that all the conditions in it can only be jointly disabled. A *strategy* is a set of allowable actions such that the attacker cannot achieve the goal condition when every condition in those actions is disabled. In a more realistic network hardening, our interest is in finding the best strategy.

Albanese et al. [3] propose an algorithm to find an least-cost strategy. However, the algorithm's worst-case computational complexity is enormous. They estimate it to be $O(n^{n^d})$, where d is the maximum distance (number of edges) between initial and goal conditions, and n is the maximum in-degree of nodes in the attack graph. While the concept of allowable set is useful, this algorithm is impractical because of this computational burden.

2.5.4 Approximated-set search

As the discussions above clarify, reducing computational complexity is a critical issue in network hardening. The root cause of the computational explosion is finding the minimum solution or the least-cost solution. In principle, the number of solutions to the network hardening problem is exponential in the size of the network. Therefore, to achieve a drastic reduction in computational complexity, we must first give up on obtaining the best solution. Albanese et al. [3] propose the following [Algorithm 2.5](#) for the approximate minimum solution, which is exactly in line with this observation. Note that, in this algorithm, $\mathcal{S}$ refers to the set of all strategies, C_i denotes the initial conditions, and $\mathcal{A}$ is the set of all allowable actions. The algorithm is based on a forward search and identifies strategies to block the attacks that depart from the initial conditions and reach a single goal condition. When multiple-goal conditions are needed, just add a dummy goal condition and a dummy exploit node that aggregate the goal conditions.

The major difference of this algorithm from the prior ones is the reduction of the search scope. The $\sigma(q)$ represents the set of strategies blocking all the paths to node q . The function “TopK” in line 15 limits the size of this set to k , in order of least cost. “TopK” implicitly refers to the minimum costs of the strategies calculated in line 16. The topological sort ensures the readiness of those minimum costs before the first application of “TopK”. The scope trimming by “TopK” suppresses the computational complexity explosion effectively.

The least cost strategy obtained by this algorithm is only an approximation. However, Albanese et al. [3] report no significant deviation from the exact optimal strategy in numerical experiments. This algorithm shows excellent scalability without causing a computational explosion.

Algorithm 2.5 A forward-search algorithm for the least-cost strategy (adapted from Albanese et al. [3])

Input:

$(E \cup C, R_r \cup R_i)$: An attack graph.
 k : The optimization parameter.

Output:

$\sigma : C \cup E \rightarrow 2^S$: a mapping.
 $minCost : C \cup E \rightarrow \mathbb{R}^+$: a mapping.

```

1:  $Q \leftarrow \text{TopologicalSort}(C \cup E)$ 
2: while  $Q \neq \emptyset$  do
3:    $q \leftarrow$  a node dequeued from  $Q$ 

4:   if  $q \in C_i$  then
5:      $\sigma(q) \leftarrow \{\{A\} \mid A \in \mathcal{A}, q \in A\}$ 
6:   else if  $q \in E$  then
7:      $\sigma(q) \leftarrow \bigcup_{(c,q) \in R_r} \sigma(c)$ 
8:   else if  $q \in C \setminus C_i$  then
9:      $\sigma' \leftarrow \{\emptyset\}$ 
10:    for  $(e, q) \in R_i$  do
11:       $\sigma' \leftarrow \{S \cup S_e \mid S \in \sigma', S_e \in \sigma(e)\}$ 
12:    end for
13:     $\sigma(q) \leftarrow \sigma'$ 
14:   end if

15:    $\sigma(q) \leftarrow \text{TopK}(\sigma(q), k)$ 
16:    $minCost(q) \leftarrow \min_{S \in \sigma(q)} \text{cost}(S)$ 
17: end while

```

2.5.5 Discussions

Network hardening is computationally very expensive. It is still a tough but unignorable problem that we should solve. The following sees the breadth of issues and other attempts.

2.5.5.1 Computational complexity

Ammann et al. [6]’s algorithm for finding a single minimal solution has polynomial complexity. The algorithm by Wang et al. [166] to find the least-cost solution and the backward-search algorithm by Albanese et al. [3] to consider the allowable set have exponential complexity in the worst case. No estimates are given for the algorithm by Albanese et al. [3] to find an approximate solution, but the numerical experiments’ results show good scalability.

Is there room for significant improvement in the algorithm for exact solutions? At least, it is a tough problem. Remind that Wang et al.’s algorithm transforms the attack graph into a logical proposition in DNF and searches the least-cost disjunction. Li [88] coined such a problem as the MinCostSAT problem. It is a search problem for the true/false assignment to propositional variables that makes the concerned proposition true while considering the assignment’s cost, given the individual costs of variables. This problem is known to be NP-hard. Although the worst-case computational explosion is inevitable, we can leverage the general-purpose MinCostSAT solver. Homer et al. [47] approach this problem along with this observation.

2.5.5.2 Cycle handling

In the algorithms presented in this section, only Wang et al. [166] explicitly incorporates the considerations for cyclic attack graphs. The other algorithms apply to acyclic attack graphs. Albanese et al. [3] state that the method by Wang et al. [166] can be leveraged to handle cycles, though the detail of such an integration is not given. As discussed in [Section 2.4.5.2](#), the conversion to acyclic graphs makes the resultant graph depend on the order of cycle elimination. The use of topological sort in the approximation algorithm by Albanese et al. [3] poses a similar problem. The result of topological sort is not necessarily unique. The reasonable and stable approach for cycle handling is still an important research issue.

2.5.5.3 Multi-objective network hardening

The algorithms discussed above are for qualitative, single-objective optimization. To date, there are few studies about multi-objective optimization algorithms for quantitative security metrics over unrestricted exploit-dependency graphs. The closest is the work by Stan et al. [150] using A* solver to approximate the Pareto front of Cost/Benefit optimization, but for acyclic graphs. Poolsappasit et al. [125] use a genetic algorithm (NSGA-II) for acyclic graphs. Dewri et al. [23] and Dewri et al. [24] use NSGA-II for attack trees rather than attack graphs. Ali and Khan [4] apply the particle swarm optimization technique to attack trees. Almohri et al. [5] apply Linear Programming techniques to acyclic graphs. Any of these algorithms is an approximation algorithm, and there is no guarantee for its optimality. Computational complexity is often problematic, too. The research in this field seems still immature.

2.5.5.4 Game theoretic approach

The game-theoretic approach is useful to discuss network hardening problems, too. In the approach, we regard network hardening as a game between an attacker and a defender. The attacker tries to compromise the network while the defender blocks it. An attack graph provides constraints that determine the actions available to the attacker and the defender. We can formulate this situation as a game by supplying utility and cost functions for the two.

Letchford and Vorobeychik [87] examine this approach and enumerate the important factors such as uncertainty assumption and computational complexity. Durkota et al. [30] discuss the game alike where the defender's action is the placement of honeypots to deceive the attacker. Their study also elaborates on the detail of this approach. Nguyen et al. [108] propose a sampling-based heuristic algorithm to solve another attack-graph game. Miehl et al. [94] elaborate on a very similar but different approach based on POMDP. They use Monte-Carlo sampling and the dynamical data from IDS to estimate the actual state of the network while searching for the optimal defense action online. Hu et al. [53] apply a Q-learning approach to a similar problem based on Bayesian attack graphs with an unknown utility function.

The problem of the game-theoretic approach is computational complexity. In general, the game's tree is too huge to search for the globally optimal solution. Therefore, it is almost inevitable to use some heuristics. The studies referred to above propose such heuristics and evaluate their performance.

There is another study worth noting here. The experiment by Durkota et al. [29] compares the optimal defense strategy induced from attack graphs by their algorithm and the strategies worked out by human respondents. The result shows that the optimal strategy is robust against the worst-case attacks, while it is inferior to the man-made strategies when applied to man-made attack strategies. This result implies an unexplored field.

2.6 Bayesian attack graph

A Bayesian attack graph is an attack graph defined as a kind of Bayesian network. It is a generalized form of probabilistic attack graphs, which treats the exploit-dependency graph as a particular case. Moreover, it considers the dependencies among events, discussed in [Section 2.4.4](#), during the attack probability calculation. It is also capable of updating the probability under some observed attack information. We can leverage the rich results in Bayesian network research to enhance attack graph analysis. Although it is not possible to handle cyclic attack graphs due to their acyclic definition, it is the most comprehensive framework to date for discussing attack graphs. This section elaborates on its characteristics.

2.6.1 Brief introduction to Bayesian network

Firstly, we define several terms relevant to Bayesian network. In this section, the capital variables such as $X_1, \dots, X_n$ denote Bernoulli variables. Then, a *Bayesian network* is a joint distribution of $X_1, \dots, X_n$ defined as

$$P(X_1, \dots, X_n) = \prod_k P(X_k | \text{parents}(X_k))$$

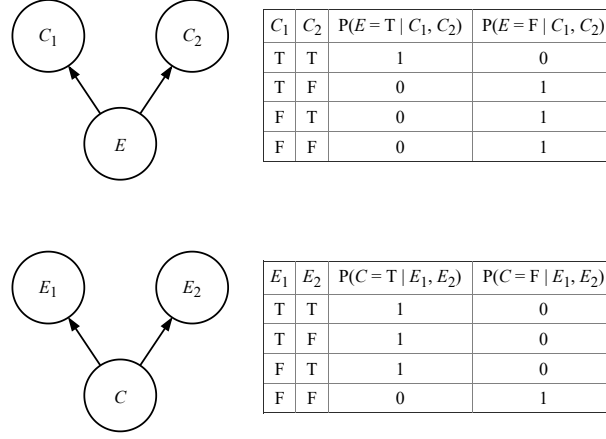


Figure 2.6.1: AND and OR combinations by CPTs

where (1) “parents(X_k)” denotes the subset of $\{X_1, \dots, X_n\}$ satisfying

$$P(X_k | X_1, \dots, X_{k-1}, X_{k+1}, \dots, X_n) = P(X_k | \text{parents}(X_k)),$$

and (2) the directed graph composed of $X_1, \dots, X_n$ and their dependencies is a DAG, i.e., the set of all ascendants of any X_k does not contain X_k . A leaf X_k without parents is supposed to have a *prior distribution* $P(X_k)$. On the other hand, every node X_k with parents has a *conditional probability distribution* $P(X_k | \text{parents}(X_k))$. The conditional probability distribution is often expressed in a *conditional probability table (CPT)*.

The accurate computation of various probabilities for a Bayesian network, along with the definition, is called *exact inference*. The computation at the expense of accuracy while suppressing the computational complexity is called *approximate inference*. The exact inference is known to be an NP-hard problem [19]. Therefore, it is inevitable to rely on approximate inferences to process large-scale networks.

Belief propagation is a well-known inference algorithm proposed by Pearl [121]. It is an exact inference algorithm applicable to Bayesian networks with polytree (also called singly-connected) structure. At the same time, it is also empirically known as an approximate inference algorithm applicable to arbitrary Bayesian networks [97]. The computational complexity of belief propagation is almost linear, although belief propagation over non-polytree networks produces approximated inference results. The textbook by Darwiche [20] provides a concise but detailed explanation of various methods.

2.6.2 Bayesian attack graph

A node other than initial conditions appearing in an attack graph represents either AND or OR combinations of other nodes connecting to itself. In a Bayesian network, a conditional probability represents the combination among nodes. If we can use CPT to represent the AND and OR combination, an attack graph can be interpreted as a Bayesian network. Frigault and Wang [36] pointed out that it is indeed possible. Figure 2.6.1 shows the CPT representations of those bindings. Note that the edge direction is different from that of an attack graph. In the case for E , an exploit node, the CPT on the right says that the variable E becomes true if and only if its two parents C_1 and C_2 are true. Similarly, In the case for C , a condition node, the CPT says that the variable C becomes

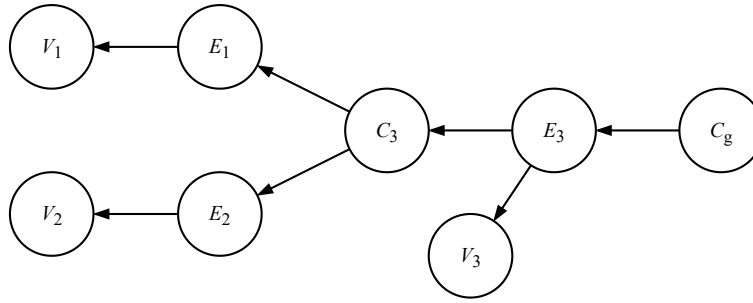


Figure 2.6.2: An example of dependent exploits

false if and only if its two parents E_1 and E_2 are false. This behavior is what we exactly need to express an attack graph as a Bayesian network. In this way, we can handle an attack graph as a specific case of Bayesian networks. Furthermore, we can use various probabilities such as 0.3 or 0.7 in the exploit node's CPT. Such a probability assignment generalizes the concept of attack graphs. Based on this observation, from now on, we will call a Bayesian network interpretable as a generalized acyclic attack graph a *Bayesian attack graph*.

Bayesian attack graphs have a variety of benefits. Firstly, the inference considers the dependence among attack paths. Secondly, CPT's flexibility allows us to represent complicated exploits that are difficult to represent in a qualitative attack graph. We will elaborate on the use of this flexibility in the next subsection. The third, but not the last, benefit is that we can estimate the relevant event's probability based on the observation of other variables' values. For example, when one host is found compromised, we can estimate the probability of its neighbors' compromise. A method for estimating the current state of a network based on observed evidence is called *dynamic analysis*. Peng Xie et al. [123] propose a modeling method for Bayesian attack graphs to estimate the attack probability more accurately based on the information detected by IDS.

2.6.3 Advanced exploit-dependency modeling

Suppose we have a host with three vulnerabilities named V_1 , V_2 , and V_3 , respectively. If the attacker exploits V_1 or V_2 , he can log in to the host. The login state is required to exploit V_3 . Figure 2.6.2 is a Bayesian attack graph that illustrates this situation. In this Bayesian attack graph, to achieve the goal condition C_g , the attacker must achieve at least two exploits, E_1 and E_3 , or E_2 and E_3 . Here, let us assume that it is easier to achieve E_3 after E_2 than E_3 after E_1 . In that case, how can we express the difference in difficulty? Attack graphs cannot handle cases like this. In contrast, Bayesian attack graphs can naturally represent it using the CPT, shown in Table 2.2.

The above discussion's essence is that a successful exploit has side effects that affect the success probability of other exploits. Wang et al. [169] pointed out that Bayesian attack graphs could handle such dependence considerations between exploits. Cheng et al. [13] propose an algorithm to automatically determine the dependencies between exploits by focusing on individual base metrics extracted from CVSS scores. Cheng et al. [14] propose the refined algorithm that enables this probability adjustment process within the framework of Bayesian attack graphs by adding some nodes and edges to the original Bayesian attack graph.

The powerful expressiveness of the Bayesian attack graph is also helpful in dealing with network security's complex nature. Although not covered in this chapter, there exist other thought-provoking

V_3	C_3	E_2	$P(E_3 = T V_3, C_3, E_2)$
T	T	T	0.61
T	T	F	0.55
T	F	T	0.12
T	F	F	0
F	T	T	0
F	T	F	0
F	F	T	0
F	F	F	0

Table 2.2: An example CPT with varying exploitability

studies. Frigault et al. [37] propose a security assessment method combining Bayesian attack graphs and CVSS temporal metrics. Sun et al. [152] propose a method for detecting zero-day attacks by constructing a kind of Bayesian attack graph from system logs.

2.6.4 Loopy belief propagation and Noisy OR/AND

Belief propagation applied to general Bayesian networks as an approximate inference algorithm is called *loopy belief propagation*. While the loopy belief propagation works efficiently, the computational load relevant to huge CPTs inhibits efficiency. Since CPTs allow us to set probabilities for each possible combination of the parent variables' values, we must articulate the exponential number of cases. It implies the combinatorial explosion and intractable memory consumption. In contrast, using the conditional probability determined by a specific rule makes it possible to simplify the calculation. A typical example is the use of Noisy OR/AND, which is shown below.

Noisy OR is a model that determines the conditional probability with N parent variables by N configuration parameters $c_1, \dots, c_N$. Noisy AND is another model similar to Noisy OR with $2N$ configuration parameters $c_1, \dots, c_N$ and $s_1, \dots, s_N$ for N parent variables. Each of them is defined as follows:

$$P_{\text{OR}}(X = T | \mathbf{u}) = 1 - \prod_{k \in I_T(\mathbf{u})} (1 - c_k),$$

$$P_{\text{AND}}(X = T | \mathbf{u}) = \prod_{k \in I_T(\mathbf{u})} c_k \prod_{k \in I_F(\mathbf{u})} s_k,$$

where $\mathbf{u}$ denotes the set of observed values, such as $(U_1 = T, U_2 = T, U_3 = F)$, for the parents variables of X , and c_k and s_k are the configuration parameters. The term $I_T(\mathbf{u})$ and $I_F(\mathbf{u})$ are the set of indices corresponding to the observed values in the parents. For example, if $\mathbf{u} = (U_1 = T, U_2 = T, U_3 = F)$, then $I_T(\mathbf{u}) = \{1, 2\}$ and $I_F(\mathbf{u}) = \{3\}$. Roughly speaking, the configuration parameters c_k in Noisy OR express the contribution of the parent facts to X . If one c_k has a value which equals to one, then the observation of $U_k = T$ assures that $X = T$. If every c_k is less than one, X will not be guaranteed to be T even when every $U_k = T$. The alike applies to Noisy AND.

Pearl [121] discusses the use of Noisy OR/AND in the belief propagation. It simplifies the equations for the message passing. The computation is quite efficient since the transformed expression contains only $O(N)$ operations, where N is the number of parent variables.

Muñoz-González et al. [96] report the performance of exact inference using Noisy OR/AND. Furthermore, Muñoz-González et al. [95] investigate the performance with the combination of loopy

belief propagation and (Leaky) Noisy OR/AND. Compared to the brief introduction in this chapter, their report examines the detail about the treatment of Bayesian networks. There is already a considerable amount of studies for Bayesian networks. The two studies are good starting points to understand those details from the viewpoint of attack graph analysis.

2.6.5 Discussions

Unfortunately, Bayesian-attack-graph-based practical methods seem immature due to the excessive computational burden. The following is a list of topics informative for further research.

2.6.5.1 CPT and Canonical models

The Noisy OR/AND introduced in [Section 2.6.4](#) is a kind of *canonical models*. Although we introduced Noisy OR/AND to reduce the computational complexity, the canonical model is usually preferred for its ease of modeling. Suppose that a variable in a Bayesian network has N parent variables. In this case, the CPT for this variable has 2^N rows, and the number of rows increases exponentially as N increases. It is not easy to determine the appropriate probabilities for each row. The canonical model provides a solution to this challenge since it determines the CPT with a limited number of parameters, such as N . Diez and Druzdzel [25] provide a concise description of canonical models.

2.6.5.2 Reliability of inference

Attack graph analysis is the automation of logical reasoning based on input data, and the reliability of the analysis depends on the reliability of the input data. The situation is no different in Bayesian attack graphs. Rather, it is more challenging to ensure the reliability of the input data. Qualitative attack graph analysis only requires clarifying the pre/postconditions of the exploit. For Bayesian attack graphs, we need to add probabilistic evaluations of exploits.

In the literature, the CVSS score is widely used for probability assignment. The caveat is that it is essentially qualitative and does not solve the issue. For example, suppose we have two probability values A and B , based on the CVSS score, and they satisfy that $A = 2B$. If A and B are reliable as probabilities, we can estimate the events corresponding to A as occurring twice as often as those of B . However, it is groundless.

Generally speaking, we cannot take the various probabilities obtained from a Bayesian attack graph at face value. In practice, we need additional careful treatment for the results.

2.6.5.3 Network hardening

Most of the methods in [Section 2.5](#) do not apply to Bayesian attack graphs. While these methods assume that the effect of security controls reflects in binary, boolean values, Bayesian attack graphs require them to be probabilities.

One way to solve the network hardening problem for Bayesian attack graphs is to apply a generic solver. For example, Poolsappasit et al. [125] propose applying the well-known genetic algorithm NSGA-II to Bayesian attack graphs. There are general-purpose algorithms for multi-objective optimization, such as genetic algorithms and swarm optimization. However, all of these algorithms are iterative. In the iterations, the tentative solution must be evaluated repeatedly. This means

that we have to infer the probabilities on the Bayesian attack graph many times, and the whole computational complexity is enormous. Furthermore, general-purpose algorithms tend to be slow to converge, which makes this approach infeasible. This field of research needs further development.

2.6.5.4 Uncertain graph

Finally, we would like to refer to the studies by Nguyen et al. [107], Nguyen and Nicol [106]. Their studies are based on uncertain graphs; an uncertain graph is a probabilistic distribution of deterministic graphs. This framework defines a cyclic attack graph as a probabilistic distribution of acyclic attack graphs. When we want to calculate the reachability to a node, in theory, we extract every possible acyclic graph and check the path's existence on each graph. Then, determine the reachability as the weighted sum according to each acyclic graph's existence probability. The two studies elaborate on the Monte-Carlo sampling strategy for uncertain graphs. Though it seems not scalable so far, it is a very different, challenging approach. The uncertain graph provides a formalism that can generalize the exploit dependency graphs with probability in a way other than Bayesian networks.

2.7 Conclusion

In this chapter, we summarized the major contributions in the field of attack graph analysis. An attack graph is a kind of cyclic AND/OR graphs enabling qualitative network security analysis. The semi-naive evaluation of a Datalog program generates an attack graph efficiently. While an attack graph comprises thousands of nodes, intelligible visualization techniques such as hierarchical representation are available. Decision-makers can also use various network security metrics based on attack graphs. These metrics clarify the probabilistic or qualitative evaluation of the risk posture of the concerned network. Furthermore, several network hardening algorithms can search for optimal security controls to prevent undesirable events. From the theoretical point of view, the Bayesian attack graph seems promising since it can generalize the concept of attack graphs with probabilistic information. The remaining key issues in this field include handling cycles, AND combinations of conditions, and intractable computational complexity.

From the viewpoint of this study, we would like to focus on the following issues:

1. the lack of quantitative network security metrics applicable to cyclic attack graphs since it is the basis for cost-benefit analysis that will be required in practical information security management,
2. the lack of a multi-objective network hardening algorithm with practical performance that is essential due to the multi-faceted nature of information security risks, and
3. the lack of detailed information, such as threat statistics, required for quantitative analysis and network hardening.

Overcoming these issues makes attack graph analysis a practical risk analysis approach suitable for the basis of secured risk analysis. In the subsequent chapters, we examine approaches for them.

Chapter 3

Network security metric

3.1 Introduction

Due to resource limitations, we need to prioritize cybersecurity investments in general. The basis for such prioritization is the cost-benefit analysis of available security options. Unfortunately, attack graphs lack the quantitative information required for cost-benefit analysis. In this chapter, our interest is in the likelihood of a successful attack — the reachability, which is quantitative.

There are many studies approaching reachabilities in the broad sense. For example, Mehta et al. [92] proposed a method processing an attack graph as a kind of Markov chain, but an attack graph is an AND/OR graph, and AND combinations of events are not expressible as a Markov chain. Sawilla and Ou [137] propose a proxy metric considering AND combinations of events. However, the metric is not interpretable as a probability. Bayesian attack graphs [91, 36, 123] can calculate reachabilities as probabilities while handling AND/OR combinations of events. The remained problems are the scalability and the handling of cycles in attack graphs; Bayesian attack graphs are defined as acyclic graphs, although general attack graphs are cyclic. It is still challenging to analyze the security of a network with the realistic scale and cyclic network topology.

In this chapter, we propose yet another approach named reachability-graph analysis. A reachability graph is a directed graph obtained from an attack graph. It enables us to compute reachabilities as probabilities, not as proxy metrics. It is extremely fast to generate, thus applicable to large-scale network security analysis. Also, it applies to attack graphs with cycles that Bayesian attack graphs cannot handle. We examine its performance and characteristics, including configurable filter functions used to define reachabilities.

The major contributions in this chapter are as follows:

1. To the author's best knowledge, the proposed algorithm is the first scalable one that calculates probabilities and applies to cyclic attack graphs with AND/OR nodes.
2. This chapter also articulates a class of extensible metrics provided with a suite of mathematical proofs. It enables further development of security metrics with the desirable properties above.

[Section 3.2](#) refers to the relevant prior studies and quickly summarizes them. [Section 3.3](#) defines the problem we solve in this chapter, with the definition of attack graphs. [Section 3.4](#) explains two approaches: the Bayesian attack graph and the reachability graph. The latter is our proposed approach. In [Section 3.5](#), we examine these approaches with two numerical experiments. The first

experiment compares the computational efficiency of our approach with Bayesian attack graphs. The second experiment shows the performance of our approach applied to large-scale networks with cyclic topology. Then, after discussing the relevant theoretical topics, including the computational complexity and the extensibility of our approach, we make conclusions.

3.2 Related work

An attack graph describes the qualitative inference results about the potential attacks. In essence, an attack graph is a kind of AND/OR graph. It is a directed graph where each node in it denotes AND/OR bindings of its parents. Attack graphs lack any quantitative information such as successful attack likelihood or monetary loss. We need some additional approaches to make relevant risks comparable.

Idika and Bhargava [56] propose to use the combination of simple metrics such as the mean length of attack paths, the standard deviation of them, and the count of them. Though, their metrics are qualitative inherently. For example, can we determine if a network's risk is twice as high when we see the doubled number of attack paths? The algorithm by Mehta et al. [92] computes the pseudo PageRank like Google's. Each node's rank metric in an attack graph denotes the expected probability of the coupled event occurrence. The critical limitation is that the metric cannot handle AND combinations of events. Sawilla and Ou [137] propose another metric named AssetRank. This metric considers the AND/OR combinations of events. On the other hand, the value of AssetRank is not a probability. Wang et al. [169] define a set of intuitive calculation rules for event occurrence probabilities. Their approach can handle both of AND/OR combinations of events. However, it assumes that every event on the graph is independent. The algorithm proposed by Homer et al. [49, 50] overcomes all of these issues. The problem left is that the computational complexity explodes in the worst case.

To date, the Bayesian attack graph [91, 36, 123] seems to be the most sophisticated approach for attack-graph based security metrics calculation. A Bayesian attack graph is an attack graph represented as a Bayesian network [121] and can compute the occurrence probability of the non-independent events coupled with nodes.

A Bayesian network is a joint probability distribution expressed as a DAG (Directed Acyclic Graph) of stochastic variables. Belief propagation is a well-known inference algorithm proposed by Pearl [121]. It is an exact inference algorithm applicable to Bayesian networks with polytree structure. At the same time, it is also empirically known as an approximate inference algorithm applicable to arbitrary Bayesian networks [97]. The computational complexity of belief propagation is almost linear in many cases.

The key idea in the Bayesian attack graph is to leverage conditional probability distribution. In a Bayesian network, we can formulate a mapping from the AND/OR combination of events in an attack graph to the corresponding node with an appropriately defined conditional distribution.

Liu and Man [91] proposed the earliest attempt for the Bayesian attack graph. Each variable in their Bayesian attack graph corresponds to a host's state, such as the host is compromised or not. Their approach does not care about the difference between AND/OR combinations of events. Frigault and Wang [36] proposed to leverage the flexibility of conditional probability distributions to represent the combination of events. Peng Xie et al. [123] examined the use of Noisy-OR and

Noisy-AND to map attack graph nodes to Bayesian network nodes. Meanwhile, these prior studies only discussed the theoretical and limited cases of problems. The studies by Poolsappasit et al. [125] and Muñoz-González et al. [95] are Bayesian attack graph applications to more realistic scenarios.

As is mentioned above, various researchers have discussed the Bayesian attack graph. However, there remain the following two issues.

The first and prominent issue is scalability. In realistic scenarios, we have to analyze a network composed of thousands of computers. Nevertheless, none of these Bayesian prior studies examined the analysis for such large-scale networks. The exact inference on a Bayesian network is an NP-hard problem [19]. Therefore, it is inevitable to depend on the approximate inference algorithms to process large-scale networks. Regretfully, those approximations are not sufficiently fast. Furthermore, it could be erroneous, and there is no guarantee for the algorithm to terminate in general.

The second issue is the treatment of cycles. By definition, a Bayesian network is a DAG. In contrast, an attack graph can contain cycles. Frigault and Wang [36] assume that the input attack graph is a DAG. Peng Xie et al. [123] never mentioned the cycles in an attack graph. Poolsappasit et al. [125] assume that it is ignorable. Muñoz-González et al. [95] refer to the cycles cut algorithm in Wang et al. [169], while Wang's approach requires multiple, transformed graph evaluations implicitly. In theory, we can regard the approach by Homer et al. [49, 50] as a kind of Bayesian attack graph with graph conversion preprocess from cyclic to acyclic, and this conversion also accompanies costly computation alike.

As a whole, we can say that the Bayesian attack graph has considerable weaknesses in the application to practical cases, including large scale, cyclic attack-graphs. We solve these issues with another approach named reachability graph.

3.3 Problem Settings

3.3.1 Attack Graph

At first, let us remind the definition of attack graphs. We follow the definition in [3].

Definition 1 (Attack graph). Given a set of *exploits* E , a set of *security conditions* C where $E \cap C = \emptyset$, a *require*-relation $R_r \subseteq C \times E$, and an *imply*-relation $R_i \subseteq E \times C$, an *attack graph* G is the connected, directed graph $G = (E \cup C, R_r \cup R_i)$, where $E \cup C$ is the node and $R_r \cup R_i$ is the edge set. For each attack graph $G = (E \cup C, R_r \cup R_i)$, *initial conditions* refer to the subset $C_i \subseteq C$ composed of every condition $c \in C$ satisfying $\nexists e \in E$ such that $(e, c) \in R_i$, i.e., the set of all the leaves in the graph, whereas *intermediate conditions* refer to the subset $C \setminus C_i$.

Usually, we use initial conditions as the representation of vulnerabilities of the hosts on a network. The exploits are the nodes representing the AND nodes in an attack graph. The intermediate conditions are the nodes corresponding to the OR nodes. See Figure 3.3.1 shows an example.

This attack graph implies that the occurrence of c_4 depends on the occurrence of e_1 or e_2 . Also, e_1 depends on c_1 and c_2 , and e_2 depends on c_2 and c_3 . The following formulae in propositional logic represent their relationship:

$$c_4 = e_1 \vee e_2,$$

$$e_1 = c_1 \wedge c_2,$$

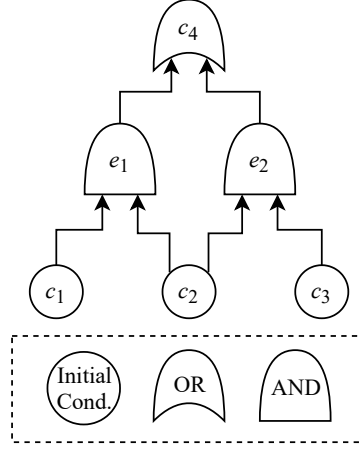


Figure 3.3.1: A small example of an attack graph

$$e_2 = c_2 \wedge c_3.$$

With this graph, we can extract two “paths” for the security condition c_4 . That is, $c_1 \wedge c_2 \rightarrow e_1 \rightarrow c_4$ and $c_2 \wedge c_3 \rightarrow e_2 \rightarrow c_4$.

We use an attack graph as an input to generate other graph structures discussed in [Section 3.4](#). Various algorithms are proposed to generate an attack graph [90, 142, 52]. Most of the algorithms have the polynomial computational-complexity in the size of the network to be analyzed. Based on these prior studies, we assume that the attack graph in need is given.

3.3.2 Network reachability analysis

Our interest is in the probabilistic quantification of the risks implied by an attack graph. Suppose every node in an attack graph is a Bernoulli variable, i.e., each variable takes a value either True or False. If the node c_1 has an assigned probability of 0.9 for $c_1 = \text{True}$, then we interpret it as that the occurrence probability of event c_1 is 0.9. We call that probability *reachability* and denote the reachability to node x by “reachability(x)”. With this reachability concept, the following definition clarifies the problem which we try to solve:

Definition 3 (Network reachability analysis). Given an attack graph $G = (E \cup C, R_r \cup R_i)$, and *initial reachabilities* $I \subseteq C_i \times [0, 1]$ satisfying $\{c \mid (c, r) \in I\} = C_i$ and $\forall (c_k, r_k), (c'_k, r'_k) \in I, c_k = c'_k \implies r_k = r'_k$. Let us define a problem in which we identify the “reachability(x)” for every $x \in E \cup C$. We call this problem a *network reachability analysis*.

This definition is incomplete in the sense that there is a freedom of choice for the “natural” assignment of reachabilities to nodes in an attack graph. We leverage this flexibility to examine several different approaches to the problem.

3.4 Approach

3.4.1 Bayesian attack graph

As preparation, we define several terms relevant to Bayesian network. In this subsection, we use capital variables such as $X_1, \dots, X_n$ to denote Bernoulli variables. Then, a *Bayesian network* is a

joint distribution of $X_1, \dots, X_n$ defined as

$$P(X_1, \dots, X_n) = \prod_k P(X_k | \text{parents}(X_k))$$

where (1) “parents(X_k)” denotes the subset of $\{X_1, \dots, X_n\}$ satisfying

$$P(X_k | X_1, \dots, X_{k-1}, X_{k+1}, \dots, X_n) = P(X_k | \text{parents}(X_k)),$$

and (2) the graph composed of $X_1, \dots, X_n$ and their dependencies is a DAG, i.e., the set of all ascendants of any X_k does not contain X_k . A leaf X_k in the graph is supposed to have a *prior distribution* $P(X_k)$. On the other hand, every node X_k with parents has a *conditional probability distribution* $P(X_k | \text{parents}(X_k))$.

In this chapter, we focus on only two types of conditional distribution: Noisy-OR and Noisy-AND. Let us define the set $\tau(X_k) = \{i | X_i \in \text{parents}(X_k), X_i = \text{True}\}$. The Noisy-OR distribution function of X_k is defined as follows:

$$P(X_k | \text{parents}(X_k)) = \begin{cases} 1 - \prod_{i \in \tau(X_k)} (1 - \mu_{k,i}) & (\tau(X_k) \neq \emptyset) \\ 0 & (\text{otherwise}) \end{cases}$$

where each $\mu_{k,i} \in [0, 1]$ is predefined for each pair composed of variable $X_i \in \text{parents}(X_k)$ and X_k . We denote this conditional distribution by “noisy-OR($X_k, \{c_{k,1}, c_{k,2}, \dots\}$)”. Similarly, The Noisy-AND distribution function of X_k is defined as follows:

$$P(X_k | \text{parents}(X_k)) = \begin{cases} \prod_{i \in \tau(X_k)} \nu_{k,i} & (\tau(X_k) = \text{parents}(X_k)) \\ 0 & (\text{otherwise}) \end{cases}$$

where each $\nu_{k,i} \in [0, 1]$ is predefined for each pair composed of variable $X_i \in \text{parents}(X_k)$ and X_k . We denote this conditional distribution by “noisy-AND($X_k, \{\nu_{k,1}, \nu_{k,2}, \dots\}$)”. Note that the definition above is a simplified case in [Section 2.6.4](#), and we use τ, μ , and ν instead of I, c , and s to avoid confusion.

With the terms introduced above, we define a Bayesian attack graph as follows. Here, we adapt the definition in Poolsappasit et al. [125] and Muñoz-González et al. [95].

Definition 4 (Bayesian attack graph). Given an attack graph $G = (E \cup C, R_r \cup R_i)$ where $E = \{e_1, \dots, e_m\}$ and $C = \{c_1, \dots, c_n\}$, and initial reachabilities $I \subseteq C_i \times [0, 1]$ satisfying $\{c_k | (c_k, r_k) \in I\} = C_i$ and $\forall (c, r), (c', r') \in I, c = c' \implies r = r'$. Let E_k be the Bernoulli variable corresponding to node e_k in E , and C_k be the Bernoulli variable corresponding to node c_k in C . Also, let $\text{parents}(E_k) = \{C_\ell | (c_\ell, e_k) \in R_r\}$ for each E_k , and $\text{parents}(C_k) = \{E_\ell | (e_\ell, c_k) \in R_i\}$ for each C_k . Then, we call the joint distribution defined as follows a Bayesian attack graph:

$$P(C_1, \dots, C_n, E_1, \dots, E_m) = \prod_{X_k \in \{C_1, \dots, C_n, E_1, \dots, E_m\}} P(X_k | \text{parents}(X_k))$$

where

$$P(X_k | \text{parents}(X_k)) = \begin{cases} P(X_k) = \begin{cases} r_k & (X_k = 1) \\ 0 & (X_k = 0) \end{cases} & (X_k \in C_i) \\ \text{noisy-OR}(X_k, \{\mu_{k,1}, \dots\}) & (X_k \in C \setminus C_i) \\ \text{noisy-AND}(X_k, \{\nu_{k,1}, \dots\}) & (X_k \in E) \end{cases}.$$

We have to define the configuration parameters $\mu_{k,i}$ and $\nu_{k,i}$ for Noisy-OR/AND to obtain a Bayesian attack graph. We omit the discussion how to determine those parameters. As our focus is on the computation after those parameters are supplied, we assume those parameters are given appropriately. Additionally, the input attack graph G is supposed to be a DAG since a Bayesian network is a DAG.

In a Bayesian attack graph, the marginal probability assigned to each node corresponds to the reachability. Various inference algorithms, such as belief propagation, can compute those probabilities. In the definition above, we map AND nodes in an attack graph to the variables with Noisy-AND probability distribution. Similarly, OR nodes are mapped to the variables with Noisy-OR distribution. As Section 4.3.2 in [121] discusses, the computational complexity of each conditional distribution's marginalization with Noisy-OR/AND scales linearly with the number of parent variables, while it is exponential in general. We use the loopy belief propagation algorithm with this efficient computation for the experiment in [Section 3.5.1](#).

3.4.2 Reachability graph

A *reachability graph*, which we propose here, is a directed graph in another form derived from an attack graph. In a reachability graph, we use the following functions to compute the reachability assigned to each node in an attack graph:

1. Max-OR function: $\text{max-OR}(\{x_1, \dots, x_n\}) = \max_k \text{reachability}(x_k)$, defined for $\forall \{x_1, \dots, x_n\} \subseteq E$.
2. Min-AND function: $\text{min-AND}(\{x_1, \dots, x_n\}) = \min_k \text{reachability}(x_k)$, defined for $\forall \{x_1, \dots, x_n\} \subseteq C$.
3. An arbitrary *filter* function $f : [0, 1] \rightarrow [0, 1]$, satisfying $\forall v_1, v_2 \in [0, 1], v_1 \leq v_2 \implies f(v_1) \leq f(v_2)$, and $\forall v \in [0, 1], f(v) \leq v$.

With the aid of these functions, we define the reachability graph as follows:

Definition 5 (Reachability graph). Given an attack graph $G = (E \cup C, R_r \cup R_i)$ where $E = \{e_1, \dots, e_m\}$ and $C = \{c_1, \dots, c_n\}$, and initial reachabilities $I \subseteq C_i \times [0, 1]$ satisfying $\{c | (c, r) \in I\} = C_i$ and $\forall (c_k, r_k), (c'_k, r'_k) \in I, c_k = c'_k \implies r_k = r'_k$. For each node $e \in E$, let $\text{parents}(e) = \{c_k | (c_k, e) \in R_r\}$. Similarly, let $\text{parents}(c) = \{e_k | (e_k, c) \in R_i\}$ for each node $c \in C \setminus C_i$. With these definitions, we categorize every node $x \in E \cup C$ into the following three types and assign reachabilities respectively:

1. External node: a node $x \in C_i$, with reachability $(x) = r_k$ where $(x, r_k) \in I$.
2. Max-OR node: a node $x \in C \setminus C_i$, with reachability $(x) = f_{[x]}(\text{max-OR}(\text{parents}(x)))$ where $f_{[x]}$ is the arbitrary filter function defined for each $x \in C \setminus C_i$.

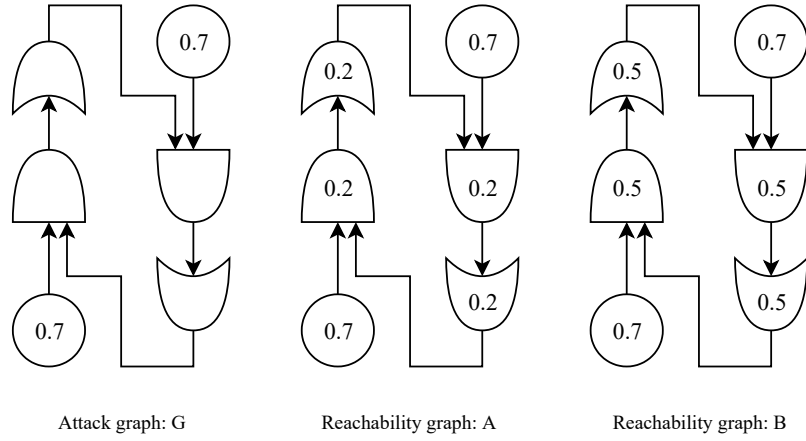


Figure 3.4.1: An example of the reachability-graph undecidability

3. Min-AND node: a node $x \in E$, with reachability $(x) = f_{[x]}(\text{min-AND}(\text{parents}(x)))$ where $f_{[x]}$ is the arbitrary filter function defined for each $x \in E$.

We call the pair of G and its associated reachability defined above as a *reachability graph*. For convenience in the later discussion, we also define $\text{children}(x) = \{e \mid (x, e) \in R_i\} \cup \{c \mid (x, c) \in R_i\}$.

The reachability graph above is an almost straightforward example following the framework given by Wang et al. [167, 168]. However, it is ill-defined. For example, in Figure 3.4.1, where we assume that the filter function $f_{[x]}$ is defined as $f_{[x]}(x) = x$, both the reachability graph A and B give the legitimate reachabilities for the attack graph G . The number in the node represents the reachability for the node.

Also, there is another problem. The initial reachabilities define the reachabilities only for external nodes. We need to compute the reachabilities for other nodes.

We define Algorithm 3.1 to compute the whole reachability graph uniquely:

The following claims hold for Algorithm 3.1:

Proposition 6. *Every temporal reachability increases monotonically during the process of the algorithm.*

Proof. As preparation, we define several terms. We call a node other than external nodes in an attack graph $G = (E \cup C, R_r \cup R_i)$ an *internal node*. In Algorithm 3.1, the temporal reachabilities of internal nodes are initialized to be zero. The algorithm updates each reachability of an internal node whenever the value is inconsistent with its parents' reachabilities. When an internal node has such inconsistency, we say the node is *unstable*. On the other hand, if the node has consistent reachability, we say it is *stable*. We call a graph containing an unstable node an *unstable graph* — otherwise, a *stable graph*. The algorithm is a procedure to derive a stable graph from an initial unstable graph. In the following discussion, we call each iteration of the while loop in the algorithm a *step*.

If all the reachabilities in the graph equal to zero, then the graph is stable. Thus, the assignment in line 5 in the algorithm intervenes in this stability by injecting non-zero positive values at external nodes. During the procedure, the first update in line 20 increases the “reachability(x)” since such an update occurs on an internal node holding external nodes with non-zero initial reachabilities as its parents. Similarly, any following update done by line 20 is based on the prior updates, including the initial reachabilities assignment, each of which is an increase of the reachability. Therefore, the

Algorithm 3.1 Reachability-graph analysis

Input: $G = (E \cup C, R_r \cup R_i)$: an attack graph to be analyzed. $I = \{(c_k, p_k) \in C_i \times [0, 1] \mid c_k \in C_i\}$: initial reachabilities. $F = \{f_{[x]}\}$: the set of filter functions defined for each $x \in E \cup C$.**Output:**reachability(x) : $(E \cup C) \rightarrow [0, 1]$: reachabilities of all the nodes in G

```

1: /* Initialize the temporal reachabilities and the update queue */
2:  $Q \leftarrow \emptyset$ 
3: for  $x \in E \cup C$  do
4:   if  $x \in C_i$  then
5:     reachability( $x$ )  $\leftarrow p$  where  $(x, p) \in I$ 
6:      $Q \leftarrow Q \cup \text{children}(x)$ 
7:   else
8:     reachability( $x$ )  $\leftarrow 0$ 
9:   end if
10: end for

11: while  $Q \neq \emptyset$  do
12:    $x \leftarrow$  take an arbitrary element in  $Q$ 
13:   /* The following max-OR/min-AND refers to the temporal "reachability" defined
      above */
14:   if  $x$  is a max-OR node then
15:      $p' \leftarrow f_{[x]}(\text{max-OR}(\text{parents}(x)))$ 
16:   else
17:      $p' \leftarrow f_{[x]}(\text{min-AND}(\text{parents}(x)))$ 
18:   end if
19:   if  $p' \neq \text{reachability}(x)$  then
20:     reachability( $x$ )  $\leftarrow p'$ 
21:      $Q \leftarrow Q \cup \text{children}(x)$ 
22:   end if
23: end while
24: return reachability

```

update is also increasing because of the monotonicity of max-OR, min-AND, and filter functions. After all, every update increases the reachabilities monotonically. $\square$

Lemma 7. *Suppose we have an input attack graph, and let C_i be the set of initial conditions, and $f_{[x]}$ be the filter function defined for each node x in the attack graph. For every temporal and final reachability to x_n , there exists a series of nodes $x_0 \in C_i, \{x_1, \dots, x_n\} \subseteq E \cup C$ satisfying*

$$\text{reachability}(x_n) = f_{[x_n]} \circ f_{[x_{n-1}]} \circ \dots \circ f_{[x_1]} \circ I(x_0)$$

and composing an acyclic path.

Proof. At first, the set $V = \{\text{reachability}(x) \mid x \in E \cup C\}$ equals $\{0\} \cup \{I(c) \mid c \in C_i\}$. Implicitly, the assignment in line 20 updates this set. Since this assignment derives from the action in either line 15 or 17, we can regard this update as the replacement of $\text{reachability}(x)$ —which is either zero or $I(x)$ —with a value $f_{[x]}(v)$ where v is a reachability taken from V . Note that v is determined by either max-OR ($\text{parents}(x)$) or min-AND ($\text{parents}(x)$), and both of them just choose an element in V . Therefore, after this update, the set V comprises zero, some $I(c)$, and a value $f_{[x]}(I(c))$. Similarly, any following update done by line 20 modifies the set V by replacing its element with a value determined by applying a filter function $f_{[x]}$ to an element in V or a value determined by replacing a portion of a filter chain in V with another existing chain in V . Thus, in any step, the set V comprises zero, some $I(c)$, and values each of which is determined by applying a chain of filter functions to $I(c)$.

Now, let us focus on the value in V such that it is expressed with a chain of filter functions and initial reachability. As the composition implies, each filter function in the chain corresponds to a node x in the graph. We want to show here that the chain is always acyclic, i.e., the chain contains no duplicative application of the same filter function.

Suppose there exists $x_n \in E \cup C$ with a reachability (x_n) such that it corresponds to a chain of filter functions with at least one duplicative application. We can take a duplicative filter function $f_{[d]}$ from the chain. Each application of $f_{[d]}$ occurs at some step of the procedure. Let us denote the first step number of the application of $f_{[d]}$ by s_1 and denote the second step number by s_2 .

The function $f_{[d]}$ takes either max-OR ($\text{parents}(d)$) as its parameter in line 15 or min-AND ($\text{parents}(d)$) as its parameter in line 17. Let v_1 denote this parameter at step s_1 . Similarly, let v_2 denote this parameter provided to $f_{[d]}$ at step s_2 . To realize the step s_2 , the condition $v_2 \neq v_1$ must be held at step s_2 due to the check “ $p' \neq \text{reachability}(d)$ ” in line 19. [Proposition 6](#) and the monotonicity of max-OR/min-AND imply that $v_1 \leq v_2$. At the same time, there exists a set of filter functions $f_1, \dots, f_k$ where $k = s_2 - s_1$ such that $v_2 = f_1 \circ \dots \circ f_k \circ f_{[d]}(v_1)$. It implies that $v_2 \leq f_{[d]}(v_1) \leq v_1$, i.e., $v_2 \leq v_1$. However, the conditions, $v_2 \neq v_1$, $v_1 \leq v_2$, and $v_2 \leq v_1$, cannot hold simultaneously. Therefore, the assumption of the duplicative application is denied, and the claim holds. $\square$

Theorem 8 (Termination of reachability graph analysis). *Given the input attack graph is finite, the procedure of reachability-graph analysis always terminates within finite steps.*

Proof. This property almost immediately follows from [Proposition 6](#) and [Lemma 7](#). First, every temporal reachability monotonically increases. Thus, if the temporal reachability to a node gets its maximum among the possible filter chains to it, it will not change thereafter. Second, due to the finiteness of the input attack graph, the set of possible filter chains to a node is also finite. Therefore,

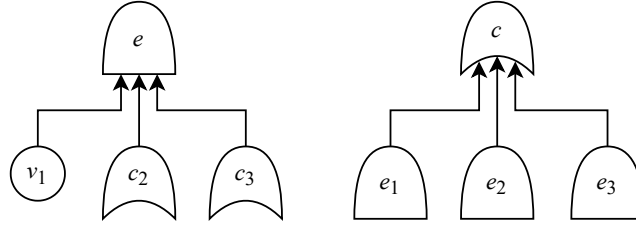


Figure 3.4.2: Typical minimal parts in reachability graphs

the update of reachabilities cannot occur infinitely; i.e., it must terminate. See also the discussion in [Section 3.6.1](#). $\square$

Theorem 9 (Confluence of reachability graph analysis). *The resultant reachabilities identified by the reachability-graph analysis is determined uniquely irrespective of the choice of update node in line 12 of the algorithm.*

Proof. Let both A and B are stable graphs obtained from the reachability-graph analysis applied to the same attack graph G and its initial reachabilities I . If $A \neq B$, then there exists a node x with different reachability between A and B . Let $A(x)$ be the reachability assigned to node x in A , and $B(x)$ be the reachability to node x in B . Without loss of generality, we can identify the node $a \in \{x \mid x \in E \cup C, B(x) < A(x)\}$ such that a is the first node satisfying $B(a) < \text{reachability}(a)$ during the procedure generating A . Let s_a be the step number in which reachability(a) exceeds $B(a)$ first time. At step s_a , the condition $\forall p \in \text{parents}(a), \text{reachability}(p) \leq B(p)$ holds because a is the first node that its reachability exceeds the reachability of the same node in B . At the same time, it also implies that $\text{reachability}(a) \leq B(a)$ due to the monotonicity of max-OR, min-AND, and filter functions. It contradicts with the requirement $B(a) < \text{reachability}(a)$. Therefore, the assumption $A \neq B$ is denied, and the claim holds. $\square$

3.4.3 The interpretation of reachabilities

Given an attack graph and its associated initial reachabilities, both Bayesian attack graphs and reachability graphs determine the reachabilities to nodes in the graph. However, the interpretation of reachabilities is different. We examine the difference here to understand the characteristics of reachability graphs.

The fundamental assumption behind the reachability graph is that the reachability will be determined by the skill level required for exploits. See [Figure 3.4.2](#), depicting typical minimal parts composing reachability graphs. For simplification, we consider the case where the filter functions are identity, i.e., $f(v) = v$.

In the figure, the exploit node e has three connections from condition nodes, v_1 , c_2 , and c_3 . The preconditions connecting to an exploit node are composed of initial and inferred conditions; v_1 corresponds to the former, and c_2 and c_3 to the latter. Following the definition, $\text{reachability}(e) = \min\{\text{reachability}(v_1), \text{reachability}(c_2), \text{reachability}(c_3)\}$. Since $\text{reachability}(v_1)$ is given initial reachability, it is kept unchanged during the reachability calculation. Also, $\text{reachability}(c_2)$ and $\text{reachability}(c_3)$ grow gradually from zero up to one during the process. This implies that the initially given $\text{reachability}(v_1)$ defines the upper limit of $\text{reachability}(e)$. In general, initial conditions are used to represent such as network topology, configuration, and vulnerabilities in the network. The assign-

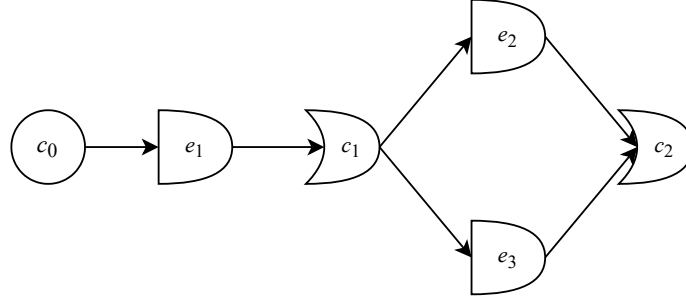


Figure 3.4.3: An input attack-graph for reachability calculation

ment of initial reachability to an initial condition denoting vulnerability indicates, on a relative scale, how many attackers can accomplish the exploit depending on that vulnerability, i.e., the required level for the exploit: the less reachability means, the harder, more difficult exploit to perform. At the same time, when an exploit node has non-zero reachability r , every precondition has some non-zero reachability no less than r . It means that top r attackers succeed in the exploit, while more or even attackers satisfy the preconditions. In other words, an exploit node sorts out all the skillful attackers from the top since any exploit requires every precondition to be met.

On the other hand, in the figure, the condition node c has the connections from e_1 , e_2 , and e_3 . Due to the nature of attack graphs as bipartite graphs, there is no initial condition node connecting to a condition node. The reachability (c) is determined solely by the reachabilities to the exploit nodes e_1 , e_2 , and e_3 and is the maximum among them. The implication is that every attacker reachable to one of its source exploits can satisfy the condition c . In other words, a condition node accepts all the skillful attackers.

As a whole, we assume that any skilled enough attacker can always proceed without fail, even when the exploits have some probabilistic nature, e.g., heap spraying. That is, each reachability in reachability graphs shows the worst-case reachability.

Let us see the detail by calculating the reachabilities for [Figure 3.4.3](#).

Firstly, we examine a simple, intuitive way of reachability assignment following [169].

Remind that an attack graph $G = (E \cup C, R_r \cup R_i)$ contains two types of nodes, the exploits E and the conditions C . Let us define a function $p : (E \cup C) \rightarrow [0, 1]$, which assign individual scores to each node in G . We define the probability P of the event implied by an exploit e or a condition c in G as follows (Be cautious in distinguishing the use of P and p , and that $P(e)$ means $P(E = \text{True})$):

$$P(e) = p(e) \prod_{c \in \text{parents}(e)} P(c), \text{ and}$$

$$P(c) = \begin{cases} p(c) & (\text{parents}(c) = \emptyset) \\ p(c) \left\{ 1 - \prod_{e \in \text{parents}(c)} (1 - P(e)) \right\} & (\text{parents}(c) \neq \emptyset) \end{cases}.$$

The definition above follows the approach by Wang et al. [169]. Let us apply this definition to the case in [Figure 3.4.3](#). Suppose that the initial event probability $P(c_0) = 1.0$, and every individual score, such as $p(e_1)$ and $p(c_1)$, equals to 0.8. Then, the probability $P(c_2)$ is calculated as follows:

$$P(c_1) = p(c_1) P(e_1) = p(c_1) p(e_1) P(c_0)$$

$$= p(c_1) p(e_1) p(c_0) = 0.8 \cdot 0.8 \cdot 1.0 = 0.64,$$

$$P(e_2) = p(e_2) P(c_1) = 0.8 \cdot 0.64 = 0.512,$$

$$P(e_3) = p(e_3) P(c_1) = 0.8 \cdot 0.64 = 0.512,$$

$$\begin{aligned} P(c_2) &= p(c_2) \{1 - (1 - P(e_2))(1 - P(e_3))\} \\ &\simeq 0.609. \end{aligned}$$

The probability $P(c_2)$ is also the reachability (c_2) determined in this approach. It seems intuitive and easily understandable.

In the Bayesian attack graph, the probability is different. While the approach above assumes that e_2 and e_3 are independent, the paths leading to them are identical until halfway through. Therefore, the two are never independent. The correct probability for c_2 considering their dependencies is as follows (Note that, in this calculation, the scores such as $p(c_2)$ and $p(e_2)$ play the role of the configuration parameters μ, ν in Noisy-OR/AND.):

$$\begin{aligned} P(c_2) &= \sum_{E_3, E_2, C_1, E_1, C_0} P(c_2 | E_2, E_3) P(E_2 | C_1) P(E_3 | C_1) P(C_1 | E_1) P(E_1 | C_0) P(C_0) \\ &\simeq 0.557. \end{aligned}$$

Now, let us see how our approach, the reachability graph analysis, handles the reachability in yet another different way. Here, we use filter functions $f_{[x]}(v) = p(x) \cdot v$. Then, the reachability to c_2 is calculated as follows (This calculation omits the detailed process of the algorithm.):

$$\text{reachability}(e_1) = f_{[e_1]}(\text{min-AND}(\{c_0\})) = 0.8 \cdot 1.0 = 0.8,$$

$$\text{reachability}(c_1) = f_{[c_1]}(\text{max-OR}(\{e_1\})) = 0.8 \cdot 0.8 = 0.64,$$

$$\text{reachability}(e_2) = f_{[e_2]}(\text{min-AND}(\{c_1\})) = 0.8 \cdot 0.64 = 0.512,$$

$$\text{reachability}(e_3) = f_{[e_3]}(\text{min-AND}(\{c_1\})) = 0.8 \cdot 0.64 = 0.512,$$

$$\begin{aligned} \text{reachability}(c_2) &= f_{[c_2]}(\text{max-OR}(\{e_2, e_3\})) = 0.8 \cdot 0.512 \\ &\simeq 0.410. \end{aligned}$$

As we can see, the reachability graph has a different probability assignment.

The difference between these three cases comes from handling events convergence at node c_2 . Figure 3.4.4 and the Venn diagram in Figure 3.4.5 explains this difference more intuitively.

Figure 3.4.5 shows the volume of overlapped attacks in node e_2 in Figure 3.4.4. By assumption, attackers start at node c_0 and penetrate deeper into the attack graph. To get through node e_2 , the attacker must achieve both of c_1 and c_2 . Then, let us assume that twelve attackers had achieved c_1 , and ten attackers had achieved c_2 . Under this circumstance, how many attackers can succeed to get to e_2 ? It depends on the case. When the two groups have no common part, no one can get to e_2 . On the other extreme, if the ten achievers are a subset of the twelve achievers, then the ten attackers

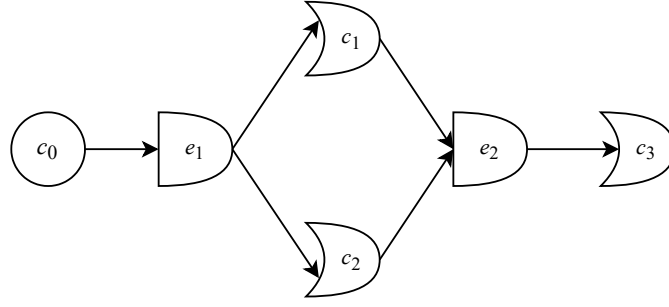


Figure 3.4.4: Another attack graph for reachability calculation

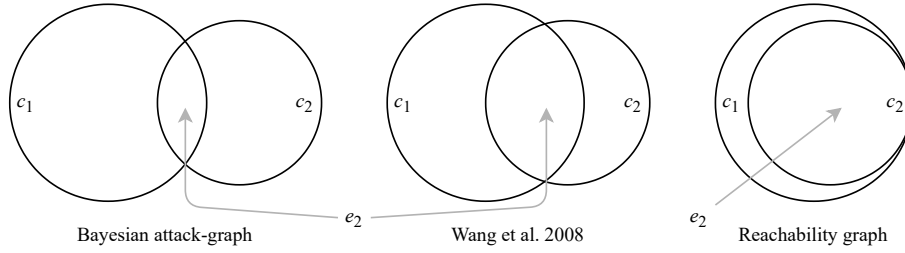


Figure 3.4.5: The overlapping volume of convergent events

will get to e_2 . [Algorithm 3.1](#) always assumes this complete overlap at AND/OR nodes in the graph. In contrast, the approach by Wang et al. [169] assumes that this overlap is limited to the volume determined as the product of reachabilities at the precondition nodes. The Bayesian attack graph determines this overlap depends on the overlaps of paths up to the AND/OR node. This difference in the overlapping volume is the root cause of the reachabilities difference. At AND nodes, the small overlap means less reachability. At OR nodes, the small overlap means more reachability since it enlarges the volume of $c_1 \cup c_2$. In reachability graph analysis, the accumulated difficulty of attacks in the most accessible path determines reachability. The reachability “0.410” indicates that 41% of the attackers going sequentially along with the best course can accomplish the goal, while other approaches compute the reachability as a weighted sum of all the relevant paths.

This difference in the reachability semantics also affects how we can decrease them. In practice, if we want to strengthen the system’s security, we have to focus on the weakest part of the system. However, in the standard approach like Bayesian attack graph, we can lower the reachabilities without focusing on that. In contrast, it is required in the reachability graph analysis since its reachabilities are determined by the path composed of the most advantageous sequence for the attacker. This point is a discreet but substantive virtue of our approach.

3.5 Experiment

3.5.1 Performance comparison

This subsection compares the computational efficiencies of the Bayesian attack graph and the reachability graph through a numerical experiment. [Figure 3.5.1](#) shows the general structure of the networks to be analyzed in this experiment.

The network comprises four layers — databases, servers, gateways, and clients — and the Internet connection. We assume that the attacker starts the penetration from the Internet. The network

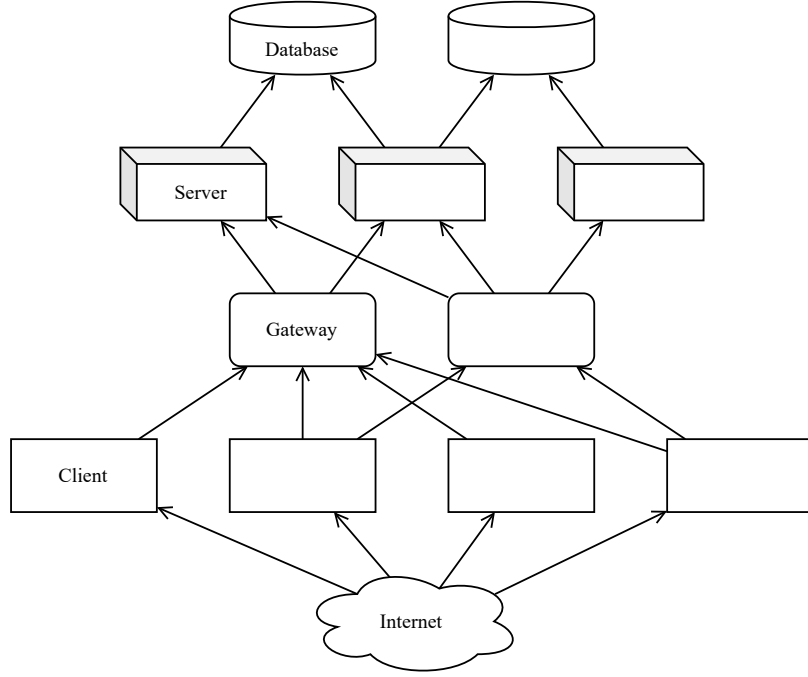


Figure 3.5.1: General structure of the experimental network

Class name	nw1	nw2	nw3	nw4	nw5
# of databases	1	2	3	3	3
# of servers	2	3	4	4	4
# of gateways	3	3	4	4	5
# of clients	24	30	48	56	80
# of clients per gateway	8	10	12	14	16
# of vulnerabilities per client	2	3	3	4	4
# of total vulnerabilities	58	86	119	155	196
# of attack graph nodes	267	420	641	821	1,114
# of attack graph edges	382	588	921	1,257	1,740
# of nodes + edges	649	1008	1,562	2,078	2,854

Table 3.1: Network size parameters for the experiment

topology forms a combination of overlapped trees, i.e., non-polytree DAG. Also, we assume each node except the Internet has at least one vulnerability.

Following the structure above, we examine five classes of networks with the number of nodes defined in Table 3.1.

The number of total vulnerabilities is the total of unique vulnerabilities over the network under this consideration. Note that we assume that client nodes share some vulnerabilities since most client computers have the same configuration. Figure 3.5.2 illustrates the vulnerabilities in a node (host) and the relationship among them as an attack graph.

This partial attack graph contains seven vulnerabilities and is composed of two layers. The attacker penetrates this node through the “In” event. If it successfully compromises this node, then it approaches to the “Out” event. To accomplish the “Out” event, it needs to exploit one of first layer vulnerabilities $v_{1,1}, v_{1,2}, v_{1,3}$ and another one of second layer vulnerabilities $v_{2,1}, v_{2,2}, v_{2,3}$. Otherwise, v_0 must be compromised. The left-most vulnerability v_0 represents an inherent vulnerability in each host, which is a simplified representation of a zero-day vulnerability. Each vulnerability has a non-

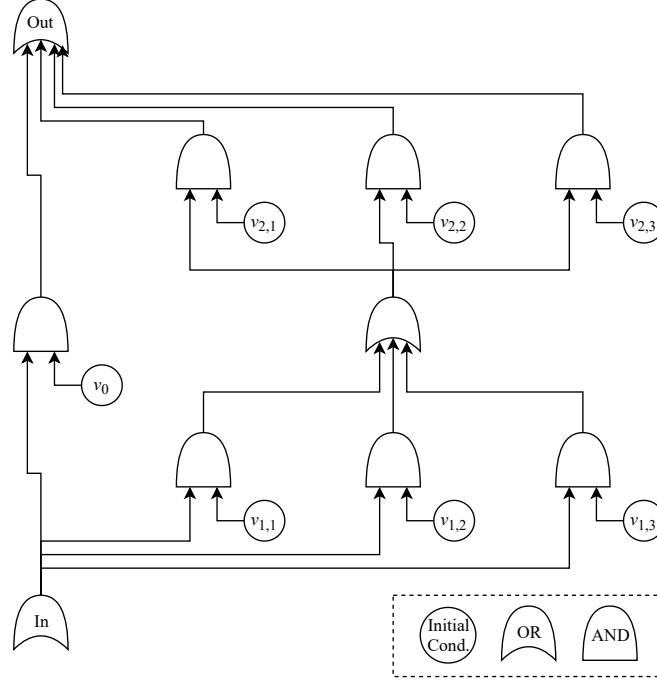


Figure 3.5.2: An example partial attack graph within a node

zero reachability given by initial reachabilities. We compose an attack graph by binding small graphs like in Figure 3.5.2; each small graph is composed of two layers with one zero-day vulnerability and other vulnerabilities.

We apply the two approaches in Section 3.4 to randomly generated attack graphs explained above. The data set comprises five classes defined in Table 3.1, where each class is a set of fifty networks. Therefore, we examine 250 cases overall. For the Bayesian attack graphs, the configuration parameters μ and ν for Noisy-OR/AND are set to one. For the reachability graphs, we assign a filter function $f_{[x]}(v) = a_x \cdot v$ for each AND node x where $a_x \in [0, 1]$ is a constant defined for x . For OR nodes, we assign the filter function $\text{ident}(v) = v$.

We compute the reachabilities to all the nodes in an attack graph by the two approaches. Both algorithms are implemented in Python 3.7 and are executed by cpython-3.7.3. The served hardware has an Intel Core i7-9750HQ 2.6GHz processor (6 cores with HyperThreading) and 32GB memory. Figure 3.5.3 to Figure 3.5.5 show the comparison results. In Figure 3.5.3 and Figure 3.5.4, the bars are the average times in seconds to calculate all the reachabilities, and the error bars represent the standard deviation.

Figure 3.5.5 shows the box plot with 4-quantiles in which the processing times are compared. Y-axis values show the ratios calculated by “the processing time in Bayesian attack graph” / “the processing time in reachability graph”. The circles are outliers with values either more than $Q3 + 1.5(Q3 - Q1)$ or less than $Q1 - 1.5(Q3 - Q1)$ where $Q3$ and $Q1$ denote the third and the first quantile, respectively.

It is easy to see that the reachability-graph analysis outperforms the Bayesian attack graph based approach. While the reachability graph analysis scales linearly in the size of the attack graph, the Bayesian attack graph based approach does not. Of course, their performance is not directly comparable since their semantics are different. Still, this clear advantage could make our approach an attractive alternative to the Bayesian attack graph.

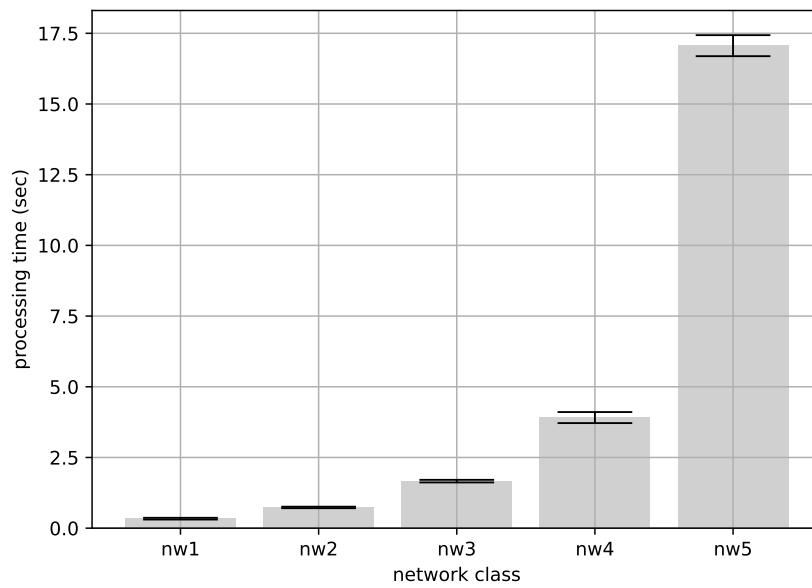


Figure 3.5.3: Processing times of Bayesian attack graphs

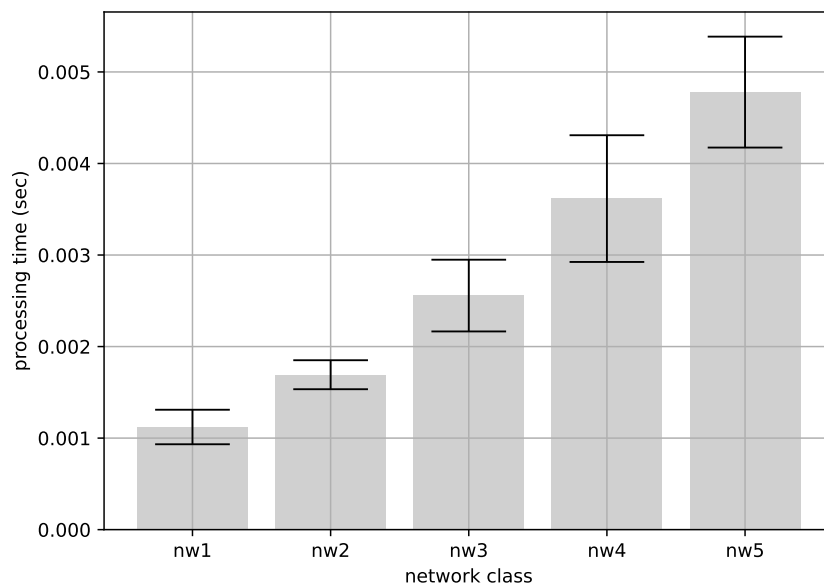


Figure 3.5.4: Processing times of reachability graphs

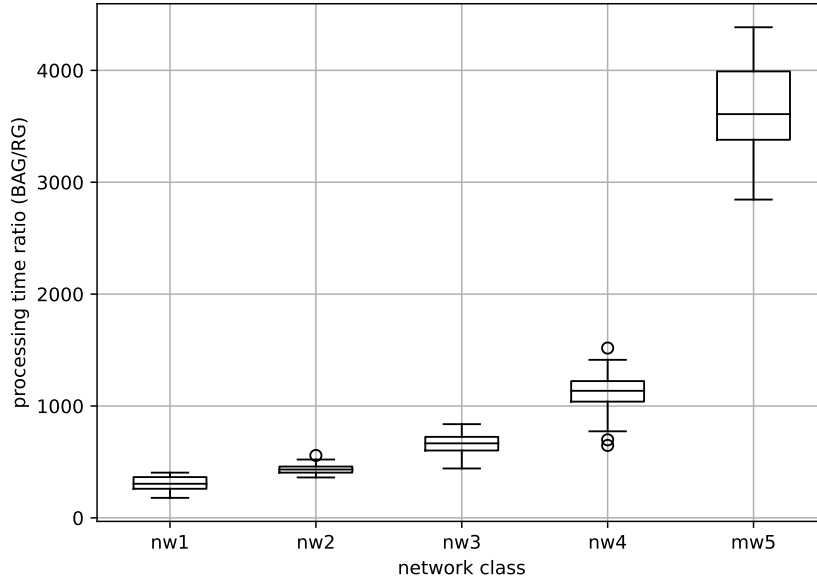


Figure 3.5.5: Comparison of processing times

Class name	nw6	nw7	nw8	nw9	nw10
# of databases	6	6	8	8	8
# of servers	8	12	16	16	16
# of gateways	12	16	24	32	64
# of clients	576	800	1,320	2,176	3,840
# of clients per gateways	48	50	55	136	240
# of vulnerabilities per client	8	12	16	20	24
# of total vulnerabilities	1,178	1,818	3,016	4,872	8,440
# of attack graph nodes	9,629	16,901	33,233	65,313	138,769
# of attack graph edges	18,702	35,990	75,496	154,536	336,264
# of nodes + edges	28,331	52,891	108,729	219,849	475,033

Table 3.2: Network size parameters for the large-scale experiment

3.5.2 Applicability to large-scale, cyclic networks

In this section, we apply the reachability-graph analysis to large-scale networks. Table 3.2 shows the size parameters of the networks.

Other experiment configurations are as same as the previous experiment, i.e., 50 cases per class. Figure 3.5.6 shows the result.

The attack graphs in the largest case have more than 130,000 nodes and 330,000 edges. Bayesian attack graph based approach cannot handle this class of networks due to its computational cost. We can say that reachability-graph analysis has the potential to analyze practical scale problems.

In addition to the above experiment, let us see how much the cycles in an attack graph affect the performance. For the purpose, we replace some of the gateway nodes in nw10 with hub nodes and cut some connections between clients and the Internet. In brief, when a node connects to a hub node, the hub node also connects to the node. As a result, the topology in (A) is transformed into (B), as shown in Figure 3.5.7. This modification increases the topological distance from the Internet

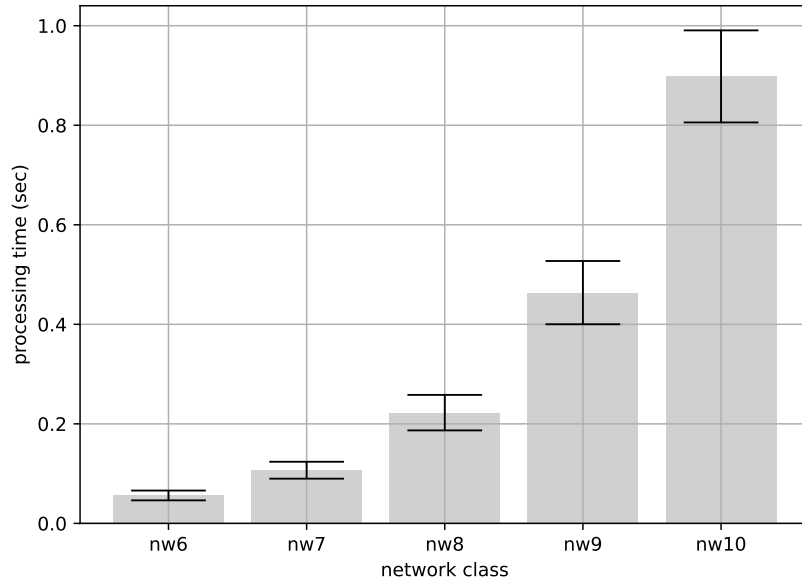


Figure 3.5.6: Processing times of large-scale reachability graphs

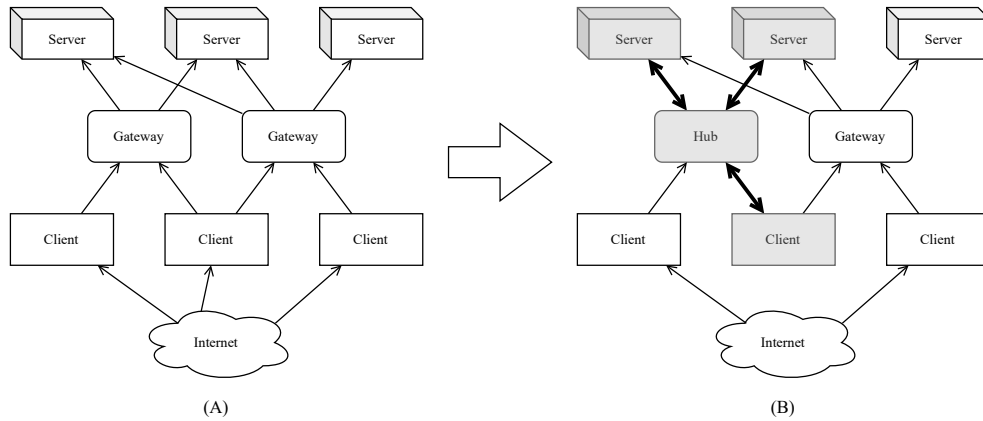


Figure 3.5.7: Replacement of a gateway with a hub

to the clients and enforces cycles handling relevant to them.

We apply this transformation to the fifty networks in nw10 while varying the number of hubs from 0 to 56, in 8 increments. Figure 3.5.8 shows the distribution of the processing times with the reachability-graph analysis of those networks.

As the number of hubs increases, the processing times are getting longer. Nevertheless, the performance degradation is limited. Even when most of the gateways are replaced with hubs, it does not exceed the doubled processing time in the baseline case where no hubs. This result is a natural consequence implied by Lemma 7. The lemma guarantees that the analysis process does not loop the graph repeatedly.

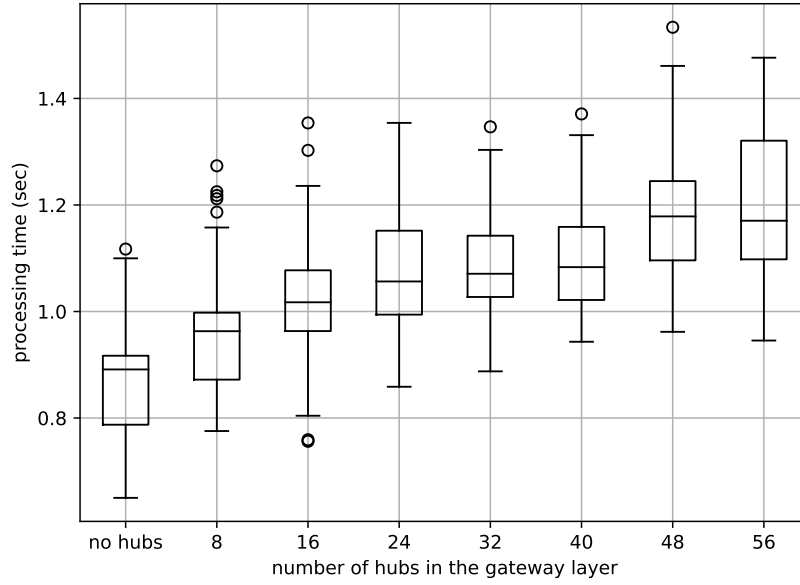


Figure 3.5.8: Processing times of cyclic reachability-graphs

3.6 Discussions

3.6.1 Computational complexity

Belief propagation applied to polytree has the computational complexity of $O(N \exp(w))$ where N denotes the number of nodes in the graph, and w is the maximum number of parent nodes subject to marginalization. Even when we use the efficient computation applicable to Noisy OR/AND, it is still $O(Nw)$. The complexity of loopy belief propagation depends on the problem. While it works as same as belief propagation in the best case, the process never stops in the worst case. In the Bayesian attack graph, the width parameter w can be considerable since it corresponds to the number of clients connected to a server or a gateway. It seems the leading cause of the non-linear behavior seen in Figure 3.5.3.

In contrast, reachability-graph analysis always terminates within finite steps. However, the worst-case computational complexity in the update count of reachabilities is $O(\mathcal{P})$, where $\mathcal{P}$ denotes the number of all the directed paths from each initial condition to nodes in the graph; it can be enormous. We can derive this estimation as follows. Lemma 7 assures that a filter chain between a node and an initial condition determines each temporal reachability. Therefore, the update count of temporal reachabilities could amount to the number of paths if the algorithm enumerates the paths from the one with the lowest reachabilities to the highest.

The worst-case complexity above is easily avoidable. Using a queue without duplication to implement Q in the algorithm enables suppressing it to $O(N^2)$, where N denotes the number of nodes. The explanation is as follows. By using the queue, the algorithm calculates reachabilities to nodes according to the hop count from the initial conditions. As a result, whenever a reachability with n filter functions is determined, every other reachability with less than n filter functions has been determined beforehand. This does not mean that the final reachability to a node is calculated only once. However, the maximum length of the filter function chains does not exceed N , and the count

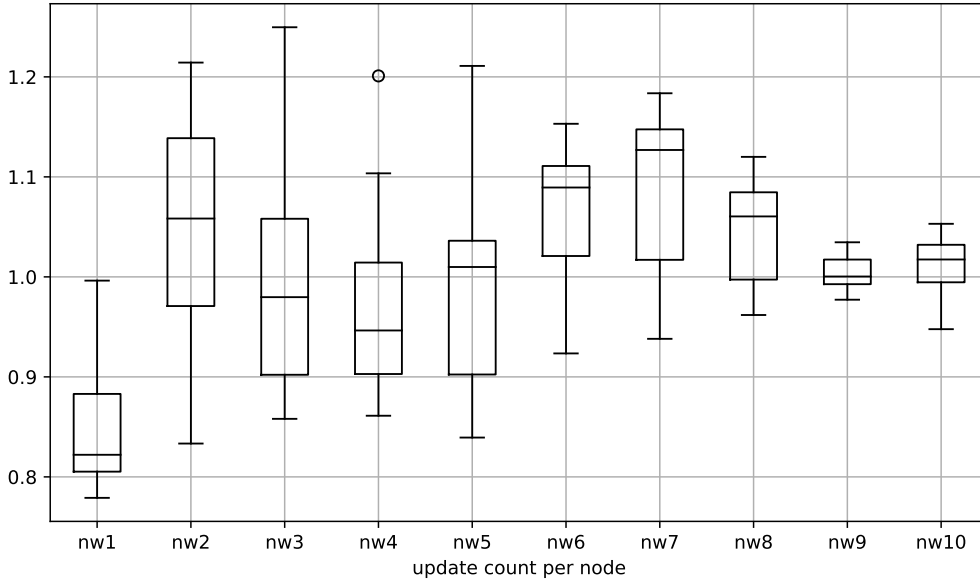


Figure 3.6.1: Update counts per node in acyclic attack graphs

of nodes processed for each iteration in the algorithm also does not exceed N , too. Thus, the total update count cannot exceed N^2 . Note that using a queue without duplication corresponds to topological sorting in acyclic graphs.

In acyclic cases, the reachability to a node will be updated only once when the node choice at line 12 in the algorithm follows the topological order, and precedence is given to appropriate nodes depending on whether the unstable node is a condition or an exploit node. There, among the group of the same type of nodes with the same hop count from initial conditions, exploit nodes are sorted in descending order of their reachability, and condition nodes are sorted in ascending. In that case, the worst-case complexity becomes $O(N)$. The program in Section 3.5 uses an unordered set to implement Q in the algorithm but roughly follows this setting since the processed graphs are almost acyclic and the exploit nodes block the depth-first calculation of reachabilities. Moreover, due to the monotonicity guaranteed by Proposition 6, any update requires the propagation of increased reachability. As the “while” iteration in Algorithm 3.1 goes further, the propagated reachability attenuates since filter functions satisfy $f(v) \leq v$. It means that the late-propagated reachabilities are getting less capable of updating even when the order of node choice does not strictly follow the topological order.

Figure 3.6.1, showing the update counts per node in the acyclic cases in Section 3.5, backs up the estimation above. The update counts per node keep around 1.0. In short, we can expect that the computational complexity in simple acyclic attack graphs is nearly linear. However, this estimation focuses on the update counts per node, and it is different from the actual processing times per node. The latter result is shown in Figure 3.6.2. The processing times get increase as the graphs get larger. Note that the size of “nw6” is about ten times larger than “nw5.” This result implies that the time complexity of the algorithm is not linear. It depends on various factors, including, but not limited to, the topological structure of the attack graph, the distribution of initial reachabilities, and the order of element extraction at line 12 in the algorithm. These properties contribute to the count

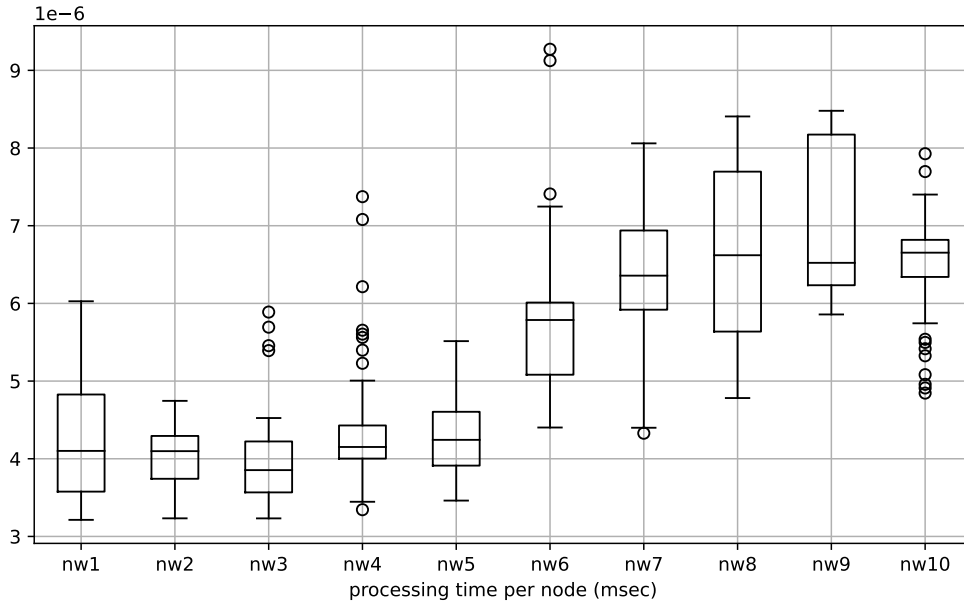


Figure 3.6.2: Processing times per node in acyclic attack graphs

of reachability calculations performed at line 15 and 17. Figure 3.6.3 shows the averaged counts of these calculations per node. We can see the increase in evaluation counts, which is the main cause of the non-linear change in the processing times.

As Figure 3.5.8 shows, cycles do not affect the discussion above so much. Figure 3.6.4 to Figure 3.6.6 clarify the performance detail.

3.6.2 Variations in filter functions

In Section 3.5, we used the simple filter function $f_{[x]}(v) = a_x \cdot v$ where $a_x \in [0, 1]$ is a constant chosen for each x . This is not the only form of filter functions available for use. The necessary and sufficient requirements for any filter function f are (1) $\forall v \in [0, 1], f(v) \leq v$, and (2) $\forall v_1, v_2 \in [0, 1], v_1 \leq v_2 \implies f(v_1) \leq f(v_2)$. One of such legitimate functions is the sigmoid function with shift

$$\varsigma_a(v) = \frac{1}{1 + e^{-a(v-c_1)}} - c_2$$

where $0 \leq a, c_1, c_2$. We can use this function to express attacks causing resource starvation because the sigmoid function's output increases drastically when the input exceeds some threshold. This behavior resembles DoS attacks. In a similar way, attacks depriving finite resources can be handled with output capped functions like as follows: $h_a(v) = \min\{a, v\}$ where $a \in [0, 1]$. This function $h_a(v)$ limits the number of successful attacks at node v . It corresponds to a situation where many attackers are trying to seize a limited number of resources such as processors, network interfaces, and other physical devices.

In this chapter, we limit reachability within $[0, 1]$. However, we can remove this limitation without losing the properties implied by Lemma 7, Theorem 8, and Theorem 9. This limitation is introduced to make reachabilities comparable with probabilities. When we use the most simple filter function, $f_{[x]}(v) = v$, and also remove that limitation, the reachability is equivalent to the skill metric discussed

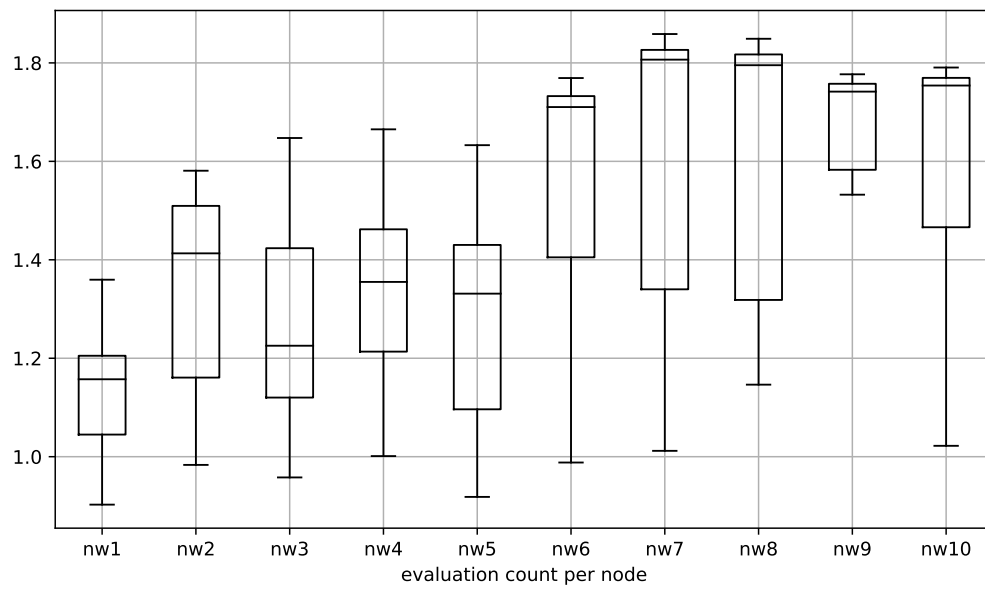


Figure 3.6.3: Evaluation counts per node in acyclic attack graphs

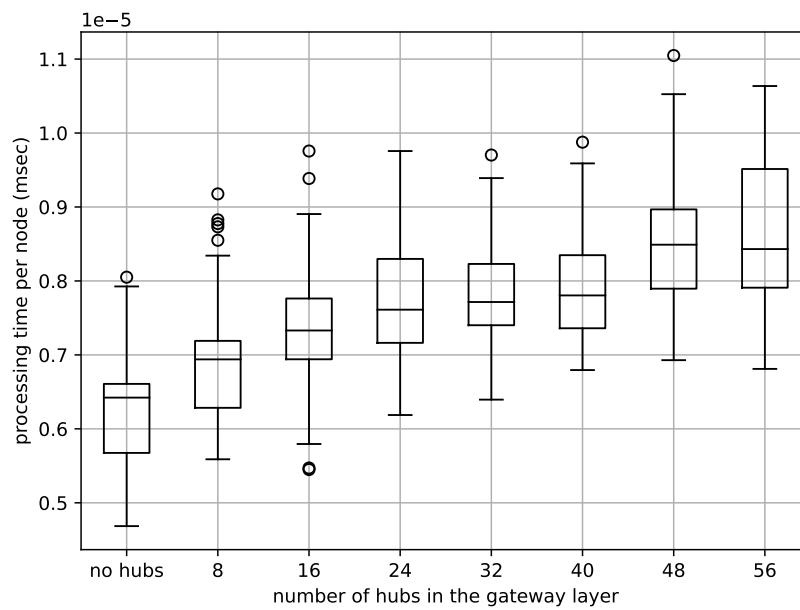


Figure 3.6.4: Processing timers per node in cyclic attack graphs

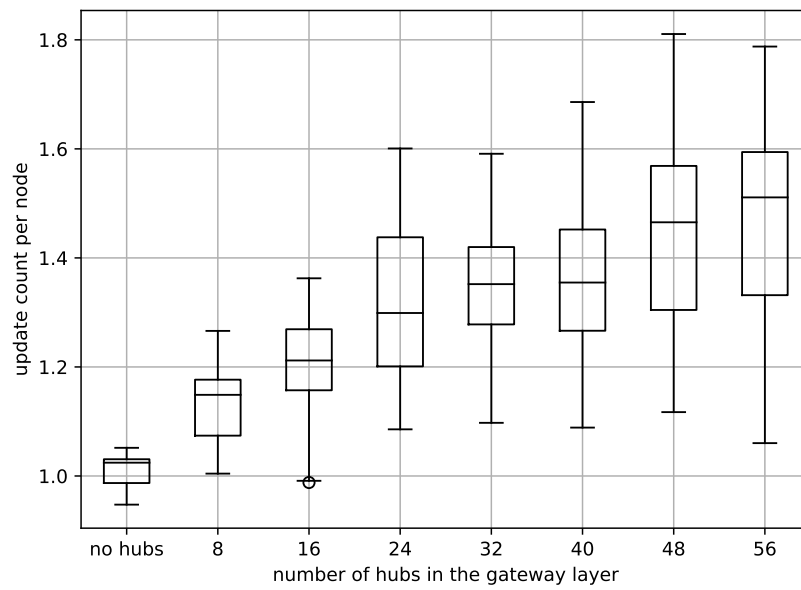


Figure 3.6.5: Update counts per node in cyclic attack graphs

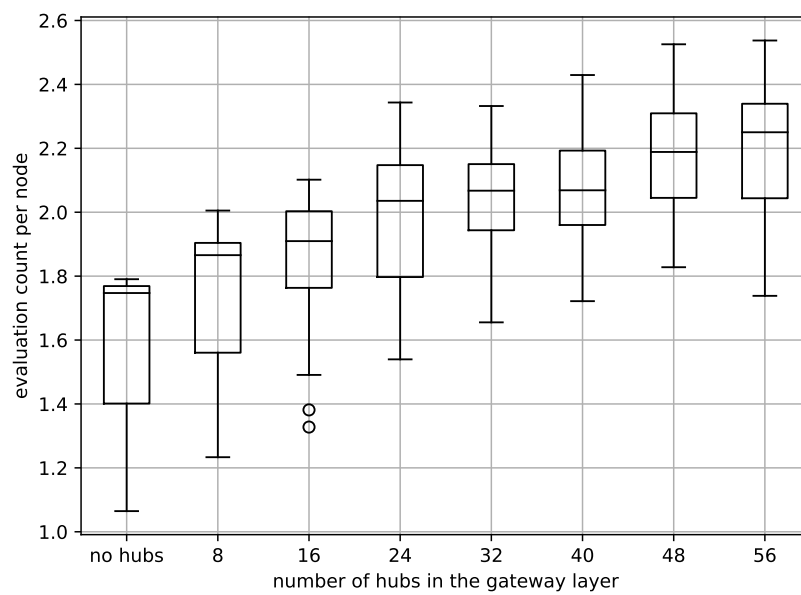


Figure 3.6.6: Evaluation counts per node in cyclic attack graphs

in [13].

3.6.3 Monotonicity of reachability-graph analysis

Let both $v = (v_1, \dots, v_n)$ and $w = (w_1, \dots, w_n)$ be vectors in $\mathbb{R}^n$. If the two vectors v and w satisfy that $v_i \leq w_i$ for every $i \in \{1, \dots, n\}$, then we say v is *less than or equals to* w and denote it by $v \leq w$. Additionally, let $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a function which satisfies $F(v) \leq F(w)$ whenever $v \leq w$. Then, we say that F is *monotonic*.

We can regard the initial reachabilities in a reachability-graph analysis as a vector. Similarly, we can interpret the process of reachability-graph analysis as a function that maps an initial reachability vector to another reachability vector representing the whole reachabilities in a graph. Under this interpretation, the reachability-graph analysis function F satisfies $F(I_1) \leq F(I_2)$ for any pair of initial reachabilities $I_1 \leq I_2$. That is, the reachability-graph analysis is monotonic about the initial reachabilities and the resultant reachabilities. This property derives immediately from [Lemma 7](#) and the monotonicity of filter functions.

3.6.4 Other related works

In this subsection, we refer to recent studies not mentioned in [Section 3.2](#). By comparing with them, we clarify why we stick to scalable, probability calculation algorithms applicable to attack graphs as cyclic AND/OR graphs.

Information security management is a kind of risk management, so it is natural to calculate risks during the process. In general, a risk is defined as the product of a probability and an impact. This explains the fundamental importance of probability calculation. For example, Chung et al. [16] propose a comprehensive framework for virtual network systems security management, in which attack probability calculation virtually equivalent to that of [169] is incorporated. The studies by Ghasemigol et al. [42] and Aksu et al. [1] also use the same calculation rules. As these prior studies indicate, probability calculation algorithms can provide a basis for other advanced studies. Our study proposes another type of probability, and it could be an alternative to the existing one to broaden our risk understandings.

Cyclic structures in an attack graph make security metrics calculation difficult. If we somehow unravel those cycles, the problem gets relatively easy. The algorithm by Wang et al. [169] contains such cycle elimination applied before probability calculation. Yousefi et al. [177] propose to unravel an attack graph to a set of attack paths; such a graph conversion enables further applications, for example, Q-learning over attack graphs [178]. However, cycle elimination might cause unintended biases since it implies the selection of specific attack paths. Also, cycle unraveling easily brings computational explosion. In one aspect, cyclic structures are the by-product of information compression to make an attack graph compact. Appropriate handling of cyclic structures is inherent and indispensable for practical attack graph analysis.

It is needless to say the importance of scalability. But, we have a trade-off between the expressiveness of problems and the scalability of applicable algorithms. HARM [51, 41] is a well-established approach, highly scalable and applicable to cyclic attack graphs; with it, we can calculate probabilities too. The essence of HARM is on its hierarchical structure. That is, it restricts the form of attack graphs. As a consequence, AND/OR combinations of nodes cannot appear freely; those locations are limited in the peripheral portions of the whole graph. This constraint makes efficient, dynamic

programming like algorithms work. At the same time, it cannot fully leverage the expressiveness of AND/OR combinations of nodes, especially in the network topological structure description. The mixture of AND/OR combinations is closely related to the rich expressiveness of Datalog used in MulVAL [118, 116, 119] and is the vital basis for practical attack templates. In short, HARM chooses the scalability at the cost of attack templates expressiveness. Conversely, our study puts more emphasis on expressiveness.

Overall, our study is the first one satisfying all the following requirements together: (1) applies to cyclic AND/OR attack graphs, (2) calculates probabilities, and (3) scales well.

3.6.5 Limitations

The experiments shown in [Section 3.5](#) are simplified miniatures of practical situations; each host is represented as a small-scaled attack graph. In practice, one host can hold hundreds or more vulnerabilities, and its attack graph gets larger too. Also, the topological structure of subject networks can be more complex. These in-situ conditions worsen the performance of our algorithm. Though our approach could outperform existing ones, it does not guarantee unconditional applicability to actual cases.

The loop belief propagation compared with our algorithm is implemented solely in Python by the author. There could be far more efficient implementation. In that case, the performance ratio shown in [Figure 3.5.5](#) changes. However, any implementation cannot overcome the non-linear complexity of Bayesian network inference.

Also, our reachability calculation cannot take into account the observed events. In other words, our approach is not capable of dynamic analysis, which is a remarkable advantage of Bayesian attack graphs. However, this does not degrade the value of our approach in static analysis. It is suitable for the risk analysis step in the information security management process.

The reliability of the inference result depends on many factors. At least, the initial reachabilities and the filter functions should be determined carefully. The parameters used in the experiments are not validated in the field. The popular way to assign those parameters in the literature is CVSS based pseudo probability calculation, e.g., [13]. Note that it is a qualitative approach inherently. There is no guarantee that the doubled value of the pseudo probability means the doubled likelihood. One practical approach is regarding every calculation as a kind of What-If analysis. In that case, the reachability-graph analysis generates quantitative results based on the assumption. Hypothetical but quantitative risk metrics should help the risk comprehension among various stakeholders.

3.7 Conclusion

This chapter has examined network security metrics based on an attack graph. While a simple attack graph applies mainly to qualitative security analysis, advanced approaches enable quantitative security analysis based on an attack graph. The Bayesian attack graph based approach is the representative one. We pointed out several issues about that approach. With that approach, large-scale network analysis is impractical due to its computational load. Also, it cannot handle cyclic structure in an attack graph. In contrast, the proposed approach — reachability-graph analysis — scales well and can handle cycles in an attack graph. Our approach shows thousands of times faster results in the experiment compared to the Bayesian attack graph based approach. Besides, we have shown

several other properties of this approach, including the termination and the confluence of analyses and the extensibility enabled by filter functions. It is also guaranteed to be monotonic to the increase in initial reachabilities. Collectively, the proposed approach is yet another tool to analyze network security in practical scenarios.

Chapter 4

Network hardening

4.1 Introduction

While reachability helps cost-benefit analysis, the multi-faceted nature of risk makes the analysis even more difficult. For example, both the leakage of customer privacy information and the service malfunction are critical risks for an enterprise. It is not easy to say which is more critical than the other. In general, we have to evaluate a system's security from multiple aspects that are not directly comparable with each other. Consequently, the evaluation criteria for countermeasures are multi-objective.

Network hardening is an optimization problem to find the best combination of countermeasures to protect a network from cyber-attacks. We regard a network as a pair of a risk function and a set of countermeasures in this problem. While an attacker exploits the vulnerabilities in the concerned network, a set of chosen countermeasures can prevent some exploits. Each countermeasure has a cost and effects; for instance, one avoids data breaches while another protects service availability. The risk function evaluates those effects; it should map the set of countermeasures to a multi-dimensional vector, as implied above. In other words, practical network hardening is a multi-objective combinatorial optimization problem.

The answer to a multi-objective optimization problem is a set of preferable solutions. Suppose we are interested in a bi-objective optimization problem in which we want to minimize the cost and maximize the benefit, and we have two solutions A and B. Both solutions could be justifiable when both A's cost and benefit are greater than B's. On the other hand, if A's cost is greater than B, and its benefit is less than B, then we have no reason to choose A. In that case, A is said to be *dominated* by B. When no other solutions dominate a solution, we call the solution a *dominant solution*. We call the set of dominant solutions the *Pareto optimal set* or the *Pareto front*. The answer for multi-objective optimization is given as the Pareto optimal set. Each Pareto optimal solution in the front has the best cost performance for its benefit. Thus, from the viewpoint of enterprise decision-makers, it becomes a convincing candidate for the actual implementation. Our interest is in the Pareto optimal set approximation for multi-objective network hardening such that it serve decision-makers a set of security investment candidates.

One more important thing to note is that the benefit protected by network hardening should be described in business vocabulary. For decision-makers who are not security experts, "the possibility of a data breach on a central database" is less intuitive than "the possibility of customer privacy infringement," even though both description denote the same incident. The appropriate form of

security description and the relevant metrics for its evaluation depend on each enterprise. Therefore, it is desirable to have a network hardening algorithm taking security landscape description in rich vocabulary as input. That is, the customizability of the security metrics and the generality of the problem description to be optimized is also concerned.

There are various approaches for describing the security landscape, such as classical planning [147, 148], attack graph [124, 70, 119], attack tree [138, 133, 83], and Markov chain [73, 136]. These approaches can describe attacker behaviors, the relevant possible security breaches, and their relation. However, powerful expressiveness tends to be coupled with intractable space complexity. For instance, classical planning can handle the broadest range of problem descriptions among the approaches mentioned above, but it can easily explode computationally. Attack tree is rather compact though it has weaknesses in expressing all the attack paths; it can be exponential in the size of the problem. As we will see in [Section 4.2.1](#), the attack graph seems to be the best-balanced approach to date.

Similarly, we have multiple approaches of network hardening as optimization problems, including, SMT solving [127], Integer (Linear) Programming (ILP) [132, 87, 139, 5, 80, 33], Markov Decision Process (MDP) [28, 93, 94], and evolutionary computations [163, 23, 24, 4]. Each approach puts some constraints on acceptable problems. It is difficult for the ILP to handle the non-linear effect of combined countermeasures while such non-linearity appears in attack graphs naturally. In contrast, MDP can use non-linear reward functions while the calculated reward is supposed to be fully ordered in general, i.e., not suitable for multi-objective optimization. From the viewpoint of the acceptable range of problems, evolutionary computations have the broadest applicability, though they are iterative approximation algorithms often slow to converge.

Furthermore, the design of network hardening algorithms is often tightly coupled with the problem definition. The change in security metrics, or objective functions, could require the rethinking of the algorithm design. Such tailoring is infeasible for non-expert users to leverage those algorithms in their network hardening jobs, each with specific characteristics.

This chapter proposes a multi-objective network hardening algorithm that enumerates the preferable combinations of countermeasures near the Pareto front. Our algorithm accepts a broad range of problem descriptions, including attack graphs, as long as the security metrics, used to define the optimization evaluation criteria, observing monotonicity; we require those metrics always increase when the corresponding set of countermeasures expands. This constraint is weak enough to enable various metrics and different forms of problem descriptions.

The major contributions in this chapter are as follows:

1. This chapter provides yet another multi-objective network hardening algorithm that can replace the use of versatile multi-objective optimization algorithms.
2. Our algorithm works as a one-pass algorithm accompanied by optional iterative refinement. The one-pass search does a gradient-based greedy search, where each gradient search gives one or more approximate Pareto optimal solutions. It is different from the ordinary gradient search in optimization problems, where a sequence of iterations identifies one optimal solution.
3. Though it is still in its early stage, numerical experiments show promising performance; it is tens of times faster than the typical genetic algorithm, NSGA-II [22], a well-known genetic algorithm popular among past studies.

Section 4.2 refers to relevant prior studies and discusses their limitations in detail. The definition of the problem we solve is given in Section 4.3. Section 4.4 explains the detail and the intuitions of our approach. In Section 4.5, we examine our approach with several numerical experiments. In these experiments, we apply our algorithms to randomly generated attack graphs and compare the performance with NSGA-II. Our approach outperforms NSGA-II by generating better solutions with less computation. After discussing several relevant topics, including the computational complexity of our approach, we make conclusions.

4.2 Related work

This chapter focuses on the multi-objective combinatorial optimization for network hardening, where the target problem is regarded as a pair of a risk-evaluation function and countermeasures. Due to its computational complexity, our goal is to approximate the Pareto optimal set of countermeasure combinations, not to identify the exact Pareto front. This section summarizes the prior studies in algorithmic security risk evaluation and multi-objective network hardening.

4.2.1 Algorithmic frameworks for risk evaluation

There are many algorithmic frameworks to evaluate network security risks. We can use those frameworks as the basis of our risk functions; in a simple case, our risk function just counts the number of adversarial events identified by a specific network security analysis algorithm. Hence, the range of risks that can be handled depends on the expressiveness of the chosen framework.

One intuitive example is the Markov chain. A Markov chain is a finite state automaton assigned state transition probabilities with an initial distribution. Each state in the chain represents a state of the concerned network. State transitions correspond to actions by the attacker. For instance, suppose we have N computers on the network, and each computer has a binary state of whether it is compromised or not. In that case, a Markov chain composed of 2^N states can represent all the possible state transitions of the whole network. As this simple case implies, due to its space inefficiency, Markov chains are infeasible even for a moderate size network.

An attack tree is a tree composed of AND-nodes and OR-nodes. Each node represents some action by the attacker or a condition such as a computer having a vulnerability. An attack tree depicts the dependencies among the conditions and actions; the nodes in a subtree show the conditions and actions required by the event corresponding to the root node of the subtree. An attack tree is far more compact than a Markov chain since each node corresponds to a specific condition or action, not the whole network state. This space efficiency comes at the cost that an attack path's order of attack steps cannot be identifiable. Moreover, it is still space inefficient; the size of an attack tree is proportional to the number of attack paths, which is exponential in the worst case.

An attack graph is a directed graph that merges the nodes corresponding to the same action or condition in an attack tree. This node merging reduces the worst-case size of an attack tree from exponential to polynomial in the size of the concerned network. It is a feasible approach from the perspective of space complexity. However, it cannot handle the negation of conditions. It excludes the inference like “the attacker succeeds when a specific countermeasure is NOT implemented.” The expressiveness matches that of Datalog [12]; it is used for facts and attack templates description to generate attack graphs.

A classical planning task subject to the STRIPS framework [32] asks the planner to find the sequence of actions reachable to predefined goals from the given initial state. In short, the chains of possible actions form a finite automaton. Therefore, the whole graph of possible actions is intractably huge as Markov chains are. Well-organized planners search only the portion of this vast space to find the paths to goals; it is inherently an approximation algorithm. However, it does not solve the worst-case complexity; it explodes when it cannot find a suitable approximated solution.

The four frameworks mentioned above are popular in algorithmic network hardening studies. None of them is complete in satisfying powerful expressiveness and tractable space efficiency.

From the feasibility point of view, attack graphs seem to be a preferable approach for it is compact even in the worst-case. In addition, many researchers have studied the practical potential of Datalog in attack graph generation. Bacic et al. [8] examine the Datalog rules design to model network infrastructures, including subnets and routers. Their effort also tries to handle the different aspects of risks, such as confidentiality, integrity, and availability. Froh and Henderson [38] extend that effort to describe the dependency of services on its infrastructure. Stan et al. [149] show realistic test results where Datalog is used to infer the attacks in the lower layer, such as the data-link layer. Sun et al. [151] discuss an approach to simulate the cloud-specific security problems like the availability damage caused by other invisible virtual machines on the same physical host. Sun et al. [153] propose a Datalog-based framework depicting the effect of infrastructure infringement on enterprise missions. Though the attack graph's expressiveness is theoretically limited as it cannot handle negation, it seems powerful enough to handle a broad range of problem descriptions.

4.2.2 Minimum-cost set cover problem

The frameworks briefed in the previous subsection can identify the goal events or conditions reachable by the attacker, corresponding to undesirable results. Therefore, one of the most simple network hardening options is eliminating those reachable goals with a cost as little as possible. We can formulate this problem as the minimum-cost set cover problem.

Jha et al. [73] have used finite automata as their framework and pointed out that it is an NP-complete problem to find the set of countermeasures blocking the paths to goals. This result indicates the inherent need for approximation algorithms. They have proposed a greedy search algorithm with provable bounds. Noel et al. [113] have proposed an exact algorithm applicable to attack graphs. Their algorithm is aware of the cost of countermeasures and finds the set of countermeasures with the minimum cost. It has exponential complexity in the worst-case. Wang et al. [166] have expanded the approach of Noel et al. [113] to support attack graphs with cyclic structure. Homer et al. [47] have recast the problem into MinCostSAT [88] and used state-of-the-art solvers. Albanese et al. [3] have designed a bottom-up greedy search algorithm approximating the minimum-cost set of countermeasures. The numerical experiments show the excellent scalability of their approach.

These approaches focus on the cost and the complete avoidance of adversarial goals. Such a problem formulation is too restrictive in practice. We do not necessarily need the complete elimination and rather a multi-objective evaluation of optimization results.

4.2.3 Satisfiability modulo theory

Rahman and Al-Shaer [126] examined the use of an SMT solver [21] to analyze network security. Roughly speaking, SMT solvers accept a class of declarative logic programs and transform them into

SAT problems, then apply powerful SAT solvers. Their framework allows multi-faceted qualitative evaluation of network security and can identify a network hardening option satisfying various constraints written in declarative logic. Further development [127] enables quantitative calculation of relevant metrics and the budget constraint in searching for hardening options. Therefore, by gradually changing those constraints, including the budget, we can obtain various network hardening options, each of which satisfies a specific budget and other security constraints.

Even though this approach has rich expressiveness in the problem description, it is inherently a SAT-based approach. SAT is known to be NP-complete. Moreover, gradually changing constraints is equivalent to doing a grid search. As a whole, it is not computationally efficient in the worst-case.

4.2.4 Markov decision process

A Markov Decision Process (MDP) is a discrete-time stochastic control process defined over a Markov chain. The implicit controller can take actions causing state transitions, and a reward function is also given that defines the immediate reward received after each state transition. We can use MDP to model an optimization problem where the controller maximizes the total reward attenuated with a discount factor.

Sarraute et al. [136] used Partially Observable MDP (POMDP) to model the attacker's behavior to mimic the optimal attack scenario for penetration testing. Durkota et al. [28] extended this idea to define a Stackelberg game in which the defender and the attacker compete. This game-theoretic formulation gives an equilibrium point where both the defender and the attacker take their own optimal strategy. The sequel study [27] leveraged the Mixed-Integer Linear Programming (MILP) technique to identify the upper bounds of the equilibrium points. The detail of the attacker/defender behaviors in this game is elaborated in [29]. However, in the game defined by Durkota et al., the defender's action is limited to distributing honeypots that detract the attacker from its true targets. Miehlung et al. [93] have proposed a more generalized attacker/defender competition based on the mixture of POMDP and a limited form of attack graphs. The defender's action corresponds to a set of countermeasures that disables some of the attacker's actions. Later on, this attempt was expanded [94] to use general attack graphs and leveraged the Partially Observable Monte-Carlo Planning (POMCP) algorithm [141] to find the defender's optimal strategy.

There are two major weaknesses in these MDP-based approaches. The first issue is that the state space of a Markov chain for network hardening is generally too vast to search all over. Therefore, optimization algorithms, such as the MILP technique by Durkota et al. [29] and the POMCP algorithm used by Miehlung et al. [94], probe a limited portion of the whole space. It brings the trade-off between the algorithm's computational efficiency and the optimality of the approximated solution. The second and critical issue is that the reward used to evaluate tentative strategies must be fully ordered. In contrast, multi-objective optimization uses multi-dimensional vectors as their evaluator, and those vectors are partially ordered. Thus, it is not suitable for multi-objective optimization.

4.2.5 Integer linear programming

An integer programming problem is an optimization problem where we try to find a vector in $\mathbb{Z}^n$, or usually in $\mathbb{N}^n$, that maximizes the objective function while following a set of constraints represented as equalities and inequalities. If the objective function and the constraints are linear, the problem is called an integer linear programming problem. Moreover, we call the problem a mixed-integer (linear)

programming problem when some elements of the solution vector are other than integer numbers. It is easy to see that the combination of countermeasures can be mapped to a vector in $\{0,1\}^n$. Therefore, we can formulate the network hardening problem as an ILP problem if we can find a way to express the problem characteristics in the linear objective function and the linear constraints.

Letchford and Vorobeychik [87] defined the network hardening problem as a Stackelberg game between the attacker and the defender and gave two independent integer linear programs in which each player tries to maximize its utility, respectively. They proposed interleaving the two programs to search the equilibrium point. Also, another heuristic approach based on greedy search was given to improve the computational efficiency. Serra et al. [139] examined another MILP formulation of the network hardening problem where the defender searches the Pareto front induced from the bi-objective optimization of seeking greater network utilities and smaller attack impacts. In contrast to these prior studies, Almohri et al. [5] introduced the concept of successful attack probability. The probability calculation makes the problem non-linear. They overcome this problem by leveraging the sequential linear programming technique in which relaxed linear problems gradually approximate the non-linear problem. However, it does not care about the cost of attacks and defenses. The study by Bhuiyan et al. [9] and Nandi et al. [98] handled a more complex problem with successful attack probabilities and costs. Khouzani et al. [80] have shown an approach to formulate a multi-objective network hardening problem with successful attack probabilities and costs as a MILP problem and used that formulation to find the Pareto front. These prior studies depend on attack graphs to identify the attacker's possible behaviors.

On the other hand, to define a network hardening integer program, Roy et al. [132] have shown using attack countermeasure trees [133], another form of attack trees twined with the defender's actions. Kordy and Widł [81] adopted attack defense trees [83] as their basis, but their approach has difficulty in accepting the attacker's repeated actions [82]. Fila and Widł [33] relaxed the limitation and broadened the applicable class of attack defense trees to direct acyclic graphs.

The rich variation of network hardening problems mentioned above indicates the powerful expressiveness of the ILP-based approach. It is, however, at the cost of the dedicated effort for appropriate integer programming. For example, both Serra et al. [139] and Almohri et al. [5] leveraged attack graphs to determine the attacker's possible actions. Nevertheless, the definition of the adopted attack graph is different, and the corresponding integer programs, too. Such a difference can also arise from the change in the adopted security metrics for evaluation. While integer (linear) programming has the potential to express a broad range of problems, it requires a careful understanding of the problem and the technique.

4.2.6 Classical planning

Speicher et al. [146] have proposed Stackelberg planning, a combination of classical planning approach and Stackelberg game where the leader's optimal strategy minimizes own cost while maximizing the cost of the follower's best response. In a Stackelberg game, we try to find equilibria of the optimal strategies of the leader and the follower. In a Stackelberg planning, we enumerate the leader's dominant strategies coupled with the follower's best responses. A Stackelberg planning aims to identify the Pareto front instead of the equilibria. For network hardening problems, the leader corresponds to the defender. The algorithm in [146] assembles the leader's every strategy and the follower's best response, eliminating the leader's dominated strategies. They applied this

framework to assess the security of email infrastructures [147]. Later on, more generalized use of this Stackelberg planning was proposed in [148], where two players compete for the best successful attack probability under budget constraints. Stan et al. [150] have applied the A* algorithm [45] to a network hardening problem defined over an attack graph where the successful attack probabilities, defense budget limitation, and the expected loss of values by successful attacks are concerned. The successful attack probabilities are used to calculate the expected loss, and the solver searches the countermeasures combination minimizing the loss within the budget.

The common characteristics of these studies are the vast state space and the use of pruning techniques for efficient search. The latter involves a kind of best-first search. However, there is no single best strategy in multi-objective optimization. Therefore, it is not straightforward to apply these approaches to such problems.

4.2.7 Evolutionary computation

In the broad sense, evolutionary computation algorithms iterate partial breadth-first search while keeping a set of tentative solutions as the candidates for the final solution. This common property makes them suitable for multi-objective optimization since the set of tentative solutions approximates the Pareto optimal set.

Viduto et al. [163] proposed to use the Tabu search algorithm [43] to solve a bi-objective network hardening problem focusing on the risk and the defense cost. Wang et al. [174] applied another algorithm called Ant-Colony optimization [26, 18] and heuristic inference to find a solution with optimal cost-benefit. Ali and Khan [4] designed a variation of Multi-Objective Comprehensive Learning Particle Swarm Optimizer (MOCLPSO) [54] and applied it to an attack tree based bi-objective network hardening problem. Dewri et al. [23] applied the NSGA-II [22] to a similar attack tree based problem. This approach was extended later to encompass advanced ideas such as searching for robust solutions and the equilibrium point, analogous to the optimal cost-benefit solution [24]. Poolsappasit et al. [125] also leveraged the NSGA-II for a similar problem with probability-assigned attack graphs. The NSGA-II is still popular in recent network hardening studies [72, 31].

The focal point in the use of evolutionary computation is performance. Generally speaking, it tends to converge slowly. Moreover, these algorithms have many configuration parameters, and parameter tuning requires a non-trivial amount of effort. Nevertheless, this class of algorithms could be the practical candidate for multi-objective network hardening due to its versatile applicability.

4.2.8 Towards the multi-objective network hardening in practice

As is discussed above, to date, most network hardening algorithms optimize single-objective or bi-objective problems. Only a few exceptions can handle multi-objective problems with the Pareto front approximation. Moreover, the algorithm design is often tightly coupled with the problem definition. The change in security metrics, or objective functions, could require the rethinking of the algorithm design. Such tailoring is infeasible for non-expert users to leverage those algorithms in their network hardening jobs, each with specific characteristics. In practice, evolutionary computation seems to be the only viable solution for multi-objective network hardening.

This intuition is backed up by Jajodia et al. [72]. While most of the past studies based on evolutionary computation focus on small-scaled, bi-objective optimization, their study handles multi-objective network hardening considering a network of hundreds of hosts. Their problem formulation

allows the defender to deploy honeypots, software patches, and service removals. The attacker's behavior is modeled with reinforcement learning. Furthermore, the proposed algorithm approximates the Pareto front with multi-objective. These sophisticated features depend on using the versatile multi-objective optimization algorithm, namely NSGA-II. However, numerical experiment results indicate the heavy computational load of the approach. Though several days of computation might be negligible for offline optimization, it is desirable to have a more efficient algorithm even more for large-scale networks.

In this chapter, we propose another approach that is a relatively scalable, multi-objective network hardening algorithm other than evolutionary computation. Our algorithm optimizes the combination of countermeasures evaluated by multi-vector valued, monotonic objective function; each vector element always increases as the given set of countermeasures expands. This monotonicity requirement is well-suited for the security metrics defined on attack graphs, as we will see in the next section.

4.3 Problem settings

This section defines the problem to be solved approximately by our approach.

4.3.1 Multi-objective network hardening

In actual attack graphs, most initial conditions represent the existence of vulnerabilities. Therefore, we can regard any subset of initial conditions as a set of vulnerabilities. We define the level of elementary cybersecurity risk of a given attack graph as follows:

Definition 10 (Uni-dimensional risk). Given an attack graph $G = (E \cup C, R_r \cup R_i)$, sets of *vulnerabilities* $V_1, V_2 \subseteq V \subseteq C_i$ where C_i denotes the set of initial conditions, and a function named “risk” which determines a value on $\mathbb{R}_+ = \{v \mid v > 0\}$ and satisfies that $V_1 \subseteq V_2 \implies \text{risk}(G, V_1) \leq \text{risk}(G, V_2)$ for any pair of vulnerabilities V_1 and V_2 on G . Then, we call the value “risk(G, V)” as a *uni-dimensional risk* determined by G, V and “risk”. Also, we call the function “risk” as a *risk function*.

As the definition tells, a risk function is monotonic about the increase of given vulnerabilities, i.e., the risk is getting higher when the vulnerabilities increase. Our approach requires risk functions to be monotonic. Several attack-graph-based metrics satisfy this requirement, such as Wang et al. [169], Homer et al. [49], Cheng et al. [13], and the one we proposed in Chapter 3.

The implicit intuition behind uni-dimensional risk is the attack graph's monotonicity with the initial conditions. An attack graph is generated from a pair of attack templates and initial conditions C_i . Suppose we have two sets of initial conditions, C_1^i and C_2^i , and generate two attack graphs, $G_1 = (C_1, R_1)$ and $G_2 = (C_2, R_2)$, from C_1^i and C_2^i with the same set of attack templates, respectively. Then, the condition, $C_1^i \subseteq C_2^i \implies C_1 \subseteq C_2 \wedge R_1 \subseteq R_2$, always holds. That is, the inclusive relation $C_1^i \subseteq C_2^i$ implies that G_1 is a subgraph of G_2 . The definition of uni-dimensional risk means that a natural definition of attack graph based risk should be proportional to the size of the attack graph to be evaluated since the inclusive relation $V_1 \subseteq V_2 \subseteq C_i$ leads to the corresponding inclusive relation between the two attack graphs derived from V_1 and V_2 .

A *patchset* for an attack graph G is an arbitrary subset $P \subseteq V$. The patchset applied vulnerabilities is the set $V \setminus P$. That is, the application of a patchset nullifies the vulnerabilities contained

in it. Note that $P_1 \subseteq P_2 \implies V \setminus P_2 \subseteq V \setminus P_1$ holds for any patchsets P_1 and P_2 . Additionally, let us define the *cost* function, “cost” : $2^{C_i} \rightarrow \mathbb{R}_+$ satisfying $P_1 \subseteq P_2 \implies \text{cost}(P_1) \leq \text{cost}(P_2)$. The combination of a patchset and a cost function represents the cost and effect of patch application.

Subsequently, we define a *security score*

$$S(G, V, \mathcal{R}, \text{cost}, P) = (\rho_1, \dots, \rho_n, \tau) \in \mathbb{R}_+^{n+1}$$

where $\mathcal{R}$ is a finite set of risk functions $\{\text{risk}_1, \dots, \text{risk}_n\}$, $\rho_k = \text{risk}_k(G, V \setminus P)$, and $\tau = \text{cost}(P)$. In other words, a security score is a tuple of (1) the residual risks after the application of a patchset P , and (2) the cost required for that patchset application. We denote by $\mathcal{S}(G, V, \mathcal{R}, \text{cost})$ the set of all the security scores determined under the given $G, V, \mathcal{R}$, and the cost function, or simply by $\mathcal{S}$ as long as there is no confusion, i.e., $\mathcal{S} = \{S(G, V, \mathcal{R}, \text{cost}, P) \mid P \subseteq C_i\}$. Note that the multiplicity of risk functions corresponds to the multiple aspects of security such as confidentiality, integrity, and availability.

Lastly, we introduce the dominance relationship between two security scores. Let $S_a = (\rho_{a,1}, \dots, \rho_{a,n}, \tau_a)$ and $S_b = (\rho_{b,1}, \dots, \rho_{b,n}, \tau_b)$ be security scores in $\mathcal{S}$. If the two scores satisfies that $\rho_{a,k} \leq \rho_{b,k}$ for $k = 1, \dots, n$ and $\tau_a \leq \tau_b$ then we say S_a *dominates* S_b . In a case where $\rho_{a,k} < \rho_{b,k}$ at some k or $\tau_a < \tau_b$ is satisfied additionally, we say S_a *completely dominates* S_b . When a security score S_a has no other security score S_b which completely dominates S_a , S_a is said to be a *dominant score* in $\mathcal{S}$. At the same time, we call a patchset with a dominant score a *dominant solution*. $\mathcal{S}^*$ denotes the set of all the dominant scores in $\mathcal{S}$. Each dominant score is optimal in the sense that there is no other score with less residual risk at that or less cost.

Now we define the multi-objective network hardening problem as follows:

Definition 11 (Multi-objective network hardening). Given an attack graph G , vulnerabilities $V \subseteq C_i$, a finite set of risk functions $\mathcal{R} = \{\text{risk}_1, \dots, \text{risk}_n\}$, and the cost function, $\text{cost} : 2^{C_i} \rightarrow \mathbb{R}_+$. We call the problem to find the set of dominant solutions $\mathcal{P}$ satisfying $\{S(G, V, \mathcal{R}, \text{cost}, P) \mid P \in \mathcal{P}\} = \mathcal{S}^*$ as a *multi-objective network hardening* problem.

Note that this problem is NP-hard since we can embed an arbitrary set-cover problem as part of it. Therefore, our approach focuses on the approximation of the Pareto front.

4.4 Approach

In this section, we define our algorithms to solve the multi-objective network hardening problem approximately and provide the key intuition behind them.

4.4.1 Monotonic gradient descent search

A security score is a function of a patchset P when we fix other parameters such as an attack graph G , vulnerabilities V , risk functions $\mathcal{R}$, and the cost function. Then, we can define the mappings $f_{\rho_1}, \dots, f_{\rho_n}, f_{\tau} : 2^{C_i} \rightarrow \mathbb{R}_+$ satisfying $S(P) = (f_{\rho_1}(P), \dots, f_{\rho_n}(P), f_{\tau}(P)) = S(G, V, \mathcal{R}, \text{cost}, P)$. All of these functions $f_{\rho_1}, \dots, f_{\rho_n}, f_{\tau}$ are monotonic, i.e., if $P_1 \subseteq P_2$ then $f_{\rho_k}(P_1) \geq f_{\rho_k}(P_2)$ holds for $k = 1, \dots, n$ and $f_{\tau}(P_1) \leq f_{\tau}(P_2)$ holds.

The monotonicity of f_{ρ_k} and f_{τ} enables the efficient search for approximated dominant solutions. To understand the reason, assume that we already have the set of all the dominant scores $\mathcal{S}^*$ for a

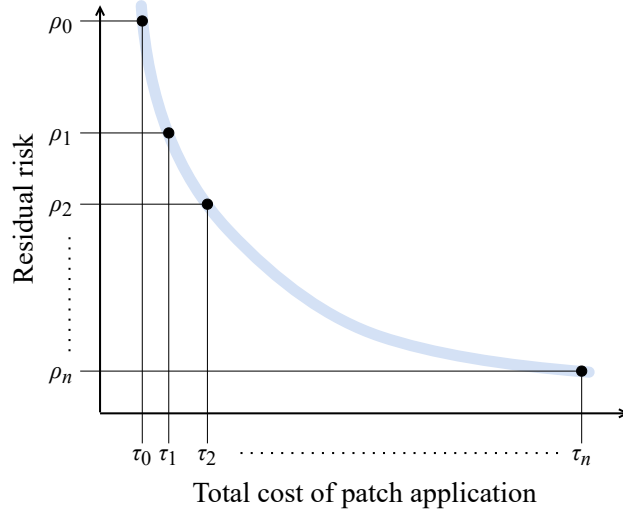


Figure 4.4.1: An example of the Pareto front

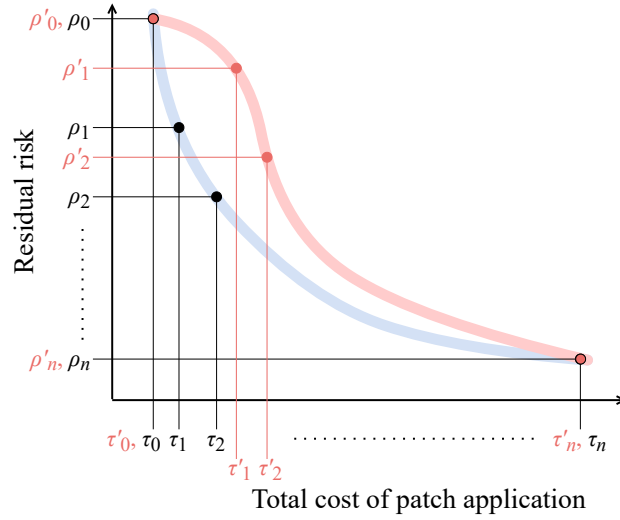


Figure 4.4.2: An example with approximated solutions overlaying

bi-objective optimization problem, i.e., $\mathcal{S}^* \subseteq \mathbb{R}_+^2$. If we plot the distribution of $\mathcal{S}^*$, then it looks like as in Figure 4.4.1.

As is depicted in the figure, $\rho_0 \geq \rho_1 \geq \dots \geq \rho_n$ and $\tau_0 \leq \tau_1 \leq \dots \leq \tau_n$ hold among $S_k = (\rho_k, \tau_k) \in \mathcal{S}^*$ as the natural consequence derived from the property of dominant scores. For each S_k , there exists at least one patchset P_k satisfying $S(G, V, \text{cost}, P_k) = S_k$. We can safely assume $P_0 = \emptyset$ and $P_n = C_i$. This is because (1) the empty set is the minimum set, and C_i is the maximum set choosable among the subsets of C_i , and (2) both f_ρ and f_τ are monotonic.

Now, suppose we have an arbitrary series of patchsets $P'_0 \subsetneq P'_1 \subsetneq \dots \subsetneq P'_n$ where $P'_0 = \emptyset$ and $P'_n = C_i$. When we overlay on Figure 4.4.1 the corresponding security-scores of this series of patchsets, it is expected that the result looks like as in Figure 4.4.2.

The overlayed series also satisfies that $\rho'_0 \geq \rho'_1 \geq \dots \geq \rho'_n$ and $\tau'_0 \leq \tau'_1 \leq \dots \leq \tau'_n$ where $(\rho'_k, \tau'_k) = S(G, V, \mathcal{R}, \text{cost}, P'_k)$ because of the monotonicity of f_ρ and f_τ . As we can see, the overlayed series is a rough approximation of the optimal series $\mathcal{S}^*$. Even when it is far from optimal, the monotonicity guarantees that $\rho'_{k-1} \geq \rho'_k$ and $\tau'_{k-1} \leq \tau'_k$ hold for $k = 1, \dots, n$ at worst. If we carefully choose each next patchset P'_k from P'_{k-1} where $P'_{k-1} \subsetneq P'_k$, then the corresponding security scores

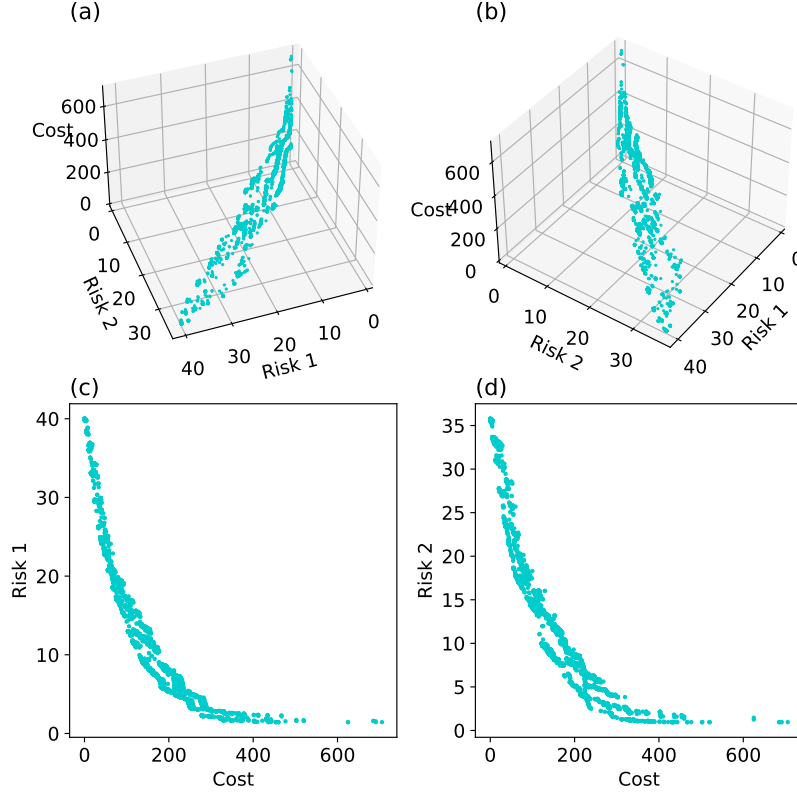


Figure 4.4.3: An example of three dimensional dominant solutions

will go right and down steadily on the plot and near to the Pareto front.

Our focus is reduced to the problem how to choose the “better” one for each next patchset P'_k from P'_{k-1} . We use gradients as the norm for the choice. Let (ρ'_k, τ'_k) and $(\rho'_{k-1}, \tau'_{k-1})$ be the security scores obtained from P'_k and P'_{k-1} , respectively. We define the gradient between P'_k and P'_{k-1} as follows:

$$\text{gradient}(P'_k, P'_{k-1}) = \frac{\rho'_k - \rho'_{k-1}}{\max\{\tau'_k - \tau'_{k-1}, \varepsilon\}}$$

where $\varepsilon > 0$ is a sufficiently small number used as the alternative denominator when $\tau'_k \simeq \tau'_{k-1}$. This gradient represents the decrease of residual risk per cost; it equals zero or negative. Therefore, for two patchset candidates P'_{k_1} and P'_{k_2} , the condition $|\text{gradient}(P'_{k_1}, P'_{k-1})| < |\text{gradient}(P'_{k_2}, P'_{k-1})|$ means that P'_{k_2} has the better cost-performance compared to P'_{k_1} . We can choose the next patchset with the best cost-performance under the constraint that $P_{k-1} \subsetneq P_k$ and $|P_k \setminus P_{k-1}| = 1$. This algorithm makes a curve like in Figure 4.4.1. We name this approach *monotonic gradient-descent search*. It is a bi-objective optimization algorithm based on greedy search.

Next, we try to leverage it for multi-objective optimization. For instance, suppose a case with $n = 2$ in Definition 11. Then, its security score is a vector in $\mathbb{R}_+^3$ composed of three axes; risk 1, risk 2, and cost. The distribution of the dominant scores could be illustrated as Figure 4.4.3.

Each green dot shows a security score. The two sub-plots, (a) and (b), in Figure 4.4.3 show the distribution in 3D space. The empty patchset brings the security score with minimum cost and maximum risk, and the patchset full of available initial conditions brings the security score with maximum cost and minimum risk. The security score of every other patchset resides between the two extremes, and it changes monotonically according to the increase of the contained elements in

the patchset. The distribution of the scores forms a spindle or a hammock-like shape due to the nature of monotonicity. The other two sub-plots, (c) and (d), show their projections onto 2D planes, risk-1 vs. cost, and risk-2 vs. cost. It is easy to see that the lower edge of each distribution in these sub-plots forms a curve similar to Figure 4.4.1. If we find a series of solutions that can form this lower edge in the projected plane, such as risk-1 vs. cost, we can expect the patchsets giving that series to be a part of the dominant solutions in the multi-objective optimization problem. In other words, we can obtain a set of multi-dimensional solutions by computing and merging the solutions searched within each plane. We leverage the monotonic gradient search algorithm to determine the solutions in a plane. This approach is analogous to the gradient-descent search with partial differentiation. Algorithm 4.1 implements the idea directly:

Algorithm 4.1 Monotonic gradient-descent search

Input:

G : an attack graph to be analyzed.
 V : a set of vulnerabilities.
 $\mathcal{R}$: a set of risk functions $\{\text{risk}_1, \dots, \text{risk}_n\}$.
 cost : a cost function $2^{C_i} \rightarrow \mathbb{R}_+$.

Output:

$\mathcal{P}$: a set of patchsets.

```

1: function MONOTONICGRADIENTSEARCH( $G, V, \mathcal{R}, \text{cost}$ )
2:    $\mathcal{P} \leftarrow \emptyset$ 
3:   for  $k \in \{1, \dots, n\}$  do
4:      $\mathcal{P} \leftarrow \mathcal{P} \cup \text{GRADIENTDESCENTSEARCH}(G, V, \text{risk}_k, \text{cost})$ 
5:   end for
6:    $\mathcal{P} \leftarrow$  dominant solutions in  $\mathcal{P}$ .
7:   return  $\mathcal{P}$ 
8: end function

9: function GRADIENTDESCENTSEARCH( $G, V, \text{risk}, \text{cost}$ )
10:   $P \leftarrow \emptyset$ 
11:   $\mathcal{P} \leftarrow \{P\}$ 
12:  while  $V \setminus P$  is not empty do
13:    /*  $G, V, \mathcal{R}$  and  $\text{cost}$  are implicitly used during the computation of  $\Delta$  and the
       gradient. */
14:     $\Delta \leftarrow$  dominant solutions in  $\{P \cup \{c\} \mid c \in V \setminus P\}$ 
15:     $\mathcal{P} \leftarrow \mathcal{P} \cup \Delta$ 
16:     $P \leftarrow \arg \max_{P' \in \Delta} |\text{gradient}(P', P)|$ 
17:  end while
18:  return dominant solutions among  $\mathcal{P}$ 
19: end function

```

Algorithm 4.1 gives a set of patchsets composed of locally dominant solutions within that set. We regard this set as the approximated dominant solutions in the sense that no redundant solution is contained in this set.

Be careful that Algorithm 4.1 is different from the ordinary gradient-based algorithms for multi-objective optimization [39]. Ordinary gradient-based algorithms start from a tentative solution and improve it iteratively to approximate a single Pareto optimal point. Therefore, those algorithms require multiple starting points and a set of sequences of iterations to form an approximated Pareto

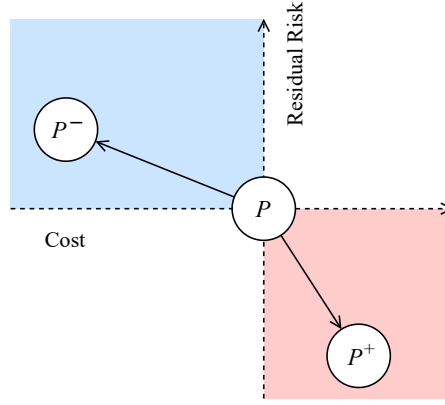
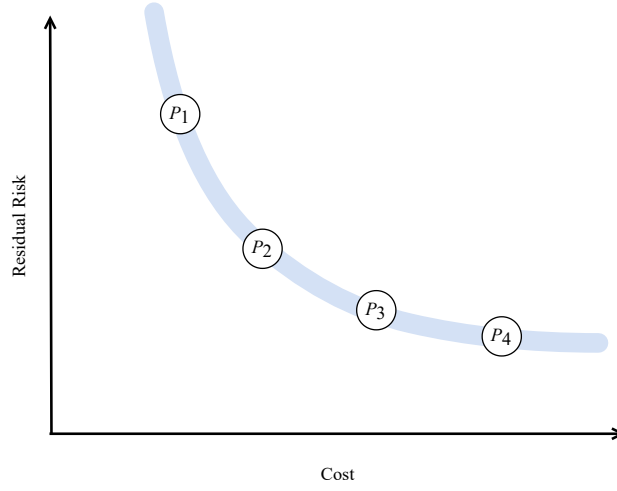
Figure 4.4.4: Interrelationships among P^- , P , and P^+ 

Figure 4.4.5: An example of an approximated Pareto front

front. In contrast, each iteration in [Algorithm 4.1](#) gives one or more approximate Pareto optimal points. [Algorithm 4.1](#) is a one-pass algorithm.

4.4.2 Iterative improvement

Well-known multi-objective optimization algorithms, such as genetic algorithms including NSGA-II, are iterative. Iterative algorithms have the potential to produce better results when we spend more computational cost. Such an improvement cannot be achieved by [Algorithm 4.1](#) since it is one-pass.

Fortunately, we can leverage the monotonicity further to explore a better approximation. Suppose we have three patchsets P^- , P , and P^+ where $P^- \subsetneq P \subsetneq P^+$, and the corresponding security scores to be the vectors in $\mathbb{R}_+^2$. [Figure 4.4.4](#) depicts the positional interrelationships among them.

Due to the monotonicity, $S(P^-)$ has less cost with greater residual risk in comparison with $S(P)$. Likewise, $S(P^+)$ has more cost with less residual risk. Therefore, any $S(P^-)$ resides in the blue region in [Figure 4.4.4](#). Also, any $S(P^+)$ resides in the red region. It means that we can obtain another patchset with a security score in the intentionally restricted region from a patchset.

Now, assume that we have a series of patchsets containing four patchsets, P_1 , P_2 , P_3 , and P_4 , which forms an approximated Pareto front shown as a blue belt in [Figure 4.4.5](#).

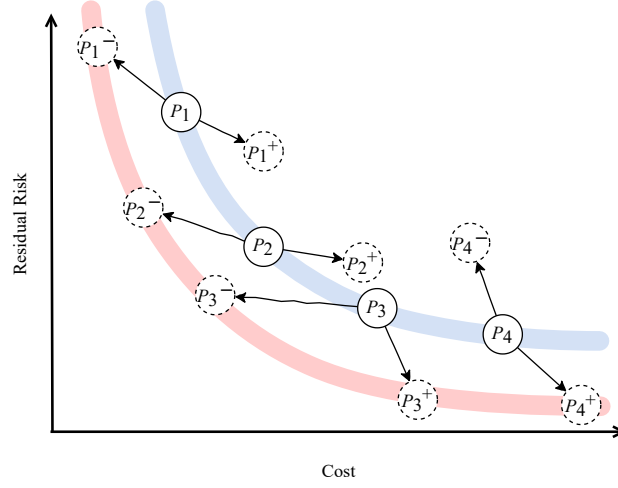


Figure 4.4.6: Interrelationships among neighbor patchsets

Since this is just an approximation, the accurate Pareto front should exist more near the axes. To explore a better approximation, define the neighbor patchsets P_k^- and P_k^+ where $P_k^- \subsetneq P_k \subsetneq P_k^+$ for $k = 1, \dots, 4$. We can easily define these neighbors by adding or removing several items from P_k . Then, the interrelationship among them can be depicted as in Figure 4.4.6.

The red-colored series forms a better approximation of the Pareto front due to the neighbors' distribution explained in Figure 4.4.4. Furthermore, this asymptotic improvement applies iteratively, i.e., the neighbor search also applies to the updated approximation. Algorithm 4.2 is the core implementation of this intuition, which generates the set of neighbors of a given patchset.

Algorithm 4.2 Neighbor solutions search algorithm

Input:

$\mathcal{P}$: a set of patchsets.
 V : a set of vulnerabilities.

Output:

$\mathcal{P}'$: a set of explored neighbor patchsets.

```

1: function NEIGHBORSOLUTIONS( $\mathcal{P}, V$ )
2:    $\mathcal{P}' \leftarrow \emptyset$ 
3:   for  $c \in V$  do
4:     for  $P \in \text{random-sample}(\mathcal{P})$  do
5:       if  $c \in P$  then
6:          $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{P \setminus \{c\}\}$ 
7:       else
8:          $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{P \cup \{c\}\}$ 
9:       end if
10:    end for
11:  end for
12:  return  $\mathcal{P}'$ 
13: end function

```

The function “random-sample” in Algorithm 4.2 takes a set of randomly chosen patchsets from the set given as the parameter. The sample size depends on the case. We can repeatedly apply Algorithm 4.2 to the approximated Pareto optimal set. Algorithm 4.1 prepares the initial approximation. With these toolings, Algorithm 4.3 performs iterative optimization.

Algorithm 4.3 Iterative solutions improvement algorithm**Input:**

r : a count of iteration.
 $\mathcal{P}$: a set of patchsets to be improved by this procedure.
 G : an attack graph to be analyzed.
 V : a set of vulnerabilities.
 $\mathcal{R}$: a set of risk functions $\{\text{risk}_1, \dots, \text{risk}_n\}$.
 cost : a cost function $2^{C_i} \rightarrow \mathbb{R}_+$.

Output:

$\mathcal{P}'$: a set of improved patchsets.

```

1:  $\mathcal{Q} \leftarrow \mathcal{P}$ 
2:  $\mathcal{P}' \leftarrow \mathcal{Q}$ 
3: for  $i \in \{1, \dots, r\}$  do
4:    $\mathcal{Q} \leftarrow \text{NEIGHBOR\_SOLUTIONS}(\mathcal{P}' \cap \mathcal{Q}, V)$ 
5:   /*  $G, V, \mathcal{R}$ , and  $\text{cost}$  are used implicitly during the computation at line 6. */
6:   Replace every  $P \in \mathcal{P}'$  with  $Q \in \mathcal{Q}$  when  $Q$  completely dominates  $P$ .
7: end for
8: return  $\mathcal{P}'$ 

```

Note that line 4 in [Algorithm 4.3](#) takes $\mathcal{P}' \cap \mathcal{Q}$ as the input to NeighborSolutions. This treatment is a simple trick to reduce the redundant neighbor searches; otherwise, the neighbor which is searched and discarded because of another neighbor's domination will be evaluated again. [Algorithm 4.3](#) gives a set of patchsets composed of locally dominant solutions as in [Algorithm 4.1](#).

4.5 Experiments

In this section, we examine the performance of our approach. We apply our approach and the NSGA-II to a set of randomly generated, multi-objective network hardening problems.

4.5.1 Common settings

The detail of the multi-objective network hardening problems to be solved in our experiments is as follows.

4.5.1.1 Network structure

[Figure 4.5.1](#) shows the general structure of the networks subject to network hardening.

The network comprises four layers — databases, servers, gateways, and clients — and the connection to the Internet. Each node in a lower layer connects to nodes in the upper layer. Each database gets connections from randomly chosen half of the servers under the constraint that every server has at least one connection to a database. Likewise, each server gets connections from half of the gateways, and each gateway gets a quarter of clients. Every client is accessible from the Internet. We assume that the attacker starts the penetration from the Internet. Also, we assume that each node except the Internet has at least one vulnerability.

Following the structure above, we examine four classes of networks with the number of nodes defined in [Table 4.1](#).

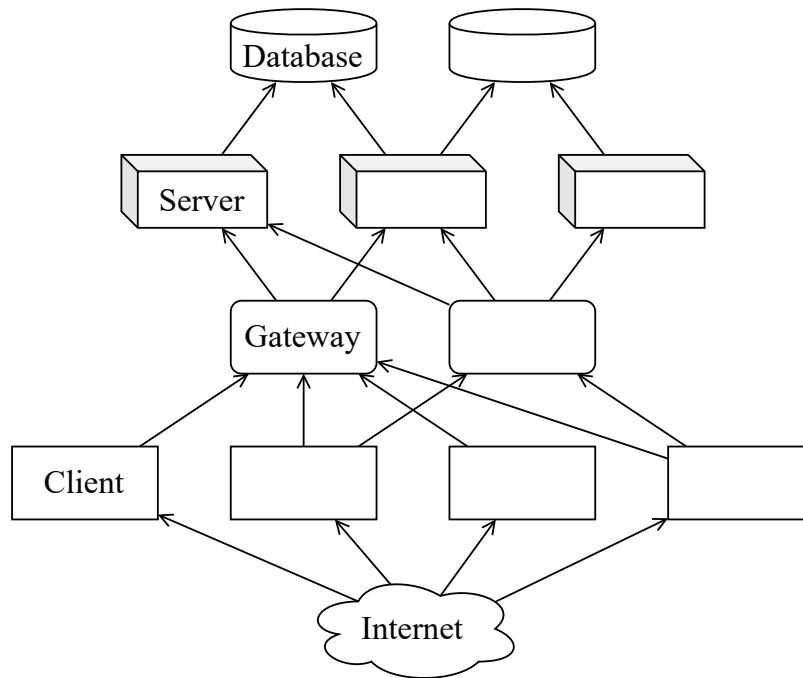


Figure 4.5.1: General structure of the experimental network

Class name	nw1	nw2	nw3	nw4	vuls./host
Databases	1	2	3	4	5
Servers	2	4	5	6	7
Gateways	1	2	2	4	7
Clients	4	8	14	18	9+5
Total vuls.	59	113	171	225	
# of AG nodes	172	337	515	680	

Table 4.1: Network size parameters for the experiment

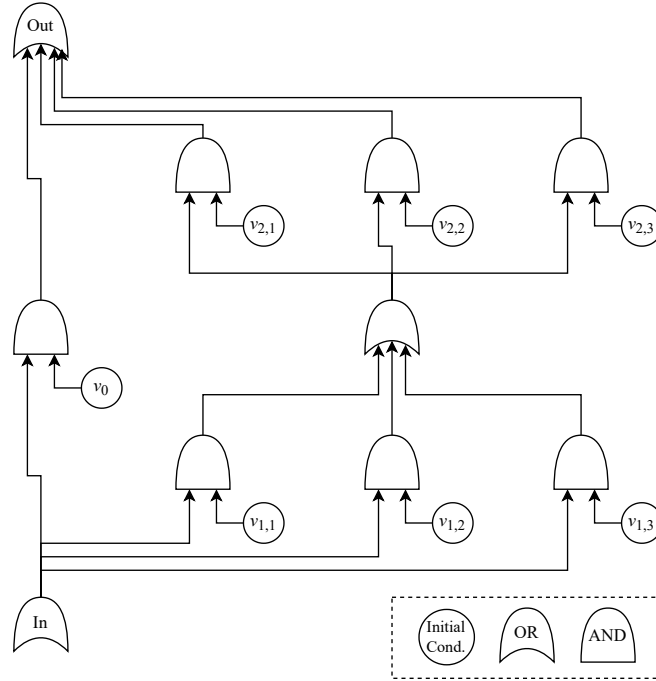


Figure 4.5.2: General structure of the attack graph per node

The number of total vulnerabilities is the total of unique vulnerabilities removable from the network under this consideration. It equals the maximum number of elements in a patchset. The number of vulnerabilities per host is shown in the right-most column, which contains one zero-day vulnerability that is supposed to be not removable. The client nodes share five vulnerabilities since most client computers have the same configuration.

4.5.1.2 Attack graph

We generate an attack graph for each host on the network, as shown in Figure 4.5.2.

The attack graph comprises two layers of patchable vulnerabilities, $v_{1,1}, v_{1,2}, \dots, v_{2,1}, v_{2,2}, \dots$, and one non-removable zero-day vulnerability, v_0 . The attacker infiltrates from the “In” node and penetrates deeper to the “Out” node. To get to the “Out” node, the attacker needs to compromise one vulnerability in the lower layer, $v_{1,1}, v_{1,2}, \dots$, and another in the upper layer, $v_{2,1}, v_{2,2}, \dots$. Alternatively, it is sufficient to compromise the zero-day vulnerability, v_0 . The chosen countermeasures aim to block this attack by nullifying these vulnerabilities. Partial nullification can reduce the attacker’s probability of reaching the “Out” node even when it does not eliminate the attack.

The attack graph of the whole network is built by combining the host-specific attack graphs explained above. For example, the “Out” nodes of servers connect to the “In” node of a database via an exploit node; this intermediate exploit node is required to keep the built attack graph bipartite. Multiple connections from the “Out” nodes to the “In” node imply that the upper node can be compromised by the attack from at least one of the lower nodes.

4.5.1.3 Assignment of attack success probabilities

We use the following calculation rule based on Wang et al. [169] to assign attack success probabilities to the nodes in the attack graph. Remind that an attack graph $G = (E \cup C, R_r \cup R_i)$ contains

two types of nodes; the exploits E and the conditions C . An exploit node represents the AND combination of its parent conditions, and a condition node represents the OR combination of its parent exploits. Let us define a function $p : (E \cup C) \rightarrow [0, 1]$, which assigns individual scores to each node in G . We define the probability P of the event implied by a node in G as follows (Be cautious in distinguishing the use of P and p , and that $P(e)$ means $P(\text{event } e \text{ occurs})$):

$$P(e) = p(e) \prod_{c \in \text{parents}(e)} P(c), \text{ and}$$

$$P(c) = p(c) \left\{ 1 - \prod_{e \in \text{parents}(c)} (1 - P(e)) \right\}.$$

We assume $p(c) = 1$ for all $c \in C_i \setminus Q$ where Q denotes the patchset in our experiments. For any $c \in Q$, we assume $p(c) = 0$. Every other $p(e)$ and $p(c)$ except the zero-day vulnerabilities has a random value in $[0.01, 1]$ subject to the uniform distribution. The exploit node of the zero-day vulnerabilities has a uniform random value in $[0.01, 0.2]$.

4.5.1.4 Risk functions and cost assignments

We assign three types of values—each corresponding to a mock of confidentiality, integrity, and availability—to network nodes and vulnerabilities to make the network hardening multi-objective. Let $CV(n)$, $IV(n)$, and $AV(n)$ be the typed asset value of a node n , and $CD(v)$, $ID(v)$, and $AD(v)$ be the typed damage quantifier of a vulnerability v , respectively. As is shown in [Section 4.5.1.2](#), each exploit node has one corresponding vulnerability in our problems. With this structure in mind, let nw denotes the set of nodes composing a network, and $(e, v) \in n$ be each pair of an exploit node e and its corresponding vulnerability v within the attack graph of node n . We define three risk functions, ρ_C, ρ_I, ρ_A as follows:

$$\begin{aligned} \rho_C(nw) &= \sum_{n \in nw} CV(n) \max_{(e,v) \in n} P(e) CD(v), \\ \rho_I(nw) &= \sum_{n \in nw} IV(n) \max_{(e,v) \in n} P(e) ID(v), \\ \rho_A(nw) &= \sum_{n \in nw} AV(n) \max_{(e,v) \in n} P(e) AD(v). \end{aligned}$$

That is, we assume that (1) the risk of a network is the sum of the risks of nodes in the network, and (2) the risk of a node is determined by the worst damage among the in-host compromises by the attacker.

The asset values, $CV(n)$, $IV(n)$, and $AV(n)$, are determined as random values subject to a normal distribution whose mean is proportional to the number of descendant hosts of the node, i.e., the upper nodes are more valuable than the lower nodes. The damage quantifiers, $CD(v)$, $ID(v)$, and $AD(v)$, are simple uniform random values on $[0.01, 1]$. The alternative denominator ε is set to be 0.00001.

Finally, the “cost” function is defined as $\text{cost}(V) = \sum_{v \in V} \text{cost}(\{v\})$ where $\text{cost}(\{v\})$ is a random value subject to a normal distribution whose mean is proportional to the asset value corresponding to the node where v resides. It means that valuable asset requires a substantial cost to protect.

The problem formulation above is adopted just for this experiment purpose. We can apply another definition as long as it satisfies the monotonicity requirement.

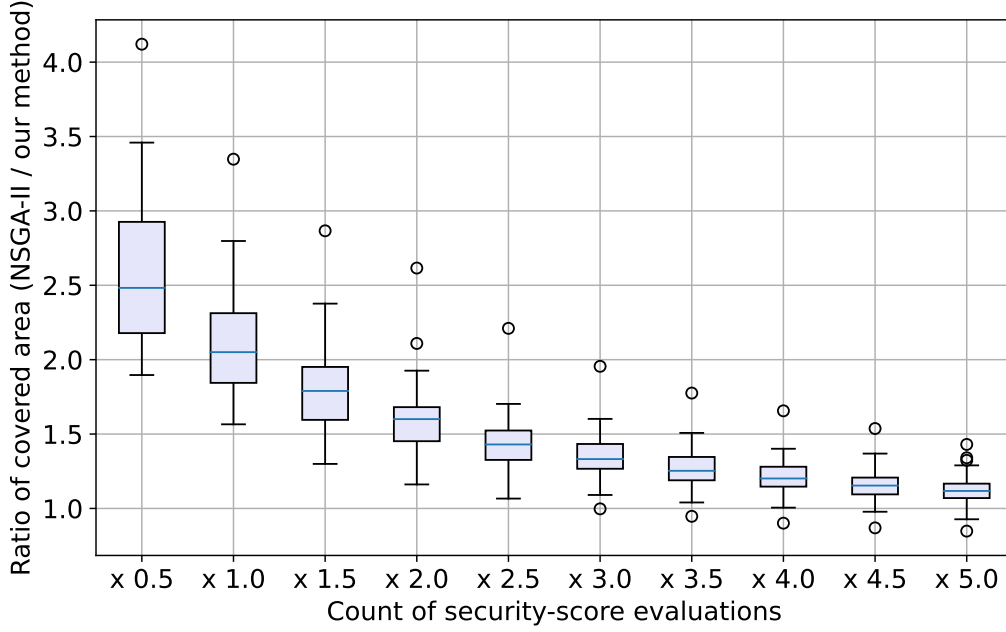


Figure 4.5.3: Comparison between NSGA-II and [Algorithm 4.1](#)

We apply our algorithms and the NSGA-II to randomly generated attack graphs explained above. The data set comprises four classes defined in [Table 4.1](#), where each class is a set of ten networks. Therefore, we examine 40 cases overall. For each case, we approximate the Pareto front. The algorithms are implemented in Python 3.7.4 and are processed by PyPy 7.3.2 in a Linux environment hosted on WSL2/Windows 10.

4.5.2 Comparison between the one-pass algorithm and NSGA-II

It is computationally impractical to identify the true Pareto front, and we cannot evaluate the optimality of the approximation. Instead, we compare our approach with the NSGA-II served by the DEAP library [35]. The implementation of NSGA-II based algorithm uses the single-point crossover and bitwise mutation. The crossover probability is 0.9, and the mutation probability is $1/n$, where n denotes the length of a gene. The population size in NSGA-II is set to $K \times V + 4$, where V denotes the number of total vulnerabilities in each network, and K is 20. The choice of K is configured to roughly match the resultant solutions count of our results and the NSGA-II results. Most of these conditions replicate Deb et al. [22]. We use hypervolume as the metric of total residual risks [176] to compare the performance of our approach and NSGA-II.

[Figure 4.5.3](#) is the box plot with 4-quantiles in which the hypervolume and the count of security-score evaluations are compared. Note that individual selection in NSGA-II accompanies the evaluation of the security score of each individual in the population.

The circles are outliers with values either more than $Q3 + 1.5(Q3 - Q1)$ or less than $Q1 - 1.5(Q3 - Q1)$ where $Q3$ and $Q1$ denote the third and the first quantile, respectively. The X-axis shows the count of security-score evaluations performed by NSGA-II based algorithm as the scaling factor to that of [Algorithm 4.1](#). The Y-axis shows the size of the hypervolume obtained by NSGA-II based algorithm as the ratio to corresponding our algorithm. For example, look at the blue-colored median bar in the middle of the plot. This bar means that NSGA-II based algorithm with 3.0 times

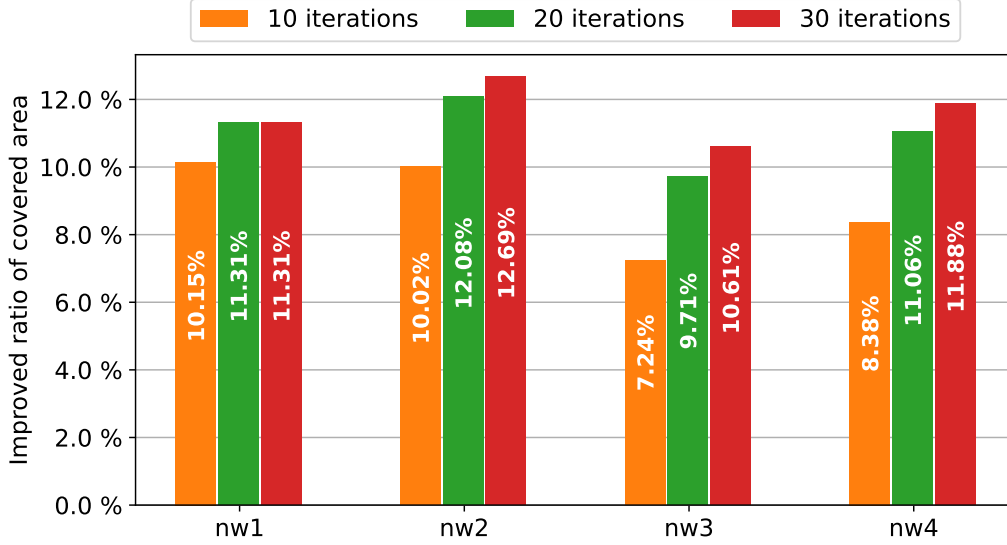


Figure 4.5.4: Performance improvements enabled by the proposed technique

security-score evaluations has, on average, the 35% or so times larger hypervolume, i.e., residual risks, compared to our algorithm's result. This result is the average of 40 cases of experiments defined in Table 4.1.

The point is that the blue-colored medians' positions stay over the ratio of 1.0. It means that the NSGA-II based solution is the inferior approximation of our result in general. However, NSGA-II is an iterative algorithm, so we could improve the result by spending more computation as Figure 4.5.3 shows. On the contrary, Algorithm 4.1 is a one-pass algorithm and cannot improve the solutions anymore.

4.5.3 Iterative improvements

Here, we compare Algorithm 4.1 and Algorithm 4.3 to see the improvements given by iteration. Figure 4.5.4 shows the comparison results.

In Figure 4.5.4, each percentage added to the plot shows the shrinkage rate of the hypervolume given by the iterative improvement, compared with the base approximation given by Algorithm 4.1. For example, the percentage of 12.69% in the middle of the figure means that the iterative improvement with $r = 30$ applied for nw2 has a 12.69% smaller hypervolume on average in comparison with Algorithm 4.1. We can read that the iteration algorithm improves the results steadily, along with the iterations increase.

Figure 4.5.5 shows the count of security score evaluations in each case. This figure illustrates the computational load in need to obtain the performance shown in Figure 4.5.4. We will revisit this figure later in Section 4.6.1.

4.5.4 Comparison between the iterative algorithm and NSGA-II

This subsection compares the result of Algorithm 4.3 with NSGA-II to explore the performance of iterative improvements. Figure 4.5.6, Figure 4.5.7, and Figure 4.5.8 are the box plots with similar settings as Figure 4.5.3. We compare 10 to 30 iterations of Algorithm 4.3 with NSGA-II. The NSGA-

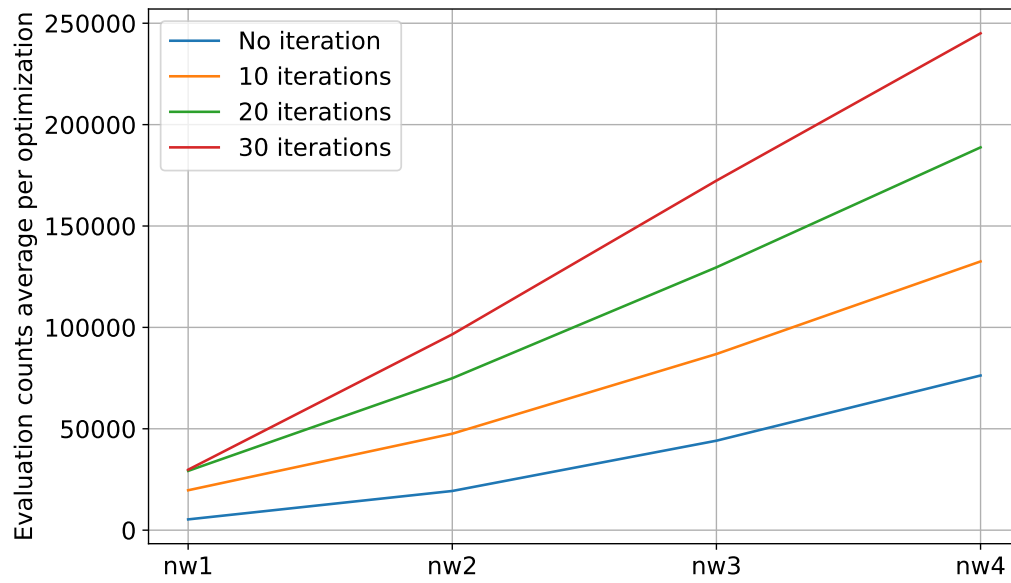


Figure 4.5.5: Security score evaluation counts in each case

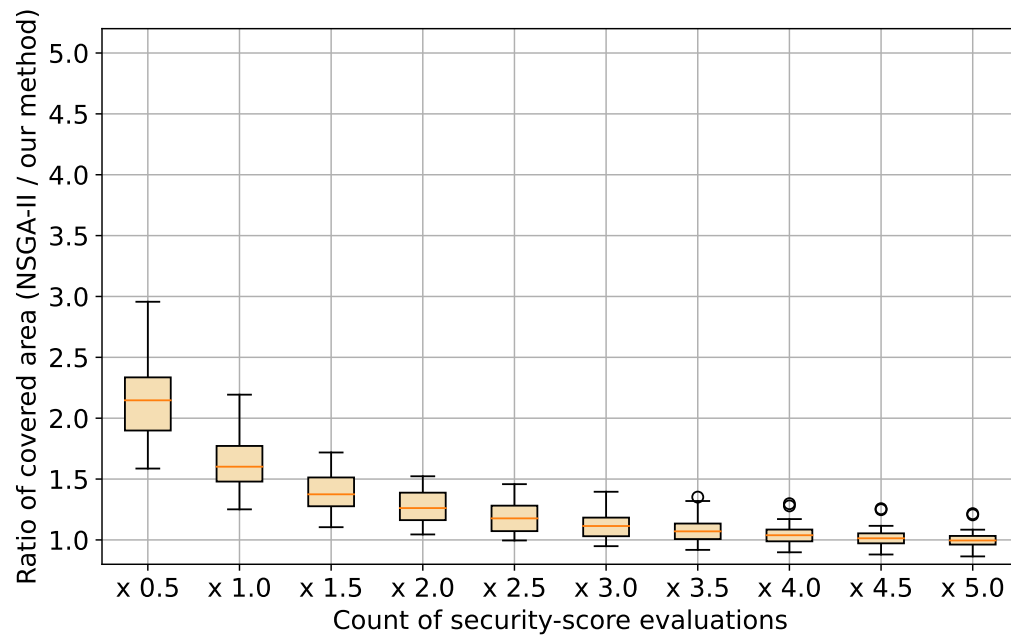


Figure 4.5.6: Comparison between NSGA-II and Algorithm 4.3 with 10 iterations

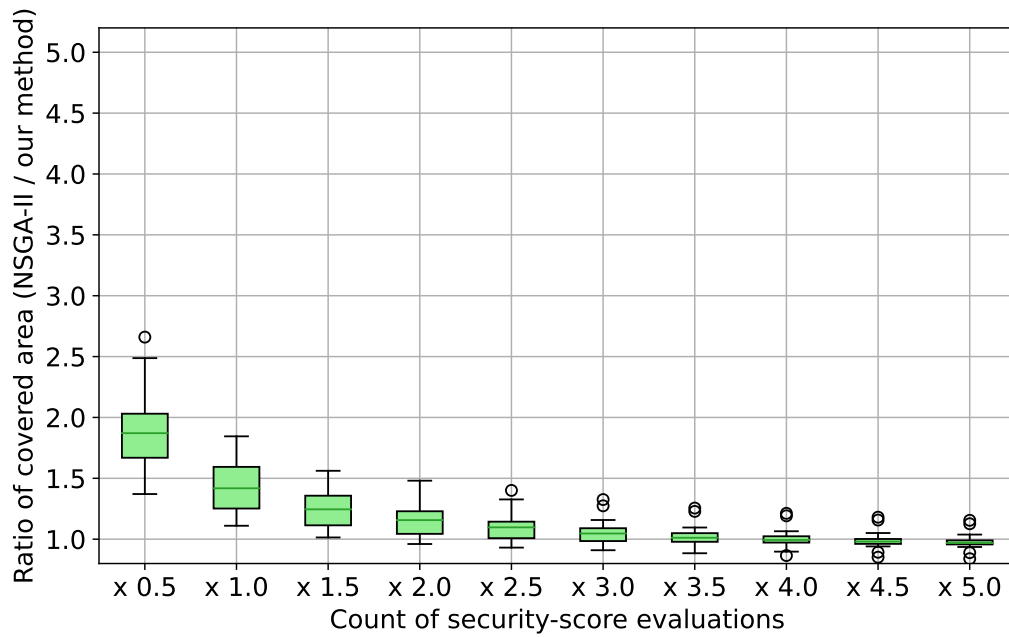


Figure 4.5.7: Comparison between NSGA-II and Algorithm 4.3 with 20 iterations

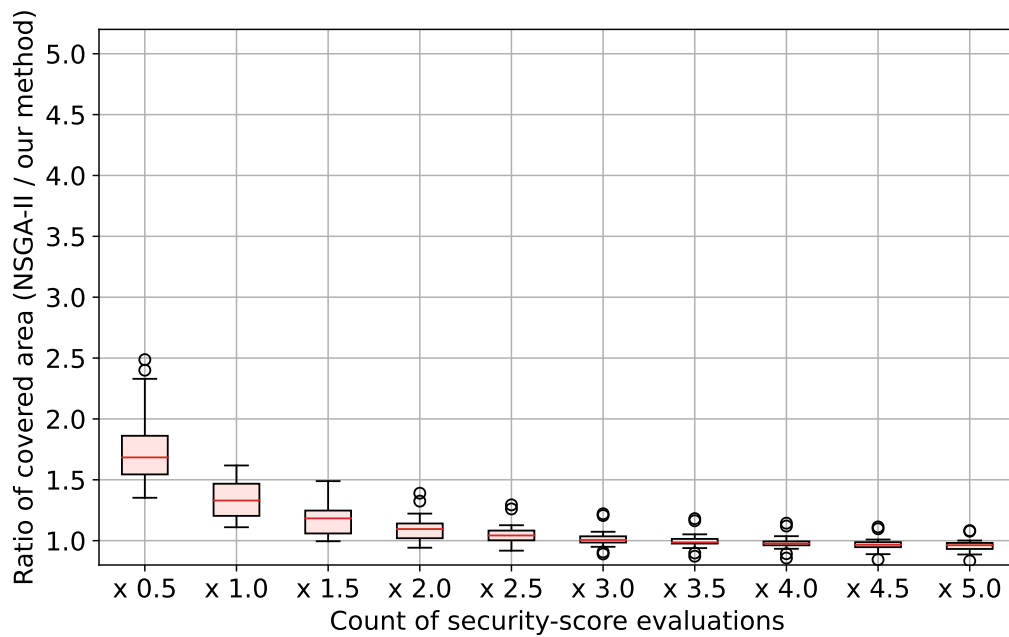


Figure 4.5.8: Comparison between NSGA-II and Algorithm 4.3 with 30 iterations

Algorithm	nw1	nw2	nw3	nw4
Alg. 1	1.9	10.7	41.9	105.2
Alg. 3 with 10 iter.	6.0	27.8	88.5	193.5
Alg. 3 with 20 iter.	8.6	42.6	129.6	269.8
Alg. 3 with 30 iter.	8.8	54.3	170.5	341.6
NSGA-II	118.6	807.8	2349.8	4526.8

Table 4.2: Average processing times in seconds of each algorithm

II results seem to converge in these figures, and the converged ratio is nearly one. If we assume that the converged solutions given by NSGA-II are truly optimal, we can interpret this result like that (1) our algorithms give almost optimal solutions, and (2) the computational cost in need to obtain semi-optimal solutions by our algorithms is substantially lower than that of the NSGA-II.

The second interpretation requires careful reading of the figures. For instance, in [Figure 4.5.8](#), it seems that the NSGA-II requires tripled counts of security score evaluations to catch up with the optimality given by 30 iterations of [Algorithm 4.3](#). [Figure 4.5.4](#) tells that the improvement enabled by 30 iterations is 10.61% to 12.69%. Simultaneously, [Figure 4.5.5](#) implies that the 30 iterations require considerably more computations than [Algorithm 4.1](#). The NSGA-II-based results at “ $\times 3.0$ ” require more than ten times the volume of score evaluations than [Algorithm 4.1](#). Therefore, if we decide to omit the optimality gain of 10.61% to 12.69% by the optional iteration, [Algorithm 4.1](#) brings semi-optimal solutions quite efficiently. In summary, these results imply the remarkable efficiency of [Algorithm 4.1](#).

4.5.5 Processing time comparison

The processing times of each algorithm are shown in [Table 4.2](#). The processing times of [Algorithm 4.3](#) contain the preprocessing time by [Algorithm 4.1](#). The chosen NSGA-II based algorithm is the one with three times more security score evaluations than [Algorithm 4.3](#) with 30 iterations; it has the even optimality of our approach. We can see the clear advantage of our algorithms.

4.6 Discussions

4.6.1 Computational complexity

The computational complexity of “GradientDescentSearch” in [Algorithm 4.1](#) is $O(N^2)$, where $N = |V|$. The amount is represented in a count of gradient or security-score evaluations, which is the dominant factor for the computational complexity in our approach. The estimation is explained as follows. Firstly, the “while” loop iterates N times since each time the size of patchset P increases by one, and the size of P cannot exceed N . Secondly, for each iteration, the gradient is computed for each element in $\{P \cup \{c\} \mid c \in V \setminus P\}$, and the size of $V \setminus P$ is $N - |P|$. Therefore, the total count of security-score evaluations is $N(N+1)/2$, i.e., $O(N^2)$. The execution of “MonotonicGradientSearch” in [Algorithm 4.1](#) is a repetition of “GradientDescentSearch”. Thus, its computational complexity is $O(MN^2)$, where M denotes the count of risk functions.

[Algorithm 4.3](#) has $O(rsN)$ complexity, where r is the iteration count, and s is the count of random samples. While we choose some fixed numbers for r and s in practice, $N = |V|$ is largely proportional to the network size. At the same time, [Algorithm 4.3](#) requires a rough approximation of

the Pareto front as its input. [Algorithm 4.1](#) generates such approximation with $O(MN^2)$ complexity. Therefore, the total computational complexity in practice is $O(rsN + MN^2)$. If we assume $rs \ll N$, then it becomes $O(MN^2)$. On the other hand, it becomes $O(rsN)$ when $N \ll rs$ holds. [Figure 4.5.5](#) verifies this approximation.

4.6.2 Applicability to other problems

The proposed algorithms do not depend on attack graphs directly. They apply to any problems with monotonic risk functions and appropriate “gradient” functions. For any problems, each solution in the determined series is guaranteed to be locally dominant at worst.

Intuitively, a patchset is a set of vulnerabilities that will be removed from the network. Adding an element to the patchset is expected to decrease the risk. It matches the expected behavior of a honeypot deployment. Thus, our approach applies to optimize the honeypot deployment strategy. However, it is not straightforward with attack graphs since a patchset application corresponds to the removal of initial conditions in an attack graph, i.e., it implies that the initial condition corresponding to a honeypot deployment induces malicious events in an attack graph. One technique to solve this irrationality in using attack graphs is stratification [134]. Stratification enables us to use negation in initial conditions; we can describe an initial condition like “when a honeypot is not deployed.”

The algorithms can also apply to problems not strictly monotonic. Monotonicity’s role is to guarantee that adding an element to a patchset always gives more cost and less residual risk. If the problem is not monotonic, then the residual risk might increase in some cases. Favorably, even when such a case arises, the filtration of locally dominant solutions removes undesirable patchset (line 14 in [Algorithm 4.1](#)). Therefore, our algorithm has some robustness to such non-monotonic cases. This property broadens the applications of our approach.

On the other hand, the algorithms are not a complete replacement of other multi-objective optimization methods, especially for problems other than network hardening. [Algorithm 4.1](#) is essentially a greedy search. The relatively better performance comes from the fact that countermeasures in a patchset are roughly independent of each other. In a network hardening, vulnerabilities on the same attack path affect each other, while those in different subnets or hosts are less influencing. This characteristic makes the problem much more effortless than the cases always requiring the search of specific, tightly coupled countermeasures. Our algorithms leverage this problem-specific fact.

4.6.3 Parameter assignments and CVSS

[Section 4.5](#) uses the risk functions, each specific to one aspect of securities, such as confidentiality, integrity, and availability. This setup implicitly assumes that we know the effect of vulnerability exploits and their relevance to these aspects. A well-known vulnerability scoring system, Common Vulnerability Scoring System [34], satisfies this assumption. “Common Vulnerability Scoring System (CVSS) captures the principal technical characteristics of software, hardware and firmware vulnerabilities. Its outputs include numerical scores indicating the severity of a vulnerability relative to other vulnerabilities.” Its impact metric contains three categorized sub-metrics for confidentiality, integrity, and availability. Even though it is qualitative essentially, we can leverage CVSS to assess vulnerabilities’ exploitability and define the coupled risk functions. The list of known vulnerabilities and their CVSS scores are available on National Vulnerability Database.

4.6.4 Limitations

The numerical experiments in this chapter are minimal, and the results do not guarantee their performance in practice. The network structures examined are a miniature of the mundane client-server system. The adopted risk functions are just a possible example of a security evaluation, though they are cared to be natural. The scale of the network hardening problem is also not sufficient to handle the field cases. Furthermore, the practical expressiveness of problems in our approach is also out of the scope of our verification.

We compared our algorithms with the NSGA-II served by the DEAP library. The experiments result could change if we use other libraries or implementations for the NSGA-II. However, in the NSGA-II, the volume of other technical treatments other than the individual evaluations, such as non-dominated sorting and crowding-distance calculation, strongly influences the total computation time. As is shown in Table 4.2, in the experiments, the average processing time of nw4 optimizations by Algorithm 4.1 is 105 seconds and by Algorithm 4.3 with 30 iterations is 341 seconds. In contrast, the NSGA-II that is evenly optimal as the latter case, with three times of security score evaluations, requires about 4,526 seconds.

As is shown in Section 4.2.7, there are a variety of evolutionary computation algorithms other than the NSGA-II applied to network hardening. We did not compare our algorithms with those others. The NSGA-II is now a classic algorithm, and the off-the-shelf implementation like the DEAP library is publicly available. We chose the NSGA-II to verify the feasibility of our approach in network hardening since that is popular among the past network hardening studies, including the recent efforts.

Though our algorithm surpasses NSGA-II, it is still insufficient to handle large-scale problems, such as the hardening of networks composed of thousands of machines. Further performance improvement is required.

4.7 Conclusion

This chapter proposed a multi-objective optimization algorithm for network security countermeasure selection based on monotonic security metrics. We pointed out that most past network hardening studies focus on single or bi-objective optimization problems with small scale, and only a few efforts based on evolutionary computations handle multi-objective network hardening. Our algorithm approximates the Pareto optimal set of countermeasures. Each solution in the set has the best cost-benefit for its cost, where the benefit is composed of residual risks of multi-dimensions. We leveraged the monotonicity of security metrics to formulate our approach. The monotonicity guarantees that the size increase of a set of countermeasures leads to an increase in cost and the decrease of residual risk. This property enables a gradient-descent search algorithm, which is computationally efficient. Our approach combines a one-pass algorithm and the optional iterative refinement algorithm. We showed its advantage by comparing it with the NSGA-II based approach and confirmed with numerical experiments that our algorithm is highly efficient than NSGA-II. The proposed algorithm has broad applicability and can be a better alternative to genetic algorithms used in past network hardening studies.

Chapter 5

Attack function analysis

5.1 Introduction

Since the research inception, there have been various definitions of attack graphs. At an early age, it was defined as finite automata, called *state enumeration graphs*. Later, the definition as cyclic directed AND/OR graphs, called *exploit dependency graphs*, was explored. Now, several researchers generalize it as a special case of Bayesian networks called *Bayesian attack graphs*. One can find even more different definitions.

The problem here is that there is no single best definition suitable for every analysis need. To date, the most matured and well-accepted definition is the exploit dependency graph. It has polynomial complexity in its representational space consumption in contrast to the exponential complexity of state enumeration graphs. However, state enumeration graphs have theoretical advantages; they are interpretable as Markov chains, Markov decision processes, and partially observable Markov decision processes. On the other hand, exploit dependency graphs lack comparable tools. Bayesian attack graphs are similar to exploit dependency graphs in their space efficiency and have strength in complex modeling such as risk analysis with partially observed attacks. Unfortunately, it is very computationally expensive to process. In practice, we have to understand the strengths and weaknesses of each attack graph form and choose the most appropriate one to use. In some cases, we might need to handle several forms simultaneously.

These attack graphs, however, originate from the sole network concerned. Each time making an attack graph, we examine the same network from various perspectives to extract the required information for the analysis. If possible, it is convenient to have a common versatile model that can be the single source of different attack graphs. Such a model might help ensure consistency among those attack graphs, too. Even though we can see much progress in this field through various reviews [84, 52, 181, 86], recent studies mainly focus on the application of attack graph analysis with a specific form of attack graphs, and it seems unique that discussing the theoretical integration of several different forms of them.

This chapter proposes using a specific function as the common versatile model. We call it an *attack function*; an attack function f is a mapping that takes and generates a set of propositions and satisfies $f(u) \subseteq f(v)$ whenever $u \subseteq v$. This formulation is a refinement and an extension of the author's prior work [188]. We show that exploit dependency graphs and state enumeration graphs are inducible from an attack function. Both induced attack graphs are guaranteed to be consistent in the sense that the inferred attack events extracted from them match each other. This fact reinforces

the theoretical legitimacy of exploit dependency graphs by clarifying their correspondence to a class of finite automata. We aim to reveal the mathematical properties and importance of attack functions behind these attack graphs.

The major contributions in this chapter are as follows:

1. This chapter shows the partial equivalence between exploit dependency graphs and state enumeration graphs. It bolsters the theoretical legitimacy of exploit dependency graphs which has never been discussed.
2. The concept of attack function analysis is introduced, which is a generalized form of attack graph analysis with theoretical potential for unexplored applications.

[Section 5.2](#) defines the two types of attack graphs, state enumeration and exploit dependency graphs, for the later discussion. Also, we compare the two forms to understand their strengths and weaknesses. [Section 5.3](#) introduces the concept of attack function. As a preparation, we shed light on exploit dependency graphs from the viewpoint of Datalog. Then, as an equivalent alternative to Datalog, we define attack functions and a procedure generating exploit dependency graphs from them. [Section 5.4](#) introduces an algorithm to generate state enumeration graphs from attack functions. Then, we focus on the relationship of the two forms of attack graphs and point out the partial equivalence. Moreover, it shows that both forms are interpretable as projections of attack functions. [Section 5.6](#) suggests the practical implication of attack functions for future study. Finally, we make our conclusion.

5.2 Two forms of attack graphs

This section explains the two primary forms of attack graphs and compares them. Most of the description is replicated from [Section 2.2.2](#).

5.2.1 State enumeration graph

In the early history of research, attack graphs have been defined as a kind of finite automata and named *state enumeration graphs* later. The following definition is taken from the study by Jha et al. [73]; it is one of the most well-cited articles in this field.

Definition 12 (State enumeration graph (adapted from [73])). Let AP be a set of atomic propositions. A state enumeration graph is a tuple $G = (S, \tau, S_0, L)$, where S is a set of states, $\tau \subseteq S \times S$ is a transition relation, $S_0 \subseteq S$ is a set of *initial states*, and $L : S \rightarrow 2^{AP}$ is a *labeling of states* with a set of propositions true in that state.

This definition implies that a state in a state enumeration graph is a set of true—or realized—atomic propositions. Each proposition corresponds to a fact such as a network reachability, vulnerability existence, and privilege escalation. In a simple case, a proposition represents that the corresponding host is compromised.

In general, the concerned propositions are proportional to the number of hosts in the concerned network. Suppose we have N computers on the network, each computer has M vulnerabilities, and each vulnerability has a state expressing whether it is exploited or not. Under this setting, the space complexity of possible states is $O(2^{MN})$. The size of a state enumeration graph is intractable even

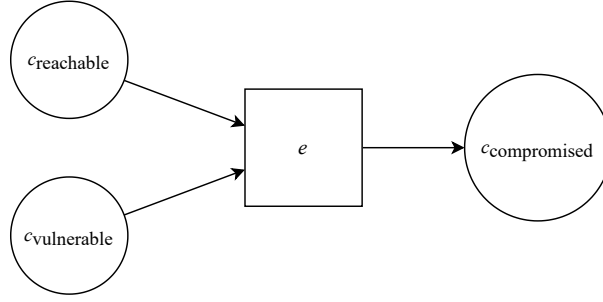


Figure 5.2.1: A minimal example of attack graph

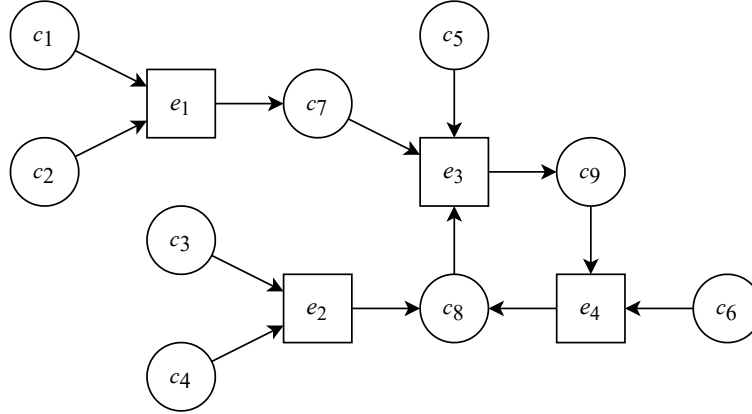


Figure 5.2.2: Another small example of attack graph

for a small-sized network with tens of machines. The earliest attempts for attack graph analysis, such as [154] and [129], adopted this form and got stacked up in the scalability problem.

5.2.2 Exploit dependency graph

The scalability problem has been solved by *exploit dependency graphs*. An exploit dependency graph is a directed bipartite graph composed of two types of nodes; *exploit nodes* and *condition nodes*. Edges run between condition nodes and exploits nodes. There are no edges connecting exploits to exploits or conditions to conditions. Each condition node denotes a fact such as a specific host is reachable from the Internet, the host has a vulnerability, and an attacker compromises the host, i.e., a condition node corresponds to an atomic proposition in state enumeration graphs. Here, let us name each condition node as $c_{\text{reachable}}$, $c_{\text{vulnerable}}$, and $c_{\text{compromised}}$, respectively. An exploit node represents the causal relationship among these conditions. Suppose we have an exploit node e , and three edges $c_{\text{reachable}} \rightarrow e$, $c_{\text{vulnerable}} \rightarrow e$, and $e \rightarrow c_{\text{compromised}}$. Then, the small subgraph in Figure 5.2.1 indicates that the two conditions, $c_{\text{reachable}}$ and $c_{\text{vulnerable}}$, are the required *preconditions* for the exploit e to realize the *postcondition* $c_{\text{compromised}}$. A postcondition could be a precondition for another exploit. Figure 5.2.2 exemplifies a usual form of attack graphs; it is a merged set of minimal attack graphs like Figure 5.2.1. We can interpret an exploit node as a description of AND combination of its preconditions, and a condition node corresponds to an OR combination of its parent exploits. In this way, an exploit-dependency graph represents the dependency relationships among conditions relevant to the concerned network's security.

The following definition is a widely accepted form of attack graphs by various researchers (Note that the content is entirely the same with Definition 1 in this dissertation):

Definition 13 (Exploit dependency graph (adapted from [166])). Given a set of *exploits* E , a set of *security conditions* C where $E \cap C = \emptyset$, a *require-relation* $R_r \subseteq C \times E$, and an *imply-relation* $R_i \subseteq E \times C$, an *attack graph* G is the connected, directed graph $G = (E \cup C, R_r \cup R_i)$, where $E \cup C$ is the node and $R_r \cup R_i$ is the edge set. For each attack graph $G = (E \cup C, R_r \cup R_i)$, *initial conditions* refer to the subset $C_i \subseteq C$ composed of every condition $c \in C$ satisfying $\nexists e \in E$ such that $(e, c) \in R_i$, i.e., the set of all the leaves in the graph, whereas *intermediate conditions* refer to the subset $C \setminus C_i$.

In state enumeration graphs, a node represents a combination of atomic propositions. In contrast, a node in exploit dependency graphs represents just one atomic proposition. This difference makes exploit dependency graphs substantially smaller. In general, the number of nodes in an exploit dependency graph is $O(N^2 + NM)$, where N is the number of machines and M is the number of vulnerabilities on each machine. Note that the term N^2 comes from the connectivities between hosts in the complete graph topology.

5.2.3 Comparison between the two forms

The polynomial complexity of exploit dependency graphs is a great advantage compared to the exponential complexity of state enumeration graphs. However, state enumeration graphs are interpretable as Markov chains, Markov decision processes, and partially observable Markov decision processes. These forms are well studied, and plenty of relevant algorithms are available for use. In contrast, exploit dependency graphs lack comparable algorithmic toolings. That is, exploit dependency graphs are not the perfect alternative for state enumeration graphs.

We also should focus on the difference in expressiveness between the two. For example, in [Figure 5.2.1](#), the attacker must achieve $c_{\text{reachable}}$ and $c_{\text{vulnerable}}$ before realizing $c_{\text{compromised}}$. In state enumeration graphs, the order between the two preconditions to be achieved can be distinguishable according to attack paths. In exploit dependency graphs, only the dependence of $c_{\text{compromised}}$ on $c_{\text{reachable}}$ and $c_{\text{vulnerable}}$ is shown, and we cannot express the attack order among preconditions.

One more thing to note is the support for condition disabling. Suppose a network with an intrusion detection system (IDS). The attacker tries to disable the IDS and then penetrate deeper into the network. After finishing the planned data theft, the attacker again turns on the IDS as a cover-up. Such an attack scenario includes the up and down of an atomic proposition corresponding to the IDS working. State enumeration graphs can express those scenarios by adding edges between the states where the IDS is on and off. Exploit dependency graphs cannot.

The fundamental assumption introduced by Ammann et al. [6] behind exploit dependency graphs is called *monotonicity* of attacks, i.e., we assume that once an atomic proposition is supposed to be true, it stands forever. In other words, we assume that the attacker never abandons gained conditions. As a result, every condition node in exploit dependency graphs is a true atomic proposition. In state enumeration graphs, false propositions can be expressed as atomic propositions missing in the labelings for states.

The comparison above indicates trade-offs between state enumeration graphs and exploit dependency graphs. For example, exploit dependency graphs are the solution if we need to reduce space consumption. When the security optimization with partially observed information matters, we should resort to state enumeration graphs. In any way, those analysis results must be consistent with each other. Therefore, it is of practical need to understand the theoretical relationship between the two forms of attack graphs. The following discussion introduces a single source of these two forms of

graphs and, based on the formulation, shows that exploit dependency graphs correspond to a class of automata that we name *attack automata*.

5.3 Attack functions for attack graph generation

This section formalizes an attack template as a monotonic mapping between proposition sets. It is easy for theoretical analysis and induces exploit-dependency graphs naturally.

5.3.1 Attack function

Datalog gives us an intuitive interpretation of exploit dependency graphs' nodes and attack templates. However, it is not suitable for further mathematical analysis. We focus on the representation of attack templates and give another definition equivalent to that in Datalog.

Let us think about the behavior of Datalog programs. The execution of a Datalog program is the repetitive application of rules to predefined and inferred facts. Each iteration adds newly inferred facts and never eliminates the existing ones. The fact-set grows monotonically, and the applied rules work as a monotonic function that takes facts as input and determines the expanded set of facts. We encode this observation into particular functions named *attack functions*.

We introduce several terms to define and explain attack functions. Let X be a non-empty finite set of true atomic propositions, and we call it a *domain*. For two domains X and Y , we call a mapping $f : 2^X \rightarrow 2^Y$ a *deduction function*; it maps a set of propositions to a set of propositions. A deduction function f satisfying $f(x_1) \subseteq f(x_2)$ for all $x_1 \subseteq x_2$ is said to be *monotonic*. When a monotonic deduction function f satisfies $f(x) \neq \emptyset$ for some x , we say that f is *valid* on x , and x is a *support* of f . If every proper subset $\hat{x}$ of a support x satisfies that $f(\hat{x}) = \emptyset$, then the support x is said to be *minimal*. When a monotonic deduction function $f : 2^X \rightarrow 2^Y$ satisfies $f(X) = \emptyset$, f is said to be *empty*.

In addition to the above, let 1^X denotes the family $\{\{x\} \mid x \in X\}$ derived from a domain X , and (u, v) be an arbitrary pair taken from $2^X \times 1^Y$. Then, we call the monotonic deduction function $\gamma_{u,v}$ defined as follows the *reduced function* from u to v :

$$\gamma_{u,v}(x) = \begin{cases} v & (u \subseteq x) \\ \emptyset & (\text{otherwise}) \end{cases}.$$

A remarkable fact is that the following theorem applies to every non-empty monotonic deduction function. (The proof is given in [Section 5.3.2](#).)

Theorem 14 (Reduced decomposition). *Let $f : 2^X \rightarrow 2^Y$ be a non-empty monotonic deduction function. Then there exists a unique finite set $\Gamma \subseteq 2^X \times 1^Y$ such that*

$$\forall x \subseteq X, f(x) = \bigcup_{(u,v) \in \Gamma} \gamma_{u,v}(x),$$

and every u of $(u, v) \in \Gamma$ is a minimal support of f .

We call the set $\Gamma \subseteq 2^X \times 1^Y$ appearing above as the *reduced decomposition* of f .

We call a non-empty monotonic deduction function $f : 2^X \rightarrow 2^Y$ an *attack function*. An attack function is equivalent to a Datalog program without facts. Each reduced function composing the

attack function corresponds to an instance in the program. The following definition equates the repetitive application of the attack function with the program's execution.

Definition 15 (Transitive image and closure). Let X be a domain, $C_i \subseteq X$ be a finite set of *initial conditions*, $f : 2^X \rightarrow 2^X$ be an attack function, and $\Gamma \subseteq 2^X \times 1^X$ be the reduced decomposition of f . Then, due to the finiteness of X and the monotonicity of f , there exists a number $K \in \mathbb{N}$ satisfying $C_{K+1} = C_K$ for the C_k defined as follows:

$$C_{k+1} = C_k \cup f(C_k), \quad C_0 = C_i.$$

We call the proposition set C_K the *transitive image* of f on C_i , and let $f^*(C_i)$ denotes it. Also, we define the following set

$$\Gamma|_{f, C_i} = \{(u, v) \mid (u, v) \in \Gamma, \gamma_{u,v}(f^*(C_i)) \neq \emptyset\},$$

and call it the *transitive closure* of f on C_i .

The transitive image is identical to the set of facts predefined and inferred by a Datalog program. At the same time, each instance in the Datalog program execution corresponds to a reduced function $\gamma_{u,v}$ extracted from the transitive closure. For each $\gamma_{u,v}$, u means the preconditions, and v indicates the postcondition. In short, we can construct an exploit dependency graph from the pair of an attack function f and the initial conditions C_i .

5.3.2 Proofs for Theorem 14

We start from several propositions as a preparation.

Proposition 16. Let $f : 2^X \rightarrow 2^Y$ and $g : 2^Y \rightarrow 2^Z$ be a monotonic deduction function, respectively. Then, the composite function $g \circ f : 2^X \rightarrow 2^Z$ is also a monotonic deduction function.

Proof. Due to the monotonicity of f , the condition $x_1 \subseteq x_2 \Rightarrow f(x_1) \subseteq f(x_2)$ holds for any x_1 and x_2 taken from 2^X . Similarly, $y_1 \subseteq y_2 \Rightarrow g(y_1) \subseteq g(y_2)$ holds for any y_1 and y_2 taken from 2^Y . Therefore, $x_1 \subseteq x_2 \Rightarrow f(x_1) \subseteq f(x_2) \Rightarrow g(f(x_1)) \subseteq g(f(x_2))$, i.e., $x_1 \subseteq x_2 \Rightarrow g \circ f(x_1) \subseteq g \circ f(x_2)$. $\square$

In the following discussion, we call a monotonic deduction function $f : 2^X \rightarrow 2^Y$ an *atomic function* if and only if $|f(X)| = 1$. Any non-empty monotonic deduction function is decomposable to a set of atomic functions as follows:

Proposition 17. For any non-empty monotonic deduction function $f : 2^X \rightarrow 2^Y$, there exists a finite set of atomic functions F such that

$$\forall x \subseteq X, f(x) = \bigcup_{\hat{f} \in F} \hat{f}(x).$$

Proof. Since every domain is a finite set of atomic propositions, $f(X)$ is also finite. In addition, for any $y \in f(X)$, the composite function $\gamma_{\{y\}, \{y\}} \circ f$ is an atomic function equivalent to $f(x) \cap \{y\}$. Thus, for any $x \subseteq X$,

$$f(x) = f(x) \cap f(X)$$

$$\begin{aligned}
 &= f(x) \cap \left(\bigcup_{y \in f(X)} \{y\} \right) \\
 &= \bigcup_{y \in f(X)} (f(x) \cap \{y\}) \\
 &= \bigcup_{y \in f(X)} \gamma_{\{y\}, \{y\}} \circ f(x).
 \end{aligned}$$

That is, the set $F = \{\gamma_{\{y\}, \{y\}} \circ f \mid y \in f(X)\}$ satisfies the claim. $\square$

Proposition 18. *For any atomic function $f : 2^X \rightarrow 2^Y$, there exists a unique reduced decomposition $\Gamma \subseteq 2^X \times 2^Y$ such that*

$$\forall x \subseteq X, f(x) = \bigcup_{(u,v) \in \Gamma} \gamma_{u,v}(x),$$

and any u of $(u,v) \in \Gamma$ is a minimal support of f .

Proof. We assume $f(X) = \{y\}$ for some $y \in Y$ without loss of generality. Let σ_0 be the set of every support of f . Then, σ_0 is finite due to the finiteness of domains and satisfies

$$\forall x \subseteq X, f(x) = \bigcup_{s \in \sigma_0} \gamma_{s, \{y\}}(x).$$

Every support in σ_0 is either minimal or not. Suppose $s_0 \in \sigma_0$ is a non-minimal support. By definition of the minimality, there exists at least one proper subset s'_0 of s_0 that is a support of f , and s'_0 must be a member of $\sigma_0 \setminus \{s_0\}$. Whenever $\gamma_{s_0, \{y\}}$ is valid on some $x \subseteq X$, $\gamma_{s'_0, \{y\}}$ is valid on the same x , too. Furthermore, whenever $f(x) = \emptyset$, x is not a support of f ; thus, none of σ_0 is contained in x . Then, the set $\sigma_1 = \sigma_0 \setminus \{s_0\}$ also satisfies

$$\forall x \subseteq X, f(x) = \bigcup_{s \in \sigma_1} \gamma_{s, \{y\}}(x).$$

Similarly, we can remove every non-minimal support from σ_0 one by one. This procedure terminates at some step since σ_0 is finite. Finally, we obtain the set of the minimal supports of f and let it be denoted by $\sigma_\perp$. This set $\sigma_\perp$ is unique since σ_0 is uniquely determined for f and $\sigma_\perp$ is the residual part of σ_0 defined by removing all the non-minimal supports. The reduced decomposition in the claim is defined as $\Gamma = \{(u, \{y\}) \mid u \in \sigma_\perp\}$. $\square$

By applying both [Proposition 17](#) and [Proposition 18](#), we can obtain the unique reduced decomposition of any non-empty monotonic deduction function. It means that [Theorem 14](#) holds.

5.4 Induced automata of exploit dependency graphs

Now, we try to clarify the relationship between exploit dependency graphs and state enumeration graphs or finite automata. The main idea is to define for each exploit dependency graph an induced automaton coupled with the same set of facts. Vice versa, we also show that each finite automaton in a specific class can be associated with exploit dependency graphs with the same set of facts. This bidirectional relationship implies that we can use exploit dependency graphs as a compact form of the class of finite automata.

5.4.1 Induced automata

Remind [Definition 12](#) and the labeling of states. The labeling function $L : S \rightarrow 2^{AP}$ maps a state in a state enumeration graph to a set of atomic propositions. Here, we assume that we can define the inverse mapping $L^{-1} : 2^{AP} \rightarrow S$. Then, it implies that (1) an attack function applies to a “state” converted by L , and (2) the facts inferred by the attack function can be recast into a state by L^{-1} . We use this property to formulate our approach.

Suppose we have a non-empty set of atomic propositions AP . [Algorithm 5.1](#) generates a state enumeration graph from a pair of an attack function $f : 2^{AP} \rightarrow 2^{AP}$ and a set of initial conditions $C_i \subseteq AP$.

Algorithm 5.1 An algorithm for induced automaton generation

Input:

- AP : a set of atomic propositions.
- $f : 2^{AP} \rightarrow 2^{AP}$: an attack function.
- $C_i \subset AP$: a set of initial conditions.

Output:

- $G = (S, \tau, S_0, L)$: a state enumeration graph.

```

1: /* This algorithm regards any element taken from  $2^{AP}$  as a state composing  $S$ .
   Thus, the identity function can represent both  $L$  and  $L^{-1}$ . */
2:  $S_0 \leftarrow \{L^{-1}(C_i)\}$ 
3:  $S \leftarrow S_0$ 
4:  $\tau \leftarrow \emptyset$ 
5:  $Q \leftarrow S_0$ 
6: while  $Q \neq \emptyset$  do
7:    $Q' \leftarrow \emptyset$ 
8:   for  $s \in Q$  do
9:     for  $c \in f(L(s))$  do
10:      if  $c \notin L(s)$  then
11:         $s' \leftarrow L^{-1}(L(s) \cup \{c\})$ 
12:         $S \leftarrow S \cup \{s'\}$ 
13:         $\tau \leftarrow \tau \cup \{(s, s')\}$ 
14:         $Q' \leftarrow Q' \cup \{s'\}$ 
15:      end if
16:    end for
17:  end for
18:   $Q \leftarrow Q'$ 
19: end while

```

We call the state enumeration graph generated by [Algorithm 5.1](#) the *induced automaton* derived from f and C_i . Any induced automaton has the following properties:

- (P1) It has one initial state $s_{\perp}$ and one maximum state $s_{\top}$ where $s_{\perp} = L^{-1}(C_i)$, and $s_{\top} = L^{-1}(f^*(C_i))$.
- (P2) Every state $s \in S$ satisfies that $L(s_{\perp}) \subseteq L(s) \subseteq L(s_{\top})$.
- (P3) Every other state is reachable from the initial state.
- (P4) The maximum state is reachable from every other state.

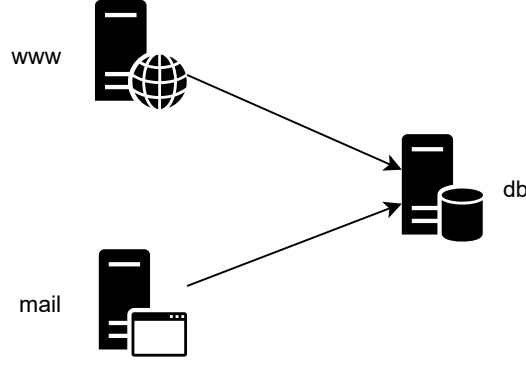


Figure 5.4.1: A mini network

(P5) Every transition is monotonic and incremental, i.e., any $(s_1, s_2) \in \tau$ satisfies that $L(s_1) \subsetneq L(s_2)$, and $|L(s_2) \setminus L(s_1)| = 1$. (This implies that the induced automaton is acyclic.)

(P6) Every transitive state inherits the transitions occurring at its ancestors, i.e.,

$$\begin{aligned} \forall (s_1, s_2) \in \tau, \forall s \in S \setminus \{s_1\}, \exists (s, s') \in \tau, \\ \text{s.t. } L(s) \cap L(s_2) = L(s_1) \Rightarrow L(s') \cap L(s_2) = L(s_2). \end{aligned}$$

The first and the second properties guarantee that all the facts inferred by f and C_i match the atomic propositions extractable from the induced automaton. Therefore, we can analyze the exploit dependency graph to identify those possible facts instead of generating and analyzing the corresponding state enumeration graph. This intuition also applies to every state enumeration graph satisfying all the properties above. Let us name such state enumeration graphs as *attack automata*. The following theorem guarantees that:

Theorem 19 (Induced dependency graph). *For every attack automaton $G = (S, \tau, S_0, L)$ defined on atomic propositions AP , there exists a pair of an attack function $f : 2^{AP} \rightarrow 2^{AP}$ and a set of initial conditions $C_i \subseteq AP$ that satisfies the following conditions: (1) $S_0 = \{L^{-1}(C_i)\}$, and (2) [Algorithm 5.1](#) generates G when applied to f and C_i .*

The proof is given in [Section 5.4.3](#).

[Theorem 19](#) guarantees that using exploit dependency graphs as a space-efficient alternative to attack automata is always legitimate when identifying all possible facts. However, it does not mean that each attack automaton has one unique exploit dependency graph as its induced counterpart. The following subsection shows an example of an attack automaton with corresponding multiple exploit dependency graphs.

5.4.2 Example and implication

Suppose we have a network composed of three hosts as depicted in [Figure 5.4.1](#). Both the host “www” and “mail” have been compromised already, and the attacker can penetrate the “db” server from either of them. We can describe this situation in Datalog as follows:

$a_1 : \text{compromised}(\text{www}).$

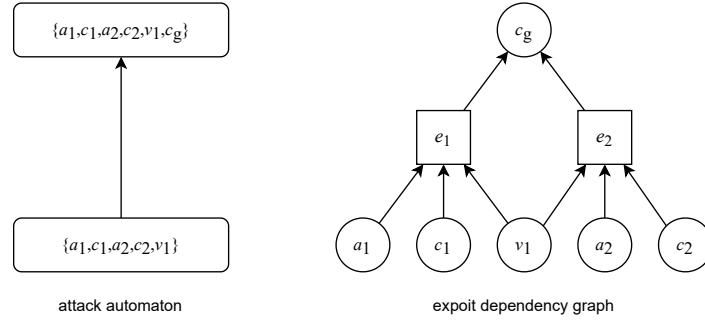


Figure 5.4.2: An example of an attack automaton

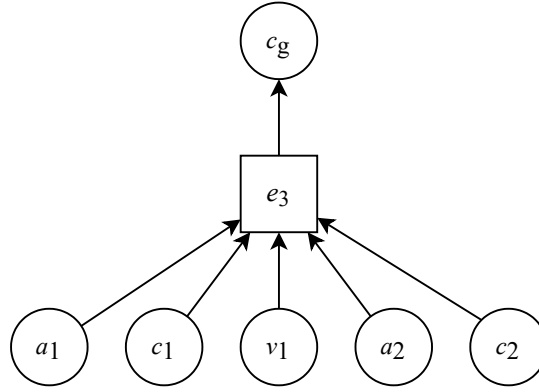


Figure 5.4.3: Another corresponding exploit dependency graph

```

a2 :compromised(mail).
c1 :connectsTo(www, db).
c2 :connectsTo(mail, db).
v1 :hasVulnerability(db, v1).
    compromised(H) :- compromised(X), connectsTo(X, H), hasVulnerability(H, V).

```

Note that we give a label to each fact in the program. Figure 5.4.2 shows the exploit dependency graph and the attack automaton $G = (S, \tau, \{s_\perp\}, L)$ derived from this program where $L(s_\perp) = \{a_1, c_1, a_2, c_2, v_1\}$.

On the other hand, there exist at least the following two attack functions f_1 and f_2 , where both induce the same G :

$$\begin{aligned}
 f_1(x) &= \gamma_{\{a_1, c_1, v_1\}, \{c_g\}}(x) \cup \gamma_{\{a_2, c_2, v_1\}, \{c_g\}}(x), \\
 f_2(x) &= \gamma_{\{a_1, c_1, a_2, c_2, v_1\}, \{c_g\}}(x).
 \end{aligned}$$

It is easy to confirm $f_1(L(s_\perp)) = f_2(L(s_\perp)) = \{c_g\}$. However, the exploit dependency graph generated from f_2 is different from that on Figure 5.4.2. See Figure 5.4.3. The difference comes from ignoring the relationship among initial conditions in the attack automaton, i.e., it has only one initial state full of all the initial conditions.

As this example shows, the relationship between exploit dependency graphs and attack automata

is not bijective. In general, for an exploit dependency graph coupled with some C_i , an infinite number of corresponding attack functions exist. Because the exploit dependency graph does not restrict the behavior of those attack functions applied to other than C_i . Similarly, an attack automaton with the initial state $L^{-1}(C_i)$ has such unbounded attack functions, too. Both forms are inherently some projections of the behavior of an attack function. When we analyze exploit dependency graphs or attack automata, we do examine specific aspects of an attack function behind those graphs.

5.4.3 Proofs for Theorem 19

Theorem 19 immediately follows from Lemma 22, and Lemma 22 is based on Lemma 20.

Lemma 20. *Algorithm 5.1 generates an attack automaton.*

Proof. It is easy to see that Algorithm 5.1 generates an automaton. Here, we need to show the induced automaton is an attack automaton, i.e., it satisfies all the properties P1 to P6 in Section 5.4. For ease of description, we transform these properties into the following equivalent ones:

(Q1) It has one initial state $s_{\perp} = L^{-1}(C_i)$.

(Q2) It has one maximum state $s_{\top}$.

(Q3) Every other state s is reachable from the initial state $s_{\perp}$ and satisfies that $L(s_{\perp}) \subseteq L(s)$.

(Q4) The maximum state $s_{\top}$ is reachable from every other state s and satisfies that $L(s) \subseteq L(s_{\top})$.

(Q5) Every transition is monotonic and incremental, i.e., any $(s_1, s_2) \in \tau$ satisfies that $L(s_1) \subsetneq L(s_2)$, and $|L(s_2) \setminus L(s_1)| = 1$.

(Q6) Every transitive state inherits the transitions occurring at its ancestors, i.e.,

$$\begin{aligned} \forall (s_1, s_2) \in \tau, \forall s \in S \setminus \{s_1\}, \exists (s, s') \in \tau, \\ \text{s.t. } L(s) \cap L(s_2) = L(s_1) \Rightarrow L(s') \cap L(s_2) = L(s_2). \end{aligned}$$

(Q7) The maximum state $s_{\top} = L^{-1}(f^*(C_i))$.

We show that the induced automaton satisfies all the properties Q1 to Q7 as follows.

(Q5) immediately follows from line 10, 11 and 13 in the algorithm.

(Q6) Every edge (s_1, s_2) is generated at line 13, and it implies that, for any state s satisfying $L(s_1) \subseteq L(s)$, $L(s_1) \subsetneq L(s_2)$ and $L(s_2) \setminus L(s_1) \subseteq f(L(s))$. When $L(s) \cap L(s_2) = L(s_1)$ holds additionally, it is equivalent to say that $L(s_1) \subseteq L(s)$ holds but $L(s)$ does not contain $L(s_2) \setminus L(s_1)$ as its part. In such cases, due to line 10 and 11, at least one edge (s, s') will be generated such that $L(s') \setminus L(s) = L(s_2) \setminus L(s_1)$, i.e., $L(s_2) \setminus L(s_1) \subseteq L(s')$. As a result,

$$\begin{aligned} L(s') \cap L(s_2) &= L(s') \cap ((L(s_2) \setminus L(s_1)) \cup L(s_1)) \\ &= (L(s') \cap (L(s_2) \setminus L(s_1))) \cup (L(s') \cap L(s_1)) \\ &= (L(s') \cap (L(s_2) \setminus L(s_1))) \cup L(s_1) && (\because L(s_1) \subseteq L(s')) \\ &= (L(s_2) \setminus L(s_1)) \cup L(s_1) && (\because L(s_2) \setminus L(s_1) \subseteq L(s')) \\ &= L(s_2). \end{aligned}$$

This is what Q6 claims.

(Q1) is guaranteed by line 2 in the algorithm.

(Q3) Q5 and the existence of the single initial state jointly guarantee that every state generated in the algorithm corresponds to an expanded set of C_i . Therefore, the initial state $s_\perp$ and every state $s \in S$ satisfy that $L(s_\perp) \subseteq L(s)$. Similarly, every state is reachable from the initial state.

The remained properties Q2, Q4, and Q7 are about the maximum state $s_\top$.

Let us name a state without out-going edges a *maximal state*. Due to Q5, every maximal state s satisfies $\bigcup_{r \in \rho(s)} L(r) \subseteq L(s)$, where $\rho(s)$ denotes the set of all the states reachable to s . Also, $f(L(s)) \subseteq L(s)$ holds for any maximal state s .

(Q2) Let M be the set of all the maximal states in the graph. It is not empty. Here, suppose that M has at least two different states, and let s_1 and s_2 be the states, respectively. Without loss of generality, we can assume that there exists a proposition p such that (1) it satisfies $p \in L(s_1)$ and $p \notin L(s_2)$, and (2) it is determined at line 9 first among the propositions satisfying (1). The existence of p implies the existence of its support $P \subseteq L(s_1)$. Since p is the first one among its kind, $P \subseteq L(s_2)$ also holds (If not, at least one proposition q in P satisfies (1) prior to p , and it contradicts the claim that p is the first one). Then, p also must appear in $f(L(s_2))$ and be included in $L(s_2)$ since $f(L(s_2)) \subseteq L(s_2)$ holds. This contradicts the requirement $p \notin L(s_2)$. Therefore, the assumption of the existence of multiple maximal states is denied. That is, the graph has only one maximal state. Let us call it the maximum state.

(Q4) The maximum state is the only one state without outgoing edges. Every path on the graph can terminate only at the maximum state. Any state other than the maximum state has at least one outgoing edge, i.e., every other state stands in the middle of or at the start of a path on the graph. In conjunction with Q5, it means Q4.

(Q7) Q5 indicates that the set of propositions of any state on a path is larger than its predecessor state just by one. This means that any state results from chained applications of atomic functions to the initial state. Namely, for any state s , there exists a series of atomic functions $\hat{f}_1, \dots, \hat{f}_n$ decomposable from f such that $L(s) = \hat{C}_n \cup \hat{f}_n(\hat{C}_n)$ where $\hat{C}_{k+1} = \hat{C}_k \cup \hat{f}_k(\hat{C}_k)$ and $\hat{C}_0 = C_i$. Since each atomic function $\hat{f}_k$ is a part of f , and its image is restricted, $\hat{f}_k(C) \subseteq f(C)$ holds for any $C \subseteq AP$. Thus, $\hat{C}_k \subseteq C_k$ also holds, where C_k follows the definition in Definition 15. As a result, it is certain that $L(s_\top) \subseteq f^*(C_i)$.

Let R be the set of propositions determined by $f^*(C_i) \setminus L(s_\top)$, and suppose that R is not empty. Note that C_0 is excluded from R since $C_0 \subseteq L(s_\top)$. Every proposition $r_j \in R$ first appears in some C_{k_j} . Let r_0 be a proposition coupled with the least k_0 , i.e., there is no other $r_j \in R$ such that $k_j < k_0$ where $r_j \in C_{k_j}$. Then, k_0 must be greater than zero because, if not, it means that $r_0 \in C_0$ and contradicts the assumption $r_0 \in R$. On the other hand, if k_0 is positive, we can say that $C_{k_0-1} \subseteq L(s_\top)$ because $C_{k_0-1} \subseteq f^*(C_i)$ and $C_{k_0-1} \cap R = \emptyset$. However, then it means that $r_0 \in f(C_{k_0-1})$ thus $r_0 \in f(L(s_\top))$. This also contradicts the assumption that $r_0 \in R$. In conclusion, R must be empty, and Q7 holds. $\square$

Definition 21 (Induced attack function). Let $G = (S, \tau, S_0, L)$ be an attack automaton defined on atomic propositions AP . For this attack automaton, we call the following attack function $f : 2^{AP} \rightarrow 2^{AP}$ the *induced attack function* from G :

$$f(x) = \bigcup_{(s, s') \in \tau} \gamma_{L(s), L(s') \setminus L(s)}(x).$$

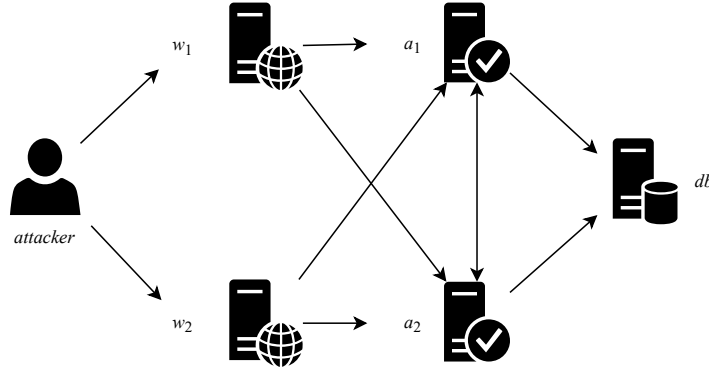


Figure 5.5.1: An example network

Lemma 22. Let $G = (S, \tau, \{s_0\}, L)$ be an attack automaton defined on atomic propositions AP , and $f : 2^{AP} \rightarrow 2^{AP}$ be the induced attack function from G . Then, the induced automaton $G' = (S', \tau', \{s'_0\}, L')$ derived from f and $L(s_0)$ equals G .

Proof. The equality $s_0 = s'_0$ follows immediately from line 2 in Algorithm 5.1. It is sufficient to show $\tau' = \tau$.

In Algorithm 5.1, τ' grows gradually from the empty set. For each iteration of the “while” loop, the state size $|L(s)|$ increments one by one.

When $|L(s)| = |L(s_0)|$, only the reduced functions $\gamma_{L(s_0), L(s') \setminus L(s_0)}$ composing f where $(s_0, s') \in \tau$ are valid on $L(s_0)$. Therefore, every edge (s_0, s') generated for τ' at line 13 is also included in τ . At the same time, all the edges $(s_0, s') \in \tau$ are replicated in τ' because every (s_0, s') is used to define the reduced functions composing f . As a result, the “while” iteration generates all the states s' in S where $|L(s')| = |L(s_0)| + 1$ and all the edges (s_0, s') appearing in τ .

Suppose Algorithm 5.1 in mid-course has generated every state s where $|L(s)| \leq n$ and all the edges connecting those states. Then, in the next “while” iteration, $|L(s)| = n$, and the reduced functions $\gamma_{L(s), L(s') \setminus L(s)}$ composing f where $(s, s') \in \tau$ are valid on $L(s)$. In the similar fashion discussed above, the iteration generates all the states s' where $|L(s')| = n + 1$ and the corresponding edges. Note that, in the iteration, other reduced functions $\gamma_{L(t), L(t') \setminus L(t)}$ where $L(t) \subseteq L(s)$ are also valid on $L(s)$. However, those functions are superseded by some $\gamma_{L(s), L(s') \setminus L(s)}$ because G is an attack automaton and satisfies the property P6 in Section 5.4. As a whole, the iteration replicates the next set of states and edges coupled with the size $n + 1$.

The claim follows from the statements above and the mathematical induction. $\square$

5.5 Example

This section exemplifies the practical implication of the coupled use of exploit dependency graphs and state enumeration graphs—whose consistency is guaranteed by the discussion up to here—by examining a small fictitious case of analysis. Figure 5.5.1 shows the network to be analyzed.

This network indicates that the *attacker* can access the two web servers, w_1 and w_2 . Both web servers access application servers, a_1 and a_2 . The application servers communicate with each other. At the backend, there is a database server, db . Every host has an exploitable vulnerability. In this example, the attacker tries to disable the entire service provided by this network. The attacker has three choices to accomplish this; (1) compromise all the web servers, (2) compromise all the

application servers, or (3) compromise the database server. At least one of these choices must be satisfied to disable the service.

The following Datalog program describes the situation above:

```
# facts
has(db, v1); has(a1, v2); has(a2, v3); has(w1, v4); has(w2, v4)

connectsTo(a1, db); connectsTo(a2, db)
connectsTo(a1, a2); connectsTo(a2, a1)
connectsTo(w1, a1); connectsTo(w1, a2)
connectsTo(w2, a1); connectsTo(w2, a2)
connectsTo(attacker, w1); connectsTo(attacker, w2)

dependsOn(s, db)
hasPrimaryAppServer(s, a1); hasSecondaryAppServer(s, a2)
hasPrimaryWebServer(s, w1); hasSecondaryWebServer(s, w2)
compromised(attacker)

# rules
compromised(X) :- compromised(D), dependsOn(X, D).
compromised(X) :- hasPrimaryAppServer(X, A1), compromised(A1),
                  hasSecondaryAppServer(X, A2), compromised(A2).
compromised(X) :- hasPrimaryWebServer(X, W1), compromised(W1),
                  hasSecondaryWebServer(X, W2), compromised(W2).
```

The attacker's goal is to accomplish "compromised(s)" where "s" indicates the service. [Figure 5.5.2](#) is the exploit dependency graph obtained from this Datalog program.

We can read various implications from this graph. The three choices correspond to the exploit nodes e_4, e_{10} , and e_{13} . Since each exploit node represents the AND combinations of its preconditions, we can prevent the attack by nullifying one of the preconditions. For example, applying a security patch for v_4 nullifies the precondition "has(w_1, v_4)", and thus the exploit e_2 . It means that "compromised(w_1)", and e_4 are also transitively removed. This sort of qualitative analysis gives us the candidates for reasonable security controls to be applied.

On the other hand, such a qualitative analysis does not tell which security treatment is the best. We need additional quantitative metrics to compare the qualitatively determined options. One of the representative quantitative metrics used in risk analysis is probability. If the given exploit dependency graph is acyclic, it is straightforward to calculate the probability while considering the AND/OR combination of conditions, e.g., [169]. Unfortunately, exploit dependency graphs are cyclic in general. As we can see, [Figure 5.5.2](#) contains a cycle composed of "compromised(a_1)", e_8 , "compromised(a_2)", and e_{12} . Even though there are several studies, such as [169, 49] and [Chapter 3](#), proposing probability calculation algorithms applicable to cyclic exploit dependency graphs, each of them has some limitations. It is not our focus to elaborate on them. Here, at least, we can say that they are different from the standard approaches for probability analysis, such as Markov chains and Bayesian networks, and lack the well-studied theoretical foundation for further application, including optimization. This is the point our induced automata can contribute. We can define an attack

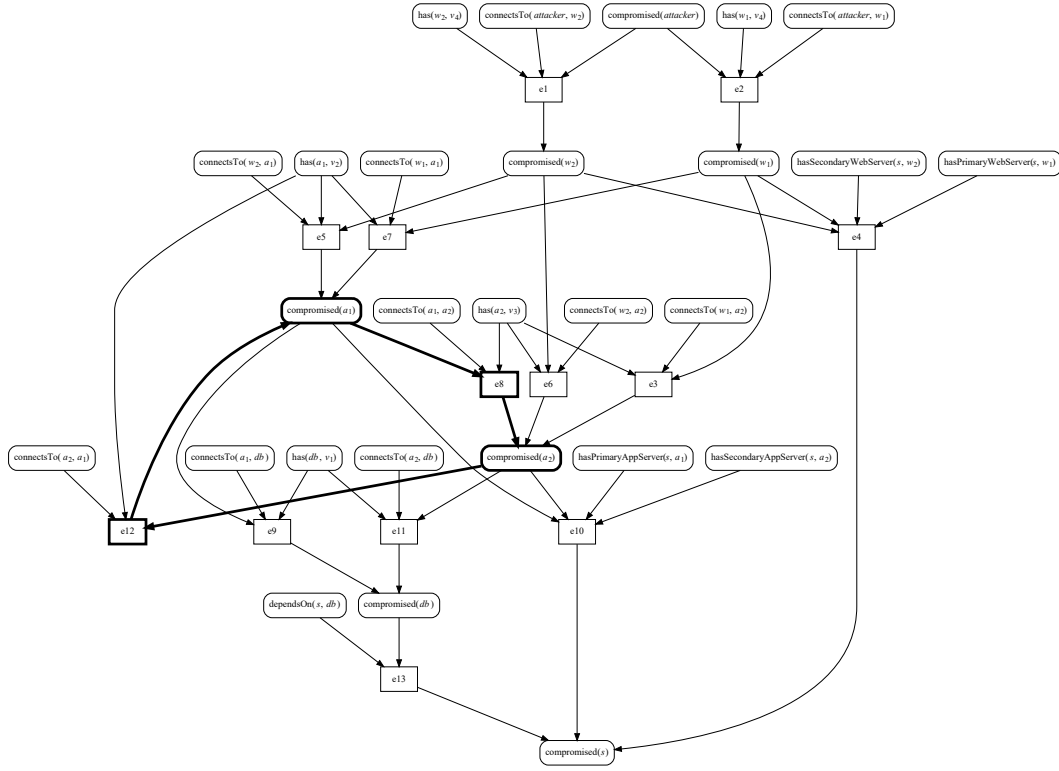


Figure 5.5.2: Exploit dependency graph

function as a Datalog program without facts, i.e., the rules extracted from the Datalog program above define an attack function. With the attack function and Algorithm 5.1, Figure 5.5.3¹ shows the induced automaton coupled with Figure 5.5.2.

Induced automata is a kind of finite automata. Thus it is easy to handle them as Markov chains when we assign appropriate parameters. We can transform this model into a Markov decision process or a partially observable Markov decision process to leverage the relevant optimization algorithms. At the same time, Figure 5.5.3 lack much information shown in Figure 5.5.2. For example, in Figure 5.5.3, we cannot identify that there are three choices required for disabling the service. Therefore, either form only is not sufficient for practical risk analysis. We need both forms in combination. Our result guarantees that we can use both forms in a manner consistent with each other.

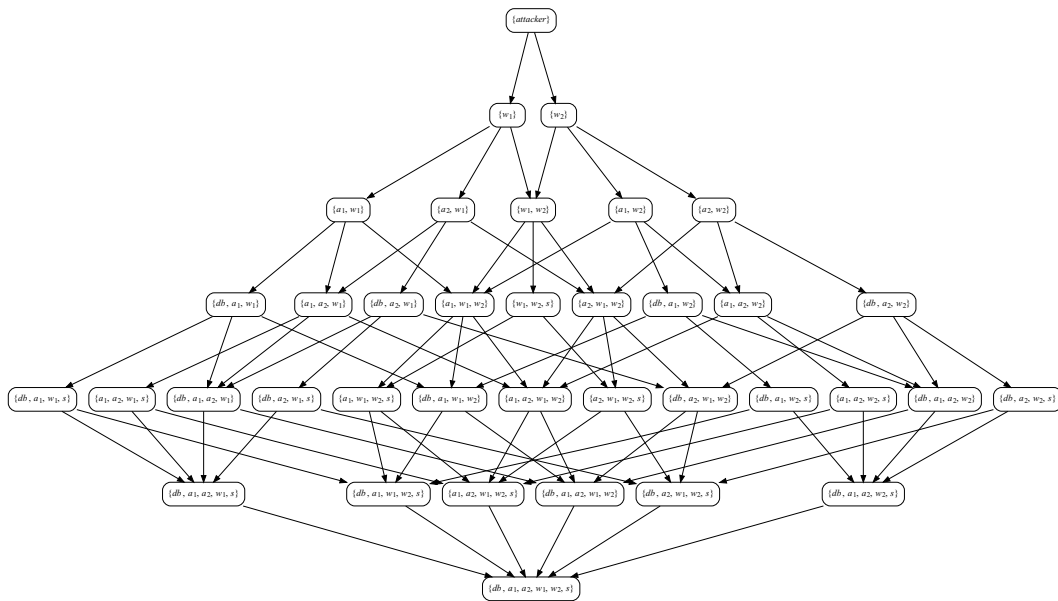
5.6 Discussions

Before closing this chapter, we would like to see the further practical implication of the attack function based approach here. The approach is not a mere theoretical tool. It has the potential for new applications in attack graph analysis.

5.6.1 Mixture of different forms of attack graphs

Every path in an attack automaton starts from the initial state and ends at the maximum state. An attack automaton is a bundle of attack paths between the two states or the two edges. We can use the labeling function to transform the two edges into a set of propositions. Those propositions are

¹The diagram is simplified by showing only the host names to be compromised in each state. For instance, “ a_1 ” is a shorthand for “ $\text{compromised}(a_1)$ ”. Facts other than “ compromised ” are omitted.



interpretable as the initial or goal conditions in an exploit dependency graph. That is, we can embed an attack automaton within an exploit dependency graph and vice versa.

5.6.2 Coarse grained analysis

Let X be a domain, and $f : 2^X \rightarrow 2^X$ be an attack function. We introduce another domain Y and two monotonic deduction functions $g : 2^Y \rightarrow 2^X$ and $g^{-1} : 2^X \rightarrow 2^Y$ satisfying that (1) $g^{-1}(y_1)$ and $g^{-1}(y_2)$ are disjoint whenever $y_1 \neq y_2$, and (2) $\cup_{y \in Y} g^{-1}(\{y\}) = X$. That is, Y and g^{-1} defines a partition of X .

In practice, each element in Y denotes an aggregated form of some $x \subseteq X$. For example, suppose we have atomic propositions $v_1, v_2, \dots, v_n$ in X , and each corresponds to a specific vulnerability with a different CVE identifier, respectively. If we know the exact vulnerability among $v_1, v_2, \dots, v_n$ on the concerned host, we can generate an attack graph. On the other hand, $y \in Y$ denotes the set of $v_1, v_2, \dots, v_n$ and corresponds to a class of vulnerabilities such as a general stack overflow vulnerability not specific to platforms. When we know that there exists a vulnerability belonging to y but do not know the specific CVE identifier, we would like to use y to generate the attack graph. In such a case, we require the resultant attack graph not to miss any consequence derived from any of $v_1, v_2, \dots, v_n$. Using the initial conditions $C_i \subseteq Y$ and the recast attack function $h : 2^Y \rightarrow 2^Y$

defined as $g^{-1} \circ f \circ g : 2^Y \rightarrow 2^Y$ enables such coarse grained analysis. Due to the monotonicity of f, g , and g^{-1} , the inclusive relationship $f^*(D_i) \subseteq g(h^*(C_i))$ is guaranteed for any $D_i \subseteq g(C_i)$.

From the mathematical viewpoint, the suggestion above is trivial. However, it has practical value in the field. Though the importance of information security should be a common understanding within every organization, there are many conflicts. While the security staffs try to allocate more budget to secure their infrastructure, the decision-makers want to spend as much money as possible on directly profitable investments. If the decreased investment in security analysis lessens the risks to be reported, the decision-makers can configure the risk level to be reported at their discretion. One can find the analogous case in the OPM Data Breach [17]. It is of practical importance to have a risk analysis method with which less costly and coarse analysis leads to a greater risk report. It could be an option for such demand to use a set of hierarchically categorized vulnerabilities and the coarse-grained analysis based on them. We will discuss this topic in the following chapter.

5.6.3 Lifting and its application

As is mentioned in Section 5.3.1, an attack function is expressed as a Datalog program without facts. Although Datalog can define arbitrary monotonic deduction function in theory, there are many difficult cases. For example, Datalog does not support the description of arithmetic operations like $a + b > c$. It is also difficult to describe the requirement that at least two of its preconditions are satisfied. The simple syntax of Datalog is sometimes too much restrictive for describing complex conditions.

There is another way to define a monotonic deduction function. Let X and Y be a domain, respectively, and $f : X \rightarrow Y$ be an arbitrary function. Then, the following function $f' : 2^X \rightarrow 2^Y$ is a monotonic deduction function: $f'(x) = \{f(v) \mid v \in x\}$. Similarly, we can transform a function $f_2 : X \times X \rightarrow Y$ into a monotonic deduction function $f'_2(x) = \{f_2(v_1, v_2) \mid v_1 \in x, v_2 \in x\}$. Let us call this kind of transformation the *lifting* of a function. By lifting, we can leverage any sort of functions to define attack functions. For example, we can lift a function written in a programming language like Python to define attack functions.

Moreover, lifting has the potential to incorporate cyber-physical system analysis. In the literature, e.g., [155], a cyber-physical system is often modeled as a discrete-time system such as $x_{k+1} = f(x_k, u_k), y_k = g(x_k, u_k)$ where x_k is the state vector of the system at clock k , u_k is the input to the system, and y_k is the output from the system. By lifting f and g with the discretization of their state space, we can generate attack graphs of this cyber-physical system. To the author's best knowledge, such an approach has never been examined.

In this chapter, we do not dig into the detail of lifting. It will be a topic in our future studies.

5.7 Conclusion

This chapter focused on the theoretical integration of different forms of attack graphs, especially exploit dependency graphs and state enumeration graphs. The central idea is the introduction of attack functions that are monotonic functions between sets. The reduced decomposition of an attack function supplied with initial conditions determines an exploit dependency graph. In addition, we have defined an algorithm to generate an induced automaton from an attack function and the initial conditions. This formulation shows the equivalence between exploit dependency graphs and attack

automata; the latter class is identical to the induced automaton. Both forms of attack graphs are inherently some projections of the behavior of an attack function. When we analyze exploit dependency graphs or attack automata, we do examine specific aspects of an attack function behind those graphs. Attack functions have the potential for further applications such as coarse-grained analysis, cyber-physical system analysis, and the mixture of two forms of attack graphs. In conclusion, we stress the importance of attack function analysis over attack graph analysis in this field.

Now, we have prepared the fundamental tools in need for formulating secured risk analysis. The next chapter integrates the discussions up to here and proposes a formal approach for secured risk analysis.

Chapter 6

Secured risk analysis

6.1 Introduction

Information security management or cybersecurity management is a managerial activity that tries to control the level of the information security risk of an enterprise. It is formulated as a kind of risk management; thus, it requires the repetition of risk assessment to identify the weak part of the enterprise to apply security controls and evaluate the degree of relevant risks. Generally, security practitioners perform the assessment, and business executives consider the security investments in need. However, while security practitioners try to lower the security risks, the business executives want to maximize the profit. Both share finite resources. Therefore the business executives have the potential motivation to underestimate the security risks as long as it enables their business investments by excluding several security investments.

The tendency above is not mere theoretical cynicism, and it is a real phenomenon seen in actual cybersecurity incidents. Let us remind the discussion in [Chapter 1](#). The Office of Personnel Management (OPM), a US federal government agency, was attacked in 2014, and over twenty million personnel records of government officials, including detailed background investigation reports of counter-terrorism officers, were breached [17]. It is pointed out by the investigation committee that the OPM inappropriately prioritized its daily operation over information security. A very similar case is the data breach in the Japan Pension Service in which 1.25 million citizen records were stolen [61, 162, 99]. The postmortem report also criticizes the executives for loosening the security rules deliberately to keep their daily business not disturbed by them. Though it is not a cybersecurity case, we can see an analogous catastrophic case in the Fukushima nuclear reactors' failure. The investigating commissioners [40] "believe that the tsunami risk was underestimated Meanwhile, with regard to severe accident countermeasures, the probabilistic safety assessment of tsunamis was deemed uncertain, so the consideration itself was pushed back and measures were not taken. As such, Tokyo Electric Power Co. used probabilistic evaluations in ways that suited them in order to avoid clarifying the tsunami risk."

The Fukushima case indicates a critical point in general risk management. Every risk management requires some risk assessment, and every risk assessment holds uncertainty up to some degree due to the nature of risk. Roughly speaking, the result of a less costly, coarse-grained risk assessment accompanies larger statistical variation; thus, the lower limit of the estimated risk is also lower than that of the costly, fine-grained risk assessment. In other words, the authority determining the level of security investment can choose the level of assessed risk at his or her discretion. In both the

cases of the OPM and the Japan Pension Service, as the reports state, the business executives were under strong political pressure to perform their business as swiftly as possible. It is natural for such people to tweak the risk assessment process to secure their daily operations even when it degrades information security.

The problem here is the vulnerability of risk assessment to the tendency mentioned above. The preferable risk assessment method should be robust against intentionally constrained resource assignments. More precisely, such a method should estimate greater risks for coarse-grained assessment. This requirement is a further advancement of the risk-weighting accountability enforcement mentioned in [Section 1.2.2](#). At the same time, [Section 1.2.3](#) implies that the method has to clarify the impact of security incidents on business activities to convince the authority to invest in security. In short, we need a robust, business-oriented security risk assessment method. Unfortunately, as discussed in [Section 6.2](#), the existing risk assessment approaches do not satisfy these requirements.

This theoretical study proposes an information-security risk assessment approach based on mathematically proven robustness. At the core, a model-based risk analysis method guarantees that an additional, thus expected to be costly, fine-grained risk analysis always gives a refined subset of the possible incidents compared to the relatively coarse-grained analysis. In other words, estimated incidents roughly decrease with the growth of assessment cost. Moreover, our approach has powerful expressiveness in describing the dependency of business processes on IT infrastructure components due to the nature of Datalog. These characteristics contribute to the robust, business-oriented security risk assessment we want to realize.

[Section 6.2](#) refers to prior studies and points out the lack of the robustness in the existing approaches. [Section 6.3](#) defines our approach. The central tool in our approach is the set-theoretical segregation and aggregation of facts used for the risk analysis. It is an application of the attack function analysis introduced in [Chapter 5](#). [Section 6.4](#) explains our approach with an example. In the example, we model the situation coarsely at first. Then, we segregate facts and filter several of the segregated facts to refine the analysis. The latter fine-grained analysis is guaranteed to give a refined subset of the possible incidents. The example elaborates on the proposed concept in [Section 6.3](#). After discussing several relevant topics, including the obstacles in implementing our approach, we conclude.

6.2 Related work

Many information security management practices in the field follow global standards such as ISO/IEC 27000 [69] and SP-800 [102] series. The standards define the PDCA cycle and the governance structure dedicated to information security management. Though each standard has some unique characteristics, they are all a kind of risk management at their heart. Nevertheless, these standards do not provide specific approaches for risk assessment. For instance, ISO/IEC 27005 [66] only elaborates on the way to categorize risks by a matrix as an example. There is no formally recommended approach for risk assessment in the ISO/IEC 27000 series. IEC/ISO 31010 [57], a part of the general risk management standard ISO 31000 series, is a catalog of various risk assessment approaches, but almost all of them are subjective, like the Delphi method. A similar limitation applies to the SP-800 series. Risk assessment practices in the field are performed with the personal expertise of analysts and have no formal, objective foundation.

On the other hand, many security metrics are proposed in the research field. One can find a variety of them in thorough surveys such as Kordy et al. [84], Ramos et al. [128], Pendleton et al. [122]. Many of these metrics are defined with a specific algorithm to calculate them. Thus, in theory, non-expert analysts can identify the level of the security risk by working out the metrics. Unfortunately, knowing the numerical metrics is not understanding the actual risk posture. Quantitative evaluation of risks requires reliable input data as its ground though it is rare to have enough information and statistics beforehand. Verendel [161] has examined a number of said quantitative metrics and has pointed out their immaturity.

If we focus on the reliability and objectiveness of risk assessment, model-based security assessment could be the candidate. This type of approach analyzes algorithmically some formal model depicting the IT infrastructure. Attack graph analysis is a prominent example. It takes as input the network topology, system configuration, and the distribution of known vulnerabilities over the network. All these inputs are described in a formal language such as Datalog [12], and an algorithm generates a graph-theoretic structure called an attack graph; it is a kind of AND/OR graph or a merged set of fault trees. Each node in an attack graph denotes an incident or security-relevant condition like the existence of a specific vulnerability. Edges explain the causal relationships between incidents and conditions. We can identify the logically possible incidents mechanically from the given input. It is well suited for use by non-experts and for enumerating the possible incidents qualitatively.

The resultant attack graph itself has no quantitative information. For instance, we cannot identify the probability of the incidents implied by it. Several researchers have proposed quantitative metrics deriving from attack graphs. Wang et al. [169] propose a set of intuitive calculation rules of probabilities considering the AND/OR combination of nodes within an essentially acyclic attack graph. Sawilla and Ou [137] define an algorithm applicable to cyclic attack graphs to calculate a metric named AssetRank that is similar to probability and PageRank by Google. Also, [Chapter 3](#) gave an algorithm computing a legitimate probabilistic metric over a cyclic attack graph. These additional quantitative metrics help us use attack graphs as the basis of quantitative risk assessment, even though we still need to refine the input quality.

There are also assistances for business risk assessment based on attack graphs. Bacic et al. [8] expand the default rules given by Ou [116] to model the multiple aspects of risks, such as confidentiality, integrity, and availability. Froh and Henderson [38] designed a Datalog library to accommodate the dependency between services and their infrastructure. Sun et al. [153] propose a Datalog-based framework depicting the effect of infrastructure infringement on enterprise missions. These prior efforts indicate the powerful extensibility of attack graph analysis to support business risk assessment.

Even though these signs of progress are promising, they lack the robustness we need in this study. The intrinsic limitation in attack graph analysis is that it requires the distribution of vulnerabilities to be known beforehand. Zero-day exploits easily break this assumption. In practice, the lack of accurate information on system configuration also inhibits the analysis. Due to the nature of attack graph analysis, if it brings a smaller set of initial conditions, lack of information results in less size of the generated attack graph, i.e., the underestimation of incidents. Intentional constraints on analysis resources and information easily deteriorate the analysis result by decreasing the possible incidents.

This chapter proposes a robust approach based on the mathematical property of Datalog-based inference. Our approach guarantees that the limited input always produces a greater estimation of incidents. To the author's best knowledge, this property is what has never been seen in past studies.

6.3 Approach

This section elaborates on our approach. Our approach put its basis on attack graph analysis. We extend it with two new concepts; segregation and aggregation of facts. Based on the concepts, a robust risk assessment process model is proposed.

6.3.1 Monotonicity, aggregation and segregation

From a mathematical point of view, the execution of a Datalog program takes a set of facts as input and then generates the deduced set of facts. Let P be a Datalog program without facts, S_0 be the input facts for P , and $P(S_0)$ be the set of all the deduced facts by executing P on S_0 . Then, the relationship $S_0 \subseteq S'_0 \Rightarrow P(S_0) \subseteq P(S'_0)$ always holds. Considering that the input facts for attack graph generation indicate the set of vulnerabilities, this *monotonicity* implies a natural tendency that the increased vulnerabilities trigger increased exploits.

Before proceeding with our discussion further, let us define the following one operator and the two operations applicable to Datalog programs:

Definition 23 (Constant replacement operator). Let C be the set of constants used in Datalog programs, and $\gamma : C \rightarrow C$ be a function satisfying $\gamma(c) = \gamma \circ \gamma(c)$ for any $c \in C$. Then, we call γ a *constant replacement operator*.

Definition 24 (Aggregation of facts). Let C be the set of constants used in Datalog programs, and $\gamma : C \rightarrow C$ be a constant replacement operator. We call the operation an *aggregation of facts* by which replacing every constant $c \in C$ appearing in a given Datalog program with $\gamma(c)$.

At a glance, an aggregation of facts only replaces constants and does not aggregate facts. However, replacing constants sometimes maps two different facts to the same single fact. This brings the aggregation of facts. Note that the constant replacement also applies to the constants in rules in the given program.

Additionally, we define the following reversed operation of the aggregation of facts:

Definition 25 (Segregation of facts). Let $\gamma : C \rightarrow C$ be a constant replacement operator, $\gamma^{-1} : \text{Im } \gamma \rightarrow 2^C$ be the inverse mapping of γ such that $\gamma^{-1}(c) = \{c' \mid c' \in C, \gamma(c') = c\}$, P be a Datalog program as the set of facts and rules where every constant $c \in P$ satisfies $c \in \text{Im } \gamma$, and $C_P = \{c_1, c_2, \dots, c_n\}$ be the multiset of all constants in P satisfying $\{c_k\} \subsetneq \gamma^{-1}(c_k)$ where each k corresponds to the order of appearance in the program P . Also, $P[c \rightarrow c']$ denotes the program obtained by replacing only one constant c appearing in P with c' . With these settings, we define the following recurring formula:

$$P_0 = P,$$

$$P_{k+1} = \bigcup_{c' \in \gamma^{-1}(c_{k+1}) \setminus \{c_{k+1}\}} P_k [c_{k+1} \rightarrow c'].$$

We call the operation to obtain P_n from the pair of P and γ as the *segregation of facts*.

Here, let us see a tiny example to understand the behavior of aggregation and segregation. Suppose we have the following three facts in Datalog:

`connectsTo(inet, gw).`

```
connectsTo(gw, httpd).
connectsTo(gw, smtpd).
```

Additionally, let γ be a constant replacement operator such that

$$\gamma(\text{gw}) = \gamma(\text{httpd}) = \gamma(\text{smtpd}) = \gamma(\text{dmz}) = \text{dmz}.$$

This replacement operator implies that the three machines, `gw`, `httpd`, and `smtpd`, belong to the same subnet named `dmz`. The result of the aggregation of the facts above by γ is as follows:

```
connectsTo(inet, dmz).
connectsTo(dmz, dmz).
```

The second and third facts become the same fact by the aggregation and are merged into one. On the contrary, the segregation of the two aggregated facts shown above is as follows:

<code>connectsTo(inet, gw).</code>	<code>connectsTo(httpd, gw).</code>
<code>connectsTo(inet, httpd).</code>	<code>connectsTo(httpd, httpd).</code>
<code>connectsTo(inet, smtpd).</code>	<code>connectsTo(httpd, smtpd).</code>
<code>connectsTo(gw, gw).</code>	<code>connectsTo(smtpd, gw).</code>
<code>connectsTo(gw, httpd).</code>	<code>connectsTo(smtpd, httpd).</code>
<code>connectsTo(gw, smtpd).</code>	<code>connectsTo(smtpd, smtpd).</code>

In essence, `dmz` is expanded to three constants, `gw`, `httpd`, and `smtpd`. Accordingly, the fact “`connectsTo(dmz, dmz)`” is expanded to nine facts based on the product $\{\text{gw}, \text{httpd}, \text{smtpd}\} \times \{\text{gw}, \text{httpd}, \text{smtpd}\}$. Note that this segregated form contains the three facts before the aggregation.

If we regard the first three facts as the true nature of the network to be analyzed, the aggregated form corresponds to the coarse-grained representation since it lacks the detailed structure within the subnet. At the same time, the segregated form is the most conservative assumption about the subnet when we decompose it into more fine-grained components. This way, aggregation and segregation of facts can be tied to the concept of analysis granularity.

The following claim holds about aggregation and segregation. The proof is given in [Section 6.3.3](#). As one can see there, the proof is a by-product of the attack function analysis discussed in [Chapter 5](#).

Theorem 26. *Let P be a Datalog program, S_0 be a set of facts, and γ be a constant replacement operator satisfying that $\gamma[S_0] = S_0$ and $\gamma[P] = P$. Here, $\gamma[P]$ denotes the aggregated form of P , and $\gamma^{-1}[P]$ denotes the segregated form of P . Then, the following equations hold:*

$$\gamma[\gamma^{-1}[P]](S_0) = \gamma[\gamma^{-1}[P](\gamma^{-1}[S_0])] = P(S_0).$$

This theorem means that the aggregation after segregation does not change the set of deducible facts in the basic sense. This property and the monotonicity of Datalog imply that if we segregate a given initial program and remove facts unmatched to reality, the deducible facts are guaranteed to be the subset of that of the initial program. In other words, combining segregation and facts removal decreases the estimated risk events. It corresponds to the refinement of the given coarse-grained assessment expressed in P and S_0 , and the guaranteed decrease of deducible incidents is what we need in this study to secure the risk assessment.

6.3.2 Secured business risk assessment

We use the discussion in the previous subsection to define the secured risk assessment, a risk assessment guaranteed to decrease the estimated risk events when we spend more effort on it. It is an iterative process shown below:

1. Describe the situation to be analyzed as a coarse-grained Datalog program. It is safe to start this first step with limited information.
2. Execute the program, obtain the attack graph, and then analyze it to see the risk level.
3. Finish the assessment if the analysis result is satisfactory, e.g., the risk level is lower enough, or it is sufficiently convincing to consider security control options.
4. If not, choose the coarse-grained facts that should be elaborated, segregate them, and remove inappropriate facts not matched to the situation.
5. Return to the second step.

The iteration of this process incurs cost while reducing the qualitatively estimated risk events. The point is that the gradual decrease of estimated risk events is mathematically guaranteed. If we use a quantitative risk metric proportional to the partial order of estimated risk events, the metric also decreases. The active incorporation of coarse-grained modeling also complements the lack of input information. These are the unique characteristics of this approach not seen in the past studies.

6.3.3 Proofs for Theorem 26

The proof for Theorem 26 is two fold. At first, we elaborate on the properties of attack functions introduced in Chapter 5. Then, we leverage them in the segregation of Datalog programs.

6.3.3.1 About attack functions

Definition 27 (Aggregation function and segregation function). Let X be a domain. When a function $\alpha : X \rightarrow X$ satisfies that $\forall x \in X, \alpha \circ \alpha(x) = \alpha(x)$, we call α an *aggregation function*. For any aggregation function α , X_α denotes the image $\{\alpha(x) \mid x \in X\}$ of α , and we call the inverse mapping $\alpha^{-1} : X_\alpha \rightarrow 2^X$ defined as follows a *segregation function*:

$$\alpha^{-1}(x) = \{w \mid w \in X, \alpha(w) = x\}.$$

We also define the application of these functions to an arbitrary set $W \subseteq X$ as follows:

$$\begin{aligned} \alpha(W) &= \{\alpha(w) \mid w \in W\}, \\ \alpha^{-1}(W) &= \{x \mid x \in X, \alpha(x) \in W\}. \end{aligned}$$

Definition 28 (Narrowed segregation function). Let $\alpha : X \rightarrow X$ be an aggregation function, and $\alpha^{-1} : X_\alpha \rightarrow 2^X$ be the coupled segregation function. When a function $\beta^{-1} : X_\alpha \rightarrow 2^X$ satisfies $\emptyset \neq \beta^{-1}(x) \subseteq \alpha^{-1}(x)$ for any $x \in X_\alpha$, we call β^{-1} a *narrowed segregation function* coupled with α .

Proposition 29. Let $\alpha : X \rightarrow X$ be an aggregation function, and $\alpha^{-1} : X_\alpha \rightarrow 2^X$ be the coupled segregation function. Then, the following property holds:

$$\forall W \subseteq X_\alpha, \alpha(\alpha^{-1}(W)) = W.$$

Proof. By definition, $\alpha^{-1}(\{w\}) = \{x \mid x \in X, \alpha(x) \in \{w\}\} = \{x \mid x \in X, \alpha(x) = w\}$ for any $w \in X_\alpha$. Note that this set, $\alpha^{-1}(\{w\})$, never be empty since $w \in X_\alpha$. Thus,

$$\alpha(\alpha^{-1}(\{w\})) = \{\alpha(x) \mid x \in X, \alpha(x) = w\} = \{w\}.$$

Accordingly, for any $W \subseteq X_\alpha$,

$$\begin{aligned} \alpha(\alpha^{-1}(W)) &= \{\alpha(x) \mid x \in X, \alpha(x) \in W\} \\ &= \bigcup_{w \in W} \{\alpha(x) \mid x \in X, \alpha(x) = w\} \\ &= \bigcup_{w \in W} \{w\} \\ &= W. \end{aligned}$$

□

Corollary 30. *Let $\alpha : X \rightarrow X$ be an aggregation function, and $\beta^{-1} : X_\alpha \rightarrow 2^X$ be a coupled narrowed segregation function. Then, the following property holds:*

$$\forall W \subseteq X_\alpha, \alpha(\beta^{-1}(W)) = W.$$

Theorem 31. *Let $\alpha : X \rightarrow X$ be an aggregation function, $\alpha^{-1} : X_\alpha \rightarrow 2^X$ be the coupled segregation function, and $f : 2^{X_\alpha} \rightarrow 2^{X_\alpha}$ be an attack function. For simplicity, we require f to satisfy $S \subseteq f(S)$ for any $S \subseteq X_\alpha$. Then, the following equation holds for any $S_0 \subseteq X_\alpha$:*

$$f^*(S_0) = \alpha\left((\alpha^{-1} \circ f \circ \alpha)^*(\alpha^{-1}(S_0))\right).$$

Proof. The right-hand side of the claimed equation is a chain of function applications as follows:

$$\alpha\left((\alpha^{-1} \circ f \circ \alpha) \circ \dots \circ (\alpha^{-1} \circ f \circ \alpha)(\alpha^{-1}(S_0))\right).$$

We can interpret this chain as follows:

$$(\alpha \circ \alpha^{-1}) \circ f \circ (\alpha \circ \alpha^{-1}) \circ \dots \circ (\alpha \circ \alpha^{-1}) \circ f \circ (\alpha \circ \alpha^{-1})(S_0).$$

With this interpretation and [Proposition 29](#), it is equivalent to the following:

$$f \circ f \circ \dots \circ f(S_0).$$

This is what the left-hand side means.

□

Corollary 32. *Let $\alpha : X \rightarrow X$ be an aggregation function, $\beta^{-1} : X_\alpha \rightarrow 2^X$ be a narrowed segregation function coupled with α , and $f : 2^{X_\alpha} \rightarrow 2^{X_\alpha}$ be an attack function satisfying $S \subseteq f(S)$ for any*

$S \subseteq X_\alpha$. Then, the following equation holds for any $S_0 \subseteq X_\alpha$:

$$f^*(S_0) = \alpha \left((\beta^{-1} \circ f \circ \alpha)^* (\beta^{-1}(S_0)) \right).$$

Lemma 33. *Let α be an aggregation function, and β^{-1} be a narrowed segregation function coupled with α . Then, for any two sets $S, S' \subseteq X_\alpha$, the following claim holds:*

$$S \subseteq S' \Leftrightarrow \beta^{-1}(S) \subseteq \beta^{-1}(S').$$

Proof. $\Rightarrow$) Due to the definition of β^{-1} , $\beta^{-1}(S_1 \cup S_2) = \beta^{-1}(S_1) \cup \beta^{-1}(S_2)$ holds for any two sets $S_1, S_2 \subseteq X_\alpha$. Also, $S' = (S' \setminus S) \cup S$ since $S \subseteq S'$. Therefore,

$$\beta^{-1}(S') = \beta^{-1}((S' \setminus S) \cup S) = \beta^{-1}(S' \setminus S) \cup \beta^{-1}(S) \supseteq \beta^{-1}(S).$$

$\Leftarrow$) Suppose $S \not\subseteq S'$, i.e., $S \setminus S'$ is not empty. On the other hand, $S = (S \setminus S') \cup (S \cap S')$, and $S' = (S' \setminus S) \cup (S \cap S')$. Thus,

$$\begin{aligned} \beta^{-1}(S) &= \beta^{-1}((S \setminus S') \cup (S \cap S')) = \beta^{-1}(S \setminus S') \cup \beta^{-1}(S \cap S'), \\ \beta^{-1}(S') &= \beta^{-1}((S' \setminus S) \cup (S \cap S')) = \beta^{-1}(S' \setminus S) \cup \beta^{-1}(S \cap S'). \end{aligned}$$

Be cautious that $\beta^{-1}(S \setminus S') \cap \beta^{-1}(S \cap S') = \beta^{-1}(S' \setminus S) \cap \beta^{-1}(S \cap S') = \emptyset$ since β^{-1} is defined as a narrowed inverse mapping of α . Therefore, under the assumptions that $\beta^{-1}(S) \subseteq \beta^{-1}(S')$ and $S \setminus S' \neq \emptyset$, it implies that $\emptyset \neq \beta^{-1}(S \setminus S') \subseteq \beta^{-1}(S' \setminus S)$. Then, there must be at least two elements $s \in S \setminus S'$ and $s' \in S' \setminus S$ such that $\beta^{-1}(s) \cap \beta^{-1}(s') \neq \emptyset$. Note that $s \neq s'$ since they are taken from $S \setminus S'$ and $S' \setminus S$, respectively. However, due to the nature of β^{-1} as before, it cannot be possible, i.e., s and s' must be equivalent whenever $\beta^{-1}(s) \cap \beta^{-1}(s') \neq \emptyset$. Therefore, $S \setminus S'$ is required to be empty, and it means $S \subseteq S'$. $\square$

Proposition 34. *Let $\alpha : X \rightarrow X$ be an aggregation function, β^{-1} be a narrowed segregation function coupled with α . Then, the following property holds:*

$$\forall W \subseteq X, \exists V \subseteq X_\alpha \text{ s.t. } W = \beta^{-1}(V) \Rightarrow W = \beta^{-1} \circ \alpha(W).$$

Proof. The claim is shown as follows:

$$\begin{aligned} \beta^{-1} \circ \alpha(W) &= \beta^{-1} \circ \alpha(\beta^{-1}(V)) \\ &= \beta^{-1}(V) \\ &= W. \end{aligned}$$

$\square$

6.3.3.2 About Datalog program segregations

Let P be a Datalog program, S be a set of facts, and γ be a constant replacement operator. In the following discussion, $\gamma[P]$ denotes the aggregated form of P by γ , and $\gamma^{-1}[P]$ denotes the segregated form of P by γ .

Lemma 35. Let “body” be the body part of a rule in a Datalog program P , $S \subseteq X$ be a set of facts given to the program, and $\gamma : X \rightarrow X$ be a constant replacement operator satisfying that $\gamma[S] = S$ and $\gamma[P] = P$. The set of legitimate substitutions of constants for variables in “body” on S is denoted by $\Sigma(S, \text{body})$. For any $\sigma \in \Sigma(S, \text{body})$, we write the set of facts determined by the substitution as $\sigma(\text{body})$, i.e., $\sigma(\text{body}) \subseteq S$. With these settings, the following two claims holds:

1. $\forall \sigma \in \Sigma(S, \text{body}), \forall \text{body}' \in \gamma^{-1}[\text{body}], \exists \sigma' \in \Sigma(\gamma^{-1}[S], \text{body}') \text{ s.t. } \sigma'(\text{body}') \subseteq \gamma^{-1}[\sigma(\text{body})]$.
2. $\forall \text{body}' \in \gamma^{-1}[\text{body}], \forall \sigma' \in \Sigma(\gamma^{-1}[S], \text{body}'), \exists \sigma \in \Sigma(S, \text{body}) \text{ s.t. } \sigma'(\text{body}') \subseteq \gamma^{-1}[\sigma(\text{body})]$.

Proof. In the following discussion, we use the property that γ and γ^{-1} applied to a set of facts work as an aggregation and a narrowed segregation function, respectively. For example, $\gamma[\gamma^{-1}[S']] = S'$ holds for any fact sets $S' \subseteq S$ because [Corollary 30](#) applies.

1) Since σ defines a substitution of constants for variables in *body*, we can equate σ as a set of pairs of variables and constants. For example, if we have $\sigma = \{(X, c), (Y, d)\}$, let σ denotes the operation that substitutes c for X and d for Y where c and d are constants, and X and Y are the variables in *body*. We write $\sigma = \{(X_1, c_1), \dots, (X_n, c_n)\}$ where $X_1, \dots, X_n$ are all the variables appearing in *body*, and $c_1, \dots, c_n$ are the constants that will be assigned to the paired variables.

By definition, segregation of facts does not affect the variables in rules. It replaces constants in a rule while duplicating the rule to enumerate all the possible combination of the replaced constants. Therefore, the set of variables in $\text{body}' \in \gamma^{-1}[\text{body}]$ is the same as that in *body*, i.e., σ and σ' have the same set of variables. Based on this observation, we define σ' as $\{(X_1, d_1), \dots, (X_n, d_n)\}$ where d_k for $1 \leq k \leq n$ is a constant arbitrarily taken from $\gamma^{-1}(c_k)$. Note that $\gamma^{-1}(c_k)$ is always non-empty due to that $\gamma[S] = S$ and c_k is taken from the constants in S .

Then, $\sigma'(\text{body}')$ is a set of facts where each fact is determined by replacing constants in *body* by γ^{-1} or assigning constants to variables in *body* according to σ' where the constants taken from $\gamma^{-1}(c_k)$ for $1 \leq k \leq n$. In other words, $\sigma'(\text{body}')$ is a segregated form of $\sigma(\text{body})$, i.e., $\sigma'(\text{body}') \subseteq \gamma^{-1}[\sigma(\text{body})]$. At the same time, $\sigma(\text{body}) \subseteq S$ holds due to the nature of σ . Therefore, [Lemma 33](#) guarantees $\gamma^{-1}[\sigma(\text{body})] \subseteq \gamma^{-1}[S]$, i.e., $\sigma' \in \Sigma(\gamma^{-1}[S], \text{body}')$.

2) We write $\sigma' = \{(X_1, d_1), \dots, (X_n, d_n)\}$ where $X_1, \dots, X_n$ are all the variables appearing in *body*, and $d_1, \dots, d_n$ are the constants that will be assigned to the paired variables. Since any d_k is taken from the constants in $\gamma^{-1}[S]$, there exists c_k contained in S such that $d_k \in \gamma^{-1}(c_k)$. For these $c_1, \dots, c_n$, we define $\sigma = \{(X_1, c_1), \dots, (X_n, c_n)\}$. Considering that $\text{body}' \in \gamma^{-1}[\text{body}]$, and every constant d in *body'* satisfies $d \in \gamma^{-1}(c)$ where c is the corresponding constant in *body*, the definition of σ implies that $\sigma'(\text{body}') \subseteq \gamma^{-1}[\sigma(\text{body})]$. On the other hand, for every *body'*, $\sigma'(\text{body}') \subseteq \gamma^{-1}[S]$ is guaranteed by the definition of σ' . By applying γ to both sides, we obtain $\gamma[\sigma'(\text{body}')] \subseteq \gamma[\gamma^{-1}[S]] = S$. Additionally, $\gamma[\sigma'(\text{body}')] = \sigma(\text{body})$ holds since $\gamma[\sigma'] = \sigma$ and $\gamma[\text{body}'] = \text{body}$. Therefore, the substitution is legitimate, i.e., $\sigma \in \Sigma(S, \text{body})$. $\square$

Lemma 36. Let P be a Datalog program, S be a set of facts, and γ be a constant replacement operator satisfying that $\gamma[S] = S$ and $\gamma[P] = P$. Also, let $\mu[P, S]$ denote the set of facts that is the expanded S by adding the facts determined by the unification of P with S . Note that this notation implies that $S \subseteq \mu[P, S]$. Then, the following equations hold:

1. $\gamma^{-1}[\mu[P, S]] = \mu[\gamma^{-1}[P], \gamma^{-1}[S]]$.

$$2. \mu[P, S] = \gamma[\mu[\gamma^{-1}[P], \gamma^{-1}[S]]].$$

Proof. The first equation almost immediately follows from [Lemma 35](#). Because, the lemma implies that the body parts determined by the left-hand side and that of the right-hand side are equal. Therefore, their corresponding heads also match with each other. Also, we can obtain the second equation by applying γ to the both sides of the first equation. $\square$

Theorem 26. *Let P be a Datalog program, S_0 be a set of facts, and γ be a constant replacement operator satisfying that $\gamma[S_0] = S_0$ and $\gamma[P] = P$. Here, $\gamma[P]$ denotes the aggregated form of P , and $\gamma^{-1}[P]$ denotes the segregated form of P . Then, the following equations hold:*

$$\gamma[\gamma^{-1}[P]](S_0) = \gamma[\gamma^{-1}[P](\gamma^{-1}[S_0])] = P(S_0).$$

Proof. To avoid the lengthy description of technical treatment, we show a sketchy proof here. It is easy to see that $\gamma[\gamma^{-1}[P]] = P$. We focus on the equation, $\gamma[\gamma^{-1}[P](\gamma^{-1}[S_0])] = P(S_0)$. The goal for the proof is to show the equivalence between the segregation of facts and the application of a narrowed segregation function. If it stands, [Corollary 32](#) guarantees the claim.

First of all, γ works as an aggregation function when applied to the set of facts since it maps a fact to another fact while satisfying $\gamma = \gamma \circ \gamma$. Similarly, $\gamma^{-1}[S]$ works as a (narrowed) segregation function when S is a set of facts since every modification by γ^{-1} is reversible by γ . Let us introduce α and β^{-1} where each corresponds to the aggregation function and the narrowed segregation function determined by γ and γ^{-1} , respectively. When S is a set of facts, $\gamma[S] = \alpha(S)$ and $\gamma^{-1}[S] = \beta^{-1}(S)$ hold. Also, let $f : 2^{X_\alpha} \rightarrow 2^{X_\alpha}$ be the attack function, corresponding to the program P , satisfying $S \subseteq f(S)$ for any $S \subseteq X_\alpha$ for simplicity. We can define X_α as the set of facts $S_0 \cup P(S_0)$, and it is legitimate to assume the existence of f due to the monotonicity of P .

Now, let us think about the process to obtain the evaluation result of $P(S_0)$. Then, it implies that there exists some $K \in \mathbb{N}$ such that $S_K = P(S_0)$ where $S_{k+1} = \mu[P, S_k]$ for $k = 0, \dots, K-1$. Note that $\mu[P, S] = f(S)$. We obtain $\mu[\gamma^{-1}[P], \gamma^{-1}[S]]$ by applying this notation to the term $\gamma^{-1}[P](\gamma^{-1}[S])$, and it equals $f'(\beta^{-1}(S))$ where f' is an attack function corresponding to $\gamma^{-1}[P]$. If we can show that $f'(\beta^{-1}(S)) = \beta^{-1} \circ f \circ \alpha(\beta^{-1}(S))$ for any $S \subseteq X_\alpha$, it means that [Corollary 32](#) applies to the segregation of facts.

[Lemma 36](#) guarantees that $\gamma[\mu[\gamma^{-1}[P], \gamma^{-1}[S]]] = \mu[P, S]$. In addition,

$$\begin{aligned} \gamma[\mu[\gamma^{-1}[P], \gamma^{-1}[S]]] &= \gamma[f'(\beta^{-1}(S))] = \alpha \circ f'(\beta^{-1}(S)), \\ \mu[P, S] &= f(S) = f \circ \alpha(\beta^{-1}(S)). \end{aligned}$$

That is,

$$\alpha \circ f'(\beta^{-1}(S)) = f \circ \alpha(\beta^{-1}(S)).$$

By applying β^{-1} to the both sides of the equation above, we get the following equation:

$$\beta^{-1} \circ \alpha \circ f'(\beta^{-1}(S)) = \beta^{-1} \circ f \circ \alpha(\beta^{-1}(S)).$$

At the same time, $\mu[\gamma^{-1}[P], \gamma^{-1}[S]] = \gamma^{-1}[\mu[P, S]]$ is also guaranteed by [Lemma 36](#). Therefore,

$$\mu[\gamma^{-1}[P], \gamma^{-1}[S]] = f'(\beta^{-1}(S))$$

$$= \gamma^{-1} [f(S)] = \beta^{-1} (f(S)).$$

That is,

$$f'(\beta^{-1}(S)) = \beta^{-1}(f(S)).$$

Note that $f(S) \subseteq X_\alpha$. With this observation and [Proposition 34](#) indicate $\beta^{-1} \circ \alpha \circ f'(\beta^{-1}(S)) = f'(\beta^{-1}(S))$. Consequently, $f'(\beta^{-1}[S]) = \beta^{-1} \circ f \circ \alpha(\beta^{-1}[S])$. $\square$

6.4 Example

In this section, we see a tiny but practical use case of the segregation of facts in the context of secured risk assessment. We omit the detail and focus only on the essential part to clarify the points.

[Figure 6.4.1](#) is a diagram representation of Datalog facts. For instance, the part “business-[enabledBy]-app” corresponds to the term “`enabledBy(business, app)`” in a Datalog program. The program corresponding to this diagram depicts a situation where a “business” is enabled by some “app,” and the “app” depends on a network infrastructure containing some vulnerabilities. We use the following three rules to analyze the situation.

```

compromised(X)
:- compromised(Y), connectsTo(Y,X),
   hasVulnerability(X,V).

compromised(S)
:- hostedOn(S,H), compromised(H).

compromised(B)
:- enabledBy(B,S), compromised(S).

```

As the red-colored part in the diagram indicates, the attack graph analysis applied to this situation infers the possible attack path from the “attacker” to the “business.”¹

However, this analysis is rough since we do not distinguish the business’s specific processes and the various applications enabling them. [Figure 6.4.1](#) implies that every business will be affected by the potential attack. It might be a too conservative estimation. To refine the analysis, we introduce a constant replacement operator γ satisfying the following properties:

```

business =  $\gamma$ (accounting)
          =  $\gamma$ (sales)
          =  $\gamma$ (business),

app =  $\gamma$ (accounting_system)
     =  $\gamma$ (sales_management_system)
     =  $\gamma$ (app).

```

With this operator, we can segregate [Figure 6.4.1](#) into the fine-grained form as shown in [Figure 6.4.2](#).

¹Do not confuse the diagram with an attack graph. We omit the specific attack graphs obtained from the diagrams in this chapter. Attack graphs are implicitly referred to determine the facts exploited for potential attacks.

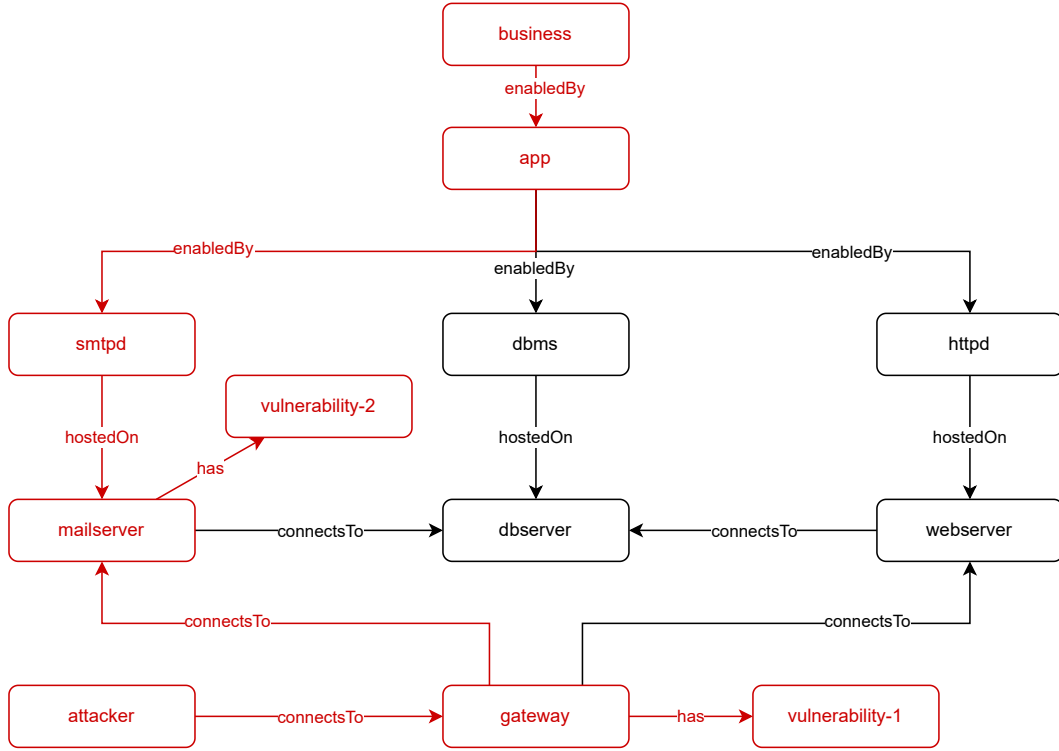


Figure 6.4.1: Initial coarse-grained program diagram

Note that this transform corresponds to the “ $\gamma^{-1}[S_0]$ ” part in [Theorem 26](#).²

This segregated form contains every possible relation between the segregated constants. If we examine the actual state of the business and its use of applications, we could remove several of them. For example, if we know that only the “sales_management_system” uses “smtpd,” and the “accounting_system” does not, we can remove the “enabledBy” relation between the “accounting_system” and the “smtpd.” It corresponds to a removal of a fact in Datalog programs. [Figure 6.4.3](#) is the exemplified diagram of the facts that applied this sort of reduction.

Now, the attack graph analysis applied to this refined form identifies that “sales” could be affected by “attacker” while “accounting” is not. Considering that, in [Figure 6.4.1](#), the “business” aggregates both “sales” and “accounting,” the fine-grained analysis dismisses the risk event to “accounting.” That is, the refinement of the analysis decreases the estimated incidents.

6.5 Discussions

This chapter demonstrated the concept of our approach by a small example. At its heart, our approach is based on a risk assessment algorithm that guarantees the refinement of the analysis monotonically decreases the estimated incidents. In other words, limited information leads to a coarse-grained assessment with greater incidents estimation. This property encourages anti-motivated business executives to invest in risk assessment proactively. To the author’s knowledge, this secured monotonicity is the unique characteristic of our approach in this field. On the other hand, it is still in its early stage of development. We discuss several topics for further advancement here.

²Since the rules used here do not contain constants, $\gamma^{-1}[P] = P$.

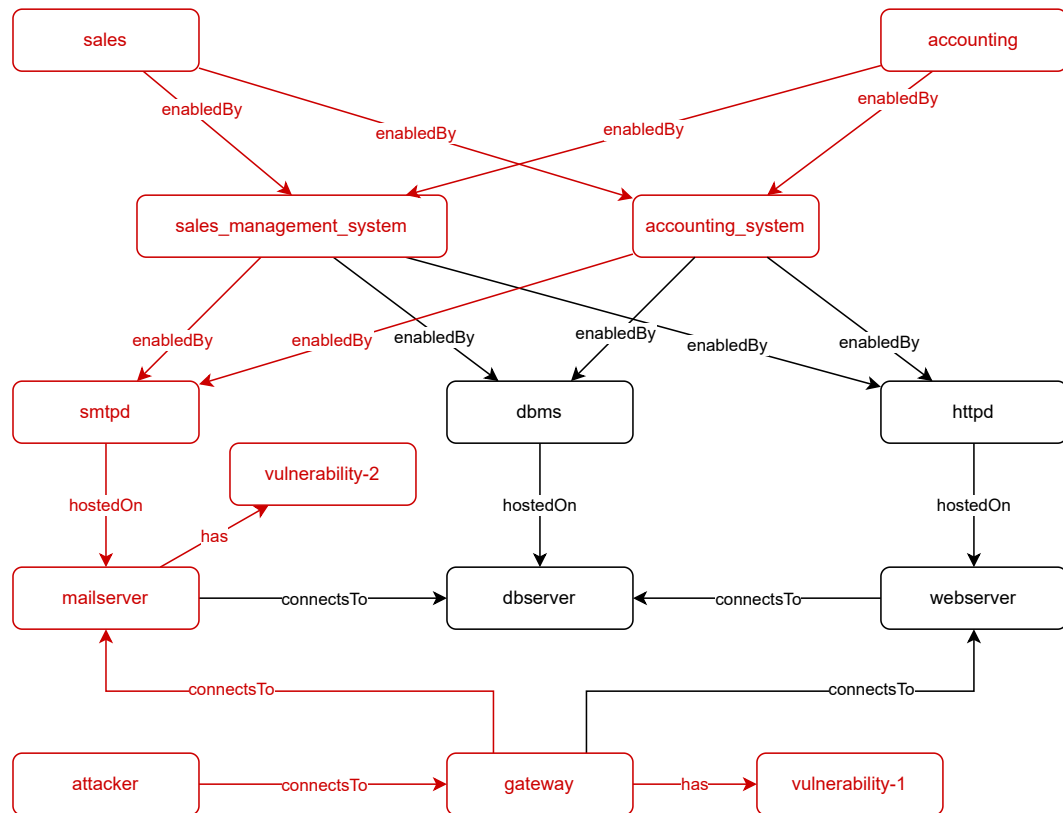


Figure 6.4.2: Segregated form

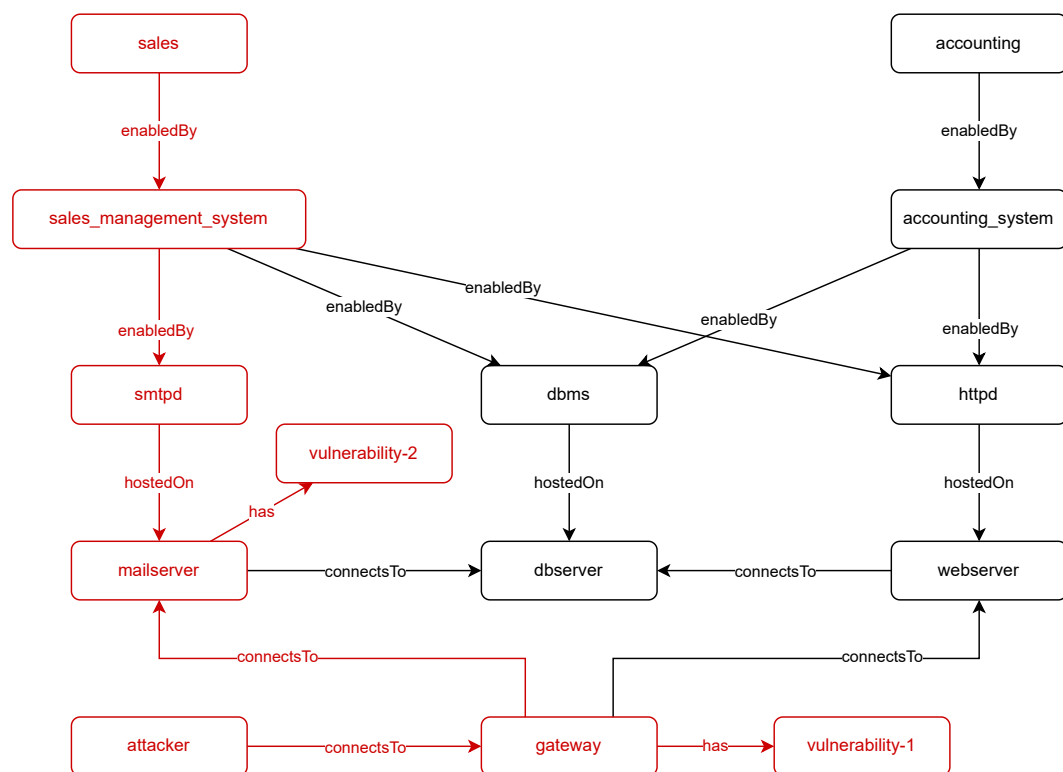


Figure 6.4.3: Fine-grained program diagram

6.5.1 Importance of vocabulary development

As the example shows, our approach can handle the dependencies of business processes on IT infrastructures on the strength of Datalog’s powerful expressiveness. It requires a carefully designed set of predicates and constants, and such design itself requires non-trivial effort. At this point, enterprise architecture practices can be a valuable source for design. Enterprise architecture is a whole picture of an enterprise, including physical infrastructure, application services, business processes, and strategic objectives. Various ontologies, such as TOGAF [157] and ArchiMate [158], are dedicated to enterprise architecture description. We can use these already tested frameworks as our basis. Notably, ArchiMate has a formal definition suitable for expressing in Datalog; the diagrams in ArchiMate closely resemble the diagrams in this chapter.

6.5.2 Need for granularity free quantitative metrics

The robustness enabled by our approach is just qualitative. The set-theoretical monotonicity does not necessarily match the monotonicity of quantitative security metrics obtained from those incident sets. Even though quantitative metrics for attack graph analysis are proposed, our approach does not guarantee the monotonic decrease of those metrics according to the refinement. It depends on the metric design. Our approach guarantees that the refined attack graph is a sub-graph of the segregated form. Therefore, a metric that monotonically decreases when applied to a sub-graph of an attack graph is a preferable candidate for quantitative assessment in conjunction with our approach. One of the most simple metrics satisfying this property is the number of deduced facts counted under the same granularity. Several other attack-graph-based metrics also satisfy this property, such as [169], [13], and the one in [Chapter 3](#). The process in [Section 6.3.2](#) implicitly assumes using such a metric to evaluate the risk level. However, it is desirable to have metrics applicable without leveling the granularity. We need further research.

6.5.3 Integration with information security risk treatment

Information security risk assessment is a gear of information security management, and we have to plan and perform information security risk treatment and then feedback on the result to the next assessment. The smooth cycle of assessment and treatment needs a systematic approach in treatment planning. That is the point network hardening algorithms can help.

As our approach premises the use of attack graphs and the segregation and aggregation of facts as its basis, not every network hardening algorithm applies well. It is desirable to have a hardening algorithm affine to those requirements. The one proposed in [Chapter 4](#) is a candidate. When it is used in conjunction with the metric in [Chapter 3](#), it applies to cyclic attack graphs and works efficiently. However, the performance is not sufficient enough to handle large-scale problems. Further development is needed.

6.5.4 Testing in the field

In essence, the proposed approach provides a formal foundation for secured risk modeling in the sense that the volume of estimated incidents is proportional to its analysis granularity. The approach does not give any specific evaluation criteria for information security risks. Risk is a multi-faceted and complex concept; there are many ways to define and evaluate it. The focal point of this study is to

formally clarify another viewpoint about risks and to formulate a framework where we can introduce more specific metrics and rich descriptions. In other word, this study is just a theoretical suggestion.

The proposed approach is not tested in the field. Since the motivation of this study is to secure the information security management process from stakeholders' conflicts, the test should clarify the effect on decision-making under such conflicts. It is challenging to show such managerial effects in the field. Our future research will focus on this topic.

6.6 Conclusion

This chapter proposed an information risk assessment approach guaranteeing that the refined assessment always gives less incidents estimation. This property secures the risk assessment process from the conflicts between stakeholders; each has a different motivation and competes for finite resources. Our approach is based on the aggregation and the segregation of facts appearing in Datalog programs. Datalog programs are used to generate attack graphs, and attack graph analysis is one of the model-based risk assessment approaches with qualitative inference. We can use the powerful expressiveness of Datalog and quantitative metrics designed for attack graphs to leverage our approach as the basis of secured business risk assessment. However, its practical use requires carefully designed ontology in Datalog, and the unguaranteed decrease in quantitative metrics for refined assessment is an issue yet to be solved. Including the field testing and a more efficient network hardening algorithm, we need further research effort to make our approach a practical tool.

Chapter 7

Summary

7.1 Contributions

In this dissertation, we have started our discussion by pointing out the misalignment between stakeholders in information security management practices. Then, we gave detailed analyses of the issue and several proposals for the solution. All the technical discussions are formulated around the monotonicity of inference. Let us take a look back at those contributions here.

[Chapter 1](#) refers to the actual data breach cases and points out the conflicts between security practitioners and business executives. The focal point in [Chapter 1](#) is the misalignment in the stakeholders' decision-making. While security practitioners try to lower security risks, business executives pursue profits. Since they share the finite resources to invest, conflicts can and do occur. Unfortunately, the current best practices and the research results seem not to handle those conflicts well. This landscape motivated us to set the research program, which enables a disciplined alignment of stakeholders. The rest of this dissertation is devoted to establishing the theoretical foundation for the secured risk analysis immune to those conflicts.

[Chapter 2](#) examines the attack graph analysis, a model-based approach for network security analysis. Attack graph analysis is essentially a qualitative analysis, and we can use it in combination with quantitative security metrics. This separation enforces accountability in risk weighing, and it makes attack graph analysis an attractive candidate for the secured risk analysis approach. [Chapter 2](#) elaborates on past studies to clarify the characteristics of this approach and the issues in practice. What we pointed out there are the following three lacks; (1) the lack of a scalable, probabilistic metric applicable to cyclic attack graphs, (2) the lack of a scalable, multi-objective network hardening algorithm, and (3) the lack of accurate information required for the actual analyses.

[Chapter 3](#) proposes a probabilistic metric applicable to cyclic attack graphs. In short, it is a metric assigning reachability probabilities to the worst-case attack paths for nodes in an attack graph. It is far more efficient to calculate compared to Bayesian attack graphs and has no problem applying it to cyclic attack graphs. On the other hand, it lacks dynamic analysis capability based on the observed incidents, which is a prominent value in Bayesian attack graphs. In contrast, our approach is suitable for static analysis, usually done during the information security risk assessment process.

[Chapter 4](#) proposes a multi-objective network hardening algorithm. It is a greedy search algorithm inspired by gradient descent search algorithms popular in optimization research. Our algorithm leverages the monotonicity of metrics given to the algorithm to approximately but steadily refine the tentative solutions. We can use the metric proposed in [Chapter 3](#) with this algorithm. Even though

there should be more improvements, our algorithm outperforms the well-known NSGA-II algorithm quite well. It could be an enabler for computer-assisted security design.

[Chapter 5](#) clarifies the theoretical foundation for using exploit dependency graphs, which are the major form of attack graphs in the research field. While there are several definitions of attack graphs, the spatial efficiency of exploit dependency graphs makes them the primary candidate for research and practical use. However, compared to the state enumeration graphs with the rich background of finite automata, it has been unclear what class of problems can be handled by exploit dependency graphs. We have shown the corresponding class of finite automata to clarify this point. Moreover, we have introduced the function form of attack graph, i.e., attack function. The concept of attack function implies the potential research area, such as the legitimate mixture of state enumeration graphs and exploit dependency graphs, lifting of functions, and coarse-grained analysis.

Finally, [Chapter 6](#) has shown a way to implement the secured risk analysis. The main contribution of this study is that we successfully define an algorithmic process to proceed with business risk analysis from coarse-grained models to more fine-grained models without underestimating the set of incidents. In other words, estimated incidents roughly decrease with the growth of assessment cost. The concept of attack function enables the proof. This mathematically guaranteed property urges business executives to invest in security more while focusing on its cost-benefits. The active incorporation of coarse-grained modeling also complements the lack of input information. This is exactly what we want as the foundation of the secured risk analysis immune to the conflicts among stakeholders.

As a whole, this study has proposed solutions to overcome the weaknesses in attack graph analysis and establish the foundation of the disciplined alignment in information security management.

7.2 Future works

Our proposed methods are far from complete, and there are many limitations. The technical ones of those are discussed in each chapter of this dissertation, and we refrain from mentioning them again here. Instead, we would like to see several directions for further research.

The most intuitive direction is building upper tiers for the disciplined alignment. The result in this dissertation is just a foundation, and much work must be done before their practical use. For example, segregation-supported ontology in Datalog is essential for implementing secured risk analysis. We also need some verification and validation criteria to assess its effectiveness. These topics are undoubtedly crucial.

From the theoretical point of view, function lifting has great potential. As is mentioned in [Section 5.6.3](#), cyber-physical systems require a different form of system description than Datalog and attack graphs. In theory, attack function analysis with the lifting of discrete-time system functions can integrate those various forms into one. If actually possible, it should be yet another research frontier.

If we broaden our horizons further, our approach could be applied to more than security. The essence of attack function analysis is that the forms of the systems under analysis can be unified into monotonic functions by lifting, thereby digitizing their behavior. Any system expressible as a function can be put into the same framework. Is this theoretical property actually useful? Can we leverage it practically? There is plenty of exciting work to be done.

Bibliography

- [1] M. Ugur Aksu, M. Hadi Dilek, E. Islam Tatli, Kemal Bicakci, H. Ibrahim Dirik, M. Umut Demirezen, and Tayfun Aykir. A quantitative CVSS-based cyber security risk assessment methodology for IT systems. *Proceedings - International Carnahan Conference on Security Technology*, 2017-Octob:1–8, 2017. ISSN 10716572. doi: 10.1109/CCST.2017.8167819.
- [2] M Ugur Aksu, Kemal Bicakci, M Hadi Dilek, A Murat Ozbayoglu, and E Islam Tatli. Automated Generation of Attack Graphs Using NVD. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, volume 2018-Janua, pages 135–142, New York, NY, USA, mar 2018. ACM. ISBN 9781450356329. doi: 10.1145/3176258.3176339.
- [3] Massimiliano Albanese, Sushil Jajodia, and Steven Noel. Time-efficient and cost-effective network hardening using attack graphs. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, pages 1–12. IEEE, jun 2012. ISBN 978-1-4673-1625-5. doi: 10.1109/DSN.2012.6263942.
- [4] Hamid Ali and Farrukh Aslam Khan. Attributed multi-objective comprehensive learning particle swarm optimization for optimal security of networks. *Applied Soft Computing*, 13(9): 3903–3921, sep 2013. ISSN 15684946. doi: 10.1016/j.asoc.2013.04.015.
- [5] Hussain M.J. Almohri, Layne T. Watson, Danfeng Yao, and Xinming Ou. Security Optimization of Dynamic Networks with Probabilistic Graph Modeling and Linear Programming. *IEEE Transactions on Dependable and Secure Computing*, 13(4):474–487, jul 2016. ISSN 1545-5971. doi: 10.1109/TDSC.2015.2411264.
- [6] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM conference on Computer and communications security - CCS '02*, page 217, New York, New York, USA, 2002. ACM Press. ISBN 1581136129. doi: 10.1145/586110.586140.
- [7] Paul Ammann, Joseph Pamula, Julie Street, and Ronald Ritchey. A Host-Based Approach to Network Attack Chaining Analysis. In *21st Annual Computer Security Applications Conference (ACSAC'05)*, pages 72–84. IEEE, 2005. ISBN 0-7695-2461-3. doi: 10.1109/CSAC.2005.6.
- [8] Eugen Bacic, Michael Froh, and Glen Henderson. MulVAL Extensions For Dynamic Asset Protection. *Defence R&D Canada*, (April), 2006.
- [9] Tanveer Hossain Bhuiyan, Apurba K. Nandi, Hugh Medal, and Mahantesh Halappanavar. Minimizing expected maximum risk from cyber-Attacks with probabilistic attack success. *2016*

- IEEE Symposium on Technologies for Homeland Security, HST 2016*, pages 1–6, 2016. doi: 10.1109/THS.2016.7568892.
- [10] Ruth Breu, Frank Innerhofer-Oberperfler, and Artsiom Yautsiukhin. Quantitative Assessment of Enterprise Security System. In *2008 Third International Conference on Availability, Reliability and Security (ARES'08)*, pages 921–928, 2008. ISBN 978-0-7695-3102-1. doi: 10.1109/ARES.2008.164.
- [11] Chen Cao, Lun Pin Yuan, Anoop Singhal, Peng Liu, Xiaoyan Sun, and Sencun Zhu. Assessing attack impact on business processes by interconnecting attack graphs and entity dependency graphs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10980 LNCS:330–348, 2018. ISSN 16113349. doi: 10.1007/978-3-319-95729-6_21.
- [12] Stefan Ceri, Georg Gottlob, and Letizia Tanca. What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166, mar 1989. ISSN 10414347. doi: 10.1109/69.43410.
- [13] Pengsu Cheng, Lingyu Wang, Sushil Jajodia, and Anoop Singhal. Aggregating CVSS Base Scores for Semantics-Rich Network Security Metrics. In *2012 IEEE 31st Symposium on Reliable Distributed Systems*, pages 31–40. IEEE, oct 2012. ISBN 978-1-4673-2397-0. doi: 10.1109/SRDS.2012.4.
- [14] Pengsu Cheng, Lingyu Wang, Sushil Jajodia, and Anoop Singhal. Refining CVSS-Based Network Security Metrics by Examining the Base Scores. In *Network Security Metrics*, pages 25–52. Springer International Publishing, Cham, 2017. doi: 10.1007/978-3-319-66505-4_2.
- [15] Matthew Chu, Kyle Ingols, Richard Lippmann, Seth Webster, and Stephen Boyer. Visualizing attack graphs, reachability, and trust relationships with NAVIGATOR. In *Proceedings of the Seventh International Symposium on Visualization for Cyber Security - VizSec '10*, pages 22–33, New York, New York, USA, 2010. ACM Press. ISBN 9781450300131. doi: 10.1145/1850795.1850798.
- [16] Chun Jen Chung, Pankaj Khatkar, Tianyi Xing, Jeongkeun Lee, and Dijiang Huang. NICE: Network intrusion detection and countermeasure selection in virtual network systems. *IEEE Transactions on Dependable and Secure Computing*, 10(4):198–211, 2013. ISSN 15455971. doi: 10.1109/TDSC.2013.8.
- [17] Committee on Oversight and Government Reform. The OPM Data Breach: How the Government Jeopardized Our National Security for More than a Generation. Technical report, U.S. House of Representatives, 2016.
- [18] Oscar Cordon, Francisco Herrera, and Thomas Stützle. A review on the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware & Soft Computing*, 9:141—175, 2002.
- [19] Paul Dagum and Michael Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, mar 1993. ISSN 00043702. doi: 10.1016/0004-3702(93)90036-B.

- [20] Adnan Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge university press, 2009. ISBN 9781139478908.
- [21] Leonardo de Moura and Nikolaj Bjørner. Z3: An Efficient SMT Solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008. doi: 10.1007/978-3-540-78800-3_24.
- [22] Kalyanmoy Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, apr 2002. ISSN 1089778X. doi: 10.1109/4235.996017.
- [23] Rinku Dewri, Nayot Poolsappasit, Indrajit Ray, and Darrell Whitley. Optimal security hardening using multi-objective optimization on attack tree models of networks. In *Proceedings of the 14th ACM conference on Computer and communications security - CCS '07*, page 204, New York, New York, USA, 2007. ACM Press. ISBN 9781595937032. doi: 10.1145/1315245.1315272.
- [24] Rinku Dewri, Indrajit Ray, Nayot Poolsappasit, and Darrell Whitley. Optimal security hardening on attack tree models of networks: a cost-benefit analysis. *International Journal of Information Security*, 11(3):167–188, jun 2012. ISSN 1615-5262. doi: 10.1007/s10207-012-0160-y.
- [25] Francisco J. Diez and Marek J. Druzdzel. Canonical Probabilistic Models for Knowledge Engineering. Technical report, UNED, Madrid, Spain, 2006.
- [26] Marco Dorigo, Vittorio Maniezzo, and Alberto Coloni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, feb 1996. ISSN 1083-4419. doi: 10.1109/3477.484436.
- [27] Karel Durkota, Viliam Lisý, Branislav Bošanský, and Christopher Kiekintveld. Approximate solutions for attack graph games with imperfect information. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9406:228–249, 2015. ISSN 16113349. doi: 10.1007/978-3-319-25594-1_13.
- [28] Karel Durkota, Viliam Lisy, Branislav Bosansky, and Christopher Kiekintveld. Optimal network security hardening using attack graph games. *IJCAI International Joint Conference on Artificial Intelligence*, 2015-Janua(Ijcai):526–532, 2015. ISSN 10450823.
- [29] Karel Durkota, Viliam Lisy, Christopher Kiekintveld, Branislav Bosansky, and Michal Pechoucek. Case studies of network defense with attack graph games. *IEEE Intelligent Systems*, 31(5):24–30, 2016. ISSN 15411672. doi: 10.1109/MIS.2016.74.
- [30] Karel Durkota, Viliam Lisý, Branislav Bošanský, Christopher Kiekintveld, and Michal Pěchouček. Hardening networks against strategic attackers using attack graph games. *Computers and Security*, 87:101578, nov 2019. ISSN 01674048. doi: 10.1016/j.cose.2019.101578.
- [31] Simon Yusuf Enoch, Jin B. Hong, Mengmeng Ge, Khaled M. Khan, and Dong Seong Kim. Multi-Objective Security Hardening Optimisation for Dynamic Networks. *IEEE International Conference on Communications*, 2019-May, 2019. ISSN 15503607. doi: 10.1109/ICC.2019.8761984.

- [32] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971. ISSN 00043702. doi: 10.1016/0004-3702(71)90010-5.
- [33] Barbara Fila and Wojciech Widel. Exploiting attack-defense trees to find an optimal set of countermeasures. *Proceedings - IEEE Computer Security Foundations Symposium*, 2020-June: 395–410, 2020. ISSN 19401434. doi: 10.1109/CSF49147.2020.00035.
- [34] FIRST. Common Vulnerability Scoring System version 3.1: Specification Document Revision 1. pages 1–28, 2019.
- [35] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13(Jul):2171–2175, 2012.
- [36] Marcel Frigault and Lingyu Wang. Measuring Network Security Using Bayesian Network-Based Attack Graphs. In *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pages 698–703. IEEE, 2008. ISBN 978-0-7695-3262-2. doi: 10.1109/COMPSAC.2008.88.
- [37] Marcel Frigault, Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring network security using dynamic bayesian network. In *Proceedings of the 4th ACM workshop on Quality of protection - QoP '08*, page 23, New York, New York, USA, 2008. ACM Press. ISBN 9781605583211. doi: 10.1145/1456362.1456368.
- [38] Michael Froh and Glen Henderson. MulVAL extensions II Defence R&D Canada-Ottawa. (August), 2009.
- [39] Ellen H. Fukuda and Luis Mauricio Graña Drummond. A survey on multiobjective descent methods. *Pesquisa Operacional*, 34(3):585–620, 2014. ISSN 16785142. doi: 10.1590/0101-7438.2014.034.03.0585.
- [40] Fukushima Nuclear Accident Independent Investigation Commission. The official report of the Fukushima nuclear accident independent investigation commission. Technical report, The National Diet of Japan, 2012.
- [41] Mengmeng Ge, Jin B. Hong, Walter Guttman, and Dong Seong Kim. A framework for automating security analysis of the internet of things. *Journal of Network and Computer Applications*, 83(April 2016):12–27, 2017. ISSN 10958592. doi: 10.1016/j.jnca.2017.01.033.
- [42] Mohammad Ghasemigol, Abbas Ghaemi-Bafghi, and Hassan Takabi. A comprehensive approach for network attack forecasting. *Computers and Security*, 58:83–105, 2016. ISSN 01674048. doi: 10.1016/j.cose.2015.11.005.
- [43] Fred Glover and Manuel Laguna. Tabu Search. In *Handbook of Combinatorial Optimization*, pages 2093–2229. Springer US, Boston, MA, 1998. doi: 10.1007/978-1-4613-0303-9_33.
- [44] Tom Gonda, Guy Shani, Rami Puzis, and Bracha Shapira. Ranking Vulnerability Fixes Using Planning Graph Analysis. 2017.

- [45] Peter Hart, Nils Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. ISSN 0536-1567. doi: 10.1109/TSSC.1968.300136.
- [46] Hannes Holm, Khurram Shahzad, Markus Buschle, and Mathias Ekstedt. P2CySeMoL: Predictive, Probabilistic Cyber Security Modeling Language. *IEEE Transactions on Dependable and Secure Computing*, 12(6):626–639, 2015.
- [47] John Homer, X Ou, and MA McQueen. From Attack Graphs to Automated Configuration Management-An Iterative Approach. Technical report, 2008.
- [48] John Homer, Ashok Varikuti, Xinming Ou, and Miles A. McQueen. Improving Attack Graph Visualization through Data Reduction and Attack Grouping. In *Visualization for Computer Security*, pages 68–79. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 3540859314. doi: 10.1007/978-3-540-85933-8_7.
- [49] John Homer, Xinming Ou, and David Schmidt. A sound and practical approach to quantifying security risk in enterprise networks. *Kansas State University Technical Report*, pages 1–15, 2009.
- [50] John Homer, Su Zhang, Xinming Ou, David Schmidt, Yanhui Du, S. Raj Rajagopalan, and Anoop Singhal. Aggregating vulnerability metrics in enterprise networks using attack graphs. *Journal of Computer Security*, 21(4):561–597, sep 2013. ISSN 18758924. doi: 10.3233/JCS-130475.
- [51] Jin B. Hong and Dong Seong Kim. Towards scalable security analysis using multi-layered security models. *Journal of Network and Computer Applications*, 75:156–168, 2016. ISSN 10958592. doi: 10.1016/j.jnca.2016.08.024.
- [52] Jin B. Hong, Dong Seong Kim, Chun-Jen Chung, and Dijiang Huang. A survey on the usability and practical applications of Graphical Security Models. *Computer Science Review*, 26:1–16, nov 2017. ISSN 15740137. doi: 10.1016/j.cosrev.2017.09.001.
- [53] Zhisheng Hu, Minghui Zhu, and Peng Liu. Online algorithms for adaptive cyber defense on Bayesian attack graphs. *MTD 2017 - Proceedings of the 2017 Workshop on Moving Target Defense, co-located with CCS 2017*, 2017-Janua:99–109, 2017. doi: 10.1145/3140549.3140556.
- [54] V.L. Huang, P.N. Suganthan, and J.J. Liang. Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems. *International Journal of Intelligent Systems*, 21(2):209–226, feb 2006. ISSN 0884-8173. doi: 10.1002/int.20128.
- [55] Martin Husák, Jana Komárková, Elias Bou-Harb, and Pavel Čeleda. Survey of attack projection, prediction, and forecasting in cyber security. *IEEE Communications Surveys and Tutorials*, 21(1):640–660, 2019. ISSN 1553877X. doi: 10.1109/COMST.2018.2871866.
- [56] Nwokedi Idika and Bharat Bhargava. Extending Attack Graph-Based Security Metrics and Aggregating Their Application. *IEEE Transactions on Dependable and Secure Computing*, 9(1):75–85, jan 2012. ISSN 1545-5971. doi: 10.1109/TDSC.2010.61.

- [57] IEC/ISO. Risk management - Risk assessment techniques, 2009. ISSN 0036-1682.
- [58] Kyle Ingols, Richard Lippmann, and Keith Piwowarski. Practical Attack Graph Generation for Network Defense. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 121–130. IEEE, dec 2006. ISBN 0-7695-2716-7. doi: 10.1109/ACSAC.2006.39.
- [59] Frank Innerhofer-Oberperfler and Ruth Breu. Using an enterprise architecture for IT risk management. In *Proceedings of the ISSA 2006 from Insight to Foresight Conference*, pages 1–12, 2006. ISBN 1-86854-636-5. doi: 10.1.1.108.6754.
- [60] Masaki Inokuchi, Yoshinobu Ohta, Shunichi Kinoshita, Tomohiko Yagyu, Orly Stan, Ron Bitton, Yuval Elovici, and Asaf Shabtai. Design procedure of knowledge base for practical attack graph generation. *AsiaCCS 2019 - Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 594–601, 2019. doi: 10.1145/3321705.3329853.
- [61] Investigation Committee on Information Leakage due to Unauthorized Access. Report on the Results of the Investigation into the Incident of Information Leakage due to Unauthorized Access. Technical report, Japan Pension Service, 2015.
- [62] ISACA. *The Business Model for Information Security*. ISACA, 2010. ISBN 978-1-60420-154-3.
- [63] ISO/IEC. Information technology - Security techniques - Information security management systems - Requirements, 2005.
- [64] ISO/IEC. Information technology - Security techniques - Code of practice for information security management, 2005.
- [65] ISO/IEC. Information technology - Security techniques - Information security management system implementation guidance, 2010.
- [66] ISO/IEC. Information technology - Security techniques - Information security risk management, 2011.
- [67] ISO/IEC. Information technology - Security techniques - Information security management systems - Requirements, 2013.
- [68] ISO/IEC. Information technology - Security techniques - Code of practice for information security controls, 2013.
- [69] ISO/IEC. Information technology - Security techniques - Information security management systems - Overview and vocabulary, 2016.
- [70] Sushil Jajodia and Steven Noel. Topological Vulnerability Analysis. In *Cyber situational awareness*, pages 139–154. Elsevier, 2010. ISBN 9780128136522. doi: 10.1007/978-1-4419-0140-8.7.
- [71] Sushil Jajodia, Steven Noel, and Brian O’Berry. Topological Analysis of Network Attack Vulnerability. In *Managing Cyber Threats*, pages 247–266. Springer-Verlag, New York, 2005. doi: 10.1007/0-387-24230-9.9.
- [72] Sushil Jajodia, Noseong Park, Edoardo Serra, and V. S. Subrahmanian. SHARE. *ACM Transactions on Internet Technology*, 18(3):1–41, aug 2018. ISSN 1533-5399. doi: 10.1145/3137571.

- [73] S. Jha, O. Sheyner, and J. Wing. Two Formal Analysis of Attack Graphs. In *Proceedings - Computer Security Foundations Workshop, 15th IEEE*, pages 49–63, 2002. ISBN 0-7695-1689-0. doi: 10.3724/SP.J.1001.2010.03584.
- [74] Pontus Johnson, Robert Lagerström, Per Närman, and Mårten Simonsson. Extended Influence Diagrams for Enterprise Architecture Analysis. In *2006 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06)*, pages 3–12, 2006. ISBN 0-7695-2558-X. doi: 10.1109/EDOC.2006.27.
- [75] Pontus Johnson, Johan Ullberg, Markus Buschle, Ulrik Franke, and Shahzad Khurram. P2AMF : Predictive , Probabilistic Architecture Modeling Framework. In *Enterprise Interoperability - Proceedings of the Fifth International IFIP Working Conference, IWEI 2013*, pages 104–117, 2013. ISBN 978-3-642-36795-3. doi: 10.1007/978-3-642-36796-0.
- [76] Pontus Johnson, Alexandre Vernotte, Mathias Ekstedt, and Robert Lagerström. pwnPr3d: An attack-graph-driven probabilistic threat-modeling approach. *Proceedings - 2016 11th International Conference on Availability, Reliability and Security, ARES 2016*, pages 278–283, 2016. doi: 10.1109/ARES.2016.77.
- [77] Pontus Johnson, Robert Lagerström, and Mathias Ekstedt. A meta language for threat modeling and attack simulations. *ACM International Conference Proceeding Series*, 2018. doi: 10.1145/3230833.3232799.
- [78] Kerem Kaynar. A taxonomy for attack graph generation and usage in network security. *Journal of Information Security and Applications*, 29:27–56, 2016. ISSN 22142126. doi: 10.1016/j.jisa.2016.02.001.
- [79] Kerem Kaynar and Fikret Sivrikaya. Distributed Attack Graph Generation. *IEEE Transactions on Dependable and Secure Computing*, 13(5):519–532, sep 2016. ISSN 1545-5971. doi: 10.1109/TDSC.2015.2423682.
- [80] MHR. Khouzani, Zhengliang Liu, and Pasquale Malacaria. Scalable min-max multi-objective cyber-security optimisation over probabilistic attack graphs. *European Journal of Operational Research*, 278(3):894–903, nov 2019. ISSN 03772217. doi: 10.1016/j.ejor.2019.04.035.
- [81] Barbara Kordy and Wojciech Wideł. How well can I secure my system? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10510 LNCS:332–347, 2017. ISSN 16113349. doi: 10.1007/978-3-319-66845-1_22.
- [82] Barbara Kordy and Wojciech Wideł. On quantitative analysis of attack defense trees with repeated labels. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10804 LNCS:325–346, 2018. ISSN 16113349. doi: 10.1007/978-3-319-89722-6_14.
- [83] Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Patrick Schweitzer. Attack-defense trees. *Journal of Logic and Computation*, 24(1):55–87, 2014. ISSN 0955792X. doi: 10.1093/logcom/exs029.

- [84] Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, 13-14: 1–38, nov 2014. ISSN 15740137. doi: 10.1016/j.cosrev.2014.07.001.
- [85] Harjinder Singh Lallie, Kurt Debattista, and Jay Bal. An Empirical Evaluation of the Effectiveness of Attack Graphs and Fault Trees in Cyber-Attack Perception. *IEEE Transactions on Information Forensics and Security*, 13(5):1110–1122, may 2018. ISSN 1556-6013. doi: 10.1109/TIFS.2017.2771238.
- [86] Harjinder Singh Lallie, Kurt Debattista, and Jay Bal. A review of attack graph and attack tree visual syntax in cyber security. *Computer Science Review*, 35:100219, feb 2020. ISSN 15740137. doi: 10.1016/j.cosrev.2019.100219.
- [87] Joshua Letchford and Yevgeniy Vorobeychik. Optimal interdiction of attack plans. In *12th International Conference on Autonomous Agents and Multiagent Systems 2013, AAMAS 2013*, volume 1, pages 199–206, 2013.
- [88] Xiao Yu Li. *Optimization algorithms for the minimum-cost satisfiability problem*. PhD thesis, North Carolina State University, Raleigh, North Carolina, 2004.
- [89] Richard Lippmann, Kyle Ingols, Chris Scott, Keith Piwowarski, Kendra Kratkiewicz, Mike Artz, and Robert Cunningham. Validating and Restoring Defense in Depth Using Attack Graphs. In *MILCOM 2006*, pages 1–10. IEEE, oct 2006. ISBN 1-4244-0617-X. doi: 10.1109/MILCOM.2006.302434.
- [90] Richard Paul Lippmann and Kyle William Ingols. An Annotated Review of Past Papers on Attack Graphs. Technical report, 2005.
- [91] Yu Liu and Hong Man. Network vulnerability assessment using Bayesian networks. In Belur V. Dasarathy, editor, *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2005*, volume 5812, page 61. International Society for Optics and Photonics, mar 2005. doi: 10.1117/12.604240.
- [92] Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund Clarke, and Jeannette Wing. Ranking Attack Graphs. In *International Workshop on Recent Advances in Intrusion Detection*, pages 127–144. 2006. ISBN 354039723X. doi: 10.1007/11856214_7.
- [93] Erik Miebling, Mohammad Rasouli, and Demosthenis Teneketzis. Optimal defense policies for partially observable spreading processes on Bayesian attack graphs. *MTD 2015 - Proceedings of the 2nd ACM Workshop on Moving Target Defense, co-located with: CCS 2015*, pages 67–76, 2015. doi: 10.1145/280475.28084828.
- [94] Erik Miebling, Mohammad Rasouli, and Demosthenis Teneketzis. A POMDP Approach to the Dynamic Defense of Large-Scale Cyber Networks. *IEEE Transactions on Information Forensics and Security*, 13(10):2490–2505, 2018. ISSN 15566013. doi: 10.1109/TIFS.2018.2819967.
- [95] Luis Muñoz-González, Daniele Sgandurra, Andrea Paudice, and Emil C. Lupu. Efficient Attack Graph Analysis through Approximate Inference. *ACM Transactions on Privacy and Security*, 20(3):1–30, aug 2017. ISSN 2471-2566. doi: 10.1145/3105760.

- [96] Luis Muñoz-González, Daniele Sgandurra, Martin Barrere, and Emil C. Lupu. Exact Inference Techniques for the Analysis of Bayesian Attack Graphs. *IEEE Transactions on Dependable and Secure Computing*, 16(2):231–244, mar 2019. ISSN 1545-5971. doi: 10.1109/TDSC.2016.2627033.
- [97] Kevin P Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.
- [98] Apurba K. Nandi, Hugh R. Medal, and Satish Vadlamani. Interdicting attack graphs to protect organizations from cyber attacks: A bi-level defender-attacker model. *Computers and Operations Research*, 75:118–131, 2016. ISSN 03050548. doi: 10.1016/j.cor.2016.05.005.
- [99] National center of Incident readiness and Strategy for Cybersecurity. Results of the investigation into the cause of the personal information leak at the Japan Pension Service. Technical report, Cybersecurity Strategic Headquarters, 2015.
- [100] National Institute of Standards and Technology. Performance measurement guide for information security. Technical Report July, National Institute of Standards and Technology, Gaithersburg, MD, 2008.
- [101] National Institute of Standards and Technology. Managing Information Security Risk: Organization, Mission, and Information System View. Technical Report March, National Institute of Standards and Technology, Gaithersburg, MD, 2011.
- [102] National Institute of Standards and Technology. Guide for conducting risk assessments, 2012. ISSN 01641212.
- [103] National Institute of Standards and Technology. Security and Privacy Controls for Federal Information Systems and Organizations. Technical Report September 2020, National Institute of Standards and Technology, Gaithersburg, MD, apr 2013.
- [104] National Institute of Standards and Technology. Assessing Security and Privacy Controls in Federal Information Systems and Organizations:. Technical Report December 2014, National Institute of Standards and Technology, Gaithersburg, MD, dec 2014.
- [105] National Institute of Standards and Technology. Risk management framework for information systems and organizations:. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, dec 2018.
- [106] Hoang Hai Nguyen and David M. Nicol. Estimating loss due to cyber-attack in the presence of uncertainty. *Proceedings - 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2020*, pages 361–369, 2020. doi: 10.1109/TrustCom50675.2020.00057.
- [107] Hoang Hai Nguyen, Kartik Palani, and David M. Nicol. An approach to incorporating uncertainty in network security analysis. *ACM International Conference Proceeding Series*, Part F1271:74–84, 2017. doi: 10.1145/3055305.3055308.

- [108] Thanh H. Nguyen, Mason Wright, Michael P. Wellman, and Satinder Singh. Multistage Attack Graph Security Games: Heuristic Strategies, with Empirical Game-Theoretic Analysis. *Security and Communication Networks*, 2018:1–28, dec 2018. ISSN 1939-0114. doi: 10.1155/2018/2864873.
- [109] S. Noel, E. Harley, K. H. Tam, M. Limiero, and M. Share. *CyGraph: Graph-Based Analytics and Visualization for Cybersecurity*, volume 35. Elsevier B.V., 1 edition, 2016. ISBN 9780444637444. doi: 10.1016/bs.host.2016.07.001.
- [110] Steven Noel and Sushil Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security - VizSEC/DMSEC '04*, page 109, New York, New York, USA, 2004. ACM Press. ISBN 1581139748. doi: 10.1145/1029208.1029225.
- [111] Steven Noel and Sushil Jajodia. Understanding Complex Network Attack Graphs through Clustered Adjacency Matrices. In *21st Annual Computer Security Applications Conference (ACSAC'05)*, pages 160–169. IEEE, 2005. ISBN 0-7695-2461-3. doi: 10.1109/CSAC.2005.58.
- [112] Steven Noel and Sushil Jajodia. Metrics suite for network attack graph analytics. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference on - CISR '14*, volume 10, pages 5–8, New York, New York, USA, 2014. ACM Press. ISBN 9781450328128. doi: 10.1145/2602087.2602117.
- [113] Steven Noel, Sushil Jajodia, Brian O’Berry, and Michael Jacobs. Efficient minimum-cost network hardening via exploit dependency graphs. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, pages 86–95. IEEE, 2003. ISBN 0-7695-2041-3. doi: 10.1109/CSAC.2003.1254313.
- [114] Steven Noel, Eric Robertson, and Sushil Jajodia. Correlating Intrusion Events and Building Attack Scenarios Through Attack Graph Distances. In *20th Annual Computer Security Applications Conference*, pages 350–359. IEEE, 2004. ISBN 0-7695-2252-1. doi: 10.1109/CSAC.2004.11.
- [115] Steven Noel, Sushil Jajodia, Lingyu Wang, and Anoop Singhal. Measuring Security Risk of Networks Using Attack Graphs. *International Journal of Next Generation Computing*, 1(1): 135–147, 2010. ISSN 0976-5034.
- [116] Xinming Ou. *A logic-programming approach to network security analysis*. PhD thesis, Princeton University, USA, 2005.
- [117] Xinming Ou and Anoop Singhal. The Need for Quantifying Security. In *Kansas State University Technical Report*, pages 1–3. 2012. doi: 10.1007/978-1-4614-1860-3_1.
- [118] Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. MulVAL: a logic-based network security analyzer. In *Proceedings of the 14th conference on USENIX Security Symposium*, volume 14, 2005.
- [119] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications*

- security - CCS '06*, pages 336–345, New York, New York, USA, 2006. ACM Press. ISBN 1595935185. doi: 10.1145/1180405.1180446.
- [120] Joseph Pamula, Sushil Jajodia, Paul Ammann, and Vipin Swarup. A weakest-adversary security metric for network configuration security analysis. In *Proceedings of the 2nd ACM workshop on Quality of protection - QoP '06*, page 31, New York, New York, USA, 2006. ACM Press. ISBN 1595935533. doi: 10.1145/1179494.1179502.
- [121] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988. ISBN 9780080514895.
- [122] Marcus Pendleton, Richard Garcia-Lebron, Jin-Hee Cho, and Shouhuai Xu. A Survey on Systems Security Metrics. *ACM Computing Surveys*, 49(4):1–35, feb 2017. ISSN 0360-0300. doi: 10.1145/3005714.
- [123] Peng Xie, Jason H Li, Xinming Ou, Peng Liu, and Renato Levy. Using Bayesian networks for cyber security analysis. In *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, pages 211–220. IEEE, jun 2010. ISBN 978-1-4244-7500-1. doi: 10.1109/DSN.2010.5544924.
- [124] Cynthia Phillips and Laura Painton Swiler. A Graph-based System for Network-vulnerability Analysis. In *Proceedings of the 1998 Workshop on New Security Paradigms*, pages 71–79, 1998. ISBN 1-58113-168-2. doi: 10.1145/310889.310919.
- [125] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic Security Risk Management Using Bayesian Attack Graphs. *IEEE Transactions on Dependable and Secure Computing*, 9(1):61–74, jan 2012. ISSN 1545-5971. doi: 10.1109/TDSC.2011.34.
- [126] Mohammad Ashiqur Rahman and Ehab Al-Shaer. A formal approach for network security management based on qualitative risk analysis. *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management, IM 2013*, pages 244–251, 2013.
- [127] Mohammad Ashiqur Rahman and Ehab Al-Shaer. Automated Synthesis of Distributed Network Access Controls: A Formal Framework with Refinement. *IEEE Transactions on Parallel and Distributed Systems*, 28(2):416–430, 2017. ISSN 10459219. doi: 10.1109/TPDS.2016.2585108.
- [128] Alex Ramos, Marcella Lazar, Raimir Holanda Filho, and Joel J. P. C. Rodrigues. Model-Based Quantitative Network Security Metrics: A Survey. *IEEE Communications Surveys & Tutorials*, 19(4):2704–2734, 2017. ISSN 1553-877X. doi: 10.1109/COMST.2017.2745505.
- [129] R.W. Ritchey and Paul Ammann. Using model checking to analyze network vulnerabilities. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pages 156–165. IEEE Comput. Soc, 2000. ISBN 0-7695-0665-8. doi: 10.1109/SECPRI.2000.848453.
- [130] Sebastian Roschke, Feng Cheng, Robert Schuppenies, and Christoph Meinel. Towards Unifying Vulnerability Information for Attack Graph Construction. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5735 LNCS, pages 218–233. 2009. ISBN 3642044735. doi: 10.1007/978-3-642-04474-8_18.

- [131] Sebastian Roschke, Feng Cheng, and Christoph Meinel. A New Alert Correlation Algorithm Based on Attack Graph. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6694 LNCS, pages 58–67. 2011. ISBN 9783642213229. doi: 10.1007/978-3-642-21323-6_8.
- [132] Arpan Roy, Dong Seong Kim, and Kishor S. Trivedi. Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees. *Proceedings of the International Conference on Dependable Systems and Networks*, 2012. doi: 10.1109/DSN.2012.6263940.
- [133] Arpan Roy, Dong Seong Kim, and Kishor S. Trivedi. Attack countermeasure trees (ACT): towards unifying the constructs of attack and defense trees. *Security and Communication Networks*, 5(8):929–943, aug 2012. ISSN 19390114. doi: 10.1002/sec.299.
- [134] Diptikalyan Saha. Extending logical attack graphs for efficient vulnerability analysis. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 63–73, 2008. ISSN 15437221. doi: 10.1145/1455770.1455780.
- [135] Shuichi Sakai, Hirokazu Okumura, Masatomo Suzuki, Hidetsugu Irie, and Kengo Zenitani. Proceedings of USIS Symposium. Graduate School of Information Science and Technology, The University of Tokyo, 2018.
- [136] Carlos Sarraute, Olivier Buffet, and Jörg Hoffmann. POMDPs make better hackers: Accounting for uncertainty in penetration testing. *Proceedings of the National Conference on Artificial Intelligence*, 3:1816–1824, 2012.
- [137] Reginald E. Sawilla and Xinming Ou. Identifying Critical Attack Assets in Dependency Attack Graphs. In *European Symposium on Research in Computer Security*, pages 18–34. Springer, 2008. doi: 10.1007/978-3-540-88313-5_2.
- [138] Bruce Schneier. Attack Trees. *Secrets and Lies: Digital Security in a Networked World*, pages 318–333, 2015.
- [139] Edoardo Serra, Sushil Jajodia, Andrea Pugliese, Antonino Rullo, and V. S. Subrahmanian. Pareto-Optimal Adversarial Defense of Enterprise Systems. *ACM Transactions on Information and System Security*, 17(3):1–39, mar 2015. ISSN 1094-9224. doi: 10.1145/2699907.
- [140] Nicholas A Sherwood. *Enterprise security architecture: a business-driven approach*. CRC Press, 2005.
- [141] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, 23:2164–2172, 2010. doi: 10.5555/2997046.2997137.
- [142] Anoop Singhal and Xinming Ou. Security risk analysis of enterprise networks using probabilistic attack graphs, 2011.
- [143] Teodor Sommestad and Fredrik Sandström. An empirical test of the accuracy of an attack graph analysis tool. *Information and Computer Security*, 23(5):516–531, 2015. ISSN 20564961. doi: 10.1108/ICS-06-2014-0036.

- [144] Teodor Sommestad, Mathias Ekstedt, and Pontus Johnson. Combining defense graphs and enterprise architecture models for security analysis. In *Proceedings - 12th IEEE International Enterprise Distributed Object Computing Conference (EDOC'08)*, pages 349–355, 2008. ISBN 9780769533735. doi: 10.1109/EDOC.2008.37.
- [145] Teodor Sommestad, Mathias Ekstedt, and Pontus Johnson. Cyber security risks assessment with bayesian defense graphs and architectural models. In *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences (HICSS)*, pages 1–10, 2009. ISBN 9780769534503. doi: 10.1109/HICSS.2009.141.
- [146] Patrick Speicher, Marcel Steinmetz, Michael Backes, Jörg Hoffmann, and Robert Künnemann. Stackelberg planning: Towards effective leader-follower state space search. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 6286–6293, 2018.
- [147] Patrick Speicher, Marcel Steinmetz, Robert Kunnemann, Milivoj Simeonovski, Giancarlo Pellegrino, Jorg Hoffmann, and Michael Backes. Formally Reasoning about the Cost and Efficacy of Securing the Email Infrastructure. *Proceedings - 3rd IEEE European Symposium on Security and Privacy, EURO S and P 2018*, pages 77–91, 2018. doi: 10.1109/EuroSP.2018.00014.
- [148] Patrick Speicher, Marcel Steinmetz, Jörg Hoffmann, Michael Backes, and Robert Künnemann. Towards automated network mitigation analysis. *Proceedings of the ACM Symposium on Applied Computing*, Part F1477:1971–1978, 2019. doi: 10.1145/3297280.3297473.
- [149] Orly Stan, Ron Bitton, Michal Ezrets, Moran Dadon, Masaki Inokuchi, Ohta Yoshinobu, Yagyu Tomohiko, Yuval Elovici, and Asaf Shabtai. Extending Attack Graphs to Represent Cyber-Attacks in Communication Protocols and Modern IT Networks. *IEEE Transactions on Dependable and Secure Computing*, 5971(c):1–18, 2020. ISSN 19410018. doi: 10.1109/TDSC.2020.3041999.
- [150] Orly Stan, Ron Bitton, Michal Ezrets, Moran Dadon, Masaki Inokuchi, Yoshinobu Ohta, Tomohiko Yagyu, Yuval Elovici, and Asaf Shabtai. Heuristic Approach for Countermeasure Selection Using Attack Graphs. pages 1–16, 2021. doi: 10.1109/csf51468.2021.00003.
- [151] Xiaoyan Sun, Jun Dai, Anoop Singhal, and Peng Liu. Inferring the stealthy bridges between enterprise network islands in cloud using cross-layer bayesian networks. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 152:3–23, 2015. ISSN 18678211. doi: 10.1007/978-3-319-23829-6_1.
- [152] Xiaoyan Sun, Jun Dai, Peng Liu, Anoop Singhal, and John Yen. Towards probabilistic identification of zero-day attack paths. In *2016 IEEE Conference on Communications and Network Security (CNS)*, pages 64–72. IEEE, oct 2016. ISBN 978-1-5090-3065-1. doi: 10.1109/CNS.2016.7860471.
- [153] Xiaoyan Sun, Anoop Singhal, and Peng Liu. Towards Actionable Mission Impact Assessment in the Context of Cloud Computing. In Giovanni Livraga and Sencun Zhu, editors, *Data and Applications Security and Privacy XXXI*, pages 259–274, Cham, 2017. Springer International Publishing. ISBN 978-3-319-61176-1.

- [154] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-attack graph generation tool. *Proceedings - DARPA Information Survivability Conference and Exposition II, DISCEX 2001*, 2:307–321, 2001. doi: 10.1109/DISCEX.2001.932182.
- [155] André Teixeira, Kin Cheong Sou, Henrik Sandberg, and Karl H. Johansson. Secure Control Systems: A Quantitative Risk Management Approach. *IEEE Control Systems*, 35(1):24–45, feb 2015. ISSN 1066-033X. doi: 10.1109/MCS.2014.2364709.
- [156] The Open Group. TOGAF® and SABSA® Integration, 2011.
- [157] The Open Group. *The TOGAF® Standard, Version 9.2*. Van Haren Publishing, 2018. ISBN 1-947754-11-9.
- [158] The Open Group. *ArchiMate® 3.1 Specification*. Van Haren Publishing, 2019.
- [159] John Thorp. *The information paradox: realizing the business benefits of information technology*. McGraw-Hill Ryerson, 2003.
- [160] US Department of Defense. The DoDAF Architecture Framework Version 2.02, 2010.
- [161] Vilhelm Verendel. Quantified security is a weak hypothesis. In *Proceedings of the 2009 workshop on New security paradigms workshop - NSPW '09*, page 37, New York, New York, USA, 2009. ACM Press. ISBN 9781605588452. doi: 10.1145/1719030.1719036.
- [162] Verification Committee for the Incident of Information Leakage due to Unauthorized Access at Japan Pension Service. Verification Report. Technical report, Ministry of Health, Labour and Welfare, 2015.
- [163] Valentina Viduto, Carsten Maple, Wei Huang, and David López-Peréz. A novel risk assessment and optimisation model for a multi-objective network security countermeasure selection problem. *Decision Support Systems*, 53(3):599–610, 2012.
- [164] S Wahe and G Petersen. *Open Enterprise Security Architecture (O-ESA): A Framework and Template for Policy-driven Security*. Van Haren Publishing, 2011. ISBN 1931624941.
- [165] Lingyu Wang, Anyi Liu, and Sushil Jajodia. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer Communications*, 29(15):2917–2933, sep 2006. ISSN 01403664. doi: 10.1016/j.comcom.2006.04.001.
- [166] Lingyu Wang, Steven Noel, and Sushil Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18):3812–3824, nov 2006. ISSN 01403664. doi: 10.1016/j.comcom.2006.06.018.
- [167] Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring the Overall Security of Network Configurations Using Attack Graphs. In *Proceedings - IFIP Annual Conference on Data and Applications Security and Privacy*, pages 98–112. 2007. doi: 10.1007/978-3-540-73538-0_9.
- [168] Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Toward measuring network security using attack graphs. In *Proceedings of the 2007 ACM workshop on Quality of protection - QoP '07*, page 49, New York, New York, USA, 2007. ACM Press. ISBN 9781595938855. doi: 10.1145/1314257.1314273.

- [169] Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, and Sushil Jajodia. An Attack Graph-Based Probabilistic Security Metric. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 283–296. 2008. ISBN 978-3-540-70567-3. doi: 10.1007/978-3-540-70567-3_22.
- [170] Lingyu Wang, Sushil Jajodia, Anoop Singhal, and Steven Noel. k-Zero Day Safety: Measuring the Security Risk of Networks against Unknown Attacks. In *European Symposium on Research in Computer Security*, pages 573–587. 2010. doi: 10.1007/978-3-642-15497-3_35.
- [171] Lingyu Wang, Sushil Jajodia, Anoop Singhal, Pengsu Cheng, and Steven Noel. k-Zero Day Safety: A Network Security Metric for Measuring the Risk of Unknown Vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, 11(1):30–44, jan 2014. ISSN 1545-5971. doi: 10.1109/TDSC.2013.24.
- [172] Lingyu Wang, Mengyuan Zhang, Sushil Jajodia, Anoop Singhal, and Massimiliano Albanese. Modeling Network Diversity for Evaluating the Robustness of Networks against Zero-Day Attacks. In *Network Security Metrics*, pages 494–511. 2014. doi: 10.1007/978-3-319-11212-1_28.
- [173] Lingyu Wang, Sushil Jajodia, and Anoop Singhal. *Network Security Metrics*. Springer International Publishing, Cham, 2017. ISBN 978-3-319-66504-7. doi: 10.1007/978-3-319-66505-4.
- [174] Shuzhen Wang, Zonghua Zhang, and Youki Kadobayashi. Exploring attack graph for cost-benefit security hardening: A probabilistic approach. *Computers and Security*, 32(61100156): 158–169, 2013. ISSN 01674048. doi: 10.1016/j.cose.2012.09.013.
- [175] Wei Li and R.B. Vaughn. Cluster Security Research Involving the Modeling of Network Exploitations Using Exploitation Graphs. In *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID’06)*, volume 2, pages 26–26. IEEE, 2006. ISBN 0-7695-2585-7. doi: 10.1109/CCGRID.2006.1630921.
- [176] Lyndon While, Phil Hingston, Luigi Barone, and S. Huband. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, feb 2006. ISSN 1089-778X. doi: 10.1109/TEVC.2005.851275.
- [177] Mehdi Yousefi, Nhamo Mtetwa, Yan Zhang, and Huaglory Tianfield. A novel approach for analysis of attack graph. *2017 IEEE International Conference on Intelligence and Security Informatics: Security and Big Data, ISI 2017*, pages 7–12, 2017. doi: 10.1109/ISI.2017.8004866.
- [178] Mehdi Yousefi, Nhamo Mtetwa, Yan Zhang, and Huaglory Tianfield. A Reinforcement Learning Approach for Attack Graph Analysis. *Proceedings - 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018*, pages 212–217, 2018. doi: 10.1109/TrustCom/BigDataSE.2018.00041.
- [179] Emmanuele Zambon, Damiano Bolzoni, Sandro Etalle, and Marco Salvato. A Model Supporting Business Continuity Auditing and Planning in Information Systems. In *Second International Conference on Internet Monitoring and Protection (ICIMP 2007)*, pages 33–33, 2007. ISBN 0-7695-2911-9. doi: 10.1109/ICIMP.2007.4.

- [180] Emmanuele Zambon, Damiano Bolzoni, Sandro Etalle, and Marco Salvato. Model-based mitigation of availability risks. In *Second IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM'07)*, pages 75–83, 2007. ISBN 1424412951. doi: 10.1109/BDIM.2007.375014.
- [181] Jianping Zeng, Shuang Wu, Yanyu Chen, Rui Zeng, and Chengrong Wu. Survey of Attack Graph Analysis Methods from the Perspective of Data and Knowledge Processing. *Security and Communication Networks*, 2019, 2019. ISSN 19390122. doi: 10.1155/2019/2031063.
- [182] Kengo Zenitani. A multi-objective cost-benefit optimization algorithm for network hardening. *International Journal of Information Security*, 21(4):813–832, aug 2022. ISSN 1615-5262. doi: 10.1007/s10207-022-00586-7.
- [183] Kengo Zenitani. From attack graph analysis to attack function analysis. *Information Sciences*, (In review), 2022.
- [184] Kengo Zenitani. Towards a disciplined alignment in information security management. *Organizational Cybersecurity Journal: Practice, Process and People*, (In review), 2022.
- [185] Kengo Zenitani. A formal approach for secured risk analysis in information security management. In *2022 9th International Conference on Behavioural and Social Computing (BESC)*, pages 1–7, Matsuyama, Japan, oct 2022. IEEE. ISBN 979-8-3503-9814-4. doi: 10.1109/BESC57393.2022.9995464.
- [186] Kengo Zenitani. Attack graph analysis: An explanatory guide. *Computers and Security*, 126: 103081, 2023. ISSN 0167-4048. doi: 10.1016/j.cose.2022.103081.
- [187] Kengo Zenitani. A scalable algorithm for network reachability analysis with cyclic attack graphs. *Journal of Computer Security*, 31(1):29–55, jan 2023. ISSN 18758924. doi: 10.3233/JCS-210103.
- [188] Kengo Zenitani, Hirokazu Okumura, Hisafumi Mitsushio, Satoru Myojin, Hidetsugu Irie, and Shuichi Sakai. Security Hazard Map by Qualitative Sensitivity Analysis. In *Proceedings of the 10th International Conference on Project Management (ProMAC2016)*, pages 970–977. The Society of Project Management, 2016.
- [189] Mengyuan Zhang, Lingyu Wang, Sushil Jajodia, Anoop Singhal, and Massimiliano Albanese. Network Diversity: A Security Metric for Evaluating the Resilience of Networks Against Zero-Day Attacks. *IEEE Transactions on Information Forensics and Security*, 11(5):1071–1086, may 2016. ISSN 1556-6013. doi: 10.1109/TIFS.2016.2516916.