

Attack Graph-based Approaches for Network Security Hardening

A thesis submitted to University of Hyderabad in partial fulfillment

for the degree of

Doctor of Philosophy

by

Ghanshyam S. Bopche

Reg.No. 11IDPH03



**SCHOOL OF COMPUTER & INFORMATION SCIENCES
UNIVERSITY OF HYDERABAD
HYDERABAD -500046**

Telangana

India

January, 2017



CERTIFICATE

This is to certify that the thesis entitled “**Attack Graph-based Approaches for Network Security Hardening**” submitted by **Ghanshyam S. Bopche** bearing registration number **11IDPH03** in partial fulfillment of the requirements for award of Doctor of Philosophy in the School of Computer & Information Sciences is a bonafide work carried out by him under my supervision and guidance at IDRBT, Hyderabad.

This thesis is free from plagiarism and has not been submitted previously in part or in full to this or any other University or Institution for award of any degree or diploma.

Parts of this thesis have been published in following publications:

1. Attack graph generation, visualization and analysis: Issues and challenges (Chapter 2).
2. Exploiting domination in attack graph for enterprise network hardening (Chapter 3).
3. Exploiting curse of diversity for improved network security (Chapter 4).

Further, the student has passed the following courses towards fulfillment of course-work requirement for Ph.D:

	Course Code	Name	Credits	Pass/Fail
1	CS801	Data Structure and Algorithms	4	Pass
2	CS802	Operating System and Programming	4	Pass
3	IT811	Secure Computing	4	Pass
4	CS810	Advanced Networking	4	Pass

Supervisor	Director	Dean
Prof. Babu M. Mehtre	Dr. A. S. Ramasastry	Prof. Arun Agrawal
IDRBT, Hyderabad-500 057	IDRBT, Hyderabad-500 057	SCIS, UOH
India	India	Hyderabad -500 046, India

DECLARATION

I, **Ghanshyam S. Bopche**, hereby declare that this thesis entitled “**Attack Graph-based Approaches for Network Security Hardening**” submitted by me under the guidance and supervision of **Prof. Babu M. Mehtre**, is a bonafide research work and is free from plagiarism. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma. I hereby agree that my thesis can be deposited in Shodganga/INFLIBNET.

A report on plagiarism statistics from the University Librarian is enclosed.

Date:

Signature of the Student

(Ghanshyam S. Bopche)

Reg. No.: 11IDPH03

//Countersigned//

Signature of the Supervisor:

Dedicated To My Family and Friends

Acknowledgements

I want to say my first “thank you” to my advisor **Prof. Babu M. Mehtre** for his guidance, wisdom, and support throughout my five years at IDRBT. He introduced me to the field of information security, and most importantly, helped me in identifying the important problems of network vulnerability analysis. I am appreciative of the time and effort he has devoted to advising me over the past years, and his consistent encouragement and positive reinforcement have made it a very rewarding experience. I would also like to thank him for his encouragement and support that helped me build confidence when I feel stuck in my research. Thank you, Sir!

I especially acknowledge the **University of Hyderabad** (UoH) for its academic support. I also extend my acknowledgment to **Institute for Development and Research in Banking Technology** (IDRBT), a R&D center established by RBI, for its kind financial support and beautiful environment for conducting the research.

It is my privilege to thank **Dr. A.S. Ramasastry**, Director, IDRBT for reviewing the work and timely suggestions throughout my research. I also thank **Mr. B. Sambamurthy**, Former Director, IDRBT for extending his cooperation at the preliminary stage of my research. It is an honor for me to thank **Prof. Arun Agrawal**, Dean, School of Computer and Information Sciences (SCIS), UoH, Hyderabad for his academic support throughout research work. Thank you so much!

I would like to thank my doctoral review committee members, **Prof. V.N. Sastry** and **Prof. Atul Negi** for providing invaluable inputs and suggestions ever since I started working on my Ph.D. Their encouragement and insightful comments are truly valuable to my research. Thank you so much!

Research grows along with inspiring discussions and sharing perspectives. I would like to thank **Prof. B. L. Deekshatulu** for the many inspiring conversations we have had, and the valuable suggestions he has given. His visions in research set a model for me as to what is meaningful computer science research. Thank you, Sir!

I also highly appreciate the valuable discussion with **Prof. Sushil Jajodia** from George Mason University and **Dr. Anoop Singhal** from National Institute of Standards and Technology (NIST), USA for insightful feedback on this research at the ICISS 14 conference. Thank you so much!

I want to thank **Prof. Venu Govindaraju** for accepting me as a visiting researcher at SUNY, Buffalo in summer 2015. But, especially, I want to thank **Prof. Shambhu Upadhyaya** and **Dr. Ifeoma Nwogu** for discussion about the attack graphs. Thank you so much!

I also want to thank many more people I was lucky to find on my way: Ilaiah, Srinivas, Pradeep, Viranna, Srinu, Gopal, Shravan, Vinaya, Chandrashekhar, Anup, Deep, Vinay, Kumar Ravi, Manu, Sandhya, Sriramulu, Sitara, among others. Their friendship filled gaps of a lonely period and made me feel “at home” in the Hyderabad. Thank you all!

Throughout my doctoral study, a lot of research associates of IDRBT accompanied me. I would like to express my thanks to Pankaj, Sandeep, Naveen, Abhishek, Jayavardhan, Shashi, Sriya, Srashti, Suresh, Mohan, Shravani, Soujanya, Ganesh, and Kranti.

Finally, my deepest gratitude goes to my family for their unflagging love and support throughout my life. I am indebted to my mother and passed away father for encouraging me to follow my curiosity in science and engineering, as well as teaching me to love, appreciate and give. I would like to thank my mother **Smt. Pushpa Bopche** especially for her best support and encouragement all times in pursuing my dreams. I would also like to thank my granny (Baran Bai), elder sister (Sarika), brother-in-law (Suresh), elder brothers (Yadorao, Sanjay, Birju), sister-in-laws (Sonali, Vaishali), younger brother (Gauri, Dadu, Sonu), for their uncountable love. Thank you all!

Abstract

With the widespread use of computer networks, especially in mission-critical infrastructures, network security has become a primary concern. We would intuitively expect that such networks are well-monitored and protected. However, it is hard because of the constant discovery of new vulnerabilities, increase in the complexity and size of networks, frequent changes in the network or security configurations, and the emergence of multistage attacks. Attack graphs have been proposed in the past to uncover potential multistage, multi-host attacks in a given network. As risk assessment is an essential factor in human decision making, attack graph-based metrics are useful to security managers for allocating scarce security resources efficiently and thereby achieving best possible network hardening. For a given network, obtaining hardening recommendations from the generated attack graphs remains an open issue.

Major contributions to network security hardening proposed in this thesis are as follows:

- We have proposed an integrated comprehensive measure called Improved Relative Cumulative Risk (IRCR) for quantifying the security risk of each exploitable vulnerability in a given network. IRCR considers the necessary network risk conditions that affect the success of an adversary and hence reflect the influential level of vulnerability instances more precisely. Moreover, IRCR accurately distinguishes different instances of the same vulnerability. Based on the IRCR recommendations, an administrator can accurately determine top vulnerabilities and prioritize the vulnerability remediation activities accordingly.

- Past studies have demonstrated that through adequate diversification of the vulnerable software/services, an administrator can increase the network robustness against the zero-day attacks. We have proposed an intra-path diversity metric and two inter-path diversity metrics called *uniqueness* and *overlap* to assess the diversification level of each attack path in a resource graph. Applying such metrics to the existing network security practices produce actionable knowledge that can be utilized for increasing the system robustness against zero-day attacks. Further, we have proposed an algorithm for identification and diversification of the repeated services along the attack paths. Experimental results show that such additional metrics would undoubtedly benefit security administrators in increasing the network robustness against zero-day attacks.
- Assessing change in the attack surface of dynamic computer networks is a formidable challenge. Despite the proposal of many attack graph-based security metrics, there has been no work on assessing the temporal variation in the attack surface of dynamic networks. Therefore, we have proposed to use classical graph distance metrics based on the Maximum Common Subgraph (MCS), and Graph Edit Distance (GED) to quantify the distance between a pair of successive attack graphs generated for a dynamic network. Experimental results show that the MCS and GED-based metrics can successfully capture the temporal variations in the network attack surface and alert security administrator about the critical network/security events.
- Finally, we have proposed a Change Distribution Matrix (CDM) based technique to identify the newly introduced changes in the network attack surface. To cater to this problem, we have reduced the problem of change detection in the network attack surface to the problem of a Error-Correcting Graph Matching (ECGM). Using CDM, security administrator's can identify the newly introduced exploits and respective enabling conditions and determine portions of the attack graph that has significantly changed.

Contents

Acknowledgments	v
Abstract	vii
List of Figures	xiv
List of Tables	xviii
Acronyms	xx
Glossary	xxii
1 Introduction	1
1.1 Motivation	4
1.1.1 Vulnerability Risk Estimation	5
1.1.2 Network Hardening and the Threats of Zero-day Attacks . . .	7
1.1.3 Assessing Temporal Variations in the Network Attack Surface	8
1.1.4 Depicting Temporal Variations in the Network Attack Surface	8
1.2 Thesis Outline	9
1.3 Contributions	9
2 Review of Related Work	13
2.1 Vulnerabilities and Exploits	13
2.2 Understanding Network Attacks	16
2.2.1 Computer Networks	16
2.2.2 Network Attacks	23
2.2.2.1 Single-step Attacks	23

2.2.2.2	Multistage, Multi-host Attacks	23
2.2.3	Anatomy of a Network Attack	24
2.2.4	Protecting Against Network Attacks	26
2.3	Vulnerability Scanning	28
2.4	Common Vulnerability Scoring Systems	29
2.5	Penetration Testing	31
2.6	Attack Trees	34
2.7	Attack Graphs	38
2.7.1	Workflow of Attack Graph Construction and Analysis	39
2.7.2	Advantages of Attack Graphs	40
2.7.3	Limitations of Attack Graphs	41
2.8	Attack Graph Representations	41
2.8.1	Condition-oriented Attack Graph	42
2.8.2	Exploit-oriented Attack Graph	42
2.8.3	Condition-exploit-oriented Attack Graph	43
2.8.4	Multiple Prerequisites Attack Graph	43
2.8.5	Logical Attack Graph	43
2.8.6	Hybrid-oriented Attack Graph	44
2.8.7	Resource Graph	44
2.9	Network Security Hardening using Attack Graphs	44
2.9.1	Vulnerability Risk Assessment	45
2.9.2	Network Security Risk Assessment	47
2.9.3	Network Attack Surface Change Assessment	49
2.9.3.1	Attack Surface	50
2.9.3.2	Network Attack Surface	50
2.9.4	Discerning Temporal Variations in the Network Attack Surface	51
2.10	Attack Graphs used in this Thesis	52
2.11	Summary	52
3	A Proximity-based Approach for Quantifying the Security Risk of Vulnerabilities	56
3.1	Introduction	57
3.2	Motivation	57

3.3	Risk Conditions Pertaining to the Network Security	59
3.3.1	Attack Path Resistance	59
3.3.2	Exploit Diversity along the Attack Path	61
3.3.3	Risk of the Neighboring Vulnerabilities	62
3.3.4	The Conjunctive (AND)/Disjunctive (OR) Dependency Be- tween Neighboring Vulnerabilities	63
3.4	Measuring Vulnerability Risk	63
3.4.1	Measuring Diversity-adjusted Vulnerability Score (DVS) . . .	63
3.4.2	Measuring Neighborhood Proximity-adjusted Vulnerability Score (NPVS)	65
3.4.3	Measuring Improved Relative Cumulative Risk (IRCR)	67
3.4.4	Summary of the Parameters used in Proximity-based Risk As- sessment	68
3.5	Experimental Setup and Results	68
3.6	Computational Complexity	86
3.7	Summary	86
4	Diversification of Software/Services for Increasing the Network Robust- ness Against Zero-day Attacks	90
4.1	Introduction	91
4.2	Motivation	92
4.3	Running Example	93
4.4	Proposed Metrics	96
4.4.1	Measuring Intra-path Diversity	97
4.4.2	Measuring Inter-path Diversity	97
4.5	Results and Analysis	99
4.5.1	Use of Intra-path Diversity Metric (d) for Resource Diversifi- cation	99
4.5.2	Use of Inter-path Diversity Metrics (i.e. u and o) for Resource Diversification	105
4.6	Summary	110

5	Assessing Temporal Variations in the Network Attack Surface	112
5.1	Introduction	113
5.2	Motivation	113
5.3	Motivating Example	115
5.4	Formal Model for the Network Attack Surface (NAS)	119
5.4.1	Notion of the Network Attack Surface (NAS)	120
5.4.2	Model to Capture the NAS	121
5.4.3	Dealing with Variations in the NAS	121
5.5	Metrics for Assessing Variation in the NAS	122
5.5.1	Maximum Common Subgraph-based Distance Metric	124
5.5.2	Graph Edit Distance-based Metric	125
5.6	Experimental Setup	127
5.6.1	Description of the Network Models	127
5.6.2	Generation of the Attack Graphs	128
5.6.3	Selection of the Sampling Interval Δt	130
5.6.4	Assessing Variations in the NAS	131
5.7	Results and Analysis	131
5.7.1	The Results for the Previously Proposed Attack Graph-based Metrics	132
5.7.2	Change Assessment using MCS and GED-based Graph Distance Metrics	137
5.7.3	Assessing the Effect of Non-linear Variation in the Vulnerability Density on the Network Attack Surface	140
5.7.4	Assessing the Effect of Vulnerability Variation in the Host(s) Directly Accessible to an Adversary or Belonging to the Protection Domain.	143
5.7.5	Summary of the Results	147
5.8	Computational Complexity	151
5.9	Summary	156

6	Depicting Temporal Variations in the Network Attack Surface	162
6.1	Introduction	162
6.2	Motivation	163
6.3	Running Example	164
6.4	Formal Model for the Network Attack Surface (NAS)	169
6.4.1	Model to Capture the Network Attack Surface	169
6.4.2	Dealing with Variations in the NAS	170
6.5	Assessing Change in the Network Attack Surface	171
6.5.1	Error-correcting Graph Matching (ECGM)	171
6.5.2	Pinpointing the Root Causes	175
6.5.3	Observations	179
6.6	Experiments	182
6.7	Summary	185
7	Conclusions and Future Work	186
7.1	Summary of Contributions	187
7.2	Future Directions	188
	References	190
	List of Publications	210
A	Annexure	212

List of Figures

1.1	An example attack path	3
2.1	An example attack graph (adapted from [1])	24
2.2	CVSS v3.0 metric group (adapted from [2])	30
2.3	Basic penetration test cycle (adapted from [3], [4])	33
2.4	An example attack tree (adapted from [5])	35
2.5	An example fault tree, on the left, and its logically equivalent, on the right (adapted from [6])	36
2.6	An attack graph that contains numerous attack trees (adapted from [7]).	38
2.7	Generic work flow of attack graph construction and analysis (adapted from [8])	40
2.8	History of vulnerability evaluation methods	46
2.9	History of attack graph-based risk assessment methods	48
3.1	Network risk conditions.	60
3.2	An example network	70
3.3	An exploit-oriented (exploit-dependency) attack graph for the network configuration 1.	73
3.4	Vulnerability instances, and their respective values for IRCR and Common Vulnerability Scoring System (CVSS).	74
3.5	Vulnerability priority comparison: IRCR vs CVSS.	75
3.6	Vulnerability instances, and their respective values for IRCR and Cumulative Probability (P)	78
3.7	Vulnerability priority comparison: IRCR vs P	79

LIST OF FIGURES

3.8	Vulnerability instances, and their respective values for IRCR and Cumulative Resistance (R)	79
3.9	Vulnerability priority comparison: IRCR vs R.	80
3.10	An exploit-dependency attack graph for the network configuration 2.	82
3.11	An exploit-dependency attack graph for the network configuration 3.	84
4.1	A test network (adapted from [9])	94
4.2	A resource graph for the test network (adapted from [9])	95
4.3	A flow chart for attack path diversification	101
4.4	Uniqueness (u) and overlap (o) score of each attack path in a resource graph pre network diversification.	109
4.5	Uniqueness (u) and overlap (o) score of each attack path in a resource graph post network diversification.	110
4.6	Rate of growth of uniqueness (u) measure	110
4.7	Rate of growth of overlap (o) measure	111
5.1	Dynamic computer network.	114
5.2	An example network	116
5.3	Differential visualization of attack graphs at the node level. Each <i>exploit</i> is shown by an <i>oval</i> , <i>initial condition</i> by a <i>box</i> and <i>post-condition</i> by a simple <i>plain-text</i> . Colored pink nodes represent newly introduced changes in the NAS over Δt	117
5.4	Heterogeneous network models.	129
5.5	The effect of a linear increase in the number of vulnerabilities on the Shortest Path (SP), Number of Paths (NP), Mean of Path Lengths (MPL), and Normalized Mean of Path Lengths (NMPL) metric under different network models.	134
5.6	The effect of a linear increase in the number of vulnerabilities on the Median of Path Lengths (MDPL), Mode of Path Lengths (MoPL), Standard Deviation of Path Lengths (SDPL), Network Compromise Percentage (NCP) metric under different network models.	135
5.7	The effect of a linear increase in the number of vulnerabilities on the Weakest Adversary (WA) metric under different network models.	136

LIST OF FIGURES

5.8	The effect of a linear increase in the number of vulnerabilities on the number of edges of the generated attack graphs under different network models.	137
5.9	The effect of a linear increase in the number of vulnerabilities on the MCS-based metric under different network models	138
5.10	The effect of a linear increase in the number of vulnerabilities on the GED-based metric under different network models	139
5.11	The effect of non-linear variation in the vulnerability density on the MCS-based metric under different network models	141
5.12	The effect of non-linear variation in the vulnerability density on the GED-based metric under different network models	142
5.13	The effect of non-linear variation in the vulnerability density on the MCS and GED-based metrics for the <i>EI</i> network model.	143
5.14	The effect of maintaining constant vulnerability density (at the network level) on the MCS-based metric under different network models . . .	144
5.15	The effect of maintaining constant vulnerability density (at the network level) on the GED-based metric under different network models . . .	145
5.16	The effect of maintaining constant vulnerability density (at the network level) on the MCS-based metric under different network models. The marked text alongside each point in the plot indicates the network hosts which are responsible for the change in the network attack surface. .	146
5.17	The effect of maintaining constant vulnerability density (at the network level) on the GED-based metric under different network models. The marked text alongside each point in the plot indicates the network hosts which are responsible for the change in the network attack surface. .	147
5.18	The effect of maintaining constant vulnerability density (at the network level) on the MCS-based graph distance metric under <i>EI</i> and <i>DMZ</i> network models.	148
5.19	Computation time in microseconds for maximum common subgraph (MCS) and graph edit distance (GED) metrics.	153
5.20	Computation time for MCS	155
5.21	Computation time for GED	156

LIST OF FIGURES

6.1	A test network	165
6.2	Differential visualization of attack graphs at the node level. Colored pink nodes represent newly introduced changes over Δt as appeared only in G_2 . Colored cyan nodes (in both the attack graphs G_1 and G_2) represent the portion of the attack surface which is constant over Δt . Colored green nodes appeared only in G_1 and represented the exploits and initial conditions which are removed due to the application of network hardening procedures.	168
6.3	Simplified attack graphs for the test network at time t and $t + \Delta t$. . .	176
6.4	Change distribution matrix C	177
6.5	Reduced change distribution matrix C_R	178
6.6	Change distribution matrices: Original (C) and Reduced (C_R)	178
6.7	Clustering operations over change distribution matrix C_R	180
6.8	Graph edit distance vs. probability of graph topology change	183
6.9	Computation time in seconds for ECGM-based edit distance under different edge densities	184
A.1	Publication [1]	213
A.2	Publication [2]	214
A.3	Publication [3]	215
A.4	Publication [4]	216
A.5	Publication [5]	217
A.6	Publication [6] decision letter	218
A.7	Publication [6]	218

List of Tables

3.1	Parameters used in proximity-based risk assessment method	69
3.2	Connectivity-limiting firewall policies	70
3.3	System characteristics for the network configuration shown in Figure 3.2	71
3.4	Vulnerability attributes vectors (as per CVSS v3.0 [2] specifications.)	72
3.5	Results of proximity-based vulnerability ranking (for attack graph shown in Figure 3.3)	72
3.6	Comparative results for different risk assessment techniques for the attack graph shown in Figure 3.3.	76
3.7	Results comparison: top 5 vulnerabilities based on the different risk assessment techniques.	77
3.8	Connectivity-limiting firewall policies for the network configuration 2	81
3.9	Comparative results for different risk assessment techniques for the attack graph shown in Figure 3.10	81
3.10	Connectivity-limiting firewall policies for the network configuration 3	83
3.11	Comparative results for different risk assessment techniques for the attack graph shown in Figure 3.11	85
4.1	Attack paths in a resource graph	96
4.2	Algorithm Parameters	102
4.3	Uniqueness (u) and overlap (o) score of attack paths in a resource graph.	106
4.4	Attack paths in a resource graph post diversification of <i>http</i> service on $Host_4$	106
4.5	Uniqueness (u) and overlap (o) score of attack paths in resource graph post diversification of <i>http</i> service on $Host_4$	107

LIST OF TABLES

4.6	Attack paths in a resource graph post diversification of <i>http</i> service on <i>Host</i> ₂	108
4.7	Uniqueness (<i>u</i>) and overlap (<i>o</i>) score of attack paths in resource graph post diversification of <i>http</i> service on <i>Host</i> ₂	108
5.1	Connectivity-limiting firewall policies	116
5.2	System characteristics for the network configuration at time <i>t</i>	117
5.3	Results for the existing attack graph-based metrics for the pair of attack graphs shown in Figure 5.3	118
5.4	Connectivity-limiting firewall policies for the DMZ network model	130
5.5	Network configuration versus vulnerabilities	148
5.6	Sensitivity of the attack graph based metrics to the network security principles and network/security events.	150
5.7	The indegree (<i>id(u)</i>) and outdegree (<i>od(u)</i>) of each vertex type in logical attack graphs	153
5.8	Computation time for the MCS-based metric.	154
5.9	Computation time for GED-based metric.	155
6.1	Firewall rules for the test network at time <i>t</i>	165
6.2	Services and vulnerability description of the test network at time <i>t</i>	166
6.3	Firewall rules for the test network at time <i>t</i> + Δt . Highlighted entries represent change in firewall policies over Δt	166
6.4	Services and vulnerability description of the test network at time <i>t</i> + Δt . Highlighted entries represent newly introduced vulnerable services.	167
A.1	Fact sheet for publication [1]	212
A.2	Fact sheet for publication [2]	213
A.3	Fact sheet for publication [3]	214
A.4	Fact sheet for publication [4]	215
A.5	Fact sheet for publication [5]	216
A.6	Fact sheet for publication [6]	217

Acronyms

AT Attack Tree

CDM Change Distribution Matrix

CVSS Common Vulnerability Scoring System

CWSS Common Weakness Scoring System

DVS Diversity-adjusted Vulnerability Score

ECGM Error-Correcting Graph Matching

FT Fault Tree

GED Graph Edit Distance

IRCR Improved Relative Cumulative Risk

MCS Maximum Common Subgraph

MDPL Median of Path Lengths

MoPL Mode of Path Lengths

MPL Mean of Path Lengths

NAS Network Attack Surface

NCP Network Compromise Percentage

NDVS Normalized Diversity-adjusted Vulnerability Score

NMPL Normalized Mean of Path Lengths

NP Number of Paths

NPVS Neighborhood Proximity-adjusted Vulnerability Score

P Cumulative Probability

p Individual Probability

R Cumulative Resistance

r Individual Resistance

RCR Relative Cumulative Risk

SDPL Standard Deviation of Path Lengths

SP Shortest Path

WA Weakest Adversary

Glossary

attack graph an abstraction representing the ways an attacker can use the interdependencies among vulnerabilities to violate a security policy

attack path a sequence of steps that starts from an attackers initial state to the attackers goal state (security policy violation) in an attack graph

attack surface subset of the system's resources (methods, channels, and data) that can be potentially used by an attacker to launch an attack

breach a security policy violation

countermeasure any action performed to remove at least one vulnerability from a system

entity a computer system, a process, a person, or a collection of such entities

exploit an instance of a vulnerability specifying the preconditions and postconditions of the vulnerability with details from the network under study

hardening providing protection to an entity

network attack surface subset of network configuration and vulnerabilities (known vulnerabilities, in particular as we do not have information about the zero-day vulnerabilities) that an adversary can use to violate the network security policy

security the process of providing confidentiality, integrity and availability to an entity according to some policy

security control a countermeasure or safeguard

security metric a value produced from measuring identifiable entity attributes that affect physical, personnel, IT, or operational security

security policy a requirement that specifies the secure states of the network

vulnerability a weakness in the system that is described by its preconditions and post-conditions

zero-day vulnerability a flaw in the programming code of computer software, unknown to the computer user and software manufacturer, that can have catastrophic consequences

Chapter 1

Introduction

“If you know your enemies and know yourself, you will win hundred times in hundred battles” The Art of War, Sun Tzu [10].

With the increasing use of the Internet and information systems in all walks of modern life, the security of computer networks is of prime importance. Cyber criminals can carry out identity theft and financial fraud; steal corporate information such as intellectual property; conduct espionage to steal state and military secrets; and recruit criminals to disrupt critical infrastructure by exploiting vulnerabilities in any system connected to the Internet. At the root of almost every security incident on the Internet are one or more software vulnerabilities. Therefore, mission-critical systems may be compromised or exposed if applications are not securely designed, developed, tested, configured, and deployed. Organizations typically implement various security measures/controls to guard against known threats. In particular, network security management controls consist of:

- Perimeter defense,
- Traffic inspection and detection of anomalies and threats,
- Detection and prevention of intrusion,
- Filter, block and prevent the malicious traffic,
- Restrict insecure ports, protocols services, and connections to the external world and the Internet,

-
- Restrict, change and segment users access,
 - Vulnerability detection, prioritization, and patching, etc.

Nowadays, Cyber attacks are more sophisticated wherein adversary combines multiple network vulnerabilities to compromise the mission-critical resources protected with state-of-the-art security solutions. Therefore, compromise of one element in network infrastructure may have catastrophic effect jeopardizing the security of overall infrastructure and information. In this thesis, our focus is on network security in general and vulnerability analysis in particular.

In any organization, security risk assessment should be conducted periodically to evaluate risks and associated threats leading to loss of confidentiality, integrity, and availability of information. Threats and vulnerabilities related to the information must also be assessed for their potential impact. Therefore, information security risk assessment should be an ongoing activity, triggered when changes are made to the existing network configuration. Essentially, organizations need to monitor the overall security of their managed networks. Administrator's should know the current status of network security to determine which systems need closer and urgent attention; so that they can prioritize their hardening efforts. This observation points towards measurement and models that quantify security as a system attribute. Security metrics assess an entity's ability to provide itself with confidentiality, integrity, and availability. Such assessments are the values derived from measuring the security-relevant attributes of a system (or a network). When the entity measured is an organization, the values produced are referred to as enterprise security metrics. Security metrics play a crucial role in measuring and assuring the desired level of network security.

INFOSEC Research Council [11] has affirmed that enterprise security metrics is one of the top 8 security research priorities. This ranking is due to the compelling promises assured by the security metrics. With these metrics, it's easy for an enterprise or an organization to determine (i) whether their security is improving (or deteriorating) with time, (ii) their return on investment (ROI) on security countermeasures, and (iii) compare their security performance with other similar organizations [12]. Even though there are no comprehensive, universally accepted metrics for addressing issues mentioned above, significant progress has been made toward this end [13].

In general, security metrics are used to help achieve information security goals. The goals of information security are to detect, prevent, and recover from the potential attacks [14]. Cyber attacks, which are adversarial actions that violate network security policy, are necessarily linked to the information security goals. Given the significance of Cyber attacks to the security of information systems, a model that imitates the ways an adversary can successfully compromise a system is invaluable to the network security evaluation.

Attack Graphs [15], [16], [17], [18] have been used in the past decade to uncover potential multistage, multi-host attack paths in a given network. An attack graph is a formal, network vulnerability analysis model that represents all possible attack scenarios by which an adversary can violate a network security policy. An attack graph for a given network can be generated from the network configuration details and the known vulnerabilities present in it. It represents how an adversary could leverage dependencies (cause-consequence relationship) between the network vulnerabilities to violate a security policy. If an attacker violates a network security policy by first exploiting a Secure Shell Daemon (*sshd*) vulnerability on host h_1 and then exploit a Remote Shell Daemon (*rshd*) vulnerability on host h_2 , then it is for sure that the adversary would be benefited from the interdependency between the *sshd* vulnerability on h_1 and the *rshd* vulnerability on h_2 . Such sequence of attacker's actions constitutes an attack path. In an attack graph shown in Figure 1.1, nodes correspond to hosts and edges corresponds to the vulnerability exploits. As evident from Figure 1.1, the attack path consists of three nodes and two edges. The third node would represent the host (either internal or external to the network) from where the adversary initiated the attack on h_1 . Therefore, an attack graph for a vulnerable network configuration is a collection of potential attack paths.

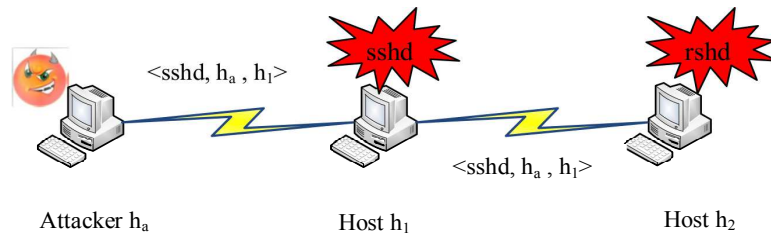


Figure 1.1: An example attack path

Analysis of the generated attack graph aims to infer security-relevant properties about the target network. Vital information in the form of most critical vulnerabilities and initial conditions (e.g. service connectivities, privileges of the software/services running with, etc.) which are crucial to the success of an adversary, assists security analysts in allocating scarce security resources efficiently. Therefore, attack graph analysis that extracts security-relevant information from the attack graph is referred to as attack graph-based security metric. As risk assessment is an essential factor in human decision making, attack graph-based metrics are useful to security managers for allocating scarce security resources efficiently and thereby achieving best possible network hardening. Attack graph-based security metrics find use in diverse security applications such as testing of unsuspecting system properties for imminent threats [16], [19], [20], [21], optimal selection of countermeasures [22], [23], optimal placement (deployment) of intrusion detection systems [24], testing of new network configurations for their security strength [25], [26], [27], etc. With the availability of scalable attack graph generation tools like NetSPA [1], MulVAL [21], and CAULDRON [28], the impact of attack graph-based security metrics for proactive network hardening will likely to grow in the coming years.

To ease the job of security managers and facilitate system hardening, in this thesis, we utilize the attack graph frameworks [16], [21], [29], [30]. We have developed new attack graph-based methods and metrics to:

- Assess the security risk posed by each exploitable vulnerability in a given network,
- Assess the diversification level of each attack path in a resource graph generated for a given network,
- Determine temporal variation in the network attack surface, and
- Effectively identify newly introduced changes in the attack surface.

1.1 Motivation

Security metrics derived from the attack graph analysis provide a quantitative basis for the management decisions that affect the protection of critical infrastructure. These metrics provide decision-making support to the network administrator's to decide on

which systems need closer and urgent attention and thereby improve network security. Here, we briefly review motivating problems that lead to our proposals.

1.1.1 Vulnerability Risk Estimation

Network security risk assessment and mitigation is imperative for the protection and maintenance of today's critical infrastructure. To enhance the security of a given network, it is crucial to evaluate network vulnerabilities for their security risk since vulnerabilities are the key to all network intrusions. Recent years have seen significant progress in software vulnerability risk estimation [2], [26], [27], [31], [32], [33]. Further, large number of vulnerability risk assessment methods were proposed by security vendors and non-profit organizations such as US-CERT [34], SANS [35], ISS X-FORCE [36], National Vulnerability Database (NVD) [37], Vupen Security [38], Secunia [39], Microsoft [40], and Symantec [41].

Standardization efforts on security metrics, such as CVSS [2], [42], [43] and Common Weakness Scoring System (CWSS) [44] focus on ranking well-known vulnerabilities and software weaknesses, respectively. Such scoring systems help administrators in measuring the severity and impact of individual vulnerabilities and hence in the process of patch management. Although popular, both CVSS and CWSS do not capture the interdependency (cause-consequence relationship) between network vulnerabilities. Therefore, they are deemed insufficient in the context of multistage, multi-host attacks. Furthermore, CVSS and CWSS measures the severity of a particular software vulnerability/weakness in isolation and hence do not capture their overall impact on the security risk of a network. Consequently, the use of such risk scoring systems often results in an imprecise risk prioritization and sub-optimal security countermeasures. Moreover, due to the dynamics in computer networks, the risk posed by the vulnerabilities changes over time. However, the CVSS does not accommodate such changes in the vulnerability risk. Because of such limitations (of the both CVSS and CWSS), a vulnerability risk assessment process requires an additional step to evaluate the risk posed by the network vulnerabilities. This can be achieved by adjusting/augmenting the CVSS Score of network vulnerabilities with the various risk conditions of the underlying network.

Existing attack graph-based security metrics such as cumulative probability [26], and cumulative resistance [27] consider the interdependency between the exploitable vulnerabilities for assessing the security risk posed by each of the exploitable vulnerability uncovered in a given network. However, they do not consider the vulnerability diversity along the attack paths. Chen et. al. [45] used diversity among the network vulnerabilities as an important risk condition to assess the security risk of a network. Whereas, Wang et. al. [30], [46] used vulnerability diversity along the attack paths to measure the robustness of a system against the zero-day attacks. Suh-Lee and Jo [33] used the proximity of un-trusted network and risk of neighboring hosts as important risk conditions to assess the security risk of each vulnerability in a given system. However, they do not consider critical risk conditions such as the cause-consequence relationship between vulnerabilities and exploit diversity along the attack paths. Work of Wang et al. [30], [46], Chen et al. [45], and Suh-Lee and Jo [33] motivated us to consider various network risk conditions for vulnerability risk assessment.

In the context of multistage, multi-host attacks, the security risk posed by a given exploitable vulnerability depends on several network risk conditions such as:

- Proximity of the target vulnerability from the attackers initial position,
- Vulnerability diversity along the attack path(s),
- The number of vulnerabilities from where the vulnerability under consideration could be directly reached and exploited.

Such risk conditions affect the success of an adversary and hence influence the risk posed by the vulnerability. Despite having an array of security metrics [2], [26], [27], [31], [32], [33] for vulnerability risk assessment, there has been no comprehensive measure which considers critical risk conditions. Consequently, the vulnerability remediation plan based on the existing security metrics often results in an ineffective application of countermeasures. Therefore, there is an urgent need for integrated comprehensive security metric that can accurately estimate the risk of each exploitable vulnerability in a dynamic network. Such metric can help security administrators in accurately determining top vulnerabilities and in prioritizing vulnerability remediation activities accordingly.

1.1.2 Network Hardening and the Threats of Zero-day Attacks

Attack graph-based security metric [25], [26], [27], [47], [48], [49], [50] provides useful information that can be acted upon and trigger appropriate action to deter potential multistage attacks. All the benefits of existing security metrics can be a potential weakness when the more secure network configuration is equally susceptible to zero-day attacks. In general, unknown vulnerabilities are considered as immeasurable due to the less predictable nature of software errors. Therefore, the research on security metrics has been hampered by difficulties in handling zero-day attacks wherein an adversary exploits multiple unknown vulnerabilities. It questions the usefulness of the existing security metrics because a more secure network configuration would be of little value if it is equally vulnerable to zero-day attacks.

Wang et. al. [30], [46] addressed the above shortcoming of existing security metrics. The authors proposed a k -zero day safety metric which essentially counts the minimal number of different zero-day vulnerabilities required to be exploited by an adversary to compromise the target resource. Larger the count, more secured the network is since the likelihood of having the unknown vulnerabilities available, applicable, and exploitable at all the same time is significantly lower. Based on the k -zero day safety metric, Wang et. al. [9] derived a new metric called *least attacking effort-based diversity metric* to measure network's capability in resisting intrusions or malware infection that employs multiple zero-day attacks. The derived metric is capable of measuring the robustness of enterprise network against the zero-day attacks. These advances [9], [30], [46] have demonstrated that through adequate diversification of the vulnerable services along the attack paths, an administrator could increase the network robustness against zero-day attacks.

Moreover, the case of misplaced diversity is prevalent in today's computer networks since the deployed network configurations are not security conscious. It results in multiple loopholes that in turn provide an adversary more ways to compromise a system. Lack of security metrics that guides administrator on diversifying enterprise network complicates the situation. The side effect of resultant unplanned or non-strategic diversification is that it provides an adversary more opportunities to compromise the system. Therefore, there is a need for suitable metrics that can tell the current diversification level of each attack path and helps in figuring out the attack paths that needs

immediate attention. Further, the technique for the identification and diversification of the repeated services along the attack paths could enhance the robustness of computer networks against the zero-day attacks.

1.1.3 Assessing Temporal Variations in the Network Attack Surface

With frequent changes in the network configuration, today's computer networks undergo continuous evolution. Consequently, there is a constant risk of information exposure to a larger threat landscape. Such ever-changing (dynamic) computer networks have varying attack surface. Essentially, the network attack surface is a subset of network configuration and vulnerabilities (known vulnerabilities, in particular as we do not have information about the zero-day vulnerabilities) that an adversary can use to compromise the network security. Constant discovery of new vulnerabilities, misconfiguration of hardware (or software) components, etc., can further intensify change in the network attack surface. Therefore, there is a pressing need to consider temporal aspects of network security. According to the standard guidelines and recommendations issued by ISO/IEC 27005 [51], and ENISA [52], [53], for maintaining the best possible security posture, computer networks have to be regularly monitored for security policy violations. Therefore, there is a need for suitable metrics to assess temporal changes in the attack surface of dynamic networks.

1.1.4 Depicting Temporal Variations in the Network Attack Surface

Assessing change in the attack surface of dynamic computer networks is indispensable to the security administrator. Even though it is possible to efficiently generate attack graphs for a realistic network, resulting graphs pose a serious challenge for human comprehension. It necessitates a visualization of the newly introduced changes in the network attack surface that are not so obvious even with the effective attack graph visualization. Therefore, techniques should be there to discern variations in the network attack surface so that the portion of the attack graph that has changed can be inferred. The technique should identify the newly introduced exploits and respective enabling

conditions in the dynamic network. Quick identification of modification in the network attack surface and hidden root causes in a time-efficient manner is crucial to the prevention of future attacks. In this thesis, we develop a technique for reliably discovering and depicting newly introduced changes in the network attack surface that can make attack graphs more understandable and useful.

1.2 Thesis Outline

Chapter 2 is a review of related work and highlights the novel aspects of the thesis compared to previous work. In Chapter 3, we propose a novel comprehensive metric called Improved Relative Cumulative Risk (IRCR) for measuring the security risk posed by exploitable vulnerabilities in a dynamic network. In Chapter 4, we propose metrics for quantifying the diversification level of each attack path in a resource graph. We also specify diversification algorithm for identification of repeated software/services along the attack paths that needs to be diversified for increasing the robustness of the network against zero-day attacks. In Chapter 5, we propose to use the classical graph distance metrics such as MCS [54] and GED [55] for measuring the temporal variation in the attack surface of dynamic networks. We also present the results of the MCS and GED based graph distance metrics on synthetic networks. In Chapter 6, we propose an error-correcting graph matching (ECGM) based change detection technique in the network attack surface. We also propose a change distribution matrix (CDM) based method to detect variations in the attack surface. Finally, Chapter 7 summarizes our contributions including future research.

1.3 Contributions

In this thesis, we have proposed new attack graph-based methods and metrics for proactive network hardening. The contributions are listed below:

- **Chapter 3** focuses on quantifying the security risk posed by exploitable vulnerabilities in a given network. To cater to this problem we have augmented the static risk score of each exploitable vulnerability, i.e. CVSS [2] with various network

risk conditions that affect the success of an adversary. We have used an exploit-dependency attack graph [56] as a network security model in conjunction with the CVSS Framework [2]. The generated attack graph captures adversarys all possible attacking strategies, and hence critical risk conditions that influence the success of an adversary. Building on the work of [26], [27], [30], [33], [45], [46], we have proposed IRCR, an integrated comprehensive measure for vulnerability risk estimation.

- First, the Diversity-adjusted Vulnerability Score (DVS) is computed for all vulnerabilities present in the generated attack graph. DVS for a given vulnerability is obtained through adjusting its CVSS Base Score by minimum path resistance score. More the path resistance, less will be the DVS value and vice versa.
- Next, Neighborhood Proximity-adjusted Vulnerability Score (NPVS) is computed for each vulnerability. NPVS for a vulnerability represents the security risk due to the neighboring vulnerabilities (immediate predecessor in the context of attack graphs) from where the target vulnerability can be directly reached and exploited. More the neighbors, more will be the attack opportunities.
- Finally, DVS and NPVS of a given exploitable vulnerability are combined to get IRCR score.

A lower value of IRCR is desirable for better security. Based on the IRCR recommendations, an administrator can accurately determine top vulnerabilities and prioritize vulnerability remediation activities accordingly. We found that the IRCR is complementary with state-of-the-art vulnerability risk scoring techniques (i.e. P [26], R [27]) on synthetic networks. Moreover, IRCR is adaptive since it automatically adjusts according to the network (or security) events.

- **Chapter 4** proposes an attack path diversification technique for increasing the network robustness against zero-day attacks. We have proposed an intra-path diversity metric and two inter-path diversity metrics called *uniqueness* and *overlap* to assess the diversification level of each attack path in a resource graph generated for a given network. Attack path(s) in which one or more vulnerabilities

could be exploited more than once will be identified. Then, we have identified all the repeated software/services along the attack paths that need to be diversified. To cater to this problem, we have developed an attack path diversification algorithm. Identified repeated services along the attack paths are replaced with functionally equivalent alternatives in such a way that an adversary should require an independent effort for exploiting each vulnerability. The efficacy and applicability of the proposed diversification technique are demonstrated through a small case study. Experimental results show that such additional metrics for determining the diversification level of each attack paths would clearly benefit administrators in increasing the network robustness against the zero-day attacks.

- **Chapter 5** focuses on assessing the temporal variation in the attack surface of dynamic networks using classical graph distance metrics. We have used logical attack graph [16], [21] as a network security model to capture the attack surface of the underlying network. Essentially, attack surface of a given computer network is a subset of network configuration and vulnerabilities that an adversary can use to compromise the target network in an incremental fashion. Since the attack graph is capable of successfully capturing the network attack surface, the distance between a pair of successive attack graphs (generated over the observed sampling interval) indicates the change in the network attack surface. We have used classical graph distance metrics based on the Maximum Common Subgraph (MCS) [54], and Graph Edit Distance (GED) [55] to quantify the distance between a pair of successive attack graphs generated for a dynamic network.

We found that the MCS and GED based graph distance metrics are competitive with state-of-the-art attack graph-based metrics on all the three different network models, viz., Flat, External-Internal, and DMZ. The MCS and GED based graph distance metrics successfully capture the temporal variation in the attack surface and also generate an alert about the security events which are responsible for such change. These graph distance metrics scale polynomially with the attack graph size. Moreover, the performance of MCS and GED based metrics is almost similar and hence the computation of one metric is enough to detect temporal variation in the network attack surface.

- **Chapter 6** focuses on the problem of effectively visualizing the newly introduced changes in the attack surface of dynamic network. We have proposed a change distribution matrix (CDM) based technique to detect the newly introduced changes in the attack surface of dynamic networks. For doing this, we have reduced the problem of change detection in the network attack surface to the error-correcting graph matching (ECGM) problem. We found that the CDM based technique identifies the root causes responsible for the incremental change in the network attack surface. Further, it identifies the newly introduced exploits and respective enabling conditions in the dynamic systems and discerns portion of the attack graph that has undergone maximum change. Such identification of an incremental change in the network attack surface and hidden root causes in a time-efficient manner is crucial to the prevention of future attacks.

Chapter 2

Review of Related Work

“Unfortunately, even with industry-best defenses, a sufficiently motivated attacker can penetrate the network” [57].

There is currently a substantial body of research in the area of attack graph generation and analysis, and much of it has focused on the problem of extracting security-relevant information from the generated attack graphs for proactive network hardening. In this Chapter, we begin by describing the background, terminologies, and tools necessary to understand the fundamental issues involved when combating computer and network intrusions. We then review related works on defending against multistage, multi-host attacks, including attack tree, attack graph. We also review existing attack graph-based security metrics. We discuss why existing work on attack graph-based metrics is not satisfactory with respect to the problems addressed in this thesis.

2.1 Vulnerabilities and Exploits

Security administrators are primarily concerned with detecting and fixing software vulnerabilities before outside attackers exploit them. Vulnerabilities are weaknesses in software packages and systems that allow the software to operate outside of designed boundaries. Exploits utilize vulnerabilities to gain otherwise unavailable information and privilege. Security databases such as the National Vulnerability Database [37],

Common Vulnerabilities and Exposures (CVE) [58] documents software-related vulnerabilities. Additionally, these databases have been used to characterize the frequency at which software vulnerabilities are found and subsequently fixed [59]. Whereas, the ExploitDB [60] database archives exploits and vulnerable software.

In general, software vulnerabilities arise from basic system complexity and bad programming practices, as is the case with most buffer overflow vulnerabilities. Other vulnerabilities, however, are inherent in software design or arise from misconfiguration. For example, FTP transmits login and authentication information “in a clear” via an unprotected channel. NVD [37] provides several distinctions between vulnerabilities such as the necessary position of an adversary relative to the target host and the consequences of an exploited vulnerability.

Exploits take advantage of software vulnerabilities to gain inaccessible information and unauthorized privilege on the victim host. Exploitation of the FTP service could involve listening, or “sniffing,” the network to obtain packets destined for another machine. The login information can then be read straight from these plain-text packets, granting an attacker the ability to log into the FTP server and masquerade as a legitimate user. Another exploit of the same FTP server could involve sending specially crafted input that causes the server to execute arbitrary commands at the privilege level of the FTP server.

Exploits are characterized based on the level of privilege provided and the necessary location of the attacker relative to the target host. Local exploits are those that can only be executed when on the same machine as the vulnerable software, whether by being physically at the terminal or logged in remotely. In contrast, remote exploits allow an attack to be executed from a remote location, only necessitating a network connection between the attacker and the vulnerable software’s port. Successful exploits provide an attacker with privilege level equal to that of the running program.

The most common and well-studied vulnerabilities and associated exploits arise when a programmer accepts input from a user and copies it into a buffer not big enough for the input. In general, right practice in such a case is to truncate the input to a size that fits within the buffer. In the past, many programmers simply made the buffer large enough to fit everything that they thought was reasonable and blindly copied the input into it. Intelligent security analysts, however, found out that it was possible to give the

incorrectly written software a piece of data larger than the buffer size, thus overflowing it, and writing to unauthorized places in memory. With a little knowledge of the computing architecture and stack, it is then possible to hijack the program to execute arbitrary code with privileges equal to that of the running program [61]. Such attack is known as a buffer overflow attack. One reason that buffer overflow attacks are so well known is that they are relatively common and easy to exploit. If the vulnerability is evident in a network program, it often can lead to remote compromises.

Another instance of incorrect input validation is evident in format string errors. The format string exploits are enabled, however, due to the improper handling of format strings, such as those passed to C functions *printf* and *sprintf*. Specially crafted format strings allow a user to write to arbitrary points in the program's memory. These sort of vulnerabilities are less common; however, they do occur, and an adversary can exploit them [62].

Logic errors comprise a nefarious class of vulnerabilities that cannot be easily pigeonholed. Some are simple mistakes in coding, such as not handling a boundary case or not ending a loop at the right time. Others can be complex, as in not providing a mechanism to handle a particular input case. Logic errors are often overlooked because the program works correctly under normal conditions and only resort to incorrect behavior with special case inputs. While buffer overflows and format string exploits usually give the adversary privilege on the machine running the vulnerable software, logic errors behave in numerous ways when exploited. Upon successful execution of an exploit, the adversary can steal information from the computer, change a local file, or simply crash the software. For example, a naive Web server that does not correctly parse the character ';' could allow an attacker to remotely download any file inside of the Web server's configuration directory. Another prime example of logic errors found in race conditions.

Several other types of computer attacks exist that don't categorize well to specific vulnerabilities. Some types of denial-of-service (DOS) attacks attempt to send requests to a piece of software faster than the system can handle them. This flooding can prevent the attacked software from responding to valid queries coming from other agents (authorized users), thus rendering it useless. While a DOS attack does not gain an attacker any sort of direct access to the network, it is incredibly easy to create, difficult to protect against, and can cause businesses to lose money due to the server downtime.

Possibly the most naive attack available, a brute force attack, attempts to enumerate through all of the possible combinations of input to try and find something that “works”. The most typical example of such attack is in password guessing. An attacker can just enumerate all of the possible combinations of letters and numbers until the right one is found. While this can take long time, brute force attacks can be speeded up with any prior knowledge. In the password guessing example, the attacker might first try all of the letters in person’s name, as well as numbers in his birthday and social security number, checking for weak passwords. Similar brute force attacks can be staged against cryptographic keys and other authentication information.

Finally, worms comprise another class of computer attacks. Worms are malicious programs that automatically exploit known system vulnerabilities and then use compromised hosts as launching station for additional automated attacks. Examples of such attacks are the Code Red [63] and Nimbda [64] worms, which propagated by exploiting Microsoft’s Internet Information Service (IIS).

2.2 Understanding Network Attacks

To secure computer networks from various types of Cyber attacks, we need to understand why they are vulnerable to attacks, and what are the potential attack steps. In this Section, we review different aspects of computer networks, Cyber attacks, and potential attackers.

2.2.1 Computer Networks

Computer network plays an important role as it binds all the information assets together and provides a means for the operational transactions where different entities can participate, exchange information and carry operations over the information by making use of specific ports, protocols, and services provided by the network. It may create the possibilities of exposure of information. Today’s computer networks are increasingly vulnerable to Cyber-related attacks for several reasons listed below:

1. Open industry standard protocols replacing vendor-specific proprietary communication protocols

Exploitation of publicly known and unknown (zero-day) software vulnerabilities is continuously increasing on an unprecedented scale with extraordinary sophistication. Critical life-sustaining infrastructures that deliver electricity and water, telecommunication, Internet and broadband connectivity, control air traffic and support our financial systems all are increasingly vulnerable to the exploitation of publicly known vulnerabilities. In the past, the safety and mission-critical systems responsible for the control of critical infrastructure used proprietary communication protocols for data transfer. However, the situation is changing due to the cost constraints [65]. Nowadays, the critical infrastructures are getting connected to the publicly open Internet by making use of Internet's core communication protocols, i.e., TCP/IP. Consequently, all parts of the network communication, i.e., critical data in the passage and at the end-points, become increasingly vulnerable to remotely exploitable vulnerabilities. A potential determined adversary can remotely access critical systems with relative ease. Such unauthorized access to the critical infrastructure can bring financial instability, damage or destruct organizations proprietary data, cause equipment damage, or worse, personal harm.

2. Standard software's replacing proprietary softwares.

An example of this aspect is commonly noticeable in Supervisory Control and Data Acquisition (SCADA) systems. Previously, SCADA systems used the proprietary software developed in-house by organizations or by a trusted third party. Nowadays popular operating systems such as Windows, Unix, and Linux are in use for building SCADA systems [66]. Additionally, SCADA systems depend on other necessary components like Web browsers. Consequently, SCADA systems and hence SCADA networks which are linked with the corporate networks) are susceptible to the publicly known vulnerabilities in Commercial-Off-The-Shelf (COTS) software [67].

3. Time-to-market priorities lead to the insecure software development and deployment.

Because of the current time-to-market demands of the software industry, software producers are shipping not so well tested products [68]. Developers are usually more concerned with the development of buyer-acceptable software at

minimum cost and with quicker time-to-market to assure gain over the competition, rather than with the effort towards testing security issues in their product. To achieve faster software development, developers utilize already available third-party software libraries that cover part of the project requirements, before attempting to develop the necessary functionality from scratch. In contrary to the above benefit, code copying results in a sharing of vulnerabilities which could be exploited by malicious attackers to gain the full privileges of the application. All the benefits of imported libraries become a potential weakness when the attacker exploits vulnerabilities in it. Deployment of not so well tested products becomes an option for the software vendors because security is an externality [69], [70], as the risks induced by the software vulnerabilities are borne (bear) by software consumers, not by software producers. Marketing initiatives may always beat the poor stature which might result from this strategy. In other words, software vendors strive to balance the costs of more secure software: added developers, fewer features, longer time-to-market in contrast to the costs of insecure software: liability to patch, occasional bad press, likely loss of sales.

4. Software monoculture.

Today's computer networks are uniform/homogeneous in nature since networking components (e.g. computing systems, software's, and networking technologies, etc.) were developed in the context of end users and operators. Furthermore, today's mass users respect standardization, predictability, and availability of the technology [71]. Computer technology manufacturers produce multitudes of identical copies of hardware or software by massive cloning of single design. Homogeneous networks are well-maintained in terms of running substantially the same software, using same computing platforms, configuring network middleboxes (e.g. switches, routers, etc.) with the same configuration rules. Such kind of uniformity simplifies the task of resource management and hence reduction in operational cost. Such prevailing network monoculture favors network management (i.e. distribution and maintenance), improved productivity, scalability, portability of user skills, fewer configuration errors, lesser user training cost in case of the job transfer [72]. Moreover, monoculture facilitates easy

debugging and improved interoperability among few different kinds of systems compared to a more diverse collection of systems.

In contrast to the above benefits, monoculture results in a sharing of vulnerabilities, which puts the entire networked system at a risk of rapidly spreading virus or other malware for example Stuxnet [73]. All the benefits of network uniformity/homogeneity become a potential weakness when systems replicate software errors/flaws because the adversary can easily exploit such flaws. All the Softwares with similar configurations can be compromised very quickly if the adversary engineers a method of compromising the security of one software. For example, in the case of botnets, a massive attack army is formed with a very low cost by exploiting the same vulnerability over and over again in a significant number of systems having a similar design and configuration. The situation is even worse when similar (homogeneous) systems are interconnected.

5. Networks are dynamic environments.

With frequent changes in the network configuration, today's computer networks undergo continuous evolution. Consequently, there is an introduction of vulnerabilities whose exploitation results in the constant risk of information exposure to a larger threat landscape. Applications are undergoing continuous innovations, several architectural ideas, and platforms are under evolution, and numerous have already been deployed. New ways of managing and setting up sessions are being implemented, and transaction processing is undergoing change with respect to the way information is handled. It makes applications vulnerable to many new types of attacks. Besides, as networks are enablers of business opportunities, the organizational ecosystem is undergoing transformation, extending its boundaries by increasingly providing access to third parties and vendors, integrating external interfaces, adopting innovations in endpoint, mobility, and wireless technologies, while relaxing norms of standardization and ownership of connecting devices. Such dynamics in the networks eventually leads to change in firewall access control policies opening up opportunities of misconfiguration and hence security compromise. Further, mobility platforms allow organizations to extend access to operational information to employees on the move and from outside the perimeter of the enterprise network. Employees may access such

information either on devices issued by the organization or on their personal devices such as laptops and PDAs. However, such need for reachability opens the doors for new Cyber threats since there is no way to differentiate legal access from malicious access if the latter uses the legitimate procedure to get into the network.

6. Computer networks are not at all vulnerability-free.

Several studies state that there is always a window of opportunity (between vulnerability disclosure and patch/fix release) available to an adversary to take advantage of the discovered vulnerabilities [74]. Even though vulnerability patches are released immediately, the administrator is typically slow in applying patches [75]. A viable explanation for this inability may be the lack of resources (security controls), available patches may have unexpected consequences, may results in the introduction of new vulnerabilities in the network [76]. Further, to achieve the primary objective of an enterprise (e.g. profitability), an administrator may be unable to fix certain vulnerabilities. Before applying a patch in production level system, the patch has to pass certain tests to ensure that it does not introduce other vulnerabilities or adversely affect other parts of the system. If the vulnerability patch does not pass the necessary tests, the production level systems would have to continue running with known vulnerabilities. Another coarse solution such as closing ports, shutting down service deemed unsuitable because it hurts primary organizational objectives.

Additional study, such as the one on Data Breach by Verizon [77], which investigated 800 cases of security breach and data compromise incidents between 2004 and 2011, discovered what they called “unknown unknowns”. Verizon [77] found that many of the data breach cases investigated involved some unknown to the organization: unknown/unclaimed assets (systems), unknown network connections, unknown data, and anonymous/hidden accounts on the organizational assets. Therefore, despite the availability of fixes for the reported vulnerabilities, it may also happen that the organizations have unknowns which contain easily exploitable vulnerabilities.

The above-stated facts lead us to the conclusion that it is highly unlikely to have a computer network, as a whole, which will forever be free from vulnerabilities.

7. Humans contribute to making networks even more vulnerable.

As a human being, we are good at identifying threats from the physical domain than threats from the digital domain. It is because threats in a physical domain are distinct and visible. On the other hand, threats are not evident at all in the digital domain because it can come from all over the globe. Even though a network is assumed to be 100% secure, the likelihood of humans making it insecure is very high [78]. According to the IBM's Cyber Security Intelligence Index [79], 95% of all security incidents involve human error. Many of these events result in successful security attacks from external attackers who prey on human weakness to lure insiders within organizations to provide them with access to sensitive information unwittingly. These mistakes are costly since they involve insiders who often have access to the most confidential information. The greatest impacts of such successful security attacks involving insiders are exposure of sensitive data, theft of intellectual property and the introduction of malware. Most of the security threats that directly result from insiders are the product of innocent mistakes rather than malicious abuse of privileges. Human error is also a factor in other security incidents caused by insiders who are the most trusted and highly skilled, such as system and network administrators. According to IBM's report, some of the most commonly recorded forms of human error caused by such employees are system misconfigurations, poor patch management practices and the use of default names and passwords.

As there is always a trade-off between usability and security, users tend to choose the former one, and hence security becomes a secondary requirement. Besides, the human interest factor is also being exploited by attackers and plays a significant part in successful security attacks seen today, but it is not always attributed to mistakes made by insiders. As humans are easily convinced and faithful, that is why many of these attacks involve social engineering techniques to lure individually targeted users into making mistakes. According to Verizon's "2013 Data Breach Investigations Report [80]", 95% of advanced and targeted attacks involved spear phishing scams with emails containing malicious attachments that can cause malware to be downloaded onto the users computing device. It gives

attackers a foothold into the organization from which they can move laterally in search of valuable information, such as intellectual property.

To conclude, Humans are the weakest link in the security ecosystem of an organization. As attackers focus on such weakest link to circumvent most sophisticated security countermeasures/defense put in place; human-related vulnerabilities contribute towards many security incidents faced by today's organizations. Therefore, humans are a source of exposures, which indirectly makes networks vulnerable.

8. Threats are increasingly endangering networks.

The evaluation of threats is generally performed based on the following considerations (see e.g. [81], [82]): (i) threat agent motive, (ii) threat agent's skills, and knowledge, (iii) cost in terms of time, effort and resources a threat agent has to bear in order to execute a particular kind of attack, and (iv) opportunity to launch an attack. However, the ever-growing use of automation while staging Cyber attacks and the advent of the Internet have put factor (ii) and (iii) stated above under a different perspective, difficult to reason about.

Let's consider publicly available resources, i.e., hacking tools such as backdoor tools, kernel-level rootkits, distributed password crackers, war driving tools that threat agents (adversary) can use. More information about these tools is given in [57], [83]. The selection of such ready-to-use tools for different purposes reveals that the adversary can use these tools with low-to-moderate expertise, knowledge, and skills, at least a lot less than what is required to build these tools. Thus, there is a vast army of attackers out there, with the necessary expertise, skills, and knowledge, capable of following step-by-step procedures on how to use such hacking tools. Additionally, increasing use of automation in attacks also puts time and effort under a different dimension. It doesn't matter if a password cracker takes a day to crack 1000 passwords [78]. It is because the adversary is not regulating or executing the password cracking process manually. If at the end of the day, the cracker reports one password cracked, the adversary attempt results in success, and the adversary can penetrate or proceed to infiltrate a network. Therefore, the most significant factors determining threat seem to be attackers motive and opportunity to attack. Attack motivation will identify

the strategy chosen by the adversary, and the ultimate objective of the intrusion. Since attack plan and attack target are unknown, defenders need tools to reason about network attacks. We assume the worst-case scenario that there will always be motivated determined adversary willing to take opportunities to compromise targets embedded well within the network. Again, it is up to the organizations to track such opportunities; it means to identify potential multi-step attacks.

2.2.2 Network Attacks

In the preceding Section, we saw how various elements contribute to making computer networks vulnerable to many new types of attacks. In this Section, our focus is on describing single-step and multi-step attacks.

2.2.2.1 Single-step Attacks

Attacks sometimes exploit a single vulnerability. Worms, for example, automated, self-propagating attack software, often search for only a single vulnerability in a host and exploit it if found. Novice computer attackers tend to act in much the same fashion, exploiting one of the few vulnerabilities to get some sort of access to a network.

2.2.2.2 Multistage, Multi-host Attacks

Proficient network attackers, however, often have a goal in mind when attacking a network, and craft their exploits to reach the goal in the easiest way. Besides, expert attackers tend to go through many actions to both limit visibility to Intrusion Detection Systems and quickly exploit targets. Finally, a network attacker often leaves behind a malicious software to provide a permanent “back door” into the network. This multi-component attack can be considered a path on a graph of all the possible attack routes through a network, otherwise known as an attack graph [84], [85].

Figure 2.1 illustrates an example of such multistage, multi-host attacks. An adversary at a given network location can progressively compromise vulnerable hosts, using information about the software vulnerabilities and network reachability. The example in Figure 2.1 shows that an attacker can first use a remotely exploitable buffer overflow vulnerability to achieve administrator-level privileges on a web server. A database

server can then be compromised at the user level with a remote exploit from the web server, and this privilege can be raised to administrator level by exploiting two new local vulnerabilities. Other sequences of exploits are also possible, as indicated by the unlabeled parts of the graph.

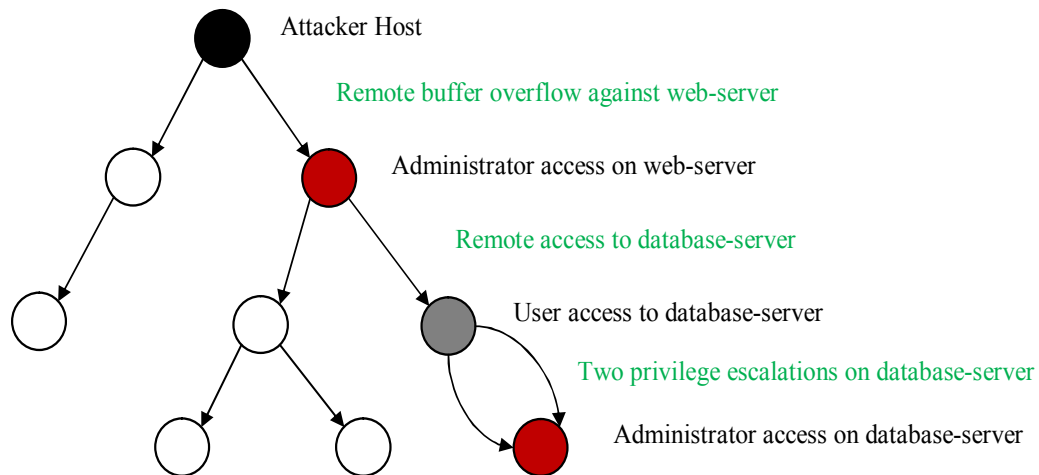


Figure 2.1: An example attack graph (adapted from [1])

2.2.3 Anatomy of a Network Attack

An adversary external to an enterprise network performs a series of atomic attacks to reach the desired goal. The execution order and duration of these attack steps depends on several aspects including attacker skill set, type of vulnerability to be exploited, the amount of prior information about the target network, and starting location of the adversary. The adversarial steps range from finding out about the target network via ports scan, running exploits against the vulnerabilities present in the network, removing attack traces, and installing malicious software's such as back doors and Trojan to guarantee easy access to the network at a later date.

There are four basic actions an adversary has to take while performing a computer attack:

1. **Prepare:** During this stage, the adversary collects network configuration information using a port scanner, vulnerability scanners (banner grabbing), and sniffers. The most important piece of information is network hosts IP addresses,

operating systems, and open ports with their associated listening software type and version.

2. **Exploit:** The adversary has to identify a vulnerable piece of software and attempt to exploit it during this stage. Upon successful completion of this phase, adversary gains privileges and information required. An adversary may execute multiple atomic attacks during this phase.
3. **Leave behind:** Once the adversary has obtained the required access level via exploitation, she often installs malicious software to allow easy access to the network in future. Such “leave behinds” might be Trojan software, network sniffers, or additional back-door network services.
4. **Cleanup:** At this stage, the adversary attempts to clean up any attack traces or pieces of evidence left by actions completed in the previous stages. This includes restarting daemons crashed during exploitation, cleaning logs, and other information, and installing modified system software designed to hide the presence of other software from normal system commands such as *ps* and *top*.

The existence and duration of each phase of an attack depends partially on the skill and nature of the adversary. At the bottom of the skill set is the “script kiddie,” a novice computer user who downloads sample exploits from ExploitDB [60] and BugTraq [86] or other vulnerability websites. Then use these exploits and attempt to run them on entire network, without regard to whether the target system is running the vulnerable software. This behavior is usually very “noisy,” meaning that it is readily detectable by most intrusion detection systems and appears in most of the log files on the target computer. Besides, if the exploit fails, it often causes the vulnerable software to crash, leaving a trail of failed attacks that is easily traceable back to the attacker. Script kiddies spend a little time in any stage other than the exploitation stage, as they are more entranced by the “coolness” of “hacking” and have no real intrusion goal other than compromising many systems.

As an attacker gains knowledge, the amount of time spent in preparation and cleanup phases increases considerably. Also, the types of attacks executed begin to favor those that do not crash the target software, leave little trace that the machine was ever attacked, and are invisible to the intrusion detection systems. Finally, the number

of attacks within the exploit phase also increases, as the skilled attacker will often use a series of seemingly low-level attacks to upgrade access levels continually.

Once an adversary advances beyond the realm of the script kiddie, the intruder often has a specific goal in exploiting the network. Whether this goal is to deface a website, obtain pre-release software, or something more nefarious like stealing credit card numbers, the attacks are more likely to specifically target at a certain objective, rather than only trying to get into the network. This goal-directed behavior, coupled with attacker's knowledge, leads to multi-component attack paths that traverse several hosts on the path to the goal state.

Other factors, such as the initial location of the attacker, prior network information, and type of exploits also determine the amount of time spent in each stage of an attack. For example, an attacker that begins an attack within an organization's internal network has access to a much broader range of network information and connectivity, allowing for quick preparation, whereas an outside attacker requires more time to determine the same amount of network information. Similarly, different exploits require more or less cleanup, depending on how noisy the exploit was, and whether it crashed the vulnerable software.

2.2.4 Protecting Against Network Attacks

A plethora of both hardware and software has been developed to combat network attacks at many different levels. Vulnerability scanners attempt to inform an administrator about single-point software vulnerabilities once a system is installed and operational. Intrusion Detection Systems (IDSs) monitor the current system and network behavior and raise an alert when some network packet or host behavior violates certain conditions. Firewalls, both in hardware and software, enforce a given security policy by blocking certain types of network access.

Numerous ways exist to protect against computer attacks. The most obvious solution is to simply fix the problem in the software and eliminate the vulnerability. While this is also the best option, several hurdles exist in changing the vulnerable software. First of all, attempting to understand and correct all of the hundreds of programs that run on all of the computers in a network is often infeasible. Also, not all software packages are distributed with source code, so it is likely that only software vendor has

the required access to change the behavior of the software. Even if software vendors fix all of the problems within their respective software, there is still a window of time (sometimes large, depending on the software vendor) where the unfixed, vulnerable software is being used, and the exploit is known [59]. Once an update is distributed, it is then necessary for the network administrators to find all of the computers that are running the previously vulnerable software and apply the update.

While fixing vulnerabilities in software is the “correct” way to protect against network attacks, the feasibility of using this as the only form of network protection is highly questionable. Other solutions have been developed to protect against software exploits in the face of numerous software vulnerabilities, even those that are still unknown. One trivial and straightforward mechanism is just to deny access to the software from unknown sources. This can be accomplished by totally disconnecting the internal network from any outside networks. As mentioned previously, this solution is inconvenient and often impossible for most organizations to achieve. Firewalls and their derivatives provide a more common way of denying access to particular software. Packet-based firewalls block network traffic based on criteria about network packets, including source (origin) and destination IP addresses, port numbers, and, in the case of more complicated firewalls, protocol type, and packet content. Firewall rules can be set up to block access to software that is running on a particular host-port combination, actually blocking access to that piece of software across the firewall. While these rules are quite useful, correctly setting up and managing a firewall is a difficult task, especially for large internal networks and complex security policies.

To discover potential attack paths in a network, one must not only examine configuration parameters on every network element machines, firewalls, routers, etc. but also consider all possible interactions among them. Conducting this multi-host, multistage vulnerability analysis by human beings is error-prone and labor-intensive. Automating this assessment process is important given the fact that the window between the time a vulnerability is reported to the time it is exploited on a large scale has diminished substantially [87]. Defenders of networks and systems can now plan on having only days to deploy countermeasures in the protection of the vulnerable systems and services that are connected to public networks. To exacerbate the situation, networks being used in organizations are getting bigger and more complex.

In the rest of this Chapter, we provide an overview of how individual host-only vulnerabilities can be identified (2.3) and prioritized (Section 2.4) while designing vulnerability mitigation plan. Next, we take a brief look at how the problem of identifying all plausible multistage, multi-host attacks has been approached in the literature. In particular, there are three main streams of work, directly related to this topic: (i) Penetration Testing, reviewed in Section 2.5, (ii) Attack Trees, reviewed in Section 2.6, and (iii) Attack Graphs, reviewed in Section 2.7. Section 2.8 give an overview of attack graph representations that have been reported in the literature. Finally, in Section 2.9, we review existing attack graph-based metrics proposed in the literature.

2.3 Vulnerability Scanning

Vulnerability scanners attempt to inform network administrator about single-point software vulnerabilities in an operational enterprise network. Well-known vulnerability scanners such as Nessus [88], GFI LanGuard [89], and Retina [90] uses the total number of discovered (scanned) vulnerabilities as the prime indicator of the security risk of a system (or overall network) [91]. For an enterprise network of reasonable size, vulnerability scanner enumerates a large number of vulnerabilities. In practice, patching all vulnerabilities in a network is the mission impossible for the administrator. When there are so many vulnerabilities to fix, an administrator needs to identify those which really matter most in securing the critical resources. One needs to prioritize network vulnerabilities based on their risk level and remediate those that pose the greatest risk [42]. Limitations of current vulnerability scanners are as follows:

- Generate overwhelming amount of data in the form of laundry list of vulnerabilities.
- Do not answer the question “Can an outside attacker obtain access to the mission-critical resources (digital jewels)?”
- If there are so many vulnerabilities to patch, vulnerability scanners do not answer the question “where does a security administrator should start?”
- No indication of how vulnerabilities can be combined.
- Vulnerabilities considered in isolation may seem acceptable risk, but attackers can combine them to produce devastating results.

2.4 Common Vulnerability Scoring Systems

The Common Vulnerability Scoring System (CVSS) [2], [42], [43] provides an open framework for communicating the characteristics and impact of security vulnerabilities [37]. It enables IT managers, application vendors, security vendors, vulnerability bulletin providers, and security researchers to speak a common language of scoring publicly known information security vulnerabilities [43]. Currently, both the CVSS versions (i.e. Version 2.0 [43] and Version 3.0 [2]) are in use and they provide an adaptable, standardized method for vulnerability risk scoring. Rather than categorizing vulnerabilities on a general basis or providing a general model of evaluating security, the CVSS Version 2.0 and Version 3.0 offers relatively accurate metrics for vulnerability risk evaluation. They provide information about vulnerabilities at the operational level and leave stakeholders to add information specific to their own explanation and needs [32]. Further, CVSS provides a crucial information regarding the pre-conditions required for the successful exploitation of a particular vulnerability and also the information about the probable post-condition (i.e. the access level gained by the adversary post vulnerability exploitation). Such information can then be used to construct an attack graph, which shows all possible attack paths in a network.

The Forum of Incident Response and Security Teams (FIRST) has announced CVSS v3.0 [2] wherein a new scoring system including new metrics such as Scope (S) and User Interaction (UI) are introduced. These sub-metrics allows analysts to customize CVSS scores based on the host that has been affected in the organization, making it contextual when required to be. Despite the proposal of such sub-metrics (i.e. User Interaction and Scope in CVSS v3.0), the problem of scoring security risk posed by network vulnerabilities in the context of a multistage, multi-host attack remains an open issue. The CVSS consists of three types of metrics: Base Metric, Temporal Metric, and Environmental Metric as shown in the Figure 2.2. Each of the above metrics is composed of several sub-metrics (explained in [2]).

Various studies examined the validity of CVSS 2.0 [43] from different perspectives such as (i) by examining the distribution of vulnerability severity levels [31], [92] in vulnerability database such as IBM ISS X-Force [36], NVD [37], and Vupen Security [38], (ii) by analyzing the vulnerabilities that are actually exploited in the wild [93], [94], and (iii) by measuring the time required to compromise the system [95].

2.4 Common Vulnerability Scoring Systems

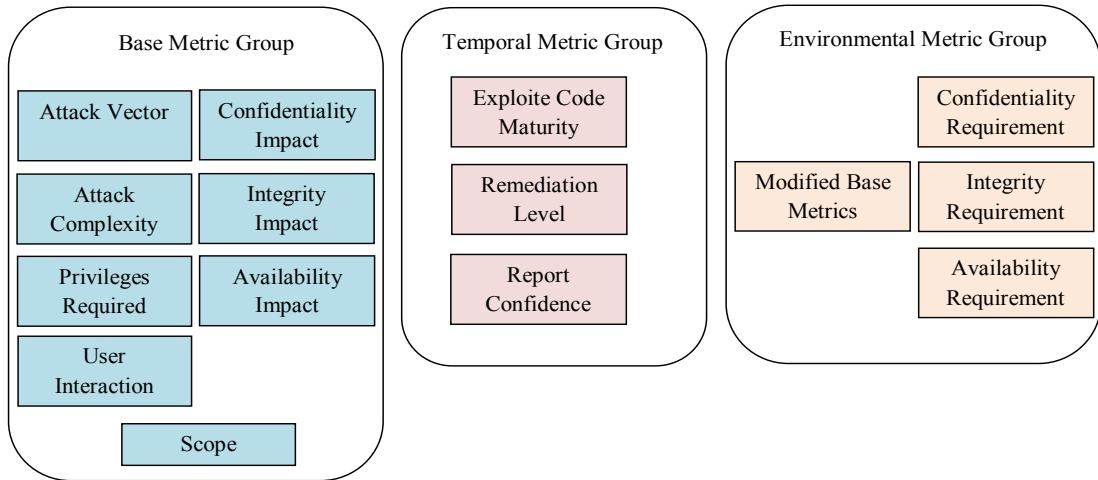


Figure 2.2: CVSS v3.0 metric group (adapted from [2])

However, these studies do not attempt to investigate the concrete reasons behind why their measurements are different from what is expected from the CVSS. Zhao et al. [32] proposed a hybrid ranking approach to estimate the influential level of vulnerabilities in a dynamic environment. These efforts (i.e. [31], [32], [92], [93], [94], [95]) contributed significantly to how the security analysts understand the network vulnerabilities, assign a numerical score to the known vulnerabilities, and how to assess the risk of vulnerability exploitation.

Despite widespread use throughout the security industry, CVSS (both v2.0 [43] and v3.0 [2]) has the following limitations:

- Not all vulnerabilities in a network are exploitable due to the absence of one or more enabling conditions. However, CVSS does not say anything about whether the vulnerability present in the system is exploitable or not. Therefore, in today's resource-constrained environment, patching of temporarily inactive vulnerabilities having higher CVSS score is of no value.
- CVSS scores each of the discovered vulnerability in isolation and does not consider the interdependencies between them. Therefore, in the context of multi-stage, multi-host attacks, estimating security risk based on the individual CVSS score is misleading.
- As CVSS measures the severity (risk) of each network vulnerability in isolation, it does not capture the impact of a particular vulnerability on the overall network

security. In particular, CVSS cannot fully deliver optimal countermeasures for the system under consideration [32].

- Due to the network (or security) events such as system reconfiguration, addition/removal of vulnerable host/services, an introduction of the new vulnerabilities, etc. the security risks posed by the exploitable vulnerabilities varies over time. However, CVSS does not accommodate such change in the risks posed by the vulnerabilities. Further, CVSS is not sensitive to the network security principles such as network partitioning/segregation, and defense-in-depth.
- Even though CVSS Temporal and Environmental Metric have been there; they have nothing to do with day-to-day network dynamics and its impact on the security risks posed by the vulnerabilities.

To summarize: CVSS Base Score is good at modeling the known factors that affect the success of an adversary such as likelihood (or easiness) of an exploit, and its severity. However, when the administrator confirms the vulnerability exposure in a particular network environment, she needs to combine the static CVSS Score with other network parameters (or risk conditions). The security risk posed by the vulnerability is influenced by the various network conditions (discussed in the Section 3.3). Because of above stated limitations (of the CVSS), a vulnerability risk assessment process requires an additional step to evaluate the risk posed by the network vulnerabilities, and this can be done by combining the CVSS Score with the various security parameters (risk conditions) of the underlying network. In Chapter 3, we propose a novel comprehensive metric called improved relative cumulative risk (IRCR) for measuring the security risk of exploitable vulnerabilities in a dynamic network.

2.5 Penetration Testing

Penetration Testing, usually called Pen Testing, is a technique to check security strength [3] of a target network under evaluation, either for quality assurance purposes or certifying compliance with well-established regulations. It is commonly performed by legal security professionals who imitates the adversarys choice of vulnerability exploitation, using the same set of attack tools and techniques, to evade security defense/controls put in place. Targets of assessment can be entire organizational network

or segment of it (i.e. operating zones of users), and the objective is to evaluate the network against the risk of information compromise, theft or espionage. Other targets of Pen Testing can be a host, a network device such as switch or router, a web application, or any other critical resource of particular concern.

Unlike most security assurance methods, Pen Testing is a comprehensive method [3]. In other words, to say that a penetration tester considers network vulnerabilities from diverse domains, for, e.g. the physical, telecommunications (both wireless and networked communication), and humans instead of analyzing a target under one particular aspect [96]. In any organization, Pen testing can be conducted in several ways: they can be executed in a black or white box manner, overt or covert.

Compared to the white box, a black box type of test is more appropriate to simulate attacks from outside of the target organization. In this case, penetration testers have no idea about the systems that they are going to evaluate. They are provided with a minimum, publicly available, set of information about the target under evaluation, and asked to acquire the remaining information needed to launch attacks the same ways as remote adversary do [97]. At the very first stage, pen testers need to determine a way in the network perimeter firewall and then, if successful; they can try to reach, and compromise other hosts within the network [98]. Alternatively, a white box type of pen testing is more relevant for simulating insider threats. In this case, pen testers are provided with the level of information and privileges equal to an employee, and the objective is to determine all plausible attack paths to reach the critical information the organization employee is not authorized to access.

Overt pen test happens under the watchful eye of an organizations IT staff so-called Blue Team and typically focus on specific systems. It is usually performed internally when the IT personnel have full knowledge about the testing [98]. On the contrary, a covert pen test preferred only when the upper management responsible for the initiative has full knowledge about the testing. It is commonly performed by Red Teams from trusted third parties, with or without prior warnings. Each of the two techniques has pros and cons. Red Teams have advantages regarding expertise, speed, and methodology. Besides, separation of duties among who is responsible for the target under evaluation and who actually performs the evaluation is also important [97]. Blue Teams have advantages related to secrecy and cost since outsourced third party pen

testers are expensive. Moreover, they have more knowledge about the target under evaluation and, therefore, tend to be more likely to find additional attack paths.

A pen testing method has four stages and is supported by the Flaw Hypothesis Methodology (FHM) [3], as depicted in Figure 2.3:

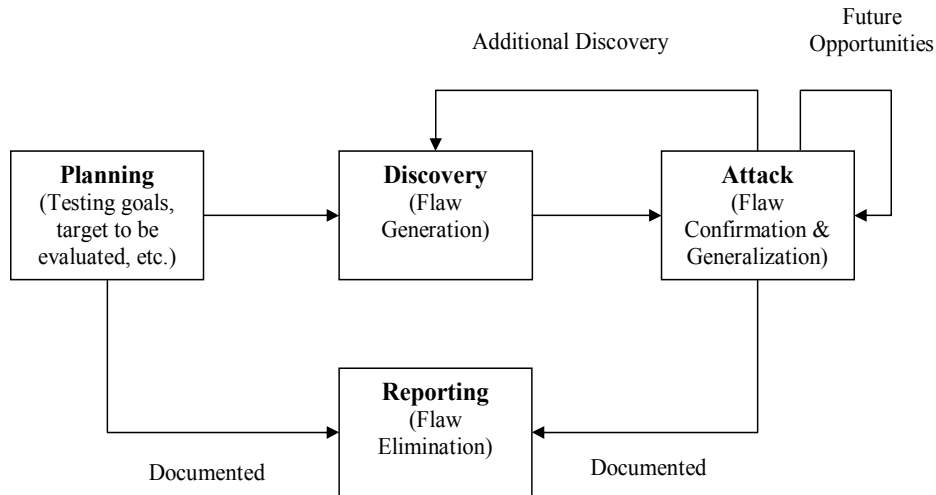


Figure 2.3: Basic penetration test cycle (adapted from [3], [4])

1. A planning phase where the scope of the pen testing, appropriate conditions for completion and attack vector to be considered (e.g. social engineering) should be agreed.
2. A discovery phase for information reconnaissance about the target under assessment and flaw discovery where vulnerabilities are uncovered. This stage assisted by tools and techniques for identification of network topology and configuration, and for vulnerability scanning [99].
3. An attack phase where the penetration tester confirms and validates flaws by exploiting them to assess the risk, they pose if exploited by the real adversary. There is a feedback loop between Attack and Discovery phase since exploitation leads to the discovery of many additional vulnerabilities. Besides, the attack phase itself can also be iterative because successful exploitation of one vulnerability might open opportunities for further exploitations [98]. Exploit kits and detailed instructions about vulnerability exploitations found in specialized fo-

runs like *www.securityfocus.com* [100] and ExploitDB [60] play a major role at this stage.

4. A reporting phase where pen test findings presented and recommendations for flaw elimination are drawn, supported by considerations of the risk posed by the identified flaws. It also documents planning required during entire penetration test cycle.

Essentially, penetration testing is an empirical method which aims at evaluating target security by uncovering flaws in it [3]. Nothing guarantees its completeness and, therefore, it is an assessment exercise which is part of the security governance cycle [7] and should regularly be performed. As with program/application testing, Pen Tests are quite useful in showing the presence of vulnerabilities, but they are unable to guarantee their absence [101]. Pen tests are expensive, time-consuming, labor-intensive, and entirely dependent on the skills and technical expertise of the testers [97]. Additionally, they are constrained by time or budget, unlike the adversary who might, e.g. leave an exploitation process running long after testers budget has expired [102].

2.6 Attack Trees

As discussed in the earlier Section, the Attack Phase (shown in Figure 2.3) is iterative, i.e. the successful exploitation of one vulnerability lays the groundwork for subsequent exploits. Additionally, the Figure shows a feedback loop between an Attack Phase and the Discovery Phase, since the exploitation of vulnerabilities leads to the discovery of many additional vulnerabilities. Both loops bring the determined adversary closer to attaining her goal by violating network security policy. Attack Trees present a structured, top-down way to organize adversary's sub-goals (maybe obtained as a result of Pen Testing) which signify means to accomplish a final goal. Accordingly, the root of the attack tree is an adversary's ultimate goal she wants to achieve, and the leaves are the sub-goals which contribute towards their parent's goal. A successful attack is a path/trace from a leaf node to the root node of the generated attack tree, and as it occurs in any tree data structure by definition, each sub-goal in the attack tree can only have one parent-goal [103].

Attack trees (AT) are in fact AND-OR tree structures used in graphical security modeling. Figure 2.4 depicts a sample attack tree where leaf nodes are alternatives (OR-gates), i.e. only one sub-goal is sufficient to achieve its parent goal unless explicit AND-gates signifies that all sub-goals are needed to accomplish their parent goal.

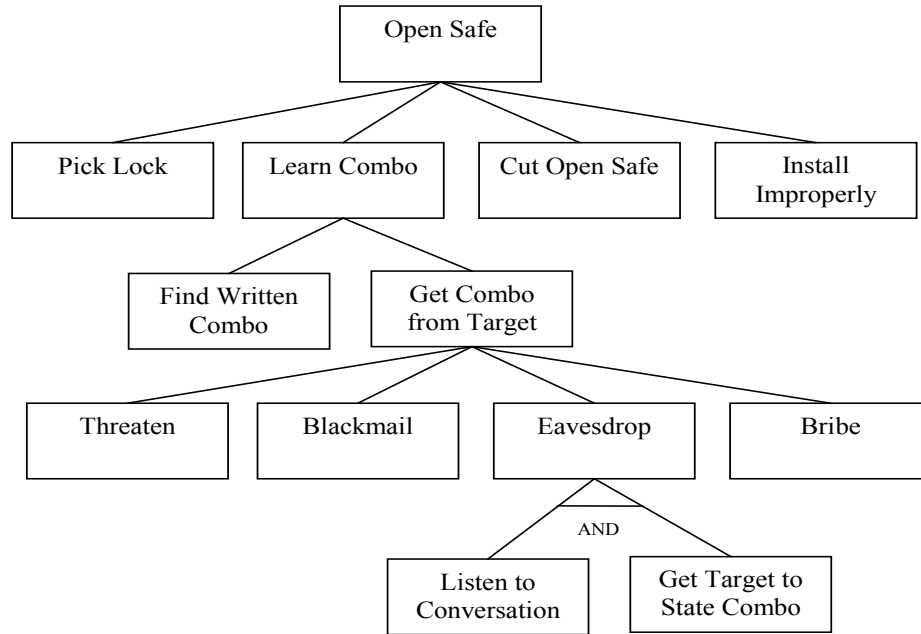


Figure 2.4: An example attack tree (adapted from [5])

Attack Tree (AT) is a variation of Fault Tree (FT) applied in the domain of Information Security by Bruce Schneier [5]. In reliability engineering, the root node of a Fault Tree (FT) signifies a system failure, i.e. an undesired event, and leaves indicates natural causes which contribute to the parent failure, i.e. elementary observable failures. The construction of both AT and FT requires deductive reasoning, i.e. thinking backward looking for actual causes of an incident to be avoided.

Figure 2.5 (on the left) depicts an example Fault Tree which is a graphical representation of the following Boolean Expression.

$$G_0 = VF \cup [(FP_1 \cup EF) \cap (FP_2 \cup EF)]$$

One can analyze Fault Tree both qualitatively and quantitatively. Qualitative analysis involves reducing the FT into an equivalent one containing only minimum cut sets

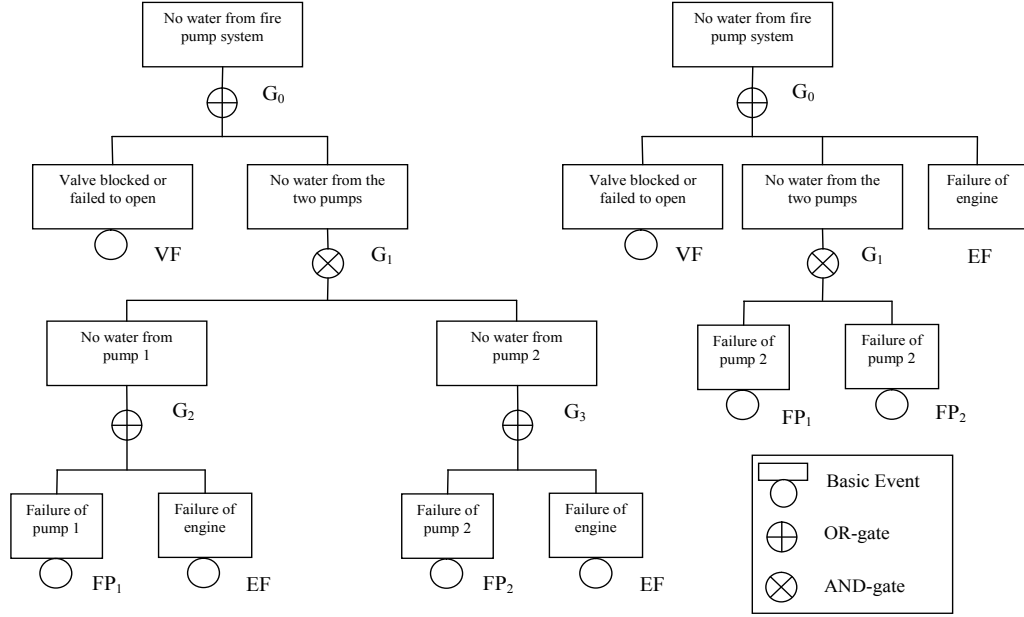


Figure 2.5: An example fault tree, on the left, and its logically equivalent, on the right (adapted from [6])

[104]. By deriving a Boolean expression from the FT, top-down substitutions are possible, reducing the fault tree into an equivalent one containing only minimum cut sets. For example, as shown in Figure 2.5 the reduction of the FT on the left side is shown on the right side and is represented by the Boolean expression:

$$G_0 = VF \cup (FP_1 \cap FP_2) \cup EF$$

Therefore, the minimum cut sets (MCS) of the original fault tree are: $MCS_1 = VF$, $MCS_2 = FP_1 \cap FP_2$ and $MCS_3 = EF$. It means that if any one of the MCS_i happens, the top level event happens. In order to prevent the top level event from happening, all MCS_i needs to be prevented.

In fact, minimum cut sets (MCS's) reveal basic events (leaves) responsible for the top event (root) to occur and do not provide complete paths through the fault tree, from leaves to root. It always works fine for the Dependable Systems; however, it might not work for the system analysis where Security is relevant rather than Safety. Moreover, such analysis also depends on the intended application, for example, Helmer et al. [105] use fault tree to identify and analyze security requirements for intrusion

detection system (IDS). Authors used minimum cut set to decide on “what components of a distributed system (modeled as leaves) must be supervised to uncover the intrusion (modeled as root)” [105].

Furthermore, one can perform quantitative analysis of FT wherein she can determine the probability of the top event (root) to occur, based on given probabilities of the underlying events (i.e. leaves) to happen. Hence, leaves encompass system faults that are observable events associated e.g. with hardware component failures, software errors, human errors, or any other applicable events which can cause the undesired top event [106], but for which failure data should exist [6]. Also, the top event in FT should be explicit (unambiguous) and clear enough to answer 3W like what, where, and when [6]. In the domain of network security, it is quite hard to obtain quantitative data about leaves of an AT but, once available, FT techniques also apply to propagate values to the root of the tree.

In general, Fault Tree Analysis techniques belong to the category of Probabilistic Risk Assessment (PRA). Tree-based PRA methods are all scenario-based approaches commonly used for risk evaluation. Therefore, the effectiveness of the obtained results depends on the accurate identification of significant failure scenarios [107]. In the domain of Dependable Systems, it is quite easier to work in the failure space because usually, a few number of fault trees covers all the critical scenarios [107]. However, in the domain of Network Security, this assumption does not hold always.

In the domain of network security, attack trees are being used heavily to assess security risk, and reason about countermeasures [108]. Attack tree based commercial tools such as SecurelTree [109] from Amenaza Technologies has been proposed for the network security risk assessment and also to reason about the potential countermeasures. Whereas, the Electric Power Research Institute (EPRI), USA [110], [111] makes use of attack trees for modeling selected failure scenarios in the smart grid. SQUARE (System Quality Requirements Engineering) methodology [112], [113] used Attack Trees for providing a high-level picture of the nature of potential attacks on a system.

Since the attack tree based risk assessment techniques are all nothing but scenario-based approaches, the effectiveness of evaluation is highly dependent on the identification of all possible attack scenarios. In the domain of network security wherein the multi-host, multistage attacks are more prominent; it is impossible to cover all attack

scenarios with few number of attack trees. In fact, several potential targets in a network can be reached via different attack paths. Therefore, the spectrum of adversaries goal is usually very high. As a result, attack trees are efficient in capturing particular attack scenario, not the network as a whole. The drawback of using the attack tree for risk assessment is that it contains more subjective nodes, the amount of information it requires is not available in practice, therefore, making it expert-specific, and applicable to only completely known scenarios.

2.7 Attack Graphs

An attack graph is a network security model that can be viewed as a structure which contains numerous attack trees. In particular, in an attack graph (a) one node can have more than one parent, for example, one initial condition (or post-condition) can act as a pre-condition for more than one exploit, (b) multiple attackers and multiple targets (goals) can be represented succinctly, and (c) both inductive reasoning (forward from cause to consequences, i.e. from an attacker's initial position to a target) and deductive reasoning (backward from consequences to the responsible causes, i.e. from an attacker's goal condition to an initial position) are possible. Such semantic of having multiple parents for a node is not possible with the use of attack trees. Figure 2.6 (adapted from [7]) illustrates these observations.

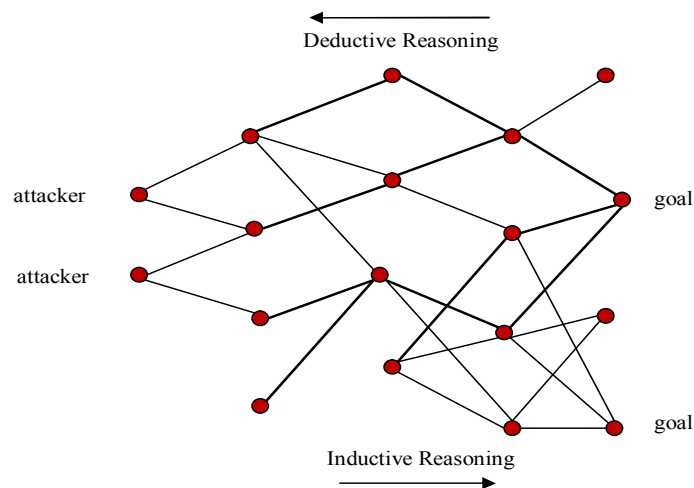


Figure 2.6: An attack graph that contains numerous attack trees (adapted from [7]).

To understand and prevent multistage, multi-host attacks, researchers proposed attack graphs [15], [16], [17], [18], [19], [56], [114]. An attack graph is a formal technique used to uncover potential attack paths in a network which allow adversary to compromise the network security. Primarily, attack graph takes into account network configuration details, vulnerability details, security advisory and succinctly depicts all potential attack paths which allow adversary to reach goals. It imitates the adversary's choice of vulnerability exploitation. Attack graphs possess two main characteristics. First, exploitable network vulnerabilities are always represented. It is motivated by the understanding that "combinations of exploits (vulnerabilities) are the typical means by which an attacker breaks into a network" [15]. Second, and derived from the first, traversing the attack graph provides potential attack paths. Moreover, graphs are a data structure that accommodates dynamics to different extents, such as pre- and postconditions attached to its nodes, very much used in the domain of attack graphs.

2.7.1 Workflow of Attack Graph Construction and Analysis

Attack graphs have been proposed for years as a formal modeling tool to detect complex multistage, multi-host attack scenarios [28]. To obtain the complete understanding of how attack graphs are constructed and utilized, a generic workflow [8] is proposed in the literature (as shown in Figure 2.7). This workflow consists of three dependent phases, namely, (1) *Information Gathering*, (2) *Attack Graph Construction*, and (3) *Attack Graph Visualization & Analysis*. For the construction of an attack graph, some initial information is required: including information about the systems under attack (i.e. information about individual hosts, network connectivity between hosts and the services functioning on these hosts) and the information about the vulnerabilities exploited by an adversary during attack attempts. This input information is separately collected from different sources and then unified in an information-gathering phase. In the second phase, an attack graph is constructed from the gathered input information. Finally, processing of attack graph is done in the visualization and analysis phase. This phase consists of two sub-phases, namely, visualization and analysis, which are independent of each other and need to be repeatedly performed.

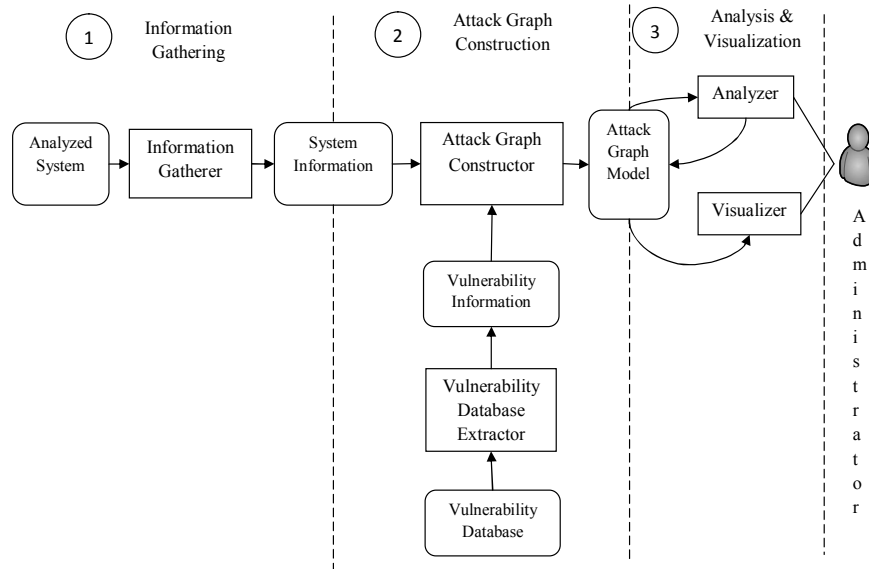


Figure 2.7: Generic work flow of attack graph construction and analysis (adapted from [8])

2.7.2 Advantages of Attack Graphs

Attack graphs yield a number of advantages. Many of them are based on the ability to link single attacks together and form attack paths. Additional advantages result from the visualization possibilities facilitated by the graph data structure. Several of the following examples have been described in greater detail in the seminal work by Oleg Sheyner [20].

The construction of attack graphs allows a number of analyses which are not possible if attacks are regarded as atomic and therefore are not related to one another. A prominent example is the identification of the shortest path an attacker has to take to reach the goal of an attack. Having the network structure and existing vulnerabilities on hosts available, it is possible to compute the minimal number and the kind of steps required to fulfill the attackers objectives. This knowledge can be beneficial to both parties. An attacker would greatly profit, because the necessary effort can be reduced to a minimum. But the defenders benefits are important as well. With the gained knowledge protective measurements can be conducted, such as the deployment of an intrusion detection system along this path. At other times, not the shortest, but the path of least resistance is more interesting. It describes the attacks which cause the

minimum attention, e.g. by not crashing systems or triggering alerts from firewalls. Sometimes pivotal points of a network can be identified, hosts which any attack would have to pass. Securing these hosts can greatly increase the chance of breaking any attack to a high-value asset located in a network. These studies can also be used in the design phase of a computer network, because they allow a benefits-costs analysis of different approaches to defend the network before cost-intensive field tests have to be conducted. Another use case for attack graphs is the correlation of ostensible 'uninteresting' events logged by an intrusion detection system. With the knowledge of possible attack paths, these events can be linked and may form two a series of steps on an attack path.

The visualization of attack graphs also benefits from the graph data structure properties. It is known that the graphical representation of complex issues can increase the understanding and reduce the time to identify points of interest. Another advantage is the possibility to abstract from the overwhelming amount data by identifying connected components in graphs. For example similar hosts, that is hosts with the same configuration in the same section of a network, can be grouped into a unique instance. This can lead to an reduction of hundreds of hosts to a single visual node. Last but not least, novice system users are able to understand the system and threats it is exposed to much faster.

2.7.3 Limitations of Attack Graphs

1. Current attack graph analysis is based completely on known vulnerability information (comprehensive vulnerability database).
2. Attack graphs does not model availability attacks such as Denial of Service (DoS) and Distributed Denial of Service (DDoS)
3. Attack graph cannot effectively address passive attacks such as phishing.
4. Attack graphs cannot model zero-day vulnerabilities and hence zero-day attacks.

2.8 Attack Graph Representations

An attack graph is a collection of attack paths that are composed of conditions, exploits, or some combination of conditions and exploits. Based on these possible combinations,

we classify the attack graphs into three broad categories: condition-oriented attack graphs, exploit-oriented attack graphs, and condition-exploit-oriented attack graphs. It is possible to convert one attack graph representation into another representation when all conditions and exploits are known.

2.8.1 Condition-oriented Attack Graph

In a condition-oriented attack graph, a node represents a subset of the network state, and an edge represents an exploit (or group of exploits) that moves the network from one state to another state. An exploit is a realized vulnerability. A vulnerability specifies only its preconditions and consequences. An exploit details the specific network machines or software involved in realizing a vulnerability. A state is a network attribute or a set of network attributes. Network attributes include hosts, host connectivity, available software at hosts, access rights at hosts, and any other network characteristic deemed relevant to the modeler. A state may be represented at various abstraction levels. Abstraction levels include the following: a single condition [15], a host [115], a host and privilege level [116], groups of hosts (e.g., subnetwork) [47], and the entire network [19]. One or more predicates represent each abstraction level. In condition-oriented attack graphs, predicates are used to describe vulnerabilities. That is, their preconditions and postconditions describe vulnerabilities. When the prerequisites for a vulnerability are satisfied, exploits cause more conditions (i.e., postconditions) to become true. These postconditions become available as preconditions for other vulnerabilities.

2.8.2 Exploit-oriented Attack Graph

An exploit-oriented attack graph is the reverse of a condition-oriented graph with respect to the nodes and edges. A state is represented by the edges of the graph, and the exploits are represented by the nodes of the graph [29]. Exploit-oriented attack graphs may be referred to as exploit dependency graphs. A typical representation of exploit-oriented attack graphs is to have unlabeled edges. The exploit-oriented attack graphs initial state(s) and the goal state(s) of the network are special nodes. Initial states are the exploit nodes with null preconditions and true postconditions. The goal states are the exploit nodes with true preconditions and null postconditions.

2.8.3 Condition-exploit-oriented Attack Graph

In a condition-exploit-oriented attack graph, state and exploits are represented by nodes [29]. An edge may relate a state and an exploit, or an exploit and a state. An edge may not connect a state and another state directly or link an exploit and another exploit directly. When a state precedes an exploit in the attack graph, it is considered a precondition for the exploit. When a state follows an exploit in the attack graph, it is regarded as a postcondition of the exploit.

2.8.4 Multiple Prerequisites Attack Graph

The multiple prerequisites attack graph is an attack graph where there are three types of nodes: states, prerequisites, and vulnerabilities [114]. State nodes represent the host and the access level obtained by the attacker. Prerequisite nodes represent the preconditions required for the attacker to realize a vulnerability, which is represented by a vulnerability node. The Multiple Prerequisite attack graph may be used with or without goal-orientation. This attack graph has been shown to have efficient run times in practice [114].

2.8.5 Logical Attack Graph

The logical attack graph [16], [21] is a goal-oriented attack graph that has two types of nodes: fact nodes and derivative nodes. Also, there are two kinds of fact nodes: primitive fact nodes and derivative fact nodes. Primitive fact nodes have no preconditions and are unconditionally true (facts). Derivative fact nodes have preconditions. However, derivative fact nodes are not directly connected to primitive fact nodes. Derivative fact nodes connect directly to derivative nodes. Derivative nodes connect directly to primitive fact nodes. The set of primitive fact nodes making a derivative node true form a conjunction. The set of derivative nodes that make a derivative fact node true form a disjunction. Also, because edges represent the “depends on” relation, an edge appears between two nodes if the source node requires the destination node to be true in order for the source node to be realized. Thus, the source node “depends” on the destination node. The logical attack graph has been shown to have efficient run times for practical use [16], [21].

2.8.6 Hybrid-oriented Attack Graph

The hybrid-oriented attack graph is described in [29]. A node represents a single condition. When multiple preconditions precede an exploit, the conjunction of these preconditions is required to realize the exploit. When state follows an exploit in the attack graph, it is considered a postcondition of the exploit. If an exploit provides multiple postconditions, it may provide any one postcondition; that is, exploits provide a disjunction of postconditions. This attack graph may or may not be used in a goal-oriented manner. The hybrid-oriented attack graph is expected to scale to hundreds of hosts [4].

2.8.7 Resource Graph

The resource graph [30], [46] is syntactically equivalent to an attack graph. The resource graph models causal relationships between network resources rather than vulnerabilities. It is also called as zero-day attack graph. For generating a resource graph, all resources in a network (i.e. services) are assumed to be vulnerable to zero-day attacks. In other words, each service running over the network believed to have potential zero-day vulnerabilities instead of known reported vulnerabilities. Vertices in the resource graph are zero-day exploits, their pre- and post-conditions. There is an AND dependency between the pre-conditions and OR dependency between exploits. A goal-oriented resource graph (zero-day attack graph) generated for the given network.

2.9 Network Security Hardening using Attack Graphs

The most important problem in network security management is to uncover potential multi-host, multistage attack scenarios due to software vulnerabilities and software (or hardware) misconfiguration. Attack graph provides a concise/succinct way of representing/displaying all possible sequence of attacks (multistage attack scenarios) that an adversary (malicious user) can execute to obtain his/her desired goal such as remotely achieving root undetected on a critical host machine. Attack graph shows/reveals non-obvious security problems in a realistic sample network. Attack graph can be used in combination with other network/host security components to secure the target networks and make them immune to the possible attacks. Attack graph aid an administrator in planning a secure network.

2.9 Network Security Hardening using Attack Graphs

The attack graphs and their associated statistics, such as number of hosts compromised and attacker privilege levels, allow a network administrator to determine likely intrusion paths and extrapolate this data to determine the current and future security of the network given past software vulnerability frequencies. As the attack graphs are displayed in near real-time, an administrator can change the network topology slightly, re-compute the graphs for the new topology, and compare the graphs produced from different configurations. This allows an administrator to weigh network security against other factors, such as hardware costs and ease of maintenance.

Attack graph analysis that extracts security-relevant information from the attack graph is referred to as attack graph-based security metric. As risk assessment is an essential factor in human decision making, attack graph-based metrics are useful to security managers for allocating scarce security resources efficiently and thereby achieving best possible network hardening. Essentially, network hardening is the process of securely configuring network devices (e.g. end user devices, servers, and infrastructure devices such as routers, firewalls, etc.) for reducing network attack surface [17], [117].

In this thesis, our focus is on deriving network hardening recommendations from the generated attack graphs. In the following subsections, we review literature related to recent directions in attack graph-based network hardening, including vulnerability risk assessment, network risk assessment, and attack surface change assessment. We discuss why existing work on attack graph-based metrics is not satisfactory for the problems addressed in the thesis.

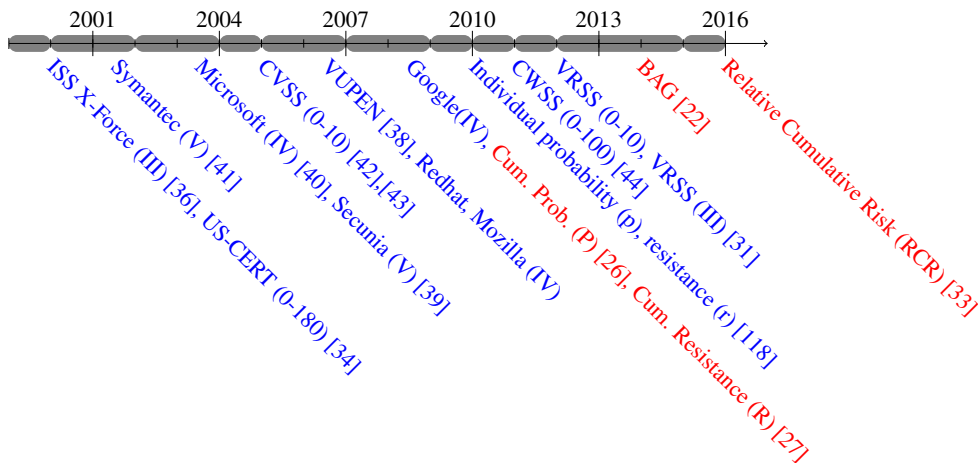
2.9.1 Vulnerability Risk Assessment

To enhance the security of a given system, it is crucial for the security managers to evaluate existing vulnerabilities for their risks since vulnerabilities are the key to all network intrusions. Large number of vulnerability risk assessment methods were developed by security vendors and non-profit organizations such as US-CERT [34], SANS [35], ISS X-FORCE, National Vulnerability Database (NVD) [37], Vupen Security [38], Secunia [39], Microsoft [40], and Symantec [41]. All different vulnerability risk assessment systems are listed on the timeline, as shown in Figure 2.8. Standardization efforts on security metrics, such as CVSS [2] and CWSS [44] focus on ranking well-known vulnerabilities and software weaknesses, respectively. Such scoring systems

2.9 Network Security Hardening using Attack Graphs

help administrators in measuring the severity and impact of individual vulnerabilities, and hence in the process of patch management.

Figure 2.8: History of vulnerability evaluation methods



Although popular [119], [120], both CVSS and CWSS do not capture the interdependency (cause-consequence relationship) between vulnerabilities and hence are deemed insufficient in the context of multistage attacks. Furthermore, CVSS and CWSS measures the severity of a particular software vulnerability/weakness in isolation and hence do not capture their overall impact on the security risk of a network [30]. Consequently, the use of such risk scoring systems often results in an imprecise risk prioritization and sub-optimal security countermeasures.

Therefore, a vulnerability risk assessment process requires an additional step to evaluate the risk posed by the network vulnerabilities. It can be achieved by combining the CVSS Base Score of vulnerabilities with the various network risk conditions.

Existing attack graph-based security metrics such as cumulative probability (P) [26], and cumulative resistance (R) [27] consider the interdependency between the exploitable vulnerabilities for assessing the security risk posed by each of the exploitable vulnerability uncovered in a given network. However, they do not consider the vulnerability diversity along the attack paths as a risk condition. Chen et. al. [45] used diversity among the network vulnerabilities as an important risk condition to assess the security risk of a network. Whereas, Wang et. al. [30], [46] used vulnerability diversity along the attack path to measure the robustness of a system against the zero-day attacks.

Suh-Lee and Jo [33] used the proximity of un-trusted network and risk of neighboring hosts as important risk conditions to assess the security risk of each vulnerability in a given system. However, they do not consider critical network risk conditions such as the cause-consequence relationship between vulnerabilities and exploit diversity along the attack paths.

Work of Wang et al. [30], [46], Chen et al. [45], and Suh-Lee and Jo [33] motivated us to consider various network risk conditions for vulnerability risk assessment. Such risk conditions include, but are not limited to (i) the proximity of the target vulnerability from the attacker's initial position, (ii) vulnerability diversity along the attack path(s), (iii) risk of the neighboring vulnerabilities (here immediate predecessor vulnerabilities in the context of attack graphs) from where the target vulnerability can be reached and exploited, etc. Such risk conditions affect the success of an adversary and hence the risks posed by the network vulnerabilities.

2.9.2 Network Security Risk Assessment

Network security risk assessment and mitigation is imperative for the protection and maintenance of today's critical infrastructure. In particular, risk assessment techniques help security managers in deciding appropriate countermeasures to deter any plausible attack. Many approaches for attack graph analysis [15], [16], [114], [121], have been proposed to determine all potential attack paths and the probability (P) [26], and resistance (R) [27], [122] of the attack paths being realized are used to estimate the network security risk. A large number of metrics based on the attack graph constructs such as graph size, connectivity [123], attack path length [25], [47], [48], etc. were proposed for assessing the security strength of a given network. All different attack graph-based network risk assessment techniques are listed on the timeline, as shown in Figure 2.9.

Pamula et al. [50] proposed security risk estimation methods based on the weakest link principle. The authors assessed the strength of a given network in terms of the minimum number required initial conditions an adversary should possess to compromise the network successfully. Tupper and Heywood [126] proposed a VEA-bility metric to measure the security strength of different potential network configurations. The vulnerability and exploitability dimensions of VEA-bility metric are the function of CVSS sub-metrics. Chen et al. [45] considered the length of all potential attack

2.9 Network Security Hardening using Attack Graphs

Figure 2.9: History of attack graph-based risk assessment methods

1998	•	Shortest path distance metric [47]
1999	•	Number of attack paths [48]
2006	•	Mean of path lengths [49]
2006	•	Weakest Adversary [50]
2006	•	Network Compromise Percentage [124]
2007	•	Cumulative Resistance (R) [27]
2008	•	Cumulative Probability (P) [26]
2012	•	Suite of attack path length-based metrics [125], [25]
2014	•	Metrics suite network attack graph analytics [123]
2014	•	<i>k-zero day safety metric</i> [46], [30]

paths and the number of different kinds of exploitable vulnerabilities in the network for measuring the network security risk.

Keramati et al. [127], [128] proposed a network security metric which in turn considers the impact and exploitability of all potential attack scenarios. Based on the recommendation of the prioritization algorithm, the subset of initial conditions and vulnerabilities of the most significant attack paths is chosen for removal by implementing the required countermeasures at minimum cost. Alhomidi and Reed [129] used a genetic algorithm to determine the optimal attack path likely to be taken by an adversary. Network hardening solutions in terms of a minimal critical set of exploits are proposed in [19], [15], [117], and [130]. Yigit et al. [131] calculated the risk of an entire network by considering the success probabilities of all the attack paths. In order to effectively harden the network, the above stated approaches (i.e. [45], [127], [128], [131]) consider the effective cost of the removal of each exploit and initial condition. Bhattacharya and Ghosh [132] proposed an analytical framework for measuring the network security risk by taking into account the cost of successfully exploiting each vulnerability in all potential attack scenarios. Dai et al. [133] proposed a risk flow graph (RFG) based approach to determine the security risk posed by the adversary to the critical enterprise resources. Poolsappasit et al. [22], Kundu and Ghosh [23] formulated the problem of effective network immunization as a multi-objective optimization problem (MOOP).

Attack graph-based security metric provides useful information that can be acted upon and trigger appropriate action to deter potential multistage attacks [134, 135]. All the benefits of existing metrics become a potential weakness when the more secure network configuration is equally susceptible to the zero-day attacks. In general, unknown (zero-day) vulnerabilities are considered immeasurable due to the less predictable nature of software errors. Therefore, the research on security metrics has been hampered by difficulties in handling the zero-day attacks wherein an adversary exploits multiple zero-day vulnerabilities. It questions the usefulness of the existing security metrics because a more secure network configuration would be of little value if it is equally vulnerable to zero-day attacks. Wang et al. [30], [46] addressed the shortcoming of existing security metrics. They proposed k -zero day safety metric which essentially counts the number of different zero-day vulnerabilities to be exploited to compromise the target resource. Larger the count, more secured the network is since the likelihood of having the unknown vulnerabilities available, applicable, and exploitable at all the same time is significantly lower. Based on the k -zero day safety metric, Wang et al. [9] proposed least attacking effort-based diversity metric to measure network's capability in resisting intrusions or malware infection that employ multiple zero-day attacks. The metric is capable of measuring the robustness of the enterprise network against the zero-day attacks.

These recent advances (i.e. [9], [30], [46]) demonstrate the promise of increasing robustness of computer systems in the face of potential zero-day attacks through efficient diversification of the vulnerable services. Therefore, additional metrics for determining the diversification level of each attack paths would undoubtedly benefit such systems. Further, algorithms for the identification and diversification of the repeated services along the attack paths could enhance the usefulness of such approaches.

2.9.3 Network Attack Surface Change Assessment

With frequent changes in the network configuration, today's computer networks undergo continuous evolution. Such ever-changing (dynamic) computer networks have varying attack surface. Constant discovery of new vulnerabilities, misconfiguration of hardware (or software) components, for example, badly installed firewalls, loose

access control policies, etc., can further intensify change in the network attack surface. Therefore, there is a pressing need to consider the temporal aspects of security while monitoring network security performance. According to the standard guidelines and recommendations issued by ISO/IEC27005 [51] and ENISA [52], for maintaining the best possible security posture of a given computer network, it has to be regularly monitored for network security policy violations.

2.9.3.1 Attack Surface

Howard et al. [136], [137] introduced the first informal notion of the attack surface. They measured how likely the Windows operating system is vulnerable to attacks based on their degree of exposure. A similar study was conducted by [138], for measuring the attack surface of Linux-based operating systems proved that the notion of an attack surface is promising while comparing two systems in terms of their security. In a follow-up work, Manadhata and Wing [139] established a generalized formal notion of software attack surface based on system's entry point and exit point framework which in turn identifies the relevant resources that contribute to the vulnerability exposure. Essentially, the above stated attack surface measurement techniques (i.e. [137], [138], [139]) are capable of measuring attack surface of a single software under consideration.

2.9.3.2 Network Attack Surface

Wang et al. [30] applied the notion of attack surface to the complete network of computer systems. Instead of focusing on the inner details of local services and applications, their focus is on the interfaces like remotely exploitable services. The system they considered is an entire computer network and respective attack surface is termed as a *network attack surface*. Sun and Jajodia [140] defined system's attack surface as a set of ways by which an adversary can enter into the system and compromise the security of a system. According to Cybenko et al. [71], network attack surface is the network channel for attackers to connect to the system and invoke the system's functionality. Such channels consist of vulnerable network configurations and existing exploitable vulnerabilities.

Cowley et al. [141] identified network topology factors such as network partitioning, and network reachability/service connectivity as an important network characteris-

tics (components) for measuring network security risk. Essentially, vulnerable service connectivities and existing vulnerabilities (well-known vulnerabilities, in particular) forms the preconditions for the incremental exploitation of remote vulnerabilities and hence acts as a basis for multistage attacks in the target network. Informally, service connectivities and vulnerabilities are the network resources used by an adversary against the network itself. In practice, not all vulnerable service connectivities, and vulnerabilities contribute equally during the network intrusion and hence we chose only those which are likely to be used by an adversary while compromising a particular host in the target network. Similar to Cybenko et al. [71], our notion of network attack surface constitutes a subset of network resources that are likely to be used by an adversary against the network itself.

Bunke et al. [142] described the application of many graph-theoretic algorithms for the analysis of dynamic enterprise networks. Showbridge et al. [143] used error-correcting graph matching (ECGM) based edit distance for monitoring the performance of telecommunication networks. Liao and Striegel [144] proposed a graph differential anomaly visualization (DAV) model in the area of network management to identify the meaningful changes and hidden anomalous activities. Awan et al. [145] proposed a framework for measuring the temporal variance in computer network risk. They used system log data collected from the university campus network to validate their framework.

Existing approaches of attack graph-based network security analysis [15], [16], [17], [18], [19], [47], [48], [117], [146], treats the monitored network as a relatively static, and takes a snapshot of the network configuration at a particular point in time. None of the previously proposed attack graph-based metric has been designed (attempt) to measure the temporal variation in the network attack surface. Therefore, proper security metrics should be there to detect a change in the network attack surface to identify problems early so that corrective actions can be taken.

2.9.4 Discerning Temporal Variations in the Network Attack Surface

Essentially, the generated attack graphs need to be processed for better visualization to depict security relevant information in the form of critical vulnerabilities, associated

pre-conditions and attack paths that are vital to the success of an adversary [134], [135], [147]. With a view to increasing attack graph usability significant efforts have been made from both academia and industry. Mehta et al. [148], Sawilla and Ou [149] proposed an attack graph ranking algorithms for reducing the attack graph complexity. Williams et al. [150], and Homer et al. [151] proposed novel visualization techniques for improving attack graph understandability. An incremental attack graph generation algorithm is proposed by Saha [152] to improve attack graph adaptability. Moreover, a number of tools for an attack graph generation were developed; examples include MulVAL [21], NetSPA [153], and CAULDRON [17].

Even though it is possible to efficiently generate attack graphs for a realistic network, resulting graphs poses a serious challenge for human comprehension. It necessitates an efficient visualization technique for depicting the newly introduced changes in the network attack surface that are not so obvious even with the effective attack graph visualization techniques. Therefore, techniques should be there to discern variations in the network attack surface so that portions of the attack graph that significantly changed can be inferred. The technique should identify the newly introduced exploits and their respective enabling conditions in the dynamic network. Quick identification of modification in the network attack surface and hidden root causes is crucial to the prevention of future attacks.

2.10 Attack Graphs used in this Thesis

In this thesis we use exploit-oriented (exploit-dependency) attack graph [29], resource graph (zero-day attack graph) [30], [46] and logical attack graph [16], [21]. We use different attack graph representations based on what representation best helps convey our aim. If we do not explicitly state the representation being used, the representation will be clear from the context.

2.11 Summary

In this Chapter we have seen three streams of research directly related to the problem of finding potential multistage, multi-host network attacks. Here we provide a

helicopter view of the conclusions we derive from this literature review.

Penetration Testing

Advantages	Disadvantages
<ul style="list-style-type: none">• Advantageous, if performed regularly.• Efficient in identifying potential attack paths• Correct in paths reported.	<ul style="list-style-type: none">• Expensive• Labor-intensives• Entirely dependent on the skills and technical competence of pen testers• Constrained by time and budget

Attack Trees

Advantages	Disadvantages
<ul style="list-style-type: none">• Structured top-down approach to organize means to achieve an attack goal (root of the tree)• Suitable for brainstorming specific scenarios• Support deductive reasoning (backward from consequences to the responsible causes, i.e. from an attackers goal condition to an initial conditions)	<ul style="list-style-type: none">• Contain more subjective nodes.• Highly dependent on the identification of all possible attack scenarios• Do not scale when numerous attack scenarios may arise• A path from each leaf to the root represents an attack path. However, in reality from a current location (e.g. a leaf) an adversary may have more than one paths to follow; this is not possible to represent with attack trees since, by definition, each node in a tree structure has an unique parent node.

Attack Graphs

Advantages

- Allows inductive and deductive reasoning about potential multistage, multi-host attacks
- Allows representing several possible goals (targets), and several initial locations
- Exploitable vulnerabilities are always represented
- Support what-if analysis
- Capture network attack surface

Disadvantages

- Graphs become too large to comprehend for reasonable size networks
- Non-vulnerable hosts are not the focus of attack graphs
- Do not represent attack and network dynamics
- Does not model Dos, DDoS, and phishing kind of attacks

Further, we have reviewed the literature related to recent directions in attack graph-based network hardening, including vulnerability risk assessment, network risk assessment, and attack surface change assessment. Here, we briefly enlist motivating problems that lead to our proposals.

1. Although researchers have proposed a significant number of metrics for vulnerability risk assessment, there has been no comprehensive measure which considers critical network risk conditions that affect the success of an adversary. Therefore, there is a pressing need of comprehensive measure that should consider the necessary network risk conditions while assessing the security risk of each exploitable vulnerability in a dynamic network.
2. Despite the proposal of a vast number of attack graph-based security metrics, a well-administered network is susceptible to the zero-day attacks. It questions the usefulness of the existing security metrics. It is because a more secure network configuration would be of little value if it is equally vulnerable to zero-day attacks. Few studies have demonstrated that through adequate diversification of the vulnerable services, an administrator could increase the network robustness against zero-day attacks. Therefore, additional metrics for determining the diversification level of each attack path would benefit such systems.

3. Finally, despite the proposal of many attack graph-based security metrics, there has been no work on assessing the temporal variation in the attack surface of dynamic networks. Therefore, there should be a metric to capture the temporal variations in the network attack surface. Furthermore, there should be a technique to effectively visualize newly introduced changes in the attack surface.

Chapter 3

A Proximity-based Approach for Quantifying the Security Risk of Vulnerabilities

“Count what is countable, measure what is measurable, and what is not measurable, make measurable.” Galileo Galilei

This Chapter presents a proximity-based measure for assessing the security risk posed by network vulnerabilities in the face of potential multistage, multi-host attacks. We propose a comprehensive metric called Improved Relative Cumulative Risk (IRCR) for risk estimation of each vulnerability through the adjustment of individual static CVSS score [2] using various network risk conditions. We consider an exploit-dependency attack graph [56] as a network security model in conjunction with the CVSS framework [2]. Attack graph for a given system captures critical risk conditions that affect the success of an adversary. These risk conditions include, but not limited to, the proximity of an exploitable vulnerability relative to the attackers initial position, exploit diversity along the attack path(s), the number of vulnerabilities (predecessors) from where the vulnerability under consideration could be directly reached and exploited, and conjunction/disjunction between predecessors. We find that the IRCR is complementary with state-of-the-art vulnerability risk scoring techniques [26], [27] on synthetic networks. Based on the IRCR recommendations, an administrator can accurately determine top vulnerabilities and prioritize vulnerability remediation activities

accordingly. Moreover, IRCR is adaptive since it automatically adjusts according to the network (or security) events.

3.1 Introduction

The purpose of vulnerability analysis and risk assessment techniques is to uncover vulnerabilities in the enterprise infrastructure and mitigate them before the security attack becomes a reality. Risk assessment techniques help security administrators in deciding appropriate countermeasures to deter any plausible attack. Traditional information security planning and management process begins with vulnerability scanning, followed by the risk assessment, and finally proactive network hardening. Estimating the risk of vulnerabilities in today's network environment has become a severe problem, primarily, because Cyber attacks have become more sophisticated wherein adversary combines multiple vulnerabilities to compromise the critical resources incrementally. Well-known vulnerability scanners like Nessus [88], GFI LanGuard [89], and Retina [90]; identify vulnerabilities in isolation, and does not capture interdependency between network vulnerabilities. However, such dependency is at the heart of ever-increasing multistage, multi-host attacks [22], [30], [56], [128], [154], [155], [156]. Further, for an enterprise network of reasonable size, vulnerability scanner enumerates a large number of vulnerabilities. In practice, patching all vulnerabilities in a given network is the mission impossible for the administrator.

When there are so many vulnerabilities to fix, an administrator needs to identify those matter most in securing the critical enterprise resources. She needs to prioritize network vulnerabilities based on their risk level and remediate those that pose the greatest risk [42]. She can mitigate the security risk by patching the top-ranked vulnerabilities. To achieve this, she needs to create the security mitigation plan from the top-ranked vulnerabilities by considering their respective security controls.

3.2 Motivation

In the context of multistage, multi-host attacks, the security risk posed by a given exploitable vulnerability depends on several network risk conditions such as

- Proximity of the target vulnerability from the attackers initial position,
- Vulnerability diversity along the attack path(s),
- The number of vulnerabilities from where the vulnerability under consideration could be directly reached and exploited.
- Conjunction/disjunction between neighboring vulnerabilities (i.e. predecessors)

Such risk conditions affect the success of an adversary and hence influence the risk posed by the vulnerability. Although a large number of metrics [2], [27], [26], [32], [31], [33] have been proposed for vulnerability risk assessment, there has been no comprehensive measure which considers the critical risk conditions mentioned above.

Standardization efforts on security metrics, such as [2], CVSS [42], [43] and CWSS [44] focus on ranking well-known vulnerabilities and software weaknesses, respectively. Such scoring systems help administrators in measuring the severity and impact of individual vulnerabilities, and hence in the process of patch management. Although popular, both CVSS and CWSS do not capture the interdependency (cause-consequence relationship) between vulnerabilities and hence are deemed insufficient in the context of multistage, multi-host attacks.

Existing attack graph-based security metrics such as cumulative probability (P) [26], and cumulative resistance (R) [27] consider the interdependency between the exploitable vulnerabilities for assessing the security risk posed by each of the exploitable vulnerability uncovered in a given network. However, they do not consider the vulnerability diversity along the attack paths. Wang et. al. [30], [46] used vulnerability diversity along the attack paths to measure the robustness of a system against the zero-day attacks. Suh-Lee and Jo [33] used the proximity of un-trusted network and risk of neighboring hosts as important risk conditions to assess the security risk of each vulnerability in a given system. However, they do not consider critical risk conditions such as the cause-consequence relationship between vulnerabilities and exploit diversity along the attack paths. Work of Wang et al. [30], [46], Chen et al. [45], Wang et al. [27], [26] and Suh-Lee and Jo [33] motivated us to consider various network risk conditions for vulnerability risk assessment. Our objective is to combine the static risk level of the network vulnerability (i.e. CVSS Base Score) with various risk conditions peculiar to the target network.

3.3 Risk Conditions Pertaining to the Network Security

To evaluate the security risk of the exploitable vulnerabilities, a suitable model that considers various network risk conditions, and which imitates the adversary's choice of vulnerability exploitation, should be utilized. There are a plethora of research work [157] on the efficient generation and ranking of the potential attack paths in the network; to model and analyze multistage, multi-host attack scenarios to provide high-level metrics for network risk assessment. For our purpose of combining the standard CVSS Base Score [2] with the network specific risk conditions, we make use of an exploit-dependency attack graph [56], which is in general space efficient and more expressive. It imitates the adversary's choice of vulnerability exploitation and successfully captures various risk conditions that affect the success of an adversary and hence influence the risk posed by the vulnerability.

The network risk conditions that are used in our proximity-based to vulnerability risk scoring to augment the static risk level (i.e. CVSS Base Score [2]) of each exploitable vulnerability in a given network are as follows:

3.3.1 Attack Path Resistance

In order to reach and exploit a particular vulnerability v in a given network, an adversary may have one or more attack paths available. According to the Wang et al. [27] network security should be measured as the smallest effort required to reach the goal. Phillips and Laura [47] proposed SP distance metric which signifies the minimum number of hurdles (here, vulnerabilities) along the attack path, an adversary has to exploit to reach and take advantage of the target vulnerability v successfully. Author's intuition behind devising shortest path (SP) metric is that: farther away the vulnerability is from the attacker's initial position, the more effort attacker has to spend to reach to the vulnerability and exploit it. The longer distance implies an adversary should have the greater endurance to reach the vulnerability v and hence lower will be the probability of vulnerability exploitation.

However, the idea of shortest attack path is misleading as it treats each type of vulnerability along the attack path(s) equally and does not capture attackers effort. Since

3.3 Risk Conditions Pertaining to the Network Security

each type of vulnerability poses different amount of resistance, an attacker has to spend different amount of effort while exploiting vulnerabilities of each kind. Therefore, the vulnerability resistance along the attack paths is the correct measure of attackers effort compared to the attack path length. The intuition behind using the least resistance path is that, given the option of different available attack paths which are reachable from the attacker's initial position to the vulnerability under consideration, an attacker will choose the path which poses least resistance [158]. Least resistance path assumes that the attacker is interested in using less effort to reach and exploit the vulnerability v . Therefore, the minimum resistance path to a targeted vulnerability is considered as an important risk condition.

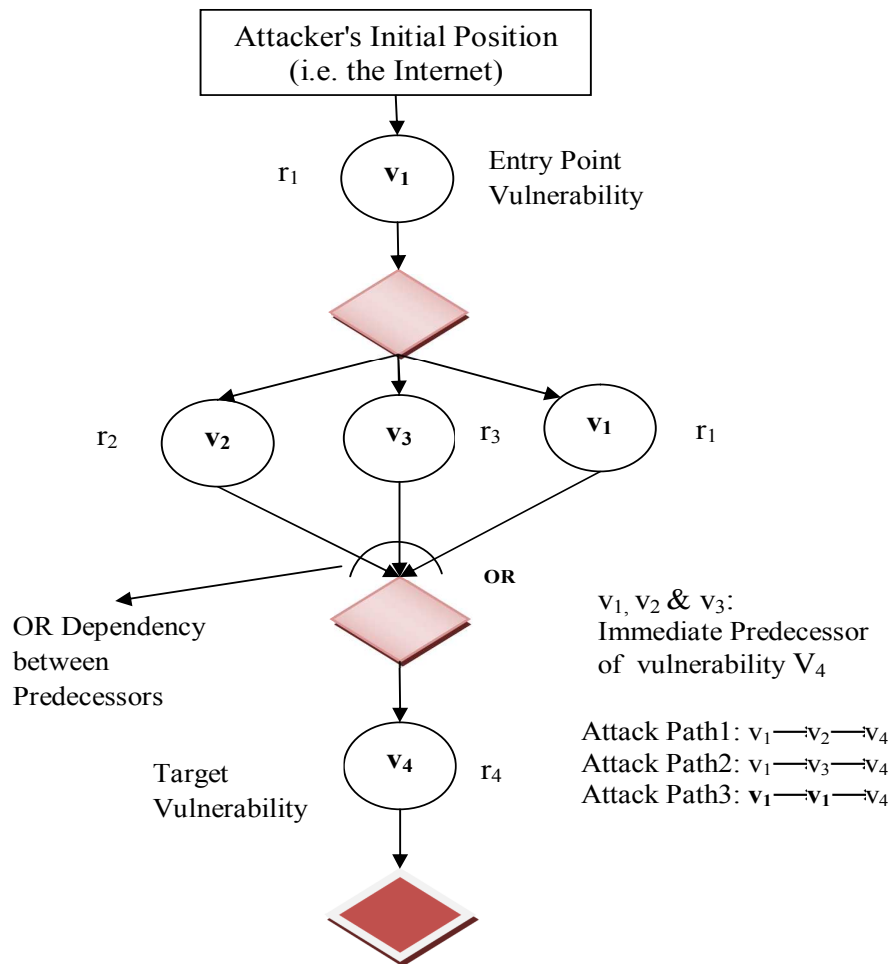


Figure 3.1: Network risk conditions.

3.3 Risk Conditions Pertaining to the Network Security

Figure 3.1 shows various network risk conditions, which are used in our proximity-based vulnerability risk assessment. Vulnerabilities v_1, v_2, v_3, v_4 are depicted by a Circle and respective post-condition by a Diamond. Here, r_i is the resistance posed by the vulnerability v_i to an attacker during its exploitation. Each vulnerability v in a network poses some level of difficulty to an attacker in terms of time, resources, and effort during its successful exploitation. Such exploitation difficulty is termed as an individual resistance (r) of a vulnerability v . In particular, $r(v)$ represents the effort put by an attacker until successful exploitation of the vulnerability [118]. Necessarily, $r(v)$ is designed in such a way that, it should provide the partial ordering on the relative difficulty of executing exploits. Higher the value of $r(v)$, it is harder to take advantage of the vulnerability v . Such resistance values can be obtained for each of the well-known vulnerability based on their CVSS Temporal Score [159].

3.3.2 Exploit Diversity along the Attack Path

Exploit diversity along the attack path(s) signifies the number of different kinds of exploits an adversary has to successfully execute to reach and exploit the target vulnerability v . Essentially, the attack path with various types of exploits requires more knowledge than the typical attack path. More kind of exploits along the attack path indicates that an adversary needs to have more knowledge about the different exploitation technologies. In completely diversified attack path, an adversary has to spend an individual and independent effort in successfully exploiting each vulnerability coming across the attack path [30].

Usually, after getting access to a particular host in a network (pivot point), an adversary performs reconnaissance/scanning from that point onward and move on to the next target machine by successfully exploiting vulnerabilities in it. An adversary may encounter one or more vulnerabilities in the next target machine(s) during the reconnaissance. If only one target host with single vulnerability, then adversary has to exploit it compulsorily. On the contrary, if there are one or more target machines with more than one vulnerabilities in it, then the attacker's decision of the next vulnerability exploitation depends on the type of vulnerability adversary encountered. If the next target vulnerability is similar to the kind of vulnerabilities which are already exploited (i.e. vulnerabilities along the attack path), then the probability of its exploitation is more.

Therefore, the number of different kinds of vulnerabilities (exploits) along the attack path also can be considered as an important risk condition. It indicates the attacker's knowledge. As shown in Figure 3.1, if an adversary chooses the third attack path, she needs to exploit the vulnerability v_1 one more time to reach the target vulnerability v_4 .

3.3.3 Risk of the Neighboring Vulnerabilities

As shown in Figure 3.1, an exploit-dependency attack graph illustrates the number of ways by which an adversary can reach and exploit the target vulnerability v_4 . For each nonentry-point vulnerability (the one which can not be exploited directly from attackers initial position) in an attack graph, there will be one or more neighbors from where the vulnerability under consideration could be directly reached and exploited. Let $PD(v_i)$ be the set of such neighboring vulnerabilities on which the exploitation of the vulnerability v_i depends.

It is important to notice that when many neighbors coexist in an attack graph, reaching the target vulnerability is easier than if only one of these neighbor exists [27]. Intuitively, the more the number of ways to arrive at a particular vulnerability means there are more chances of its exploitation. In other words, more attack opportunities mean less security because attackers will have better chance to reach the target [27]. Even though the attack paths arising from other neighbors are harder than the original (least effort) one, they nevertheless represent possibilities for attacks and thus they increase the overall probability of exploitation of the target vulnerability. That is the vulnerability with a large number of invokers (immediate predecessors in the context of attack graph) is more likely to be exploited than the one with less number of neighbors.

As shown in Figure 3.1, the vulnerability v_4 can be directly reached and exploited from three different vulnerabilities. Even though attacker will take/follow any one of the attack path, the availability of other neighbors represents possibilities of attacks, and thus they increases the overall probability of exploitation of v_4 .

3.3.4 The Conjunctive (AND)/Disjunctive (OR) Dependency Between Neighboring Vulnerabilities

Essentially, there can be either conjunctive (AND), or disjunctive (OR) dependency relationship between the exploits in an exploit-dependency attack graph [56]. The conjunctive dependency between the predecessors implies that the successor exploit cannot be successfully executed until all of the participating exploits (in an AND dependency) have been executed successfully. Whereas, the disjunctive dependency between predecessor exploits indicates that the successor can be executed successfully if any one of the predecessor vulnerability can be exploited successfully. Compared to the conjunctive (AND) dependency, disjunctive (OR) dependency between predecessor exploits leads to the higher probability of execution of the successor exploit and hence increased security risk. As shown in Figure 3.1, there is an OR dependency between the neighboring exploits (i.e. v_1, v_2, v_3) of v_4 .

3.4 Measuring Vulnerability Risk

In this Section, we will discuss how to use critical network conditions for quantifying the security risks of exploitable vulnerabilities.

3.4.1 Measuring Diversity-adjusted Vulnerability Score (DVS)

To measure the security risk of exposed vulnerability, we first need to identify the untrusted network from where the attacker starts compromising the network. There can be multiple untrusted networks from where an adversary can initiate the attack, but for the sake of simplicity, we assume a single network, i.e. the Internet. Each exploitable vulnerability in a target system is positioned a certain distance apart from the attacker's initial position and can be reachable through one or more attack paths.

According to the Wang et al. [27], network security should be measured as the smallest effort required to reach the goal. The shortest path distance (SP) [47] assumes the same amount of effort is needed to exploit each vulnerability along the attack path and only counted the number of hurdles (vulnerabilities) along the attack path(s) as a measure of attackers effort. However, each type of vulnerability poses different amount

of resistance to the adversary and hence she needs to spend different amount of effort on each vulnerability during the network intrusion. Therefore, we have considered vulnerability resistance along the attack path(s) as one of the factor while calculating the risk posed by the target vulnerability. For an attack graph shown in Figure 3.1, the strength of each of the attack path (AP_i) can be computed as:

$$PR(AP_i) = \sum_{j=1}^n r_j \quad (3.1)$$

Here r_j is the resistance posed by an individual vulnerability to the attacker. As shown in Figure 3.1, the resistance posed by the attack paths 1, 2 and 3 is simply the sum of the resistance of vulnerabilities along the attack paths. However, if there is a repetition of already exploited vulnerability along the attack path(s) (for, e.g. third attack path in Figure 3.1), then the attacker can use previously engineered exploits with little or no modification. In such case, the resistance posed by the repeated vulnerability is much smaller than the initial (original) resistance.

For the repeated vulnerability v_1 along the third attack path in Figure 3.1, the resistance value becomes $0.3 * r_1$. Such reduction in the resistance is due to the attackers acquired skills, tools, and techniques. Here, 0.3 is the effort reduction factor, and it captures the effect of vulnerability repetition along the attack path. It is the only subjective parameter used in our risk calculation method. The decision of the selection of the effort reduction factor should be carefully done, and it is heavily dependent on the types of vulnerabilities. An administrator can choose this value based on the effort required to tweak the already engineered exploit for getting the advantage of repeated vulnerability in a network. To the best of our knowledge, there is no study on how much reduction in attacker's work factor (vulnerability resistance) happens when the attacker exploits the same vulnerability second time. Such reduction in attacker's work factor could be different for different classes of vulnerabilities.

Based on the above discussion, the path resistance (PR) of the third attack path in Figure 3.1 can be calculated as:

$$PR(AP_3) = r_1 + (0.3 * r_1) + r_4$$

Once we compute the resistance posed by each of the attack path available to reach the target vulnerability v_i in an attack graph, we focus on the path with minimum

resistance value. It is because network security should always be measured as the smallest effort required to reach the goal [27]. The lower the attack path resistance, closer the vulnerability is, analogically, to the untrusted network, and therefore, higher the chances of target vulnerability exploitation. The problem of finding the distance (i.e. minimum path resistance) of vulnerability from the attacker's initial position is analogous to the single-source shortest path (SSSP) problem. For the vulnerability v_4 in example attack graph (Figure 3.1), the minimum path resistance MPR is the minimum of all attack path resistance values. That is:

$$MPR(v_4) = \min(PR(AP_1), PR(AP_2), PR(AP_3)) \quad (3.2)$$

Here, $MPR(v_i)$ indicates the minimum resistance attacker has to face during the exploitation of v_i . If there are two or more attack paths available to an attacker to reach the vulnerability v_i , then the attack path that poses minimum resistance should be considered for computing the MPR value [158]. In order to assess the security risk posed by an exploitable vulnerability v_i , we define Diversity-adjusted vulnerability score (DVS) as follows:

$$DVS(v_i) = CVSS(v_i) \times \frac{1}{MPR(v_i)} \quad (3.3)$$

A lower value of DVS is desirable for better network security. Smaller the value of MPR , higher will be the DVS of vulnerability v_i . Consider the attacker encounters the two network vulnerabilities v_1 and v_2 with same CVSS base score but different MPR values. If $MPR(v_1) > MPR(v_2)$ then $DVS(v_1) < DVS(v_2)$ and hence higher will be the chances of exploitation of vulnerability v_2 .

3.4.2 Measuring Neighborhood Proximity-adjusted Vulnerability Score (NPVS)

In an exploit-dependency attack graph shown in Figure 3.1, except the entry point vulnerabilities, other vulnerabilities may have one or more predecessor vulnerabilities. Therefore, the neighborhood $N(v_i)$ of a particular non-entry point vulnerability v_i consists of the set of vulnerabilities whose exploitation lead to the exposure of v_i , that is, $N(v_i) = \{v_j \in V : v_j v_i \in E\}$. Here, V is the set of vulnerabilities/exploits present

in the attack graph, and E is the set of edges among the exploits. By design of the exploit-dependency attack graph [56], there is a “depend on” relationship between the non-entry point vulnerability v_i and its predecessors.

Here, our focus is on the vulnerability neighborhood to understand how many ways an attacker can reach a particular vulnerability. An administrator must consider “how many predecessors are there for a particular vulnerability” and “how they are related, i.e. whether through AND dependency or OR dependency.”

We define vulnerability neighborhood as the predecessor sets PD_1, PD_2, \dots, PD_n for each exploitable vulnerability depicted in the generated attack graph such that:

$PD_i = \{a \text{ set of vulnerabilities in an attack graph from where the vulnerability } v_i \text{ can be directly reached and exploited} \}$

By definition, vulnerabilities in a predecessor set PD_i can directly trigger the vulnerability v_i . The reachability from a group of predecessor vulnerabilities to the target vulnerability v_i is explicitly governed by the vulnerable service connectivities/network access control policies. Essentially, there can be either conjunctive (AND), or disjunctive (OR) dependency relationship between the predecessor exploits. The conjunctive dependency between the predecessors implies that the successor exploit cannot be executed until all of the participating exploits (in an AND dependency) have been executed successfully. Whereas, the disjunctive OR dependency between the predecessor exploits indicates that the successor vulnerability can be exploited successfully if any one of the predecessors can be executed successfully.

It is important to notice that when many neighbors (i.e. predecessor vulnerabilities) coexist in an attack graph, reaching the target vulnerability is easier than if only one of these neighbor exists. Intuitively, the more the number of ways to arrive at a particular vulnerability means there are more chances of its exploitation. In other words, more attack opportunities mean less security because attackers will have better chance to reach the target. Even though the attack paths arising from other neighbors are harder than the original (least effort) one, they nevertheless represent possibilities for attacks and thus they increase the overall probability of exploitation of the target vulnerability. Therefore, we have considered the Normalized Diversity-adjusted Vulnerability Score (NDVS) of each neighboring vulnerability for computation of the NPVS of vulnerability under consideration. In particular, $NDVS(v_i) = \frac{DVS(v_i)}{10}$.

For a given vulnerability v_i having a predecessor set PD_i the NPVS becomes:

$$NPVS(v_i) = \begin{cases} NDVS(v_j) \times NDVS(v_k); \text{ when } v_j \text{ and } v_k \text{ are Conjunctive} \\ NDVS(v_j) + NDVS(v_k) - NDVS(v_j) \times NDVS(v_k); \\ \text{ when } v_j \text{ and } v_k \text{ are Disjunctive} \\ NDVS(v_j); \text{ when } v_j \text{ is the only neighbor of } v_i \\ 1; \text{ when } v_i \text{ is the entry point vulnerability} \end{cases} \quad (3.4)$$

For the first two cases (in Equation 3.4), vulnerability v_i is the immediate successor of the vulnerabilities v_j , and v_k . For the entry point vulnerabilities (whose exploitation does not depend on the exploitation of any other vulnerability), the predecessor set $PD_i = \{\phi\}$. Therefore, the NPVS value for such directly exploitable vulnerabilities will be the highest, i.e. 1 (as shown in Equation 3.4). The magnitude of NPVS for a vulnerability also depends on the AND-OR relationship between its predecessor vulnerabilities. Compared to the AND dependency, an OR dependency between predecessor vulnerabilities leads to the higher NPVS score of a successor. For the vulnerabilities which are not directly exploitable from the attacker's initial position, a large number of OR-ed predecessors with higher DVSs results in a higher NPVS value.

3.4.3 Measuring Improved Relative Cumulative Risk (IRCR)

To assess the cumulative risk of each exploitable vulnerability in the network, we need to take into account the DVS and NPVS values computed in the previous step (Equation 3.3 and 3.4) and aggregate them in a way that can express the security risk condition around the vulnerability in question. The improved relative cumulative risk (IRCR) of an exploitable vulnerability v_i in the given network can be computed as:

$$IRCR(v_i) = DVS(v_i) \times NPVS(v_i) \quad (3.5)$$

As shown in Equation 3.3, $DVS(v_i)$ is the individual CVSS base score of a vulnerability v_i , adjusted by the minimum path resistance value i.e. $MPR(v_i)$. The security risk because of the neighboring vulnerabilities represented by the $NPVS(v_i)$ (Equation 3.4). For a vulnerability having higher DVS, and a large number of OR dependent predecessors results in a higher IRCR. The magnitude of NPVS depends on the AND-OR relationship between neighboring vulnerabilities. For a given number of predecessors, the NPVS score of successor vulnerability is always high for the OR dependency,

and less for the *AND* dependency. Since CVSS Base Score captures the severity (impact) of the vulnerability and risk conditions capture the likelihood (probability) of vulnerability exploitation, as per the classical definition of risk, IRCR measures the security risk posed by the vulnerability.

The larger the value of IRCR, the greater will be the risk posed by the vulnerability. Therefore, the administrator decides on the patching order of vulnerabilities based on their IRCR value. In practice, patching of all vulnerabilities in the network is mission impossible for the administrator. The vulnerabilities for which no patch is available (or whose patching may hurt the business performance), the goal of the administrator's is to reduce the risk posed by them. Since the lower value of IRCR is desired for secure network configuration, the purpose of the security analyst is to decrease the IRCR. It can be done by disabling the initial conditions, patching of entry-point vulnerabilities, and network reconfiguration. Patching of entry point vulnerabilities remove all the attack paths, and hence the IRCR of all other network vulnerabilities become 0. If patches are not available for the entry point vulnerabilities, then one needs to disable the initial conditions which consequently lead to the longer attack sequence and hence the larger value of attack path resistance. Further, by disabling the initial conditions, one can reduce the cardinality of predecessor set PD_i of a vulnerability v_i . The higher the exploit diversity along the possible attack path(s), lower will be the IRCR score of the target vulnerability. There are many possible ways to increase the exploit diversity along the attack path for network hardening and have been suggested in [30], [46], [160].

3.4.4 Summary of the Parameters used in Proximity-based Risk Assessment

Table 3.1 list the definitions of the various parameters/terms introduced in this Section.

3.5 Experimental Setup and Results

In this Section, different case studies are presented to show the performance of our proximity-based approach developed for the purpose of vulnerability risk estimation. Comparisons between our proximity-based vulnerability ranking and other attack

Table 3.1: Parameters used in proximity-based risk assessment method

Parameters	Definitions
r	Individual resistance posed by the vulnerability v
$PR(AP_i)$	Path resistance along the attack path i .
$MPR(v_i)$	Minimum path resistance required to reach and exploit vulnerability v_i
DVS	Diversity-adjusted vulnerability score
PD_i	A set of vulnerabilities in an attack graph that are direct predecessor of the vulnerability v_i . Here, $i = 1 \dots n$, where n is such sets.
NDVS	Normalized diversity-adjusted vulnerability score
NPVS	Neighborhood proximity-adjusted vulnerability score
IRCR	Improved relative cumulative risk

graph-based metrics such as cumulative probability (P) [26], and cumulative resistance (R) [27] is also rendered and analyzed. Experimental results demonstrate that proposed IRCR metric can be a complementary to the current attack graph-based metrics (i.e. P [26], R [27]) in measuring the influential levels of exploitable vulnerabilities and hence can be a used for prioritizing vulnerability remediation activities.

Network configuration 1: The topology of the test network is shown in Figure 3.2, which is same as the network topology used in [32], [118]. There are Four machines located within Two subnets. $Host_3$ is attackers target machine, and *MySQL* is the critical resource running over it. The attacker is a malicious entity in the external network, and her goal is to obtain root-level privileges on $Host_3$. The job of firewalls is to separate internal network from the Internet. Firewall policies that limit connectivity in the network configuration are given in Table 3.2.

Table 3.3 shows the system characteristics for the hosts available in the example network. Table 3.4 shows nine example vulnerabilities and their basic and temporal vectors. Such kind of data is available in public vulnerability databases viz. NVD [37], Bugtraq [86], etc. Here external firewall allows any external host to only access services running on host $Host_0$. Connections to all other services/ports on other hosts are blocked. Host's within the internal network have authority to connect to only those ports specified by the firewall policies as shown in Table 3.2. The number 1, 2 and

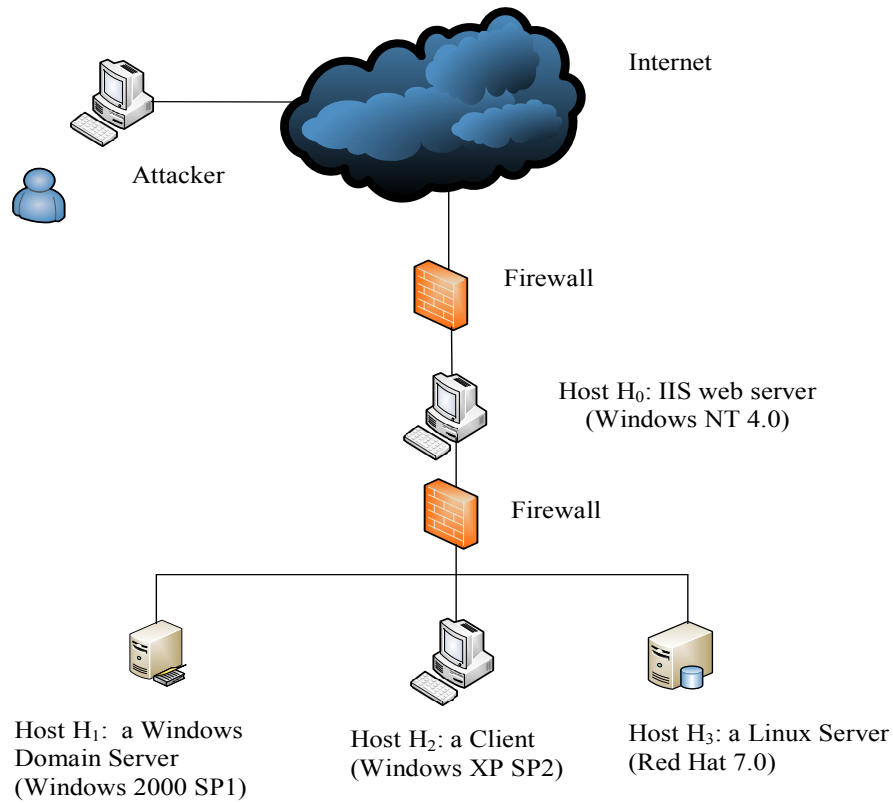


Figure 3.2: An example network

Table 3.2: Connectivity-limiting firewall policies

<i>Host</i>	<i>Attacker</i>	<i>H₀</i>	<i>H₁</i>	<i>H₂</i>	<i>H₃</i>
<i>Attacker</i>	0	1	-1	-1	-1
<i>H₀</i>	-1	0	1,2,3	1,2	1,2,3
<i>H₁</i>	-1	1	0	1,2	1,2,3
<i>H₂</i>	-1	1	-1	0	1,2,3
<i>H₃</i>	-1	1	-1	-1	0

3 represents the open services which can be referred to the numbers assigned to each host in Table 3.3; -1 represents source host is prevented from having access to any service on the destination host; 0 means a self-connection. An attack graph generated

3.5 Experimental Setup and Results

Table 3.3: System characteristics for the network configuration shown in Figure 3.2

Host	Services	Ports	Vulnerabilities	CVE IDs
$Host_0$	1. IIS_Web_Service	80	IIS buffer-overflow	CVE-2010-2370
$Host_1$	1. ftp	21	ftp-rhost overwrite	CVE-2008-1396
	2. ssh	22	ssh buffer-overflow	CVE-2002-1359
	3. rshd	514	rsh-login	CVE-1999-0180
$Host_2$	1. Netbios-ssn	139	netbios_ssn_nullsession	CVE-2003-0661
	2. rshd	514	rsh-login	CVE-1999-0180
$Host_3$	1. LICQ	5190	LICQ-remote-to-user	CVE-2001-0439
	2. Squid_proxy	80	Squid-port-scan	CVE-2001-1030
	3. MySQL_DB	3306	Local-setuid-buffer-overflow	CVE-2006-3368

for the example network is shown in Figure 3.3. Attacker's initial position is shown by a Circle, exploits (vulnerabilities) by an Oval, and respective post-conditions by a plain-text.

The primary goal of the IRCR-based vulnerability rating system is to separate vulnerabilities from each other as far as possible. Table 3.5 ¹ shows 13 vulnerability instances of the example network in Figure 3.2. As shown in Table 3.5 each vulnerability instance has assigned rank depending on their severity levels measured by the IRCR metric (Equation 3.5). The vulnerability instances are sorted in decreasing order of their IRCR score. Since the lower value of IRCR is desired for secure network configuration, vulnerabilities with higher IRCR score need to be fixed with top priority.

The primary goal of assessing the security risk of each exploitable vulnerability is to prioritize them for proactive network hardening and hence threat mitigation. Security administrators can mitigate the security risk by patching the top-ranked vulnerabilities. They need to create the security mitigation plan from the top-ranked vulnerabilities by considering their respective security controls. In this Chapter, we haven't

¹Table 3.5 shows vulnerability instances, and their respective values for CVSS Base Score, Temporal Score, Individual Probability (p), Individual Resistance (r), minimum path resistance (MPR) to reach from attacker's initial position, Diversity-adjusted vulnerability score (DVS), Neighborhood proximity-adjusted vulnerability score (NPVS), and Improved relative cumulative risk (IRCR).

3.5 Experimental Setup and Results

Table 3.4: Vulnerability attributes vectors (as per CVSS v3.0 [2] specifications.)

Vulnerability CVE IDs	Basic Vector	CVSS Base Score	Temporal Vectors	CVSS Temporal Score
CVE-2010-2370	AV:N/AC:L/PR:N/UI:N/S:C/C:N/I:L/A:N	5.8	E:F/RL:O/RC:C	5.4
CVE-2008-1396	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	5.3	E:F/RL:O/RC:C	4.9
CVE-2002-1359	AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H	10	E:H/RL:O/RC:C	9.5
CVE-1999-0180	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L	7.3	E:P/RL:O/RC:C	6.6
CVE-2003-0661	AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:N/A:N	5.8	E:F/RL:O/RC:C	5.4
CVE-2001-0439	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L	7.3	E:H/RL:O/RC:C	7
CVE-2001-1030	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L	7.3	E:P/RL:O/RC:C	6.6
CVE-2006-3368	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	5.3	E:H/RL:O/RC:C	5.1

considered security controls and their respective cost, as our goal is to quantify the risk of each vulnerability.

Table 3.5: Results of proximity-based vulnerability ranking (for attack graph shown in Figure 3.3)

Vulnerability Instance	CVE IDs	CVSS Base Score	CVSS Temporal Score	p	r	MPR	DVS	NPVS	IRCR	Rank
IIS_buffer_overflow(0,0)	CVE-2010-2370	5.8	5.4	0.54	1.85	1.85	3.14	1.00	3.135	1
SSH_buffer_overflow(0,1)	CVE-2002-1359	10	9.5	0.95	1.05	2.90	3.44	0.31	1.079	2
Squid_port_scan(1,3)	CVE-2001-1030	7.3	6.6	0.66	1.52	4.42	1.65	0.43	0.715	3
Squid_port_scan(0,3)	CVE-2001-1030	7.3	6.6	0.66	1.52	3.37	2.17	0.31	0.680	4
Netbios-ssnnull session(1,2)	CVE-2003-0661	5.8	5.4	0.54	1.85	4.76	1.22	0.43	0.528	5
Netbios-ssnnull session(0,2)	CVE-2003-0661	5.8	5.4	0.54	1.85	3.70	1.57	0.31	0.491	6
ftp_rhost(0,1)	CVE-2008-1396	5.3	4.9	0.49	2.04	3.89	1.36	0.31	0.427	7
Squid_port_scan(2,3)	CVE-2001-1030	7.3	6.6	0.66	1.52	5.22	1.40	0.26	0.363	8
LICQ_remote_to_user(0,3)	CVE-2001-0439	7.3	7	0.7	1.43	4.80	1.52	0.22	0.330	9
Local-setuid buffer-overflow(3,3)	CVE-2006-3368	5.3	5.1	0.51	1.96	6.76	0.78	0.34	0.266	10
LICQ_remote_to_user(1,3)	CVE-2001-0439	7.3	7	0.7	1.43	5.85	1.25	0.17	0.206	11
Rsh_login(1,1)	CVE-1999-0180	7.3	6.6	0.66	1.52	5.41	1.35	0.14	0.184	12
LICQ_remote_to_user(2,3)	CVE-2001-0439	7.3	7	0.7	1.43	6.65	1.10	0.14	0.154	13

The IRCR and the CVSS Base Score for each vulnerability instance in the test network are plotted in Figure 3.4. Along the x-axis, there are different vulnerability instances, and along the primary (left) and the secondary (right) y-axis the values of IRCR and CVSS, respectively. Figure 3.4 is a point plot and line between two vulnerability instance has no significance other than to indicate the trend. Since the vulnera-

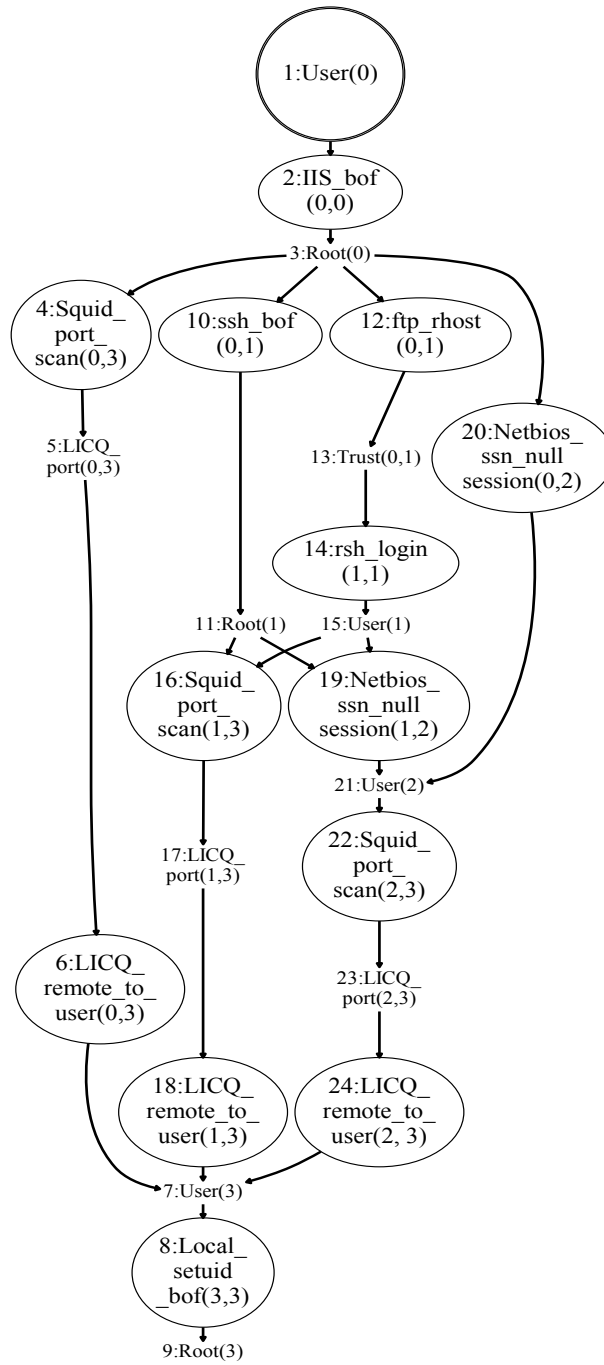


Figure 3.3: An exploit-oriented (exploit-dependency) attack graph for the network configuration 1.

bilities with the higher IRCR need to be patched with top priority, we have plotted the vulnerabilities in decreasing order of their *IRCR* score. As evident from the Figure 3.4, the values of CVSS Base Score experience disparate fluctuating trends. Taking this fact into account, one can say that exploitable vulnerability with a higher CVSS score need not pose a higher risk and vice versa. For vulnerabilities/vulnerability instances having same CVSS score, there is no way to know in which order they should be patched. It hinders the decision on taking hardening strategies. However, IRCR can reflect the influential level of vulnerability instances more precisely, as the estimation is processed by considering various network risk conditions. Proposed IRCR-based vulnerability rating system separate vulnerabilities from each other as far as possible according to their effects/influence.

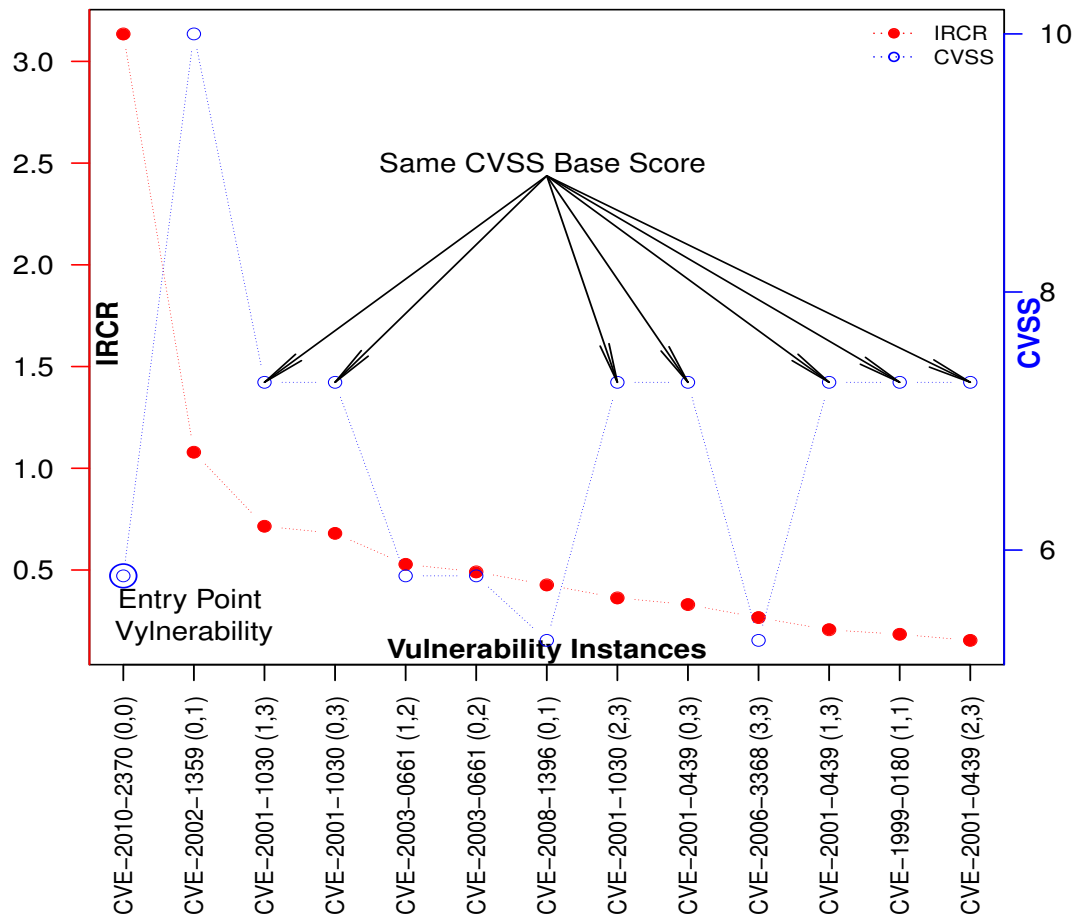


Figure 3.4: Vulnerability instances, and their respective values for IRCR and CVSS.

Vulnerability Priority Comparison

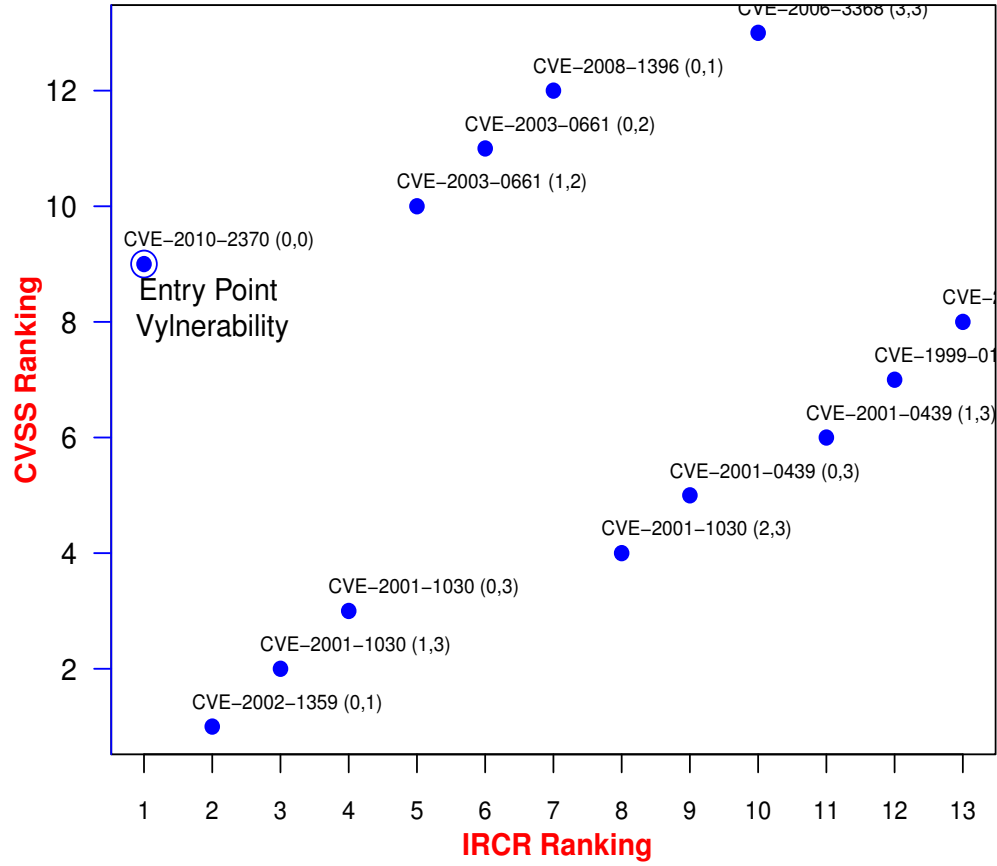


Figure 3.5: Vulnerability priority comparison: IRCR vs CVSS.

Figure 3.5 shows the patching order of the vulnerabilities based on the rating assigned by IRCR and CVSS. Along the x-axis, vulnerabilities are plotted in the decreasing order of their IRCR value. After checking Table 3.5 and Figures 3.5, 3.3, we can find that the CVSS ranking of the vulnerability *IIS_buffer_overflow* (CVE-2010-2370) is at 9th position. In contrast, through our proximity-based ranking, it comes to 1st as shown in Table 3.5, which corresponds well to the estimated attacking scenario. In attack graph of Figure 3.3, *IIS_buffer_overflow(0,0)* is the first vulnerability to be triggered by an adversary and it is the very critical for vulnerability instance to stand out that no adversary can ignore or bypass while deploying attacks. Without issuing this vulnerability (i.e. CVE-2010-2370) successfully, no other vulnerabilities can be

3.5 Experimental Setup and Results

exploited.

Two vulnerabilities *Squid_port_scan* and *LICQ_remote_to_user*, share the same CVSS Base Score of 7.3 and given the same rank. These sort of rankings can frequently happen in CVSS, confusing on patching order and hindering decision on taking hardening strategies. Checking ranking results, 3 vulnerability instances of *Squid_port_scan* and 3 instances of *LICQ_remote_to_user* have been ranked in Table 3.5. It is obvious that all the instances of *Squid_port_scan* are ranked prior the instances of *LICQ_remote_to_user*. It indicates that based on the view of whole security level, the elimination of each *Squid_port_scan* can provide more secure enhancement than the elimination of *LICQ_remote_to_user* instances.

Table 3.6: Comparative results for different risk assessment techniques for the attack graph shown in Figure 3.3.

Vulnerability Instance	CVE IDs	CVSS Base Score	IRCR	Cumulative Resistance (R)	Cumulative Probability (P)
IIS_buffer_overflow(0,0)	CVE-2010-2370	5.8	3.135	1.852	0.540
SSH_buffer_overflow(0,1)	CVE-2002-1359	10	1.079	2.904	0.513
Squid_port_scan(1,3)	CVE-2001-1030	7.3	0.715	3.405	0.395
Squid_port_scan(0,3)	CVE-2001-1030	7.3	0.680	3.367	0.356
Netbios-ssnnull session(1,2)	CVE-2003-0661	5.8	0.528	3.741	0.323
Netbios-ssnnull session(0,2)	CVE-2003-0661	5.8	0.491	3.704	0.292
ftp_rhost(0,1)	CVE-2008-1396	5.3	0.427	3.893	0.265
Squid_port_scan(2,3)	CVE-2001-1030	7.3	0.363	3.376	0.343
LICQ_remote_to_user(0,3)	CVE-2001-0439	7.3	0.330	4.796	0.249
Local-setuid buffer_overflow(3,3)	CVE-2006-3368	5.3	0.266	3.565	0.3
LICQ_remote_to_user(1,3)	CVE-2001-0439	7.3	0.206	4.833	0.276
Rsh_login(1,1)	CVE-1999-0180	7.3	0.184	5.408	0.175
LICQ_remote_to_user(2,3)	CVE-2001-0439	7.3	0.154	4.805	0.240

Next, we compared the IRCR metric with P [26] and R [27] as shown in Table 3.6. Here, cumulative probability P [26] measures the overall likelihood of an attacker successfully exploiting a given vulnerability from the attacker's initial position. It quantifies the attacker's likelihood of exploiting the particular vulnerability by taking into account the causal relationship between already exploited vulnerabilities. The lower the value of P, lower will be the cumulative (overall) probability of the attacker reaching and exploiting the vulnerability. Therefore, lower value of P is desired for secure network configuration. On the other hand, cumulative resistance (R) [27] represents

3.5 Experimental Setup and Results

Table 3.7: Results comparison: top 5 vulnerabilities based on the different risk assessment techniques.

Risk Assessment Technique	Top 5 vulnerabilities instance
CVSS v3.0 [2]	CVE-2002-1359 , CVE-2001-1030 (1,3), CVE-2001-1030 (0,3), CVE-2001-1030 (2,3), CVE-2001-0439 (0,3)
Probability-based Metric (P) [26]	CVE-2010-2370, CVE-2002-1359, CVE-2001-1030 (1,3), CVE-2001-1030 (0,3), CVE-2001-1030 (2,3)
Attack Resistance Metric (R) [27]	CVE-2010-2370, CVE-2002-1359, CVE-2001-1030 (0,3), CVE-2001-1030 (2,3) , CVE-2001-1030 (1,3)
Improved Relative Cumulative Risk (IRCR)	CVE-2010-2370, CVE-2002-1359, CVE-2001-1030 (1,3), CVE-2001-1030 (0,3), CVE-2003-0661 (1,2)

the overall difficulty of exploiting a particular vulnerability starting from the attacker's initial position. The higher the value of R, higher will be the resistance posed by the vulnerability to the attacker. We have chosen these two metrics for comparison because they take into account all the vulnerabilities between attacker's initial position and the target vulnerability. Further, these metrics consider the cause-consequence relationship between network vulnerabilities.

The comparative results for the top 5 vulnerability instances based on the different risk assessment techniques is presented in Table 3.7. As evident from Table 3.7, both P and R choose the vulnerability instance CVE-2001-1030 (2,3) over CVE-2003-0661 (1,2). It is because the calculation steps for both P and R are roughly similar. Further, both P and R gives higher priority to the vulnerabilities that can be reachable and exploitable through more number of attack paths. There are three ways to reach and exploit the vulnerability instance CVE-2001-1030 (2,3). Such fluctuating trend in the risk value of CVE-2001-1030 (2,3) can be easily seen in both Figures 3.6 and 3.8. However, IRCR gives priority to CVE-2003-0661 (1,2) over CVE-2001-1030 (2,3). It is because the attacker needs to spend less effort to reach and exploit CVE-2003-0661 (1,2).

Figure 3.7 shows patching order of the vulnerabilities based on the rank assigned by both IRCR and P. Whereas, the patching order of the exploitable vulnerabilities based on the rank assigned by IRCR and R is shown in Figure 3.9. For our example

attack graph (shown in Figure 3.3), metric P, R and IRCR provides a nearly similar result for the mitigation plan of top 5 vulnerabilities. It demonstrates that proposed IRCR metric can be a complementary to the current attack graph-based metrics in measuring the influential levels of exploitable vulnerabilities. As there is no repetition of vulnerabilities in any of the attack paths of example attack graph (Figure 3.3), and the calculation steps for P, R, and IRCR are roughly similar, all of them provides the close result. But this may not be true always.

Next, we have reconfigured the test network shown in Figure 3.2. The newly generated attack graph for the reconfigured network captures the effect of network/system reconfiguration, introduction of the new vulnerabilities, and network misconfiguration. The proposed proximity-based vulnerability rating system is applied to prioritize the vulnerabilities for network hardening. The result of the new comprehensive IRCR provides more score diversity than CVSS, P, R. These changes would help organizations and individuals better prioritize their responses to new vulnerabilities.

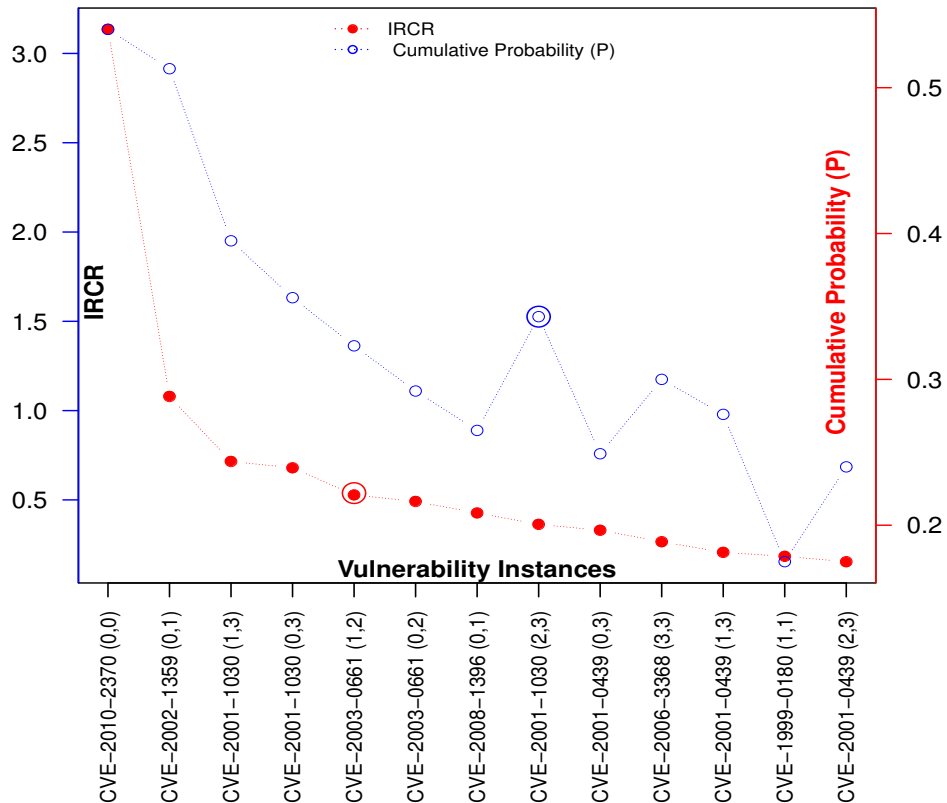


Figure 3.6: Vulnerability instances, and their respective values for IRCR and P

Vulnerability Priority Comparison

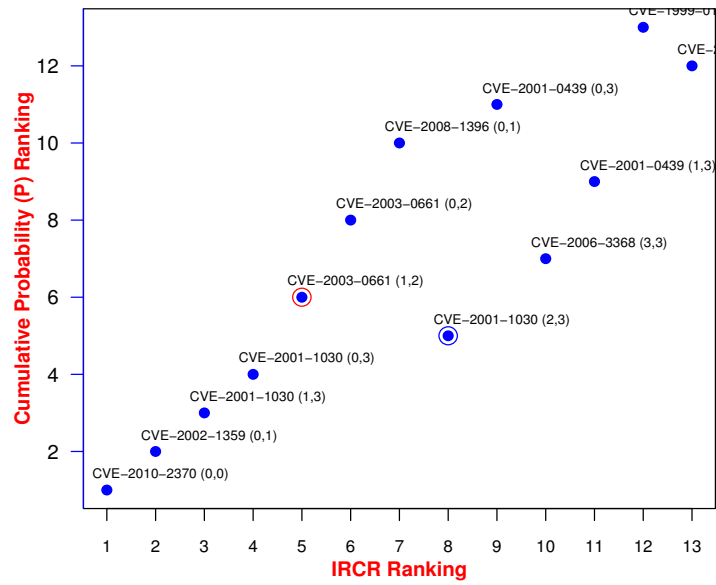


Figure 3.7: Vulnerability priority comparison: IRCR vs P

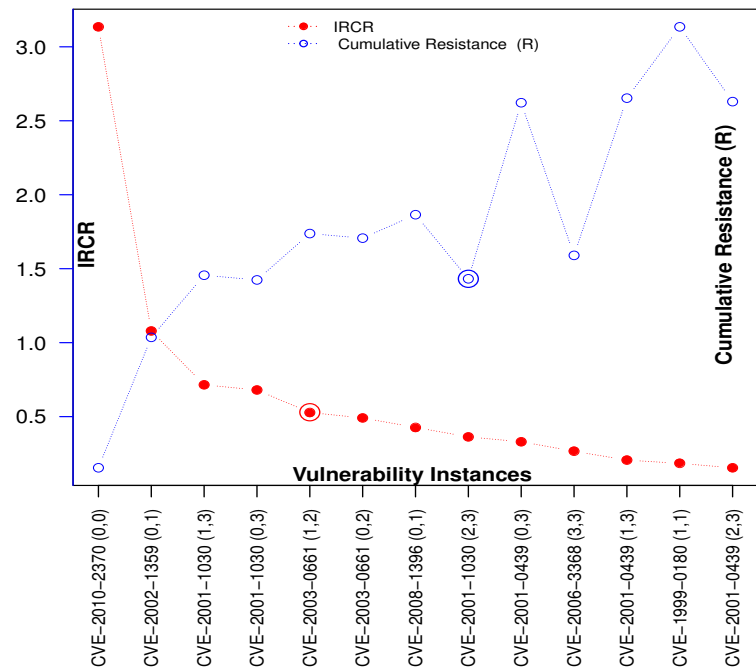


Figure 3.8: Vulnerability instances, and their respective values for IRCR and R

Vulnerability Priority Comparison

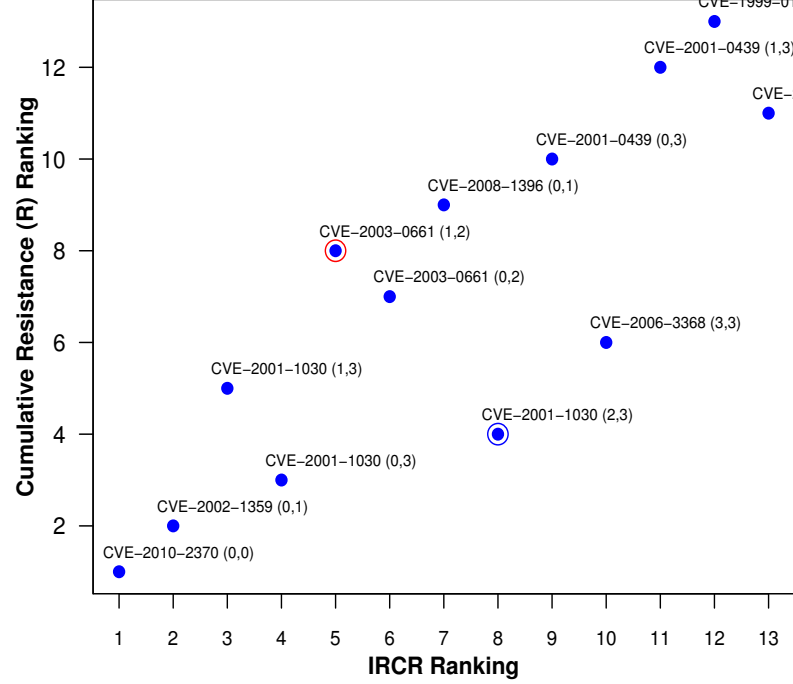


Figure 3.9: Vulnerability priority comparison: ICR vs R.

Network configuration 2: We have introduced a new vulnerable *ftp* service in *Host₂*. *ftp* service has a vulnerability CVE-2008-1396. So, there are total three vulnerable services are running on the *Host₂* (i.e. NetBIOS-ssn, rshd, and ftp). The access control policies for such changed network configuration are provided in Table 3.8. Figure 3.10 depicts the attack graph generated for the reconfigured network. It is evident from Figure 3.10 that there is a repetition of vulnerabilities along one of the attack path. Green Ovals represents such repeated vulnerabilities. The comparative results for the different risk assessment techniques for the attack graph illustrated in Figure 3.10 are provided in Table 3.9.

Because of the introduction of vulnerable *ftp* service in *Host₂*, there is an increase in the number of attack paths from 3 to 5 for a vulnerability instance *Squid_port_scan*(2, 3), i.e. CVE-2001-1030 (2,3). Such increase in the attack paths (or neighboring vulnerabilities) leads to increase in the NPVS value of a vulnerability *Squid_port_scan*(0, 3). However, there is no change in the DVS of CVE-2001-

3.5 Experimental Setup and Results

Table 3.8: Connectivity-limiting firewall policies for the network configuration 2

$Host$	$Attacker$	H_0	H_1	H_2	H_3
$Attacker$	0	1	-1	-1	-1
H_0	-1	0	1,2,3	1,2,3	1,2,3
H_1	-1	1	0	1,2,3	1,2,3
H_2	-1	1	-1	0	1,2,3
H_3	-1	1	-1	-1	0

Table 3.9: Comparative results for different risk assessment techniques for the attack graph shown in Figure 3.10

Vulnerability Instance	CVE IDs	CVSS Base Score	IRCR	Cumulative Resistance (R)	Cumulative Probability (P)
IIS_buffer-overflow(0,0)	CVE-2010-2370	5.8	3.135	1.852	0.540
SSH_buffer_overflow(0,1)	CVE-2002-1359	10	1.079	2.904	0.513
Squid_port_scan(1,3)	CVE-2001-1030	7.3	0.715	3.405	0.395
Squid_port_scan(0,3)	CVE-2001-1030	7.3	0.680	3.367	0.356
Netbios-ssnnull session(1,2)	CVE-2003-0661	5.8	0.528	3.741	0.323
Netbios-ssnnull session(0,2)	CVE-2003-0661	5.8	0.491	3.704	0.292
Squid_port_scan(2,3)	CVE-2001-1030	7.3	0.480	2.793	0.412
ftp_rhost(0,2)	CVE-2008-1396	5.3	0.427	3.893	0.265
ftp_rhost(0,1)	CVE-2008-1396	5.3	0.427	3.893	0.265
LICQ_remote_to_user(0,3)	CVE-2001-0439	7.3	0.330	4.796	0.249
Local-setuid buffer_overflow(3,3)	CVE-2006-3368	5.3	0.266	3.494	0.313
Rsh_login(2,2)	CVE-1999-0180	7.3	0.239	4.072	0.216
LICQ_remote_to_user(1,3)	CVE-2001-0439	7.3	0.206	4.833	0.276
Rsh_login(1,1)	CVE-1999-0180	7.3	0.184	5.408	0.175
LICQ_remote_to_user(2,3)	CVE-2001-0439	7.3	0.154	4.221	0.288
ftp_rhost(1,2)	CVE-2008-1396	5.3	0.119	7.449	0.086

1030 (2,3). It is because there is already least resistance path available through the vulnerability instance *Netbios - ssnnullsession(0, 2)*. Even though vulnerability repetition has occurred along one of the attack paths, the resultant path resistance is higher compared to the attack path through vulnerability instance *Netbios-ssnnull session(0,2)*. In such cases, even the repetition of vulnerabilities along the attack path does not benefit the adversary as one or more least resistance paths are readily available.

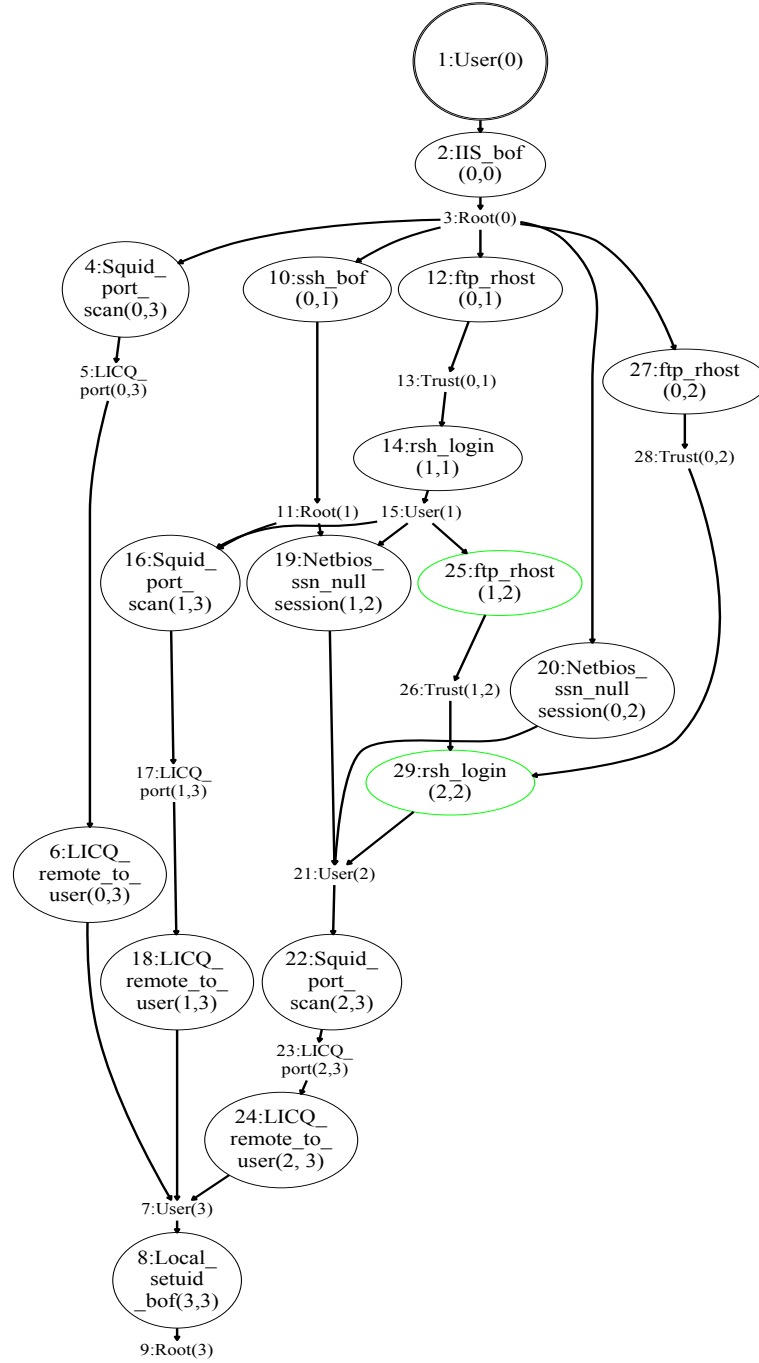


Figure 3.10: An exploit-dependency attack graph for the network configuration 2.

As evident from Table 3.6 and 3.9, because of increase in NPVS value vulnerability *Squid_port_scan*(2,3) improves its ranking by one position.

As evident from Table 3.9, both P and R favours the vulnerability instance CVE-2001-1030 (2,3) over CVE-2003-0661 (1,2). It is because both P and R gives higher priority to the vulnerabilities that can be reachable and exploitable through more number of attack paths. However, IRCR gives higher priority to CVE-2003-0661 (1,2) over CVE-2001-1030 (2,3). It is because the attacker needs to spend less effort to reach and exploit CVE-2003-0661 (1,2). In the mitigation plans of top 5 vulnerabilities (for the example attack graph illustrated in Figure 3.10) based on P, R and IRCR four vulnerabilities are common. Even though the mitigation plan is 80% same for all the three risk assessment methods, the vulnerability patching order varies.

Network configuration 3: We have introduced two more vulnerable services such as *LICQ*, and *Squid proxy* in $Host_1$ of the network configuration 2. *LICQ* has a vulnerability CVE-2001-0439, and *Squid proxy* has a CVE-2001-1030 vulnerability. So, there are complete five vulnerable services are running on $Host_1$ (i.e. ftp, ssh, rshd, LICQ, and Squid Proxy). The access control policies for the network configuration are illustrated in Table 3.10. Figure 3.11 depicts the attack graph generated for the network configuration. It is evident from Figure 3.10 that there is a repetition of vulnerabilities along many of the attack paths. Green Ovals signifies the repeated vulnerabilities along the attack paths. The comparative results for the different risk assessment techniques for the attack graph illustrated in Figure 3.11 are provided in Table 3.11.

Table 3.10: Connectivity-limiting firewall policies for the network configuration 3

$Host$	$Attacker$	H_0	H_1	H_2	H_3
$Attacker$	0	1	-1	-1	-1
H_0	-1	0	1,2,3,4,5	1,2,3	1,2,3
H_1	-1	1	0	1,2,3	1,2,3
H_2	-1	1	-1	0	1,2,3
H_3	-1	1	-1	-1	0

As evident from Table 3.11, both P and R choose the vulnerability instances CVE-2001-1030 (2,3), CVE-2003-0661 (1,2) over CVE-2001-1030 (0,1), CVE-2001-1030 (0,3). It is because both P and R gives higher priority to the vulnerabilities that can be reachable and exploitable through more number of attack paths. However, IRCR gives

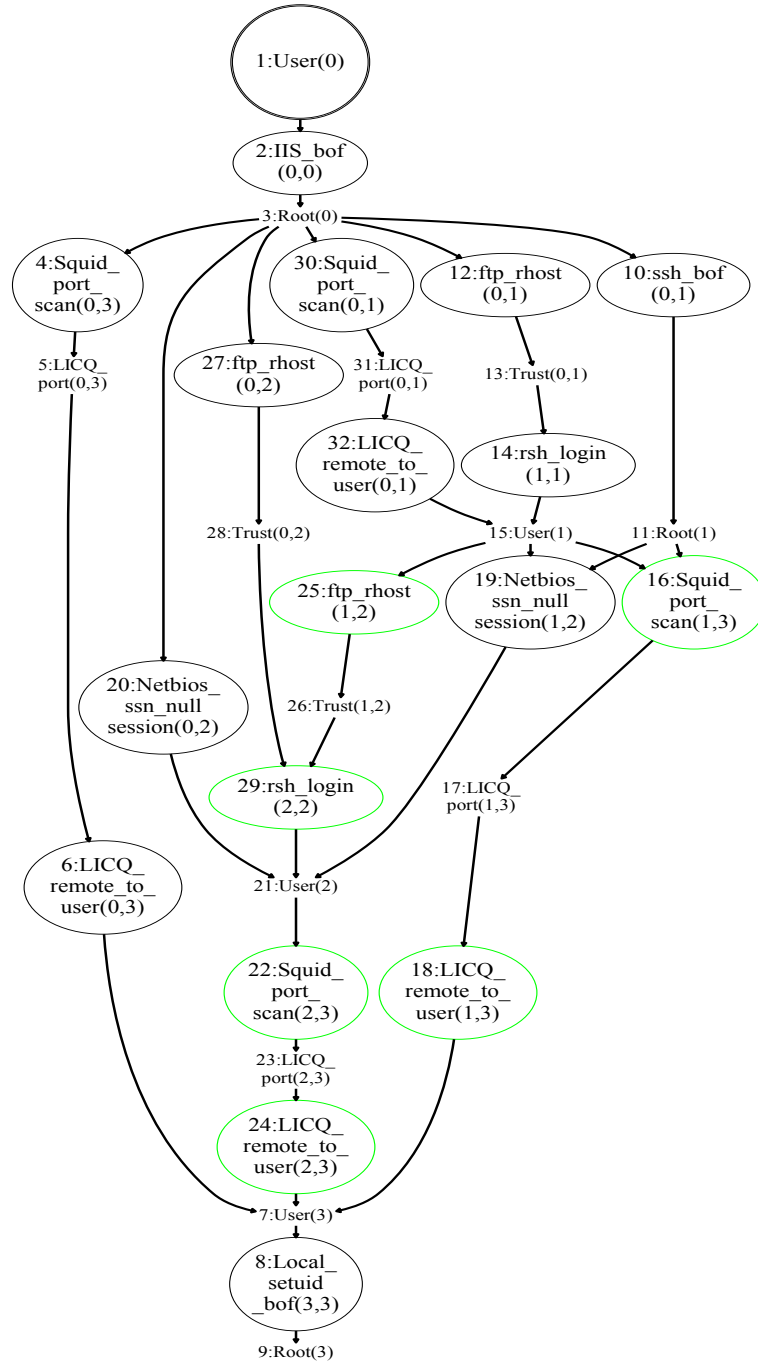


Figure 3.11: An exploit-dependency attack graph for the network configuration 3.

priority to CVE-2001-1030 (0,1), CVE-2001-1030 (0,3) over CVE-2001-1030 (2,3), CVE-2003-0661 (1,2). It is because the attacker needs to spend less effort to reach

3.5 Experimental Setup and Results

Table 3.11: Comparative results for different risk assessment techniques for the attack graph shown in Figure 3.11

Vulnerability Instance	CVE IDs	CVSS Base Score	IRCR	Cumulative Resistance (R)	Cumulative Probability (P)
IIS_buffer-overflow(0,0)	CVE-2010-2370	5.8	3.135	1.852	0.540
SSH_buffer_overflow(0,1)	CVE-2002-1359	10	1.079	2.904	0.513
Squid_port_scan(1,3)	CVE-2001-1030	7.3	0.858	2.871	0.461
Squid_port_scan(0,1)	CVE-2001-1030	7.3	0.680	3.367	0.356
Squid_port_scan(0,3)	CVE-2001-1030	7.3	0.680	3.367	0.356
Netbios-ssnnull session(1,2)	CVE-2003-0661	5.8	0.633	3.207	0.377
Squid_port_scan(2,3)	CVE-2001-1030	7.3	0.503	2.681	0.446
Netbios-ssnnull session(0,2)	CVE-2003-0661	5.8	0.491	3.704	0.292
ftp_rhost(0,2)	CVE-2008-1396	5.3	0.427	3.893	0.265
ftp_rhost(0,1)	CVE-2008-1396	5.3	0.427	3.893	0.265
LICQ_remote_to_user(0,1)	CVE-2001-0439	7.3	0.330	4.796	0.249
LICQ_remote_to_user(0,3)	CVE-2001-0439	7.3	0.330	4.796	0.249
Rsh_login(2,2)	CVE-1999-0180	7.3	0.287	3.620	0.265
Local-setuid buffer_overflow(3,3)	CVE-2006-3368	5.3	0.266	3.422	0.332
ftp_rhost(1,2)	CVE-2008-1396	5.3	0.235	4.582	0.186
LICQ_remote_to_user(1,3)	CVE-2001-0439	7.3	0.206	4.299	0.323
Rsh_login(1,1)	CVE-1999-0180	7.3	0.184	5.408	0.175
LICQ_remote_to_user(2,3)	CVE-2001-0439	7.3	0.154	4.109	0.312

and exploit vulnerability instances CVE-2001-1030 (0,1), CVE-2001-1030 (0,3). In the mitigation plans of top 5 vulnerabilities (for the example attack graph illustrated in Figure 3.11) based on P, R and IRCR three vulnerabilities are common. Even though the mitigation plan is 60% same for all the three risk assessment methods, the vulnerability patching order varies.

A vulnerability assessment is a process of identifying, quantifying, and prioritizing the vulnerabilities in a system [31], [92]. As a large number of vulnerabilities are there in a network to fix/patch, a key question for security manager is which vulnerabilities to prioritize. Therefore, the need for vulnerability prioritization in organizations is widely recognized. The primary goal of the IRCR-based vulnerability rating system is to separate vulnerabilities from each other as far as possible according to their effects/influence. Comparative analysis of IRCR with existing vulnerability rating systems demonstrates that IRCR works well. IRCR combines the advantages of state-of-the-art vulnerability rating systems such as CVSS [2], cumulative probability

(P [26], cumulative resistance (R) [27], relative cumulative risk (Relative Cumulative Risk (RCR)) [33]. The results show that the quality of the vulnerability risk score can be improved with the help of network risk conditions. The result of the new proximity-based vulnerability risk assessment method is more score diversity than CVSS, P, and R. These changes would help organizations and individuals better prioritize their responses to new vulnerabilities.

3.6 Computational Complexity

Among the available attack paths, finding the minimum resistance value path from the attacker's initial position is analogous to the single-source shortest path (SSSP) problem. In particular, the problem is similar to finding the shortest path from a single source in a directed weighted graph. The exploit-dependency attack graph, we generated for our example network is weighted directed acyclic graph (DAG). We have employed breadth-first search (BFS) algorithm to compute the minimum effort (resistance) attacker needs to spend in order to reach and exploit the vulnerability under consideration. With BFS, the proximity (minimum path resistance) of vulnerability from the attacker's initial position can be computed in $\mathcal{O}(E)$, where, E is the total number of edges in the exploit-dependency attack graph. Further, finding all predecessor sets PD_1, PD_2, \dots, PD_n for all n vulnerabilities in an attack graph can also be done in $\mathcal{O}(E)$ using the BFS. Computation of IRCR takes $\mathcal{O}(V)$, where V is the total number of exploitable vulnerabilities in the attack graph G . Since the number of vulnerabilities in the attack graph is a finite constant; hence $V = \mathcal{O}(n)$. Therefore, the computation time for the whole process of IRCR estimation is $\mathcal{O}(E)$. As the automatic generation of an attack graph is beyond the scope of this Chapter, computational complexity does not include the attack graph construction time and the time required for the vulnerability scanning.

3.7 Summary

In practice, the ultimate goal of the network administrator is to make systems free from vulnerabilities and immune to Cyber attacks. While assessing the security of an

entire network, it is not enough to consider the presence or absence of single-point (host only) vulnerabilities. Therefore, to evaluate the security of a whole network, security administrator's must examine the effects of interactions between local (host-only) vulnerabilities and find global (meta) vulnerabilities.

In this Chapter, we have proposed a comprehensive measure called Improved Relative Cumulative Risk (IRCR) for quantifying the security risk of each exploitable vulnerability in a dynamic network. IRCR incorporates CVSS Base Score, the proximity of vulnerability (in terms of path resistance) from the attackers initial position, vulnerability diversity along the attack path(s), and also the risk of the neighboring vulnerabilities (immediate predecessor from where the target vulnerability can be reached and exploited). Vulnerabilities are classified depending upon its severity levels measured by IRCR.

- Vulnerabilities with higher IRCR need to be patched with higher priority than the vulnerabilities with lower IRCR. In this way, IRCR helps administrator in deciding over the patching order of the vulnerabilities and in designing practical patching strategy. Patching of top-ranked vulnerabilities reduces the number of entry points into the network; prevent further attacks and ensures the security of the network resources. We have tested proposed IRCR metric on the synthetic network and experimental results show that IRCR efficiently computes the security risk of each exploitable vulnerability present in the system. The resultant IRCR score can be used for the direct comparison of exploitable vulnerabilities regarding their security risks, and hence for vulnerability prioritization. It helps administrators in accurately determining top vulnerabilities and in prioritizing vulnerability remediation activities accordingly.
- The proposed proximity-based risk assessment method is sensitive to the network (or security) events such as system reconfiguration, change in the access control policies, service connectivities, installation of vulnerable software/services, the disclosure of new vulnerabilities, etc. Because of such network dynamics, there could be a change in the one or more network risk conditions. Consequently, the security risk posed by the exploitable system vulnerabilities varies over the time. In the best-case scenario, the vulnerability becomes inactive due to the unavailability of one or more required initial conditions. CVSS

does not accommodate such change in the vulnerability risk level. However, for a given vulnerability, IRCR takes into account various risk conditions about the target network and therefore efficiently captures the actual risk posed by the vulnerability. In other words, to say that IRCR is sensitive to the network dynamics.

- Network security principles such as the principle of least privileges, network segregation, defense-in-depth, etc., alters one or more network risk conditions. In particular, for a given vulnerability, the network risk conditions such as vulnerability distance from the attacker's initial position, exploit diversity along the attack paths, the number of predecessor vulnerabilities, etc., are the function of network security principles. As our proposed risk assessment approach uses various network conditions, IRCR agrees well with the network security policies. However, CVSS shows little relevance to such security principles. Therefore, as a recommendation, we suggest that network risk conditions that affect the success of an adversary should be incorporated into the CVSS metric group.
- Due to the unavailability of standard test graphs in the attack graph domain, the experimental evaluation of the proposed IRCR metric, has been performed using a small test network and case studies. We have compared IRCR with previously proposed attack graph-based metrics such as cumulative probability (P) [26] and cumulative resistance (R) [27]. We have chosen these two because just like IRCR; they take into account all the vulnerabilities between attacker's initial position and the target vulnerability. Moreover, P and R captures the conjunctive/disjunctive relationship between exploitable vulnerabilities. The quantitative comparison of IRCR with other similar work is not presented because of the lack (unavailability) of the commonly accepted indicators that compares existing security evaluation metrics. Instead, we compared IRCR qualitatively with the similar work in Chapter 2.
- Since IRCR is a relative score, it should not be interpreted in an absolute sense. The IRCR of each exploitable vulnerability makes sense only when it is computed for the same network and then compared. Therefore, IRCR should not be used for comparing two or more different systems/networks in terms of their security strength.
- There are many directions for extending the work presented in this Chapter. As an immediate future work, we propose to investigate the reduction in at-

tacker's work factor (vulnerability resistance) because of repetition of vulnerabilities along the attack paths. Such reduction in attacker's work factor/effort could be different for different classes of vulnerabilities. If it is, then the reduction in attackers effort is a function of vulnerability types and their repetitiveness. Further, we want to improve the IRCR metric to make it accurate, flexible and usable. Moreover, we will use it for investigating specific aspects of network security.

Chapter 4

Diversification of Software/Services for Increasing the Network Robustness Against Zero-day Attacks

This Chapter ¹ presents metrics to assess the diversification level of attack path(s) in a resource graph. We propose a metric to compute intra-path diversity and two more metrics called *uniqueness* and *overlap*, to measure inter-path diversity. Our objective is to evaluate the quality of each attack path in terms of the resistance posed by each of them to an adversary. Uniqueness measures the quality of being the novel attack path. Such novel attack scenarios pose more resistance to the adversary during network intrusion. Whereas, overlap measures the degree of overlap in terms of shared resources, attack tools, and techniques, etc. of a particular attack path with other paths. Attack paths with higher overlap act as the focal points for network hardening. Applying such metrics to existing network security practices produce actionable knowledge that can be utilized for increasing the system robustness against zero-day attacks. We identify attack paths in which one or more vulnerabilities could be exploited more than once. Next, we identify all the repeated software/services along the attack paths that need to be diversified. To cater to this problem, we propose an attack path diversification algorithm. Identified duplicated services are replaced with functionally equivalent al-

¹This Chapter is based on the work “Exploiting curse of diversity for improved network security” published in the *Proceedings of the 4th International Conference on Advances in Computing, Communications and Informatics, ICACCI-2015*, pages 1975-1981, 2015.

ternatives in such a way that an adversary should require an individual and independent effort for exploiting each vulnerability along the attack paths. The effectiveness and usability of the proposed diversification technique are demonstrated through a small case study. Experimental results show that such additional metrics for determining the diversification level of each attack paths would undoubtedly benefit security administrators in increasing the network robustness against the zero-day attacks.

4.1 Introduction

The robustness of biological systems against the spread of disease or infection is highly dependent on the existing species diversity. Higher the species diversity more robust the system is and vice versa. Necessarily, each species in biological systems poses a unique set of immunological defenses. However, in contrary to the biological systems, computer systems are remarkably less diverse. Such lack of diversity poses serious threats (risks) to the existing homogeneous computer networks.

Essentially, an adversary learns with the initial system compromises and then applies the learned knowledge to compromise the subsequent systems in a network with less effort and time. An exploit engineered to take advantage of a particular vulnerability could be leveraged on many other systems to multiply the effect of an attack. While attackers have a low success rate, they compensate for it in volume. The existence of the same vulnerability on multiple systems in an enterprise network greatly benefits the adversary because she can gain incremental access to the critical business resources with relative ease.

If resource diversity is not well planned or not strategically defined in the deployed network configurations, it offers an adversary more opportunities to compromise the target. In particular, misplaced diversity may facilitate the attacker in compromising the security of the underlying network [9]. Employing resource diversity as a tool for increasing the system robustness against the zero-day attacks may lead to other side effects such as performance overhead, loss of usability, an increase in cost/effort required for network management, etc. We can call such side effects altogether as “Curse of Diversity”. Despite these side effects, employing the resource diversity in real time implementations is advantageous, for example, in Data Center (DC), Disaster Recovery (DR) sites, and Near Site (NS) where the security manager needs to use

different versions of software/services. Otherwise, if one site is compromised with some attack, the same attack can also be exercised against other DR sites if the same environment is maintained.

4.2 Motivation

Attack graph-based security metric provides security-relevant vital information that can be acted upon and prompt appropriate security countermeasures to deter potential multistage attacks [134], [135]. However, all the benefits of existing metrics become a potential defect when the well-secured network configuration is vulnerable to the zero-day attacks. In general, unknown (zero-day) vulnerabilities are considered immeasurable due to the less predictable nature of software errors. Therefore, the research on security metrics has been hampered by difficulties in handling the zero-day attacks wherein an adversary exploits multiple zero-day vulnerabilities. It questions the usefulness of the existing security metrics because a more secure network configuration would be of little value if it is equally vulnerable to zero-day attacks.

Wang et al. [30], [46] addressed the shortcomings of existing security metrics. They proposed k -zero day safety metric which primarily considers the minimum number of different zero-day vulnerabilities needed to be exploited by an adversary to compromise the target resource successfully. Higher the count, more robust the network is since the probability of having the unknown vulnerabilities available, applicable, and exploitable at all the same time is significantly lower. Depending on the k -zero day safety metric, Wang et al. [9] proposed least attacking effort-based diversity metric to measure network's capability in resisting intrusions or malware infection that employ multiple zero-day attacks. The metric is capable of assessing the network robustness against the zero-day attacks.

These advances (i.e. [9], [30], [46]) demonstrate that through efficient diversification of the network services security administrator can increase system robustness in the face of possible zero-day attacks. Therefore, additional metrics for determining the diversification level of each attack paths would undoubtedly benefit such systems. Further, algorithms for the identification and diversification of the repeated services along the attack paths could enhance the usefulness of such approaches.

Furthermore, the case of misplaced diversity is prevalent in today's computer networks since the deployed network configurations are not security conscious. It results in multiple loopholes that in turn provide an adversary more ways to compromise a system. Lack of security metrics that guides administrator on diversifying enterprise network complicates the situation. The side effect of resultant unplanned or non-strategic diversification is that it provides an adversary more opportunities to compromise the system. Hence, there is an urgent need of metric that can tell the current diversification level of each attack path and helps in figuring out the attack paths that needs immediate attention. The lack of such metrics in measuring the diversification level of attack paths motivates our work.

4.3 Running Example

The topology of the test network is shown in Figure 4.1, which is same as the network topology used in [9]. There are Four machines viz. $Host_1$, $Host_2$, $Host_3$, and $Host_4$ located within Two subnets. $Host_4$ is attackers target machine. The attacker on $Host_0$ is a malicious entity in the external network and her goal is to gain root-level privileges on $Host_4$ by exploiting zero-day vulnerability present in either *http* or *rsh* running over it. The job of firewalls is to separate internal network from the Internet. There are 2 filtering devices: (1) a DMZ filtering device (i.e. $Firewall_1$) to filter external network connections that are destined for DMZ network, and (2) an internal filtering device (i.e. $Firewall_2$) to filter DMZ connections, which are destined for internal networks. Each of the network hosts except $Host_0$ running services that are remotely accessible and we assume all these services have potential zero-day vulnerabilities instead of known reported vulnerabilities.

To build intuition about example network shown in Figure 4.1, we made following assumptions:

- $Host_1$ and $Host_2$ offers *http* service
- $Host_3$ offers *ssh* service
- $Host_4$ offers both *http* and *rsh* service
- $Firewall_1$ allows any external host to only access services running on host $Host_1$. Connections to all other services/ports on other hosts are blocked.

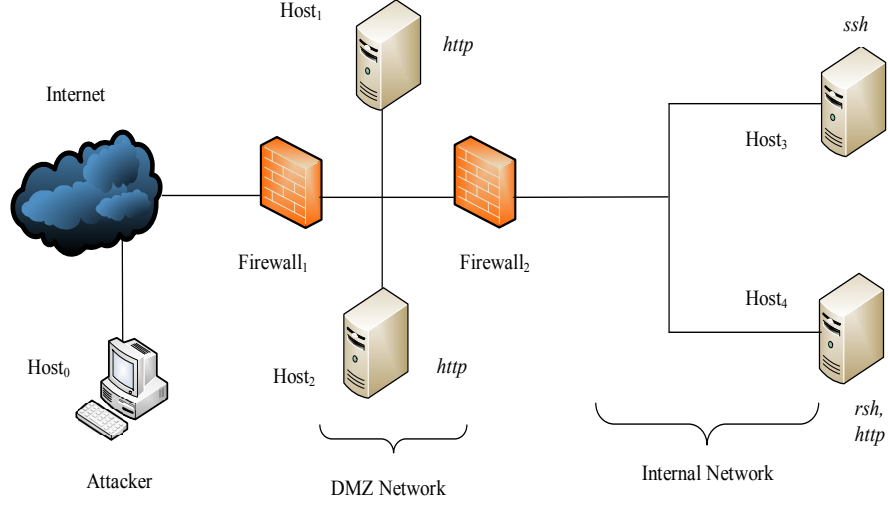


Figure 4.1: A test network (adapted from [9])

- *Firewall₂* allows Hosts within the DMZ (i.e. *Host₁*, *Host₂*) to connect to only *Host₃*.
- All resources, i.e. services and firewalls in test network are potentially vulnerable to zero-day attacks
- Security manager have enough resources to sustain adaptation or support diversification. In other words, the enterprise has adequate configuration space (opportunistic diversity [161]) available for each installed software/service.
- Attack surface metric (*ASM*) [139] is ready for use for all the software/services installed over the enterprise network.

A goal-oriented resource graph (zero-day attack graph) generated for the test network is shown in Figure 4.2. Unlike traditional attack graphs model [21], [17], the resource graph models zero-day vulnerabilities. As shown in Figure 4.2, each pair in a resource graph represents a security related condition. For example, a network connectivity $\langle source, destination \rangle$ and attackers privilege on host $\langle privilege, host \rangle$. Each triple (inside the rectangular box), i.e. $\langle resource/service, sourcehost, destinationhost \rangle$ represents the potential exploit of the resource. For the identifiers, numbers in pairs and triples specify related host. For example, $\langle user, 0 \rangle$ indicates that an adversary has root privileges on *Host₀*. The execution of vulnerability in *http* service from *Host₁* to *Host₂* is shown as $\langle http, 1, 2 \rangle$.

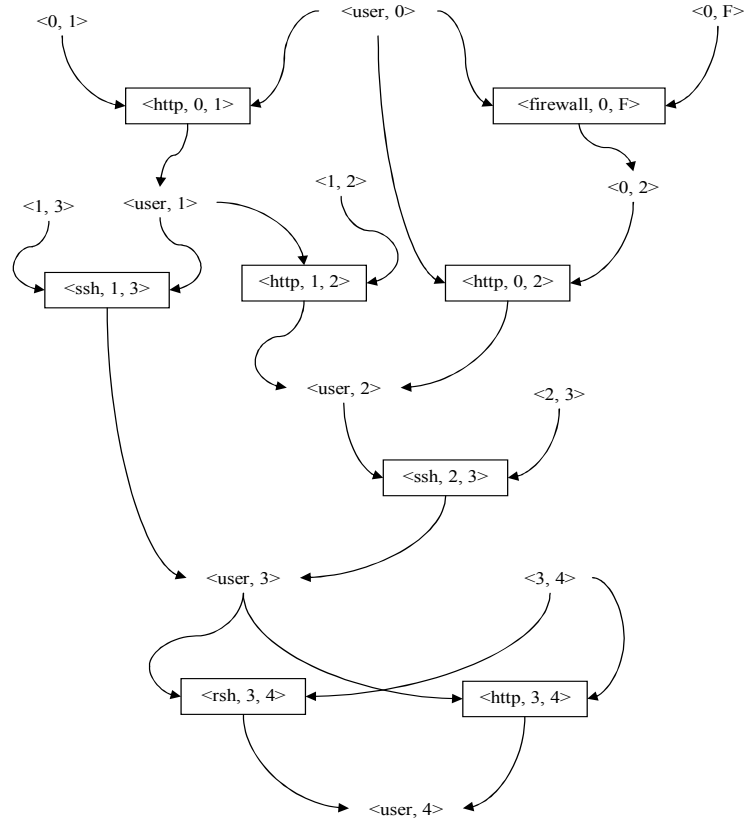


Figure 4.2: A resource graph for the test network (adapted from [9])

In resource graph, security conditions are of two types, namely *initial* and *intermediate* conditions. Initial conditions never be a post-condition of any exploit, but become a necessary pre-condition. An example of such initial conditions is accessibility rules ($\langle user, 0 \rangle$) and network configuration ($\langle 1, 2 \rangle$). Intermediate conditions can be both pre-condition and post-condition for some exploits. Edges in a resource graph point from pre-condition to a zero-day exploit and from the exploit to a post-condition.

For the successful exploitation of a zero-day vulnerability, all of its pre-conditions must be satisfied, and once exploited results in the generation of post-conditions successfully. As shown in Table 4.1, there are total Six attack paths available to an adversary in order to reach goal state (i.e. $\langle user, 4 \rangle$). Each attack path in a resource graph is a sequence of exploits chained together. The Second column (No. of Steps) in Table 4.1 indicates the number of zero-day exploits, an adversary need to successfully execute along the attack path in order to the reach the target. The number of distinct

Table 4.1: Attack paths in a resource graph

Attack Path	No. of Steps	No. of Resources	Diversity Metric (d)	Diversification Order (\mathcal{D})
1. $\langle http, 0, 1 \rangle \rightarrow \langle ssh, 1, 3 \rangle \rightarrow \langle rsh, 3, 4 \rangle$	3	3	1	N/A
2. $\langle http, 0, 1 \rangle \rightarrow \langle ssh, 1, 3 \rangle \rightarrow \langle \mathbf{http}, 3, 4 \rangle$	3	2	0.67	2
3. $\langle http, 0, 1 \rangle \rightarrow \langle \mathbf{http}, 1, 2 \rangle \rightarrow \langle ssh, 2, 3 \rangle \rightarrow \langle rsh, 3, 4 \rangle$	4	3	0.75	3
4. $\langle http, 0, 1 \rangle \rightarrow \langle \mathbf{http}, 1, 2 \rangle \rightarrow \langle ssh, 2, 3 \rangle \rightarrow \langle \mathbf{http}, 3, 4 \rangle$	4	2	0.5	1
5. $\langle firewall, 0, F \rangle \rightarrow \langle http, 0, 2 \rangle \rightarrow \langle ssh, 2, 3 \rangle \rightarrow \langle rsh, 3, 4 \rangle$	4	4	1	N/A
6. $\langle firewall, 0, F \rangle \rightarrow \langle http, 0, 2 \rangle \rightarrow \langle ssh, 2, 3 \rangle \rightarrow \langle \mathbf{http}, 3, 4 \rangle$	4	3	0.75	4

resources/services exploited in each attack path are shown in Third column (i.e. No. of Resources). As shown in Table 4.1, First and Fifth attack path is completely diversified. However, there is an exploitation of the one or more zero-day vulnerabilities more than once in the remaining attack paths.

For an enterprise network of reasonable size, the resource graph is enormous and complex. The number of vertices and edges grows combinatorially, at least quadratic in the number of hosts multiplied by the number of software/services installed on those hosts. In such scenario, determining the services that are installed on many hosts in an enterprise network is very crucial in stopping or delaying a possible attack since the exploitation of one service enable adversaries to exercise the same exploit at many places. To do this, we first need to identify attack paths which are adequately diversified and the other attack paths that need diversification. In this process of classifying the attack paths based on their intrinsic diversification, we need a good set of metrics.

4.4 Proposed Metrics

In this Section, we propose a metric to compute intra-path diversity and two more metrics called *uniqueness* and *overlap*, to measure inter-path diversity. The diversity of an attack path set, Π , can be measured in two ways:

1. Within the individual attack path $\pi_a \in \Pi$
2. Between any two attack paths $\pi_a, \pi_b \in \Pi$ available to an adversary.

Here, Π is the set of attack paths in a resource graph (as shown in Figure 4.2) generated for a given network.

4.4.1 Measuring Intra-path Diversity

Intra-path diversity metric assess the diversification level of each attack path in the resource graph generated for a given network. The metric assigns the numeric score to each attack path and rank them in terms of their diversification level. The metric is based on the number of attack steps required (i.e. the total number of vulnerabilities exploited) and the number of distinct steps (i.e. the number of vulnerabilities targeting distinct resources) in the attack path. The Intra-path diversity metric that determines the diversification level of attack path π_a is defined as:

$$d(\pi_a) = 1 - \left(\frac{\text{No. of steps} - \text{No. of resources}}{\text{No. of steps}} \right) \quad (4.1)$$

Here, $d(\pi_a)$ range on the scale from 0 to 1. For truly diversified attack paths the d value is always 1. The higher value indicates that the attack path is adequately diversified. A diversification order \mathcal{D} is assigned to each attack path based on their d value. For truly diversified attack paths, the diversification order (\mathcal{D}) is represented by N/A (as shown in Fifth column of Table 4.1). For the attack path whose d value is minimum has assigned a diversification order (\mathcal{D}) of 1 (Fourth attack path in Table 4.1). For the attack paths having same d value, we can order them either in top-to-bottom or bottom-to-top manner (attack path 3 and 6 in Table 4.1). Attack path with $\mathcal{D} = 1$ is the first candidate for diversification during the process of network hardening.

4.4.2 Measuring Inter-path Diversity

Availability of alternate attack paths to an adversary challenges the administrator's decision of focusing on the single attack path for network hardening. Such individual path-based system hardening solutions do not stop or deter an adversary from incrementally compromising the security of the target network. It is because the adversary is capable of taking alternate attack paths during real-time network intrusion. To evaluate the distinctness of exploits within individual attack path, researchers have proposed

diversity metrics [46], [30]. However, there is no metric to find out how many vulnerabilities/vulnerable resources are common to a pair of attack paths. To what extent attack paths overlap? To what degree they are unique to each other?

The origin of these questions lies in the efficient diversification of the installed software/services so that the network is more robust to both zero-day and well-known attacks. If there is a metric support to detect the shared/overlapped portion between a pair of attack paths, an administrator can identify the resources common to both the attack paths in a pair. She can determine what sort of resources, attack methods, tools, and techniques common to both the attack paths.

As administrator's goal is to evaluate the distinctness of alternative attack paths in the resource graph, she needs metrics that can measure the set difference between the adversarial actions along the attack paths. Inter-path diversity metrics can be used to evaluate the quality of each attack path in terms of the resistance posed by them to zero-day or well-known attacks. Such diversity metrics can measure the distance between attack paths using set difference or set intersection of their resources. The ultimate goal of such diversity metrics is to evaluate the performance of each attack path in terms of the amount of resistance posed by each path to an adversary during network intrusion.

In this Chapter, we evaluate the quality of each attack path using diversity metrics, which can be measured as the set difference or set intersection of network services along the attack paths. Usually, similar target serves great advantage for adversaries in launching attacks and incrementally compromising the critical enterprise resources. The goal is to diversify resources along the attack paths to deter potential adversaries while maintaining operational integrity of a system. To cater to this problem, we need diversity metrics that can measure the set difference between the adversarial actions along the attack paths.

Uniqueness (u) metric helps administrator in finding novel attack paths and is defined as:

$$u(\Pi) = \sum_{\pi_a, \pi_b \in \Pi, \pi_a \neq \pi_b} \begin{cases} 0; & \text{if } \pi_a \setminus \pi_b = \Phi \text{ or } \pi_b \subset \pi_a \\ 1; & \text{otherwise} \end{cases} \quad (4.2)$$

The uniqueness (u) measure captures the way in which paths do not subsume/absorb each other. The attack path with higher uniqueness (u) score poses more

resistance to the adversary during network intrusion.

However, the Overlap (o) takes the set intersection of the two attack paths as:

$$o(\pi_a, \pi_b) = \pi_a \cap \pi_b \quad (4.3)$$

If the sum of overlap score (o) is higher for a particular attack path, then the path shares more resources (i.e. services, attack techniques, redundant effort, etc.) with the other attack paths. Such paths with a high number of overlapping points could be the focal point for network hardening. In other words, an attack path having a greater overlap (o) score with respect to the other attack paths is critical to the network hardening.

Here, both uniqueness (u) and overlap (o) are relative scores, i.e. with respect to other attack paths.

4.5 Results and Analysis

In this Section, we demonstrate the use of proposed intra-path (d) and inter-path (u and o) diversity metrics for diversifying the resources of the test network shown in Figure 4.1.

4.5.1 Use of Intra-path Diversity Metric (d) for Resource Diversification

By traversing the resource graph shown in Figure 4.2, we have extracted all the attack paths that end up in a critical host, i.e. $Host_4$. If each of these attack paths is diversified adequately, then the adversary has to spend an individual and independent effort in successfully exploiting vulnerabilities coming across such attack paths. It greatly benefits the defenders because the adequately diversified attack path either able to stop an ongoing intrusion in between or can delay/limit the speed of network intrusion.

As we assumed the higher configuration space (i.e. significant opportunistic diversity [161]) available for each software/service installed over the network, there may be various options available (i.e. larger solution space) to an administrator while diversifying each attack paths. To reduce the complexity of the problem, we need to consider

the factors like the security strength or resistance posed by the alternative functionality equivalent software to the Cyber attacks. In real time network, there may be some software/services that may not be replaceable, as there may not be other services with similar functionality, i.e. the configuration space for such services is null. In such case, an organization has to keep running vulnerable service and live with the vulnerabilities in it.

Intra-path diversity metric (d) proposed in Section 4.4.1 assess the diversification level of each attack path in a resource graph shown in Figure 4.2. Our next objective is to determine the repeated service(s) in attack path(s) that needs to be diversified. The decision of “service replacement with other functionally equivalent alternative” is guided by the attack surface metric (ASM) [139]. The flow chart of our attack path diversification approach is shown in Figure 4.3. Accordingly, Table 4.2 shows the parameters used.

As evident from Figure 4.3, our proposal consists of two phases. In the First phase, all attack paths are extracted from the resource graph. Then, the intra-path diversity metric d is applied to each attack path to rank them based on their current diversification level. Based on d value, the diversification order \mathcal{D} is determined. Exploits are extracted separately from the attack paths for which $d = \max$ and also from the remaining attack paths which are not sufficiently diversified.

In the Second phase, based on diversification order (Fifth column of Table 4.1), the path with the lowest \mathcal{D} value will be chosen for diversification. A set of repeated services in the selected attack path are obtained. Then, the configuration space (i.e. a set of alternative software/services with similar functionalities) for such repeated services is determined. Accordingly, attack surface metric (ASM) [139] is applied to the configuration space of each repeated service to find the alternative service which replaces the original one. The intuition behind using attack surface metric (ASM) for service selection is that the newly introduced service should pose more resistance (in terms of time and effort) to an adversary during network compromise. Finally, the repeated service(s) in an attack path is replaced with the identified one using attack surface metric, and diversity metric (d) is re-calculated for all the attack paths. The above procedure is repeated until the d value is maximum for all the attack paths.

As shown in Table 4.1, Fourth column indicates the current diversification level (i.e. d) of each attack path in a resource graph. As evident from the Fourth column,

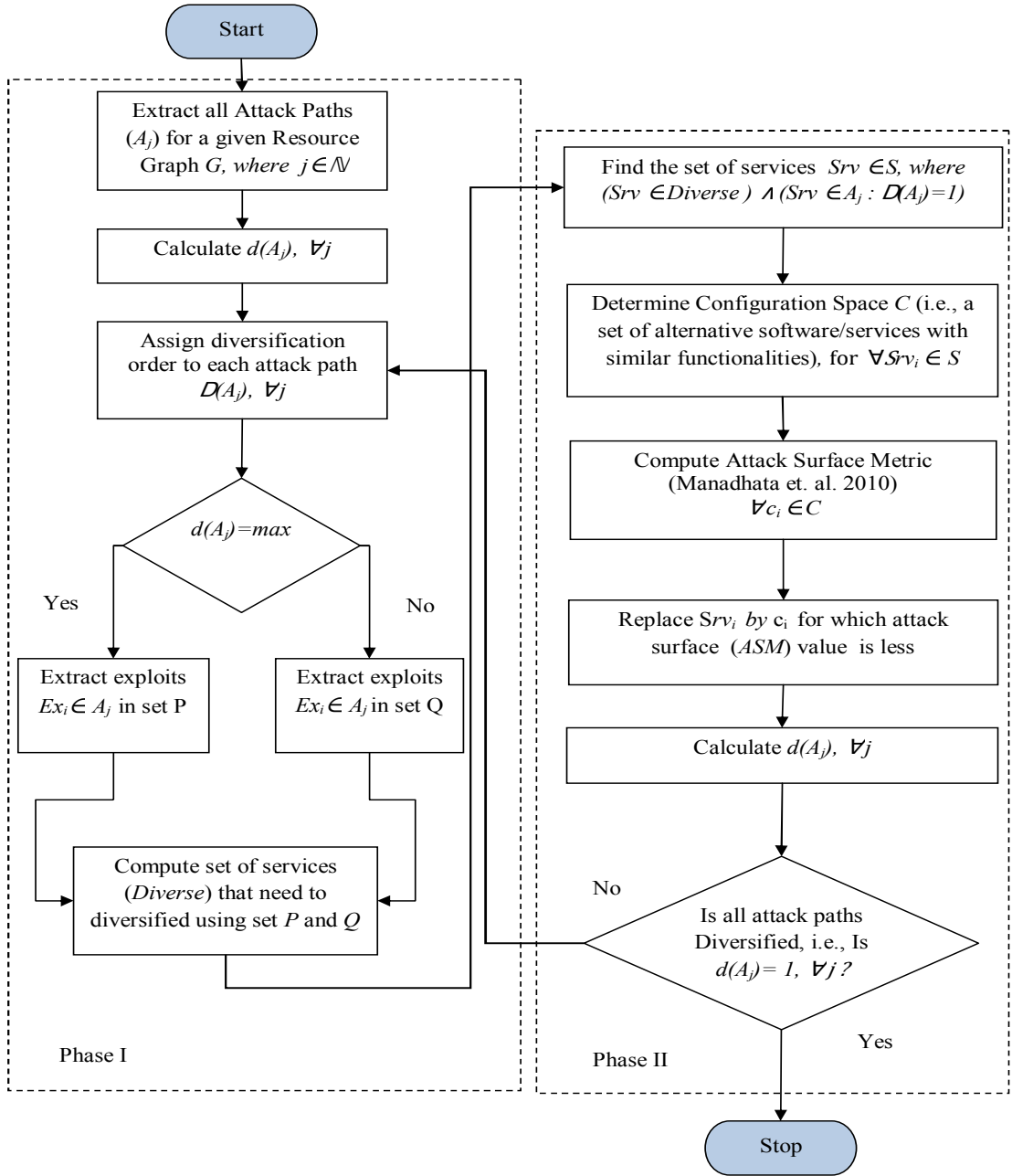


Figure 4.3: A flow chart for attack path diversification

First and Fifth attack path is truly diversified. It means, execution of each exploit along these attack paths require an independent effort. The d value for the remaining attack paths signifies how much diversification is needed (in terms of the number of service

Table 4.2: Algorithm Parameters

Abbreviation	Explanation
Ex	An exploit
Srv	Service
Src	Source Host
Dst	Destination Host
A_j	An attack path
d	Diversification Metric
ASM	Attack Surface Metric [139]
\mathcal{D}	Diversification Order

replacement) in each attack path to make them truly diverse. The lower the value of d for an attack path, more services need to be diversified in that path. The procedure $findTarget(G)$ (Algorithm 4.2) identifies the services that need to be diversified in attack paths for which the $d < 1$. The highlighted exploits in Table 4.1 represents the vulnerable services that need to be diversified for making a corresponding attack path truly diverse.

As shown in Table 4.1, the *http* service running over the $Host_2$ and $Host_4$ needs to be diversified with candidate services available in configuration space of *http*. Firstly, we select the *http* service running over the $Host_2$ for diversification. The chosen service can be replaced with either *IIS*, *Apache*, *Oracle* or *MySql* whichever feasible as per their attack surface metric (*ASM*) value. In algorithm 4.1, steps 5-12 identifies and replaces the *http* service running over $Host_2$ with suitable functionality equivalent alternative.

As evident from Table 4.1, for our test network, it is worthless to diversify the *http* service running on $Host_2$. Diversifying the *http* on $Host_2$ will not work. It is because, the shortest path (here, Fifth and Sixth attack path) is already available to an adversary to compromise the $Host_2$. Instead, the security administrator can apply the detection and prevention mechanisms for *http* service or enforce security service on it. After

Algorithm 4.1 Attack path diversification (G): diversification of software/services along the attack paths in a resource graph (G)

Input: $G \rightarrow$ resource graph generated for a test network

Output: Resource graph G with adequately diversified attack paths

- 1: Extract all the attack paths A_j from the resource graph G ▷ using backward algorithm
 - 2: Compute diversification level d of each attack path in resource graph G , i.e. $d(A_j), \forall j$
 - 3: Let $targetServices$ is a set of tuples $\langle S, H \rangle$, where each tuple represents a service S running on host H .
 - 4: $targetServices \leftarrow findTarget(G)$
 - 5: Determine the set of services Srv_i , where $(Srv_i \in targetServices) \wedge (Srv_i \in A_j | \mathcal{D}(A_j) = 1)$
 - 6: **for all** $Srv_i \in S$ **do**
 - 7: Determine Configuration Space C (Set of all alternative softwares/services with similar functionalities)
 - 8: **for all** $c_i \in C$ **do**
 - 9: Compute $ASM(c_i)$ ▷ Compute Attack Surface Metric [139] for each candidate service available in the configuration space C of service Srv_i
 - 10: **end for**
 - 11: Replace service Srv_i by c_i for which ASM is less
 - 12: **end for**
 - 13: Compute diversity metric d , i.e. $d(A_j), \forall j$
 - 14: **if** $(d(A_j) = 1, \forall j)$ **then**
 - 15: Resource graph is truly diversified
 - 16: **else**
 - 17: go to step 4
 - 18: **end if**
-

enforcing security services for *http* the exploit $\langle http, 1, 2 \rangle$ become $s\langle http, 1, 2 \rangle$. Assuming all services are mission-critical, an administrator can apply detection and prevention mechanisms and enforce security services for such redundant services. Now, both $\langle http, 0, 1 \rangle$ and $s\langle http, 1, 2 \rangle$ along the Third and Fourth attack path will pose different amount of resistance to the adversary. Same will be true for the *http* service

Algorithm 4.2 findTarget(G): Identification of repeated services along the attack paths in a given resource graph.

Input: $G \rightarrow$ resource graph generated for a test network

Output: $Diverse \rightarrow$ is a set of tuples $\langle S, H \rangle$, where each tuple represents a service S running on host H .

- 1: Let $Ex\langle Srv, Src, Dst \rangle$ be the triple representing an exploit in the resource graph. It represents the vulnerable service Srv running over the host Dst and prone to be exploited by an adversary from the host Src .
 - 2: $Diverse \leftarrow \phi$
 - 3: Assign diversification order to each attack path, i.e. $\mathcal{D}(A_j), \forall j$
 - 4: **if** ($d(A_j) = max$) **then**
 - 5: $P \leftarrow Ex_i \in A_j$
 - 6: **else**
 - 7: $Q \leftarrow Ex_i \in A_j$
 - 8: **end if**
 - 9: **for all** $Ex_i \in P$ **do**
 - 10: **for all** $Ex_j \in Q$ **do**
 - 11: **if** $\{((Srv \in Ex_i) = (Srv \in Ex_j)) \wedge ((Dst \in Ex_i) \neq (Dst \in Ex_j))\}$ **then**
 - 12: $Diverse \leftarrow \langle Srv, Dst \rangle$
 - 13: **end if**
 - 14: **end for**
 - 15: **end for**
 - 16: **return** $Diverse \triangleright$ A set of repeated services along the attack paths in a resource graph.
-

running on $Host_4$.

Once the *http* service(s) on $Host_2$ is secured, again diversity metric (d) for each path is computed and diversification order (\mathcal{D}) is determined. Attack path with smaller d is diversified, and this procedure is repeated until all the attack paths are adequately diversified. The net effect of diversifying services in each attack path is that the presence of diverse/secured services results in slower progress in the attack graph by attackers.

Algorithm 4.1 precisely identifies and diversifies all the attack paths that are not adequately diversified in the resource graph G . Proposed algorithm guarantees to terminate after a finite number of steps as the termination condition is well-defined. In particular, proposed algorithm successfully terminates after a finite number of iterations. To evaluate this, we tested the algorithm with different resource graphs (generated for test networks) with a different number of exploits and a different number of initial conditions. We found that *Attack path diversification (G)* algorithm precisely identifies all the attack paths that are not adequately diversified. Once identified, proposed algorithm diversifies/secure the repeated services along the attack paths to increase the network robustness against the zero-day attacks.

Algorithm 4.1 is conditioned on the availability of adequate/sufficient configuration space for each software/services running over the enterprise network. However, in practice, this cannot be true always. In other words, there may be some software/services that may not be replaceable, as there may not be other alternative software with similar functionalities available (i.e. configuration space is null for such services). Further, the efficacy and accuracy of proposed algorithm are dependent on the accuracy of the *ASM* metric and its availability for the variety of services in use across the enterprise networks.

4.5.2 Use of Inter-path Diversity Metrics (i.e. u and o) for Resource Diversification

In this Section, we explore how inter-path diversity metrics can also be used for network diversification.

For the resource graph shown in Figure 4.2, we have computed *uniqueness* (u) and *overlap* (o) score for each attack graph pair as shown in Table 4.3. A pair $\langle u, o \rangle$ represents the uniqueness (u) and overlap (o) of an attack path in a resource graph with respect to the other attack paths. As evident from the Table 4.3, the First, and Third attack path are unique with respect to the Sixth attack path and vice versa. In other words, the First and Sixth attack path does not subsume each other. It is also true for the attack path pair $\langle \text{Path 2}, \text{Path 6} \rangle$. Here, uniqueness (u) measures the quality of being the novel attack path. As evident from the Table 4.3, overlap (o) score for the Fifth attack path is highest among all the attack paths. The resources along this attack

path are also common to the other attack paths. It means if an adversary is capable of successfully compromising all the resources along the Fifth attack path, then she can easily follow all the other attack paths without any difficulty. An attack path with higher overlap score act as the focal point for network hardening.

Table 4.3: Uniqueness (u) and overlap (o) score of attack paths in a resource graph.

Attack Path	Path 1	Path 2	Path 3	Path 4	Path 5	Path 6	Σ
<i>Path 1</i>	/	$\langle 0, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 0, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 1, 12 \rangle$
Path 2	$\langle 0, 2 \rangle$	/	$\langle 0, 2 \rangle$	$\langle 0, 2 \rangle$	$\langle 0, 2 \rangle$	$\langle 0, 2 \rangle$	$\langle 0, 10 \rangle$
Path 3	$\langle 0, 3 \rangle$	$\langle 0, 2 \rangle$	/	$\langle 0, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 1, 12 \rangle$
Path 4	$\langle 0, 2 \rangle$	$\langle 0, 2 \rangle$	$\langle 0, 2 \rangle$	/	$\langle 0, 2 \rangle$	$\langle 0, 2 \rangle$	$\langle 0, 10 \rangle$
<i>Path 5</i>	$\langle 0, 3 \rangle$	$\langle 0, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 0, 2 \rangle$	/	$\langle 0, 3 \rangle$	$\langle 0, 13 \rangle$
Path 6	$\langle 1, 2 \rangle$	$\langle 0, 2 \rangle$	$\langle 1, 2 \rangle$	$\langle 0, 2 \rangle$	$\langle 0, 3 \rangle$	/	$\langle 2, 11 \rangle$

Table 4.4: Attack paths in a resource graph post diversification of *http* service on *Host*₄.

Attack Path	No. of Steps	No. of Resources	Diversity Metric (d)
1. $\langle http, 0, 1 \rangle \rightarrow \langle ssh, 1, 3 \rangle \rightarrow \langle rsh, 3, 4 \rangle$	3	3	1
2. $\langle http, 0, 1 \rangle \rightarrow \langle ssh, 1, 3 \rangle \rightarrow s\langle http, 3, 4 \rangle$	3	3	1
3. $\langle http, 0, 1 \rangle \rightarrow \langle \mathbf{http}, 1, 2 \rangle \rightarrow \langle ssh, 2, 3 \rangle \rightarrow \langle rsh, 3, 4 \rangle$	4	3	0.75
4. $\langle http, 0, 1 \rangle \rightarrow \langle \mathbf{http}, 1, 2 \rangle \rightarrow \langle ssh, 2, 3 \rangle \rightarrow s\langle http, 3, 4 \rangle$	4	3	0.75
5. $\langle firewall, 0, F \rangle \rightarrow \langle http, 0, 2 \rangle \rightarrow \langle ssh, 2, 3 \rangle \rightarrow \langle rsh, 3, 4 \rangle$	4	4	1
6. $\langle firewall, 0, F \rangle \rightarrow \langle http, 0, 2 \rangle \rightarrow \langle ssh, 2, 3 \rangle \rightarrow s\langle http, 3, 4 \rangle$	4	4	1

As we discussed earlier, the First and Fifth attack path in the resource graph (Table 4.1) are completely diversified. Therefore, it is pointless to diversify resources along these paths. Although the Fifth attack path is completely diversified, its uniqueness (u) score is zero (as shown in Table 4.3). It is because it subsumes all the other attack paths. As our goal is to maximize the uniqueness (u) score of each attack path, the attack

path having least uniqueness (u) score will be the first candidate for diversification. As evident from Table 4.3, both Second and Fourth attack path will be the nominees. To break the tie, we make use of intra-path diversity metric d . Consequently, the repeated services along the Fourth attack path will be the candidates for diversification.

As discussed in Section 4.5.1, it is of no use to diversify the *http* service running over $Host_2$ and $Host_4$ as well. Instead, detection and prevention mechanisms are applied to enforce the security. After enforcing security for *http* service running over $Host_2$, exploit $\langle http, 1, 2 \rangle$ becomes $s\langle http, 1, 2 \rangle$ posing different amount of resistance to the adversary. The same will be true for the *http* service running on $Host_4$. Table 4.4 shows changes in the attack paths post diversification of *http* service on $Host_4$.

Table 4.5: Uniqueness (u) and overlap (o) score of attack paths in resource graph post diversification of *http* service on $Host_4$.

Attack Path	Path 1	Path 2	Path 3	Path 4	Path 5	Path 6	Σ
<i>Path 1</i>	/	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 3, 12 \rangle$
<i>Path 2</i>	$\langle 1, 2 \rangle$	/	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 3, 12 \rangle$
Path 3	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	/	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 3, 12 \rangle$
Path 4	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	/	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 3, 12 \rangle$
<i>Path 5</i>	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	/	$\langle 1, 3 \rangle$	$\langle 3, 13 \rangle$
<i>Path 6</i>	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 3 \rangle$	/	$\langle 3, 13 \rangle$

As evident from Table 4.4, except the Third and Fourth attack path, all the other attack paths are completely diversified. Because of enforcing security service on *http* running on $Host_4$, there is an increase in the uniqueness (u) and overlap (o) score of each attack path as shown in Table 4.5. Every attack path has the same u value. It is hard for an administrator to decide on the next attack path for diversification. In such case, intra-path diversity (d) metric will be helpful. Based on d value, the Third and Fourth will be the candidate attack paths that need to be diversified. The *http* service running over $Host_2$ will be the candidate service for diversification.

Just like previously done, apply the detection and prevention mechanisms for this service or enforce security service on it. Post securing the *http* service on $Host_2$, the

Table 4.6: Attack paths in a resource graph post diversification of *http* service on *Host₂*.

Attack Path	No. of Steps	No. of Resources	Diversity Metric (d)
1. $\langle http, 0, 1 \rangle \rightarrow \langle ssh, 1, 3 \rangle \rightarrow \langle rsh, 3, 4 \rangle$	3	3	1
2. $\langle http, 0, 1 \rangle \rightarrow \langle ssh, 1, 3 \rangle \rightarrow s\langle http, 3, 4 \rangle$	3	3	1
3. $\langle http, 0, 1 \rangle \rightarrow s\langle http, 1, 2 \rangle \rightarrow \langle ssh, 2, 3 \rangle \rightarrow \langle rsh, 3, 4 \rangle$	4	4	1
4. $\langle http, 0, 1 \rangle \rightarrow s\langle http, 1, 2 \rangle \rightarrow \langle ssh, 2, 3 \rangle \rightarrow s\langle http, 3, 4 \rangle$	4	4	1
5. $\langle firewall, 0, F \rangle \rightarrow \langle http, 0, 2 \rangle \rightarrow \langle ssh, 2, 3 \rangle \rightarrow \langle rsh, 3, 4 \rangle$	4	4	1
6. $\langle firewall, 0, F \rangle \rightarrow \langle http, 0, 2 \rangle \rightarrow \langle ssh, 2, 3 \rangle \rightarrow s\langle http, 3, 4 \rangle$	4	4	1

Table 4.7: Uniqueness (*u*) and overlap (*o*) score of attack paths in resource graph post diversification of *http* service on *Host₂*.

Attack Path	Path 1	Path 2	Path 3	Path 4	Path 5	Path 6	Σ
Path 1	/	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 3, 12 \rangle$
Path 2	$\langle 1, 2 \rangle$	/	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 3, 12 \rangle$
Path 3	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	/	$\langle 1, 3 \rangle$	$\langle 1, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 4, 13 \rangle$
Path 4	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 3 \rangle$	/	$\langle 1, 2 \rangle$	$\langle 1, 3 \rangle$	$\langle 4, 13 \rangle$
Path 5	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 1, 3 \rangle$	$\langle 1, 2 \rangle$	/	$\langle 1, 3 \rangle$	$\langle 4, 13 \rangle$
Path 6	$\langle 1, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 1, 2 \rangle$	$\langle 1, 3 \rangle$	$\langle 1, 3 \rangle$	/	$\langle 4, 13 \rangle$

attack paths in the resource graph are shown in the Table 4.6. As evident, all the attack paths in a resource graph are completely diversified. Now, each vulnerability along the attack path(s) poses different amount of resistance to the adversary. Table 4.7 shows the increase in uniqueness (*u*) value of each attack path due to the application of securing *http* service on *Host₂*. The attack paths in the majority of attack path pairs in resource graph do not subsume each other. The net effect of the service diversification is that an adversary has to spend independent and individual effort in exploiting each vulnerability along the attack paths. Such software/service diversification in enterprise networks is very crucial in stopping or delaying a potential multistage, multi-host attacks.

Figure 4.4a, and 4.4b shows the Uniqueness (u) and overlap (o) score of each attack paths in resource graph pre diversification, respectively. Whereas, Figure 4.5a and 4.5b shows the Uniqueness (u) and overlap (o) score of each attack paths in resource graph post diversification, respectively.

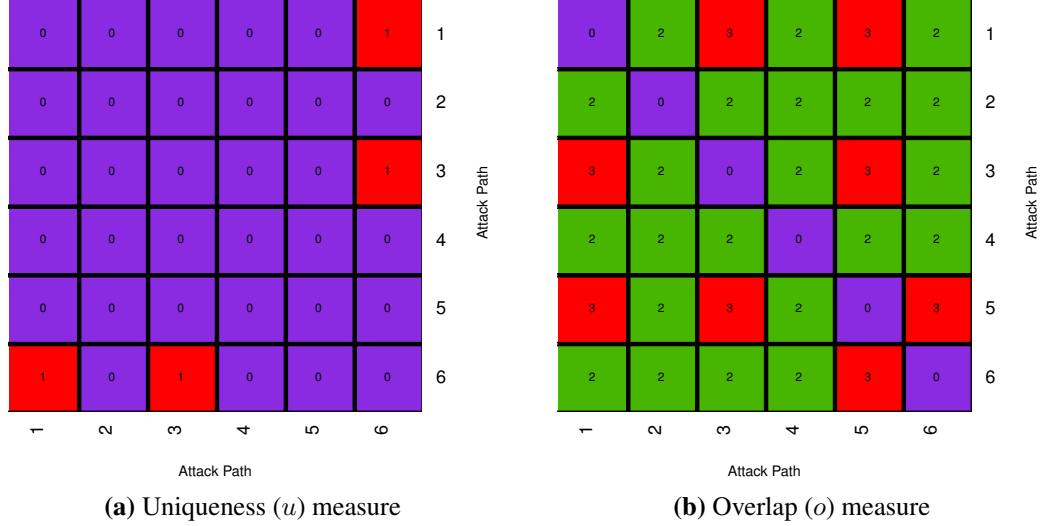


Figure 4.4: Uniqueness (u) and overlap (o) score of each attack path in a resource graph pre network diversification.

Further, Figure 4.6 and 4.7 shows the rate of growth of uniqueness (u) and overlap (o) measure, respectively in each iteration. Because of securing the *http* service on *Host₄* in the first iteration of network hardening, there is an increase in the uniqueness (u) score by a large amount for all the attack paths. It is because the uniqueness (u) captures the way in which paths do not subsume each other. As shown in Table 4.1, exploit $\langle http, 3, 4 \rangle$ is appeared in attack paths 2, 4, and 6. Therefore, the overlap (o) score of only these paths changed as shown in Figure 4.7 and it is because of set intersection property. On the other hand, there is an increase in the overlap score of Third and Fourth attack path after the second iteration, i.e. post diversification of *http* service on *Host₂*.

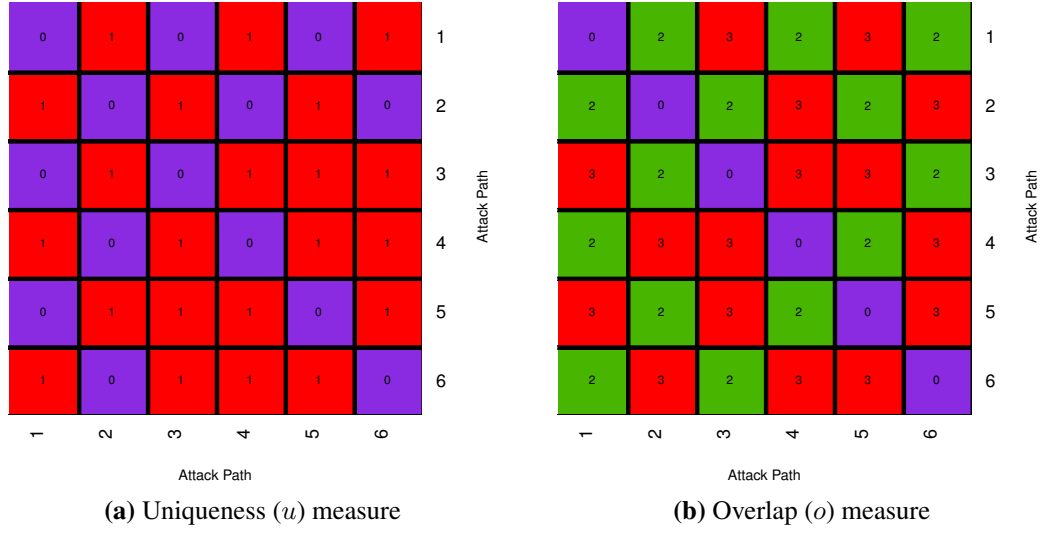


Figure 4.5: Uniqueness (u) and overlap (o) score of each attack path in a resource graph post network diversification.

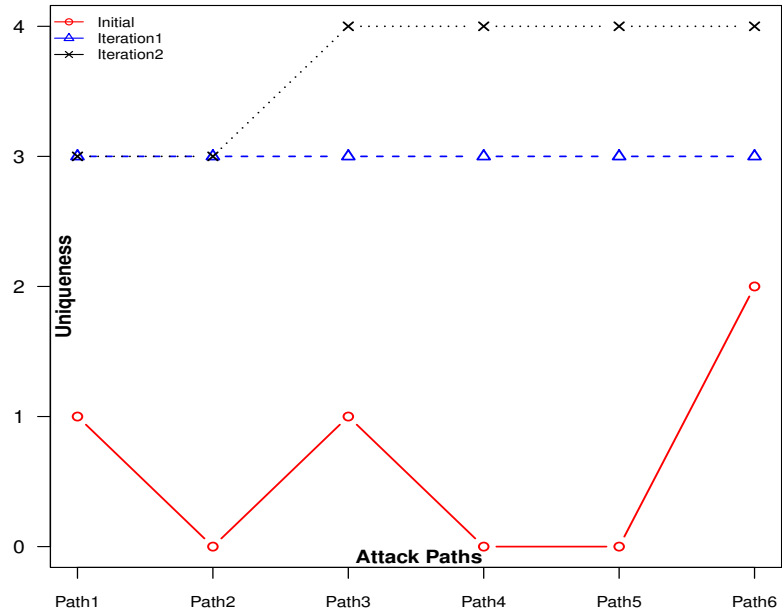


Figure 4.6: Rate of growth of uniqueness (u) measure

4.6 Summary

In this Chapter, we have proposed diversity metrics to assess the diversification level of each attack path in a resource graph generated for a given network. Further, we

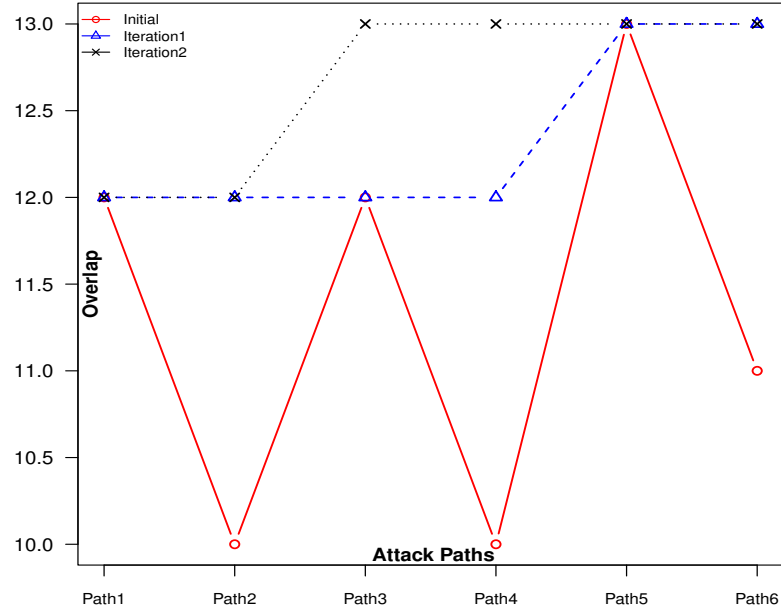


Figure 4.7: Rate of growth of overlap (o) measure

have proposed an algorithm to detect the network services that need to be diversified for increasing the robustness of enterprise networks against zero-day attacks. The decision of “service replacement with other functionally equivalent alternative candidate service” is guided by attack surface metric (*ASM*) [139]. The proposed solution provides a technique for network resource diversification by eliminating the problem caused by misplaced diversity.

In this Chapter, we assumed that an enterprise has enough resources to sustain diversification. In other words, the configuration space of installed network services is adequate/sufficient, i.e. the number of functionally equivalent alternatives are already available in sufficient numbers or quantities. All operational networks in an enterprises have some constraints like security budget, network/service uptime, and resource limitations like skill set, man-hour, etc. Consequently, our future work will be on cost-controlled network diversification.

Chapter 5

Assessing Temporal Variations in the Network Attack Surface

This Chapter ¹ focuses on the problem of assessing temporal variation in the attack surface of dynamic networks. We have used classical graph distance metrics based on the Maximum Common Subgraph (MCS), and Graph Edit Distance (GED) to quantify the distance between a pair of successive attack graphs generated for a dynamic network. We have used a *logical attack graph* [16], [21] as a security model to capture the attack surface of the underlying network. Since the attack graph is capable of successfully capturing the network attack surface, the distance between a pair of successive attack graphs (generated over the observed sampling interval) indicates the change in the network attack surface. We found that the MCS and GED based graph distance metrics are competitive with state-of-the-art attack graph-based metrics on all the three different network models, viz., Flat, External-Internal, and DMZ. The MCS and GED based metrics successfully capture the temporal variation in the attack surface and also generate an alert about the security events which are responsible for the change. Such graph distance metrics capture the impact of the network or security event(s) that cause a significant change in the network attack surface, locate most vulnerable hosts and the effect of increasing vulnerabilities further on these hosts.

¹This Chapter is based on the work “Graph similarity metrics for assessing temporal changes in attack surface of dynamic networks” published in *Computers & Security*, Elsevier, Volume 64, January 2017, Pages 16-43, (ISSN 0167-4048).

5.1 Introduction

With frequent changes in the network configuration, increasingly sophisticated and diverse application portfolios, various options of using network devices, etc., today's networks undergo continuous evolution. Consequently, there is a constant risk of information exposure to a larger threat landscape. Such ever-changing (dynamic) computer networks in terms of the size and complexity have varying attack surface. Essentially, the network attack surface is a subset of network configuration and vulnerabilities (known vulnerabilities, in particular as we do not have information about the zero-day vulnerabilities) that an adversary can use to violate the network security policy. Constant discovery of new vulnerabilities, misconfiguration of hardware (or software) components, for example, badly installed firewall, loose access control policies, etc. can further intensify change in the network attack surface. Therefore, there is a pressing need to consider temporal aspects of security while monitoring network performance. According to the standard guidelines and recommendations issued by [51] and [52], for maintaining the best possible security posture of a given computer network, it has to be monitored regularly for network security policy violations. Therefore, proper metrics should be there to quantify and present the effect of network/security events that endangers the security of the networks.

5.2 Motivation

Despite the proposal of many attack graph-based security metrics [25, 47, 48, 49, 50, 123, 124], there has been no work on assessing the temporal variation in the attack surface of dynamic networks. None of these previously proposed attack graph-based metrics designed (attempt) to measure the temporal variation in the network attack surface. Primarily, these metrics have been engineered to measure day-to-day changes in the security strength of a given network. However, securing the computer network requires an understanding at a finer granularity (i.e. At a micro level). While existing metrics in literature are comparatively coarse (i.e. for the same value of a given metric at different instants of time, underlying networks can be very distinct in attack surface), fine granular metrics, for example, graph distance metrics, can be used to quantify changes in the network attack surface. Therefore, proper metrics should be

there to detect a change in the attack surface in order to identify problems early so that corrective actions can be taken. Without such measures, the network (or security) events that cause a significant change in the network attack surface are not detectable, and hence prevent analysts from gaining awareness of the temporal aspects of network security.

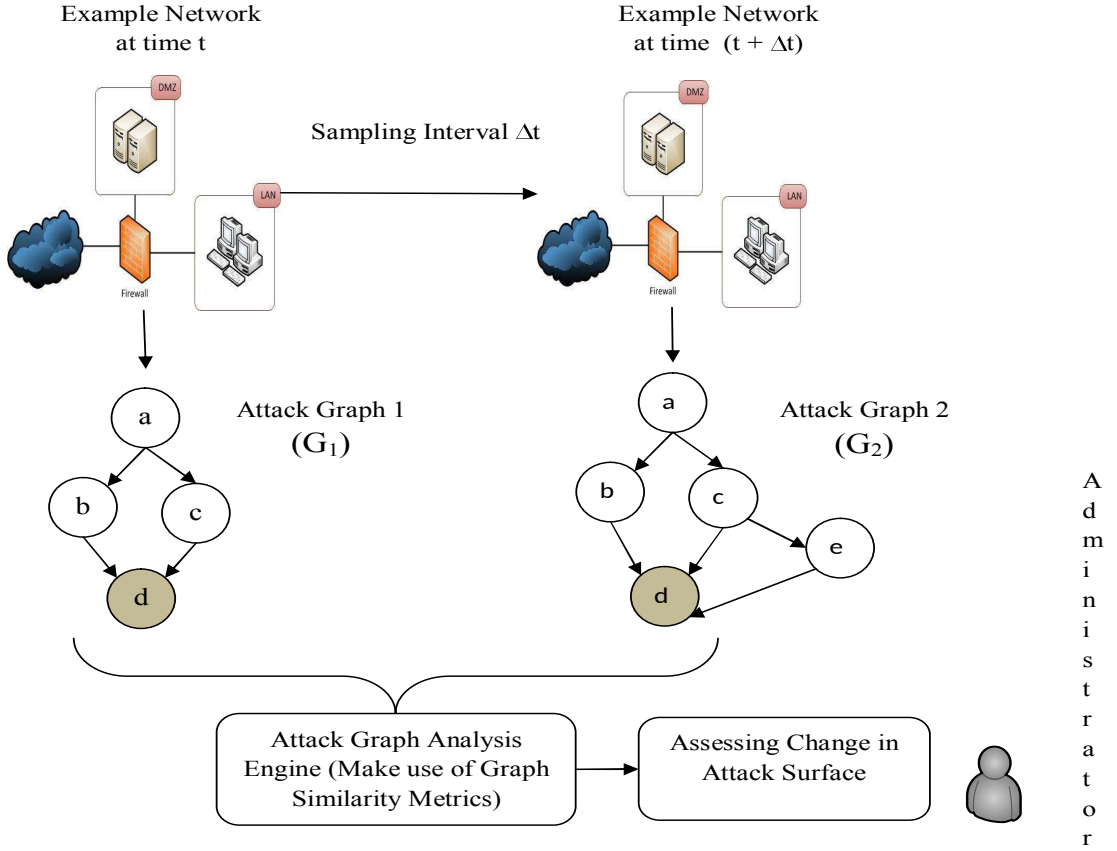


Figure 5.1: Dynamic computer network.

By observing shortcomings of the previously proposed metrics, we conclude that a change in the network attack surface should be measured with graph distance metrics. To bridge the gap, we explore classical graph distance measures based on the Maximum Common Subgraph (MCS) [54] and the Graph Edit Distance (GED) [55] to measure the distance between two successive attack graphs adjacent over the time. As shown in Figure 5.1, for a pair of attack graphs, i.e. $\langle G_1, G_2 \rangle$ generated for the same network over the sampling interval Δt , the similarity (or distance) between these

graphs naturally represents the similarity (or distance) between their respective attack surfaces. Such change measurement (in the network attack surface) is fundamental to the problem of intrusion prevention.

In this Chapter, we have adopted classical graph distance measures based on the MCS and GED to our particular problem of monitoring the security of temporal networks. It is because of their inherent capability of handling random structured graphs, where each graph is having unconstrained and unique labels for both nodes and edges [143]. The results obtained from MCS and GED based metrics assist security analysts in understanding the root causes (i.e. network/security events) responsible for the change in the network attack surface. In doing so, the analyst can gain better insight about the network attack surface and achieve a better understanding of their managed networks. Further, these metrics indicate the potential problems that are not so obvious even with the effective attack graph visualization.

5.3 Motivating Example

To build intuition about the problem we intend to solve, consider a toy example, as shown in Figure 5.2. Hosts 1, 2, and 3 comprise the internal network. Here, *Host 3* is of utmost importance as it hosts *MySQL database*. An attacker on *Host Ha* is an entity with malicious intent from the external network, and her intention is to gain root-level privileges on the Host 3. So, our primary concern is whether an attacker can obtain root privileges on the Host 3. Firewalls are put in place to allow all outbound connection requests, but inbound requests are permitted to access Host 0 only. In other words, firewalls allow any anonymous user to access service(s) running on the IIS web server only. Access to all other services running on other machines is denied or blocked. Hosts internal to the network (i.e. Hosts 1, 2, and 3) are allowed to connect to only those ports and hence services specified by the connectivity-limiting firewall policies shown in Table 5.1. Where, *All* specifies that source host may connect to the destination host on any port in order to have access to the services running on those ports, and *None* specifies that source host is prohibited from accessing any of the services running on the destination host. We assume that hosts in the example network have some initial vulnerabilities, which are summarized in Table 5.2 and indexed by *CVE* number.

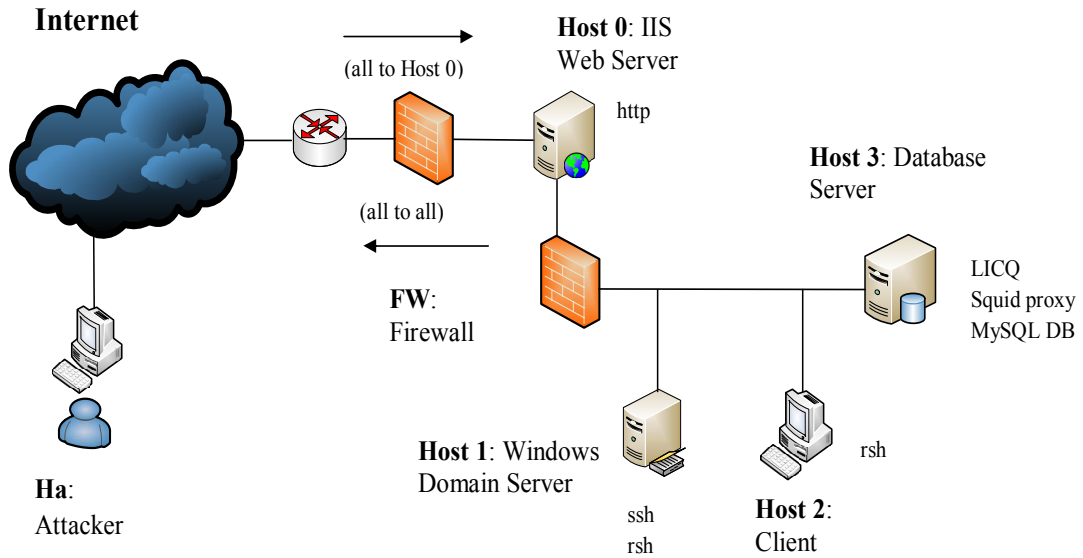


Figure 5.2: An example network

Table 5.1: Connectivity-limiting firewall policies

Host	Attacker	<i>Host0</i>	<i>Host1</i>	<i>Host2</i>	<i>Host3</i>
Attacker	localhost	All	None	None	None
<i>Host0</i>	All	localhost	All	All	Squid LICQ
<i>Host1</i>	All	IIS	localhost	All	Squid LICQ
<i>Host2</i>	All	IIS	All	localhost	Squid LICQ
<i>Host3</i>	All	IIS	All	All	localhost

In our example network, the Web Server (i.e. Host 0) is accessible to all anonymous external users, and it has a vulnerability *CVE-2010-2370*. Once exploited successfully, an attacker can obtain root-level access to the Web Server. An attack graph for this network configuration (at time t) is shown in Figure 5.3a. It shows all the attack paths available to an attacker to reach the target, i.e. Host 3.

5.3 Motivating Example

Table 5.2: System characteristics for the network configuration at time t

Host	Services	Ports	Vulnerabilities	CVE	CVSS Temporal Score
<i>Host0</i>	IIS Web Service	80	IIS Buffer Overflow	CVE-2010-2370	3.6
<i>Host1</i>	ssh	22	ssh buffer overflow	CVE-2002-1359	9.5
	rsh	514	rsh login	CVE-1999-0180	5.9
<i>Host2</i>	rsh	514	rsh login	CVE-1999-0180	5.9
<i>Host3</i>	LICQ	5190	LICQ-remote-to-user	CVE-2001-0439	7.125
	Squid proxy	80	squid-port-scan	CVE-2001-1030	5.8889
	MySQL DB	3306	local-setuid-bof	CVE-2006-3368	4.75

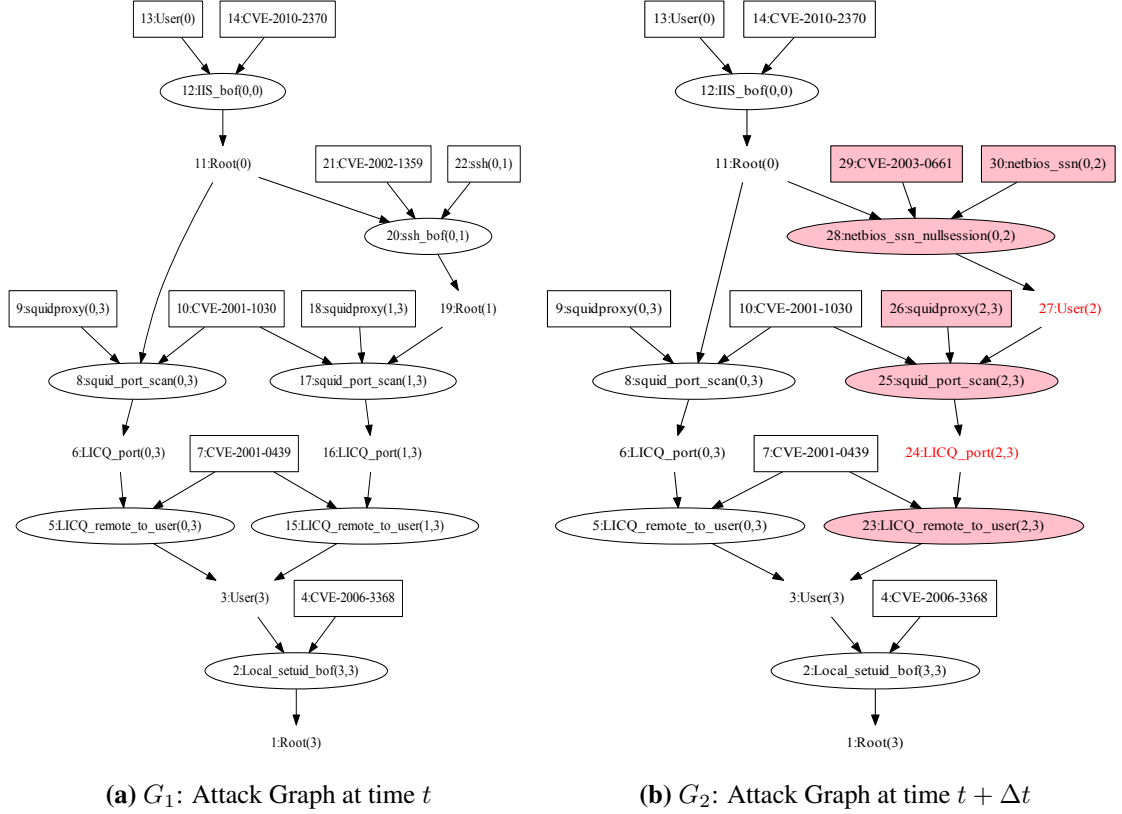


Figure 5.3: Differential visualization of attack graphs at the node level. Each *exploit* is shown by an oval, *initial condition* by a box and *post-condition* by a simple plain-text. Colored pink nodes represent newly introduced changes in the NAS over Δt .

5.3 Motivating Example

Now assume we have patched the vulnerability *CVE-2002-1359* in the *ssh* service running on the Host 1. Additionally, we have introduced a new service *NetBIOS-SSN* in Host 2 which has a vulnerability *CVE-2003-0661* (CVSS Temporal Score: 4.2868). For this network configuration, the attack graph is shown in Figure 6.2b. It is evident from Figure 5.3 that both the attack graphs (at time t and $t + \Delta t$) have two attack paths. The values for the previously proposed attack graph-based metrics are provided in Table 5.3.

Table 5.3: Results for the existing attack graph-based metrics for the pair of attack graphs shown in Figure 5.3

Attack graph-based metric	Network config. at time t (Attack Graph G_1)	Network config. at time $t + \Delta t$ (Attack Graph G_2)
Number of paths metric (NP) [48]	2	2
Shortest path metric SP[47]	4	4
Mean of path lengths (MPL) [49]	4.5	4.5
Normalized mean of path lengths (NMPL) [25]	2.25	2.25
Median of path lengths (MDPL) [25]	4.5	4.5
Mode of path lengths (MoPL) [25]	4 (minimum mode value)	4 (minimum mode value)
Standard deviation of path lengths (SDPL) [25]	0.5	0.5
Network compromise percentage (NCP) [124]	75%	75%
Weakest adversary (WA) [50]	2	2
Cumulative probability (P) [26]	0.12	0.09
Cumulative resistance (R) [27]	6.81	7.00

As evident from Table 5.3, metrics such as SP [47], NP [48], MPL [49], NCP [124], WA [50], etc. conclude that both the configurations are equally secure (have the same security strength), whereas, the P [26] and R [27] show that the two network configurations are not equally secure. Due to the introduction of a new vulnerable service NetBIOS-SSN (vulnerability: *CVE* – 2003 – 0661, Temporal Score: 4.2868) in the Host 2, there is a change in the resulting value of the probability based metrics. One can say that these two metrics (i.e. P and R) are sensitive to the changes in the network

attack surface. But what if there is an introduction of the new vulnerable service (in the Host 2) other than the above one and having the vulnerability with the temporal score of 9.5 (equal to that of the *CVE* – 2002 – 1359). In this case, even the probability based metrics fail to detect changes in the security strength of the underlying network. As these metrics (i.e. P and R) assign 1 to all true initial conditions such as vulnerable service connectivity, presence of exploitable vulnerabilities, etc., they do not capture the actual network resource involvement in the multistage attack. Therefore, they are not accurate enough to capture the change in the network attack surface.

The key observation here is that the two network configurations having the same value for the attack graph based metrics are different in their attack surface as shown in Figure 5.3. Therefore, specifically, we may ask a question, *Is the existing attack graph based metric paying-off (useful) in the context of network security change assessment?* The remainder of this Chapter is built upon this important observation and addresses the issue of detection of a change in the network attack surface at a finer granularity. We argue that the graph distance metrics could be used to quantify the variations in the network attack surface at the level of newly introduced nodes or edges in the attack graph. As the *require edge* in an attack graph captures the detail about the resource involvement (e.g. service connectivities, vulnerable services, etc.), it is quite sensitive to the changes in the attack surface. Therefore, we are proposing the use of graph distance metrics that accepts *require edge* as an input parameter.

5.4 Formal Model for the Network Attack Surface (NAS)

In this section, we formalize the notion of NAS. Next, we show how the NAS and dynamics in it are ideally captured by the logical attack graph [16], [21].

Cowley et al. [141] identified network topology factors such as network segregation/partitioning and network reachability (in terms of service connectivity) as important network characteristics for measuring the network security risk. Essentially, vulnerable service connectivities and existing vulnerabilities (well-known vulnerabilities, in particular) form the pre-conditions for the incremental exploitation of remote

vulnerabilities. Such initial conditions form the basis for multi-step network intrusion. Informally, service connectivities and exploitable technical vulnerabilities (in a software/service) are the network resources used by an adversary against the network itself. In practice, not all service connectivities and vulnerabilities contribute equally during the network intrusion and hence we chose only those which are likely to be used by an adversary while compromising the target network. Similar to [71], our notion of the network attack surface constitutes a subset of network resources that are likely to be used by an adversary against the network itself.

5.4.1 Notion of the NAS

Consider a typical computer network $\mathcal{N} = \{h_0, h_1, h_2, \dots, h_n\}$ consisting of n number of hosts, where each $h_i \in \mathcal{N}$ can potentially be the target of remote adversary in addition to the attacking host h_0 . The security configuration of a network \mathcal{N} constitute the following two facts:

1. A finite set of services $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$ and associated vulnerabilities. Here $S_i \in \mathcal{S}$ is the set of services running over the host h_i
2. Service connectivity relations among the hosts, i.e. $\mathcal{C} = \{C_1, C_2, \dots, C_n\}$, where $C_i \in \mathcal{C}$ is the set of service connectivity relations from host $h_i \in \mathcal{N}$

Above stated security configuration of the network \mathcal{N} is represented by $Conf(\mathcal{N})$. Each machine in the network hosts several services, where each service can have several well-known vulnerabilities, which in turn can be identified and exploited by an adversary. A set of vulnerabilities in the network \mathcal{N} is denoted by $V = \{V_1, V_2, \dots, V_m\}$, where m is the total number of well-known vulnerabilities in the network. Each V_i is associated with the single service on a single host. The set of exploitable vulnerabilities in the host $h_i \in \mathcal{N}$ is represented by $VN_i = \{vn_{i,1}, vn_{i,2}, \dots, vn_{i,p}\}$. The total set of exploitable vulnerabilities in the network is represented by $VN = VN_1 \cup VN_2 \cup \dots \cup VN_n$ and $VN \subseteq V$. For the remotely exploitable vulnerabilities, we need to consider service connectivity relations between the pair of hosts $\langle h_i, h_j \rangle$.

Let \mathcal{C}' be the total set of vulnerable service connectivity relations in a network and $\mathcal{C}' \subseteq \mathcal{C}$. Such contribution of VN and \mathcal{C}' reflects their likelihood of being used during the network compromise. To conclude, NAS is a subset of the network configuration

5.4 Formal Model for the Network Attack Surface (NAS)

and vulnerabilities that an attacker can exploit to reach the target. With the perception of the \mathcal{NAS} as discussed above, we define \mathcal{NAS} as follows:

Definition 1 (Network Attack Surface (NAS)). *Given a set of vulnerable service connectivities $\mathcal{C}' \subseteq \mathcal{C}$, and a set of exploitable vulnerabilities $VN \subseteq V$ that can be used to compromise the target in the network \mathcal{N} , the network attack surface \mathcal{NAS} is the set, where $\mathcal{NAS} = \{\mathcal{C}' \cup VN\}$.*

5.4.2 Model to Capture the NAS

Essentially, a logical attack graph depicts exploits and their respective enabling conditions such as vulnerable service connectivities, the presence of exploitable technical vulnerabilities, etc. A directed edge from the initial security condition to an exploit represents *require edge*, whereas an edge from the exploit to a security condition indicates an *imply edge*. So, the notion of attack surface for an entire computer network is ideally captured by the attack graph, particularly in terms of *require edges*.

Definition 2 (Attack Graph). *“Given a set of exploits e , a set of conditions c , 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 directed graph $G(e \cup c, R_r \cup R_i)$, where $(e \cup c)$ is the vertex set and $(R_r \cup R_i)$ is the edge set” [56].*

We represent the attack surface of network configuration, i.e. $\mathcal{NAS}(\text{Conf}(\mathcal{N}))$ by logical attack graph $G(e \cup c, R_r \cup R_i)$ defined above. Therefore, we can assess the variation in the attack surface of a given computer network by computing the distance between two successive attack graphs adjacent over time. In doing so, we can pinpoint important changes and dynamics which are responsible for the change in the \mathcal{NAS} .

5.4.3 Dealing with Variations in the NAS

In practice, change in the network configuration is due to the numerous actions (taken by the administrator) as a part of network maintenance and security hardening procedure. Additionally, there is a change in the \mathcal{NAS} due to some of the actions that are not under the control of an administrator. A partial list of such actions/events that lead to the significant change in the network attack surface is compiled as follows:

- Installation/removal of network devices, software, services, etc.
- Change in the access control policies
- Misconfiguration of software, services and hardware devices
- Vulnerability patching
- Disclosure of new vulnerabilities

Let M be the enumeration of all the above actions/events. After applying the action set M , the initial configuration $Conf(\mathcal{N})$ changes. We represent the resulting configuration as $Conf(\mathcal{N}) \oplus M$ and resulting network attack surface as $NAS(Conf(\mathcal{N} \oplus M))$. Such change in NAS is successfully captured by an attack graph generated at time $t + \Delta t$.

The algorithm 5.1 detects all *require edges* (i.e. The edges between the initial conditions and exploits) in an attack graph G . We have used only *require edge*, an attack graph construct/parameter, for computing graph distance metrics because it efficiently captures the contribution of network resources to the network attack surface and is sensitive to the changes in it. The next section discusses the graph distance metrics we employed for our specific problem of change assessment.

5.5 Metrics for Assessing Variation in the NAS

A plethora of graph distance metrics has been proposed in the field of pattern recognition to compare two graphs for their similarity (or dissimilarity). Two graphs are believed to be the same if there exists a graph isomorphism between them [162]. Closest to our work is that of Showbridge et al. [143], who used ECGM based edit distance to monitor the performance of telecommunication networks. Ning and Xu [163] applied error-correcting graph/subgraph isomorphism for learning attack strategies. Liao and Striegel [144] proposed a graph differential anomaly visualization (DAV) model in the area of network management to identify the meaningful changes and hidden anomalous activities. Modern measures such as Tabu [164], [165], RASCAL [166], and DeltaCon [167] have been used extensively to study the similarity/distance between chemical graphs. We focus on the classical graph distance measures such as MCS [54] and GED [55] to study the impact of day-to-day network dynamics on to the network attack surface.

Algorithm 5.1 *findEdges*: find *require edges* in an attack graph G

Input:

$G(V, E) \rightarrow$ a goal-oriented logical attack graphs generated for the enterprise network.

Output:

$Init \subset V \rightarrow$ set of initial conditions in G

$RequireEdge \subset E \rightarrow$ set of require edges in an attack graph G

```

1:  $\langle V, E \rangle \leftarrow G$ 
2: for all  $u \in V$  do
3:   if  $\left( (indegree(u) = 0) \wedge (outdegree(u) \geq 1) \right)$  then
4:      $Init \leftarrow u$ 
5:   end if
6: end for
7: for all  $u \in Init$  do
8:   if  $(u = i)$  for  $(i, j) \in E$  then
9:      $RequireEdge \leftarrow (i, j)$ 
10:  end if
11: end for

```

In order to understand the network events that cause changes in the attack graph over time, two attack graphs that are adjacent over the time need to be compared for their structural similarity. In this context, two successive attack graphs need to be matched to identify their common features. This can be achieved by looking for an exact graph/subgraph isomorphism to show graph equivalence (or inclusion). However, the base reference attack graph for the particular network configuration at time t is often modified/corrupted by the occurrence of various events (explained in Section 5.4.3) over time Δt . Furthermore, the assumption of the existence of an isomorphism between successive graphs is generally too strong. Therefore, error-correcting graph matching such as MCS [54] and GED [55] is preferred in this study over the modern measures such as Tabu [164], [165], RASCAL [166], and DeltaCon [167].

Since the MCS and GED based graph matching drops the condition that the mapping must preserve all vertices and edges, the goal is reduced to finding a *best* mapping, i.e. the one which preserves a maximum number of vertices and edges. In the context

of understanding the effect of network dynamics on the NAS, we need to match only *require edges* in the two successive attack graphs. Such *require edges* capture the network resource involvement in the exploitation of vulnerabilities during the potential multistage intrusion.

5.5.1 Maximum Common Subgraph-based Distance Metric

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be the two consecutive attack graphs generated for a given computer network over the sampling interval Δt . The intuition behind using MCS based distance metric [54] for assessing variation in the attack surface is based on the fact that successive attack graphs share portion of the subgraph structure. Therefore, MCS based distance metric is defined as,

$$MCS(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{\max(|G_1|, |G_2|)} \quad (5.1)$$

Since both the attack graphs are generated for the same network (over the sampling interval Δt), they share subgraph structure and hence certain graph properties. In the context of an attack graph, the shared properties may be in terms of nodes (i.e. *pre-conditions*, *exploits*, and *post-conditions*) and edges (i.e. *require edges* and *imply edges*). Such shared properties are due to the un-patched vulnerabilities and vulnerable service connectivities in an attack graph G_1 that persists over Δt . In practice, even though the security analyst is aware of the presence of vulnerabilities and respective enabling conditions, she cannot patch (fix) all because of the underlying constraints such as countermeasure cost, limited security budget, unavailability of patch/workarounds, delayed patches (patch Tuesday), response-time requirements, etc. Few (or all) of such un-patched vulnerabilities will straightaway appear in the attack graph (here G_2) generated for the network after Δt . It means some of the un-patched vulnerabilities are stable over Δt . Here, we call the attack graph generated at time $t + \Delta t$ as an original attack graph.

In Equation (5.1), $mcs(G_1, G_2)$ represents the shared portion of the successive attack graphs G_1 and G_2 and it can be computed by counting either the number of common edges or common nodes. However, $mcs(G_1, G_2)$ in terms of common nodes does not capture the shared subgraph structure because one initial condition may act as a

pre-condition for one or more exploits. Such scenario can be captured only if we consider the common edges while computing $mcs(G_1, G_2)$. *Require edge* in an attack graph identifies which initial condition is responsible for the successful execution of an exploit and provides local context information about the network attack surface. On the other hand, an *imply edge* identifies those conditions that are generated after the successful execution of an exploit. For our specific problem of change assessment, we consider only *require edges* because of their ability to capture attack surface elements such as vulnerable service connectivities and remotely exploitable vulnerabilities.

If two attack graphs G_1 and G_2 are exactly the same, then the size of $mcs(G_1, G_2)$ equals the original attack graph G_2 , and the distance between the attack graphs will be 0. On the contrary, if two attack graphs are totally different, the size of $mcs(G_1, G_2)$ will be zero resulting in a distance of 1. In other cases, the value of MCS based distance metric lies between 0 and 1. In practice, the network attack surface may vary (change) as a function of variation in the number of exploitable vulnerabilities. If there is an increase in the number of vulnerabilities (because of vulnerable network configuration, introduction of new vulnerability, etc.), we can say there is an increase in the network attack surface and we need to detect this scenario efficiently using the graph distance metric. On the other hand, if the vulnerabilities are patched efficiently and there is no introduction of other vulnerabilities to cancel the effect of patched vulnerabilities, we can say there is a decrease in the network attack surface. This scenario also needs to be detected efficiently in order to test the efficiency of applied countermeasures such as vulnerability patching, re-configuration (re-dimensioning) of the network, etc.

5.5.2 Graph Edit Distance-based Metric

Essentially, graph edit distance (GED) [55] is the number of edit operations that need to transform one graph into the other. For exact graph matching, one can use GED. The basic idea of using GED for our specific problem of change assessment in the network attack surface is that there is an edit cost associated to modify an attack graph G_2 such that it becomes isomorphic to the graph G_1 . Our goal is to reduce network attack surface at time $t + \Delta t$ to the attack surface at time t . There is a cost associated with each initial condition, for example, countermeasure cost, response-time requirements in terms of man-hour, etc. In order to prevent an exploit from successful execution, we

need to disable (remove) at least one initial condition. This can be done by removing *require edges* from the original attack graph G_2 . Unlike [55], we consider only the addition/removal of the *require edges* as an edit operation. Moreover, the post-conditions are the direct outcome of successful execution of an exploit, and hence they are not under the control of security analysts. Therefore, we are not considering the addition (or removal) of *implied edge* as an edit operation. The motive behind using GED is that the more the number of steps/edit operations required to transform attack graph G_2 to G_1 , the larger the variation in the network attack surface over time Δt . In this Chapter, we consider the cost of all the edit operations (i.e. addition/removal of *require edges*) equal to one. Then the normalized GED can be simplified to:

$$GED(G_1, G_2) = \frac{|G_1| + |G_2| - 2|mcs(G_1, G_2)|}{|G_1| + |G_2|} \quad (5.2)$$

Clearly, here the GED, as a measure of change in an attack surface, increases with an increase in the amount of change experienced by an enterprise network over the sampling interval Δt . If two attack graphs G_1 and G_2 match exactly the same, the numerator in Equation (5.2) is zero, and hence GED will result in a 0. On the contrary, if two attack graphs do not share a single *require edge*, GED will result in a value of 1. It means GED is bounded below by 0 when G_1 and G_2 are isomorphic (i.e. there is no change in the network attack surface), and above by 1 when the attack surface is completely changed.

For the attack graph pair $\langle G_1, G_2 \rangle$ shown in Figure 5.3, there is a total of 11 *require edges* in each graph. We do not consider the *require edges* between post-conditions and exploits, as such conditions are not under the control of an administrator and cannot be removed without removing their causes. As discussed in Section 5.4.2, only the *require edges* between initial conditions and exploits are capable of capturing the actual resource involvement in the network attack surface. Therefore, $|G_1| = |G_2| = 11$. Further, there are 8 *require edges*, which are common to both the attack graphs, i.e. $|mcs(G_1, G_2)| = 8$. Such common edges represent the shared attack surface over the sampling interval Δt . As evident from the Equations (5.1) and (5.2), the resulting value for both the MCS and GED based metrics for the attack graph pair $\langle G_1, G_2 \rangle$ is 0.27 and it represents a level of change in the network attack surface over Δt . The root causes of such change are (i) the patching of existing vulnerability *CVE* – 2002 – 1359 in the

ssh service (running over the Host 1) and (ii) introduction of a new service NetBIOS-SSN (in Host 2) which has a vulnerability *CVE* – 2003 – 0661. Here resulting value for both the metrics is the same and it is because of the same size of both the attack graphs (here 11).

5.6 Experimental Setup

To support our experiment, we consider a set of 3 network models as shown in Figure 5.4. Host T (target for an attacker) is the critical resource in each network model, and an attacker is the entity with malicious intent whose goal is to obtain a root-level access on the target Host T . An attacker is able to exploit all discovered vulnerabilities in the target network. Firewall(s) are put in place (particularly in *EI* and *DMZ* network models) to govern the service accessibility between external user (attacker) and the other hosts of the network. Hosts P , R , and S are the intermediaries between the attacker and the target Host T and each of them is reachable to the attacker under varied vulnerability densities. As evident from Figure 5.4, Three columns show different network models, viz. Flat (F), External-Internal (EI), and *DMZ*. Incremental variations (i.e. Assignment of vulnerable hosts/vulnerabilities) to each network model are shown in the rows 1, 2, 3, and 4. First target host T is added to the network and then intermediary hosts, i.e. P , R , S (between the attacker and the target T) in a linear fashion. Once the maximum number of hosts (here 4) is added to each network model, vulnerabilities are increased linearly. The maximum number of vulnerabilities on each host is restricted to 4 only.

5.6.1 Description of the Network Models

Essentially, all hosts (machines) in a Flat network (F) belong to the same protection domain and hence each host on a network can connect to any other host. As a result, an attacker can directly access services running on the target host T . In case of External-Internal (EI) network, a firewall is set up which in turn allows service connectivity to only a subset of hosts and ports in either direction. Host P is the only host an attacker can directly access. Other hosts, i.e. Hosts P , R , S , and T can connect to each other. At last, two filtering devices (i.e. Firewalls) are put in the *DMZ* network to control

inbound and outbound connection requests. A DMZ firewall filters all the connection requests coming from external network and destined for the DMZ network, whereas internal firewall filters connection requests coming from the DMZ network and destined for the internal network. In DMZ network an attacker can access services running on host P only. The firewall policies which govern service connectivity between other hosts are shown in Table 5.4. Where, *Yes* specifies that source host may connect to the destination host on any port in order to have access to the services running on those ports, and *No* specifies that source host is prohibited from accessing any of the services running over the destination.

5.6.2 Generation of the Attack Graphs

For our purpose of security change assessment, we have adopted an incremental way of network security evaluation. First, a vulnerable host is introduced in each network model incrementally, one at a time and then a goal-oriented attack graph is generated using *MulVAL* [21], [16]. Automatic generation of the attack graph is beyond the scope of this thesis, and we assume that they have been generated using tools like *MulVAL*. The attack graphs that we generated are finite and contain neither loops (cycles) nor multiple edges. The generated attack graph nodes are of two types, namely, exploits and conditions (i.e. pre and post-conditions), where each node is uniquely labeled. In practice, obtaining a full description of the difference between two attack graphs (generated over the observed sampling interval Δt) for a large network is infeasible because it requires solving the subgraph isomorphism problem which is a hard problem to solve. The essence of generating uniquely labeled attack graphs is that the graph matching algorithms for such class of uniquely labeled graphs are very efficient.

Hosts in our experiment can connect to one another on a single port, for example, *Port 80* and the number of vulnerabilities we considered are changed (altered) for this port only. Further, we assume that all the vulnerabilities are remotely exploitable. We considered reachability (i.e. vulnerable service connectivity) as one of the vital pre-conditions for successful exploitation of remote vulnerabilities. Once the vulnerability is successfully exploited, it enables an attacker to execute arbitrary code by which she can gain root-level access on the vulnerable host. In practice, a vulnerable service running with root privileges is more likely to be exploited compared to a service running

5.6 Experimental Setup



Figure 5.4: Heterogeneous network models.

with user (non-root) privileges. Therefore, we are not concerned about the problem of locally exploitable vulnerabilities in this thesis and we left this as a part of future work.

Table 5.4: Connectivity-limiting firewall policies for the DMZ network model

Host	Attacker	P	R	S	T
Attacker	localhost	Yes	No	No	No
P	Yes	localhost	Yes	No	No
R	Yes	Yes	localhost	Yes	Yes
S	Yes	No	No	localhost	Yes
T	Yes	No	Yes	Yes	localhost

5.6.3 Selection of the Sampling Interval Δt

For the experimental network observed at time t , we have generated an attack graph G_1 to capture its attack surface. Change (variation) in the attack surface of the observed network over the time Δt is captured by generating an attack graph G_2 at time $t + \Delta t$. Here Δt is the arbitrary (randomly chosen) sampling interval and is one of the parameters of utmost importance while monitoring enterprise network security. It defines the frequency of attack graph generation for the given network and hence helps in determining how frequently security measurement should be taken. Δt constitutes the time required for gathering environment specific information such as vulnerability details, network configuration details, etc., and the attack graph generation time. In particular Δt dictates the type of attacks that can be detected. In practice, Δt could be static or variable, slow or fast. However, in practice, it could be ideal to generate an attack graph when network undergoes major changes, for example, introduction of new host(s) or services, network reconfiguration, failure of security devices, etc. The decision of Δt selection should be carefully done and it is heavily dependent on the expertise of the security assessor. The sampling interval Δt should be chosen in such a way that any security event that causes significant changes in the network attack surface will be detected within an acceptable period. In our experiment, during the sampling interval Δt enough number of vulnerabilities are added to the network so that the adversary will be able to reach and compromise the target Host T . The intuition behind doing this is that the goal-oriented attack graph should be generated after every Δt . We have generated attack graphs continuously and selected only those for metric computation that

are goal-oriented. Such intuition can also help in the selection of Δt . Issues relating to the selection of an optimal sampling interval for the purpose of graph generation and attack surface measurement are beyond the scope of this thesis.

5.6.4 Assessing Variations in the NAS

For a specific time window \mathcal{T} , we have generated a number of attack graphs for a given experimental network at discrete instants of time based on Δt . It produces a sequence of goal-oriented attack graphs. Each attack graph in a sequence (except the first one) is compared for dissimilarity with its immediate predecessor using graph distance measures (discussed in Section 5.5) to report the degree of change in the network attack surface over Δt . We have used *require edge* (an attack graph feature) as a measurement variable because this feature can successfully outline variation (change) in the network attack surface. Such graph parameter is sensitive to the changes in network attack surface and hence can be mapped successfully to the external causes providing an event detection capability [143]. Graph similarity metrics (i.e. *MCS* and *GED*) we employed in our study combine such graph parameter and hence provide a single sequence (or series) that can be utilized for change analysis. We have applied these metrics to each graph pair $\langle G_i, G_j \rangle$ of the attack graph sequence generated for each network model. It generates a new sequence of numbers where each value represents a level of change in the network attack surface over Δt . In other words, such graph similarity measures provide a trend of the network security dynamic behavior as it evolves over time [143]. Once the significant change is detected in the network attack surface, the analyst needs to examine the attack graph to find out the external cause of the problem.

5.7 Results and Analysis

This section describes the results obtained for the experimental setup described in Section 5.6. For easier illustration, first we will discuss the results obtained for the previously proposed attack graph based metrics such as

- Shortest path metric [47]

- Number of paths metric [48]
- Mean of path lengths [49]
- Normalized mean of path lengths [25]
- Median of path lengths [25]
- Mode of path lengths [25]
- Standard deviation of path lengths [25]
- Network compromise percentage [124]
- Weakest adversary [50]

Then, we will discuss the results for the graph distance metrics explored in this study. The analysis is conducted in the following directions:

1. Effect of linear and non-linear variations in the vulnerability density on the network attack surface.
2. Effect of host insertion and deletion on the network attack surface.
3. Effect of vulnerability variation in the host(s), which is either directly accessible to an adversary or belonging to the protection domain.

We have adopted an incremental approach to change assessment. As shown along the x-axis in Figure 5.5a-5.5d, 5.6a-5.6d, 5.7, etc., we incrementally assign vulnerabilities to the different hosts in each network model. The expression of vulnerability assignment takes the form: jH , where j is the remotely exploitable vulnerability on the target host H . For example, the expression $4P2T$ states that for a given experimental network there are 4 remotely exploitable vulnerabilities on the host P , and 2 on the target host T , respectively. In each step either the vulnerable host (or vulnerability to the existing host) is added (to the network) and an attack graph is generated using MulVAL tool. The sequence of attack graph generated is then evaluated using previously proposed attack graph based metrics and the two graph distance metrics such as MCS and GED used in this study.

5.7.1 The Results for the Previously Proposed Attack Graph-based Metrics

As evident from Figure 5.5a, irrespective of the network models, the shortest path metric (SP) [47] remains constant even though there is an increase in the number of

vulnerabilities across the network. SP does not capture the effect of the events such as the introduction of vulnerable host(s) and an increase in the number of vulnerabilities. Therefore, SP metric is not useful for network security monitoring. The effect of above stated events on the number of paths metric (NP) [48] is shown in Figure 5.5b. From the plot it is clear that NP increases exponentially as there is a linear increase in the number of vulnerabilities. One can say that NP successfully captures the effect of variation in the network topology (security) factors. However, for the two attack graphs G_1 and G_2 (as shown in Figure 5.3), having the same number of attack paths, NP metric indicates that the underlying networks are equally secure. But this may not be the reality. Two networks having the same value of NP metric may be different in their attack surface as shown in Figure 5.3. From Figure 5.5c, it is clear that there is an increase in the mean of path length metric (MPL) [49] as a result of increase in the number of vulnerabilities across the network. Essentially, an increase in the value of MPL indicates that network security is improving as a result of increase in the number of vulnerabilities. However, such interpretation is clearly erroneous and misleading. If some of the attack paths are removed from the attack graph G_1 by patching vulnerabilities, the resulting attack graph may produce the same MPL value as the original graph G_1 . It means that the mean of path length can be the same even after network changes. Hence, as a concluding remark, MPL is not able to capture the improvement (or degradation) in the security of the underlying network. In contrast to the MPL metric, normalized mean of path length (NMPL) metric captures fine granular improvement and degradation in the network security, as shown in Figure 5.5d. But the difference in the security strengths of the different network models becomes negligible as the network becomes more and more saturated with the number of exploitable vulnerabilities.

As evident from Figure 5.6a, at various points (i.e. Horizontal points along the x-axis), the median of path length metric (MDPL) indicates that different networks are equally secure, when they are actually not. From Figure 5.6b, it is clear that with the exception of the flat network model, a mode value stays constant even though there is an increase in the number of vulnerabilities and change in the access control policies. In doing so, the mode of path lengths (MoPL) metric treats all networks equally secure. As a concluding remark, MDPL, and MoPL are not useful for assessing

5.7 Results and Analysis

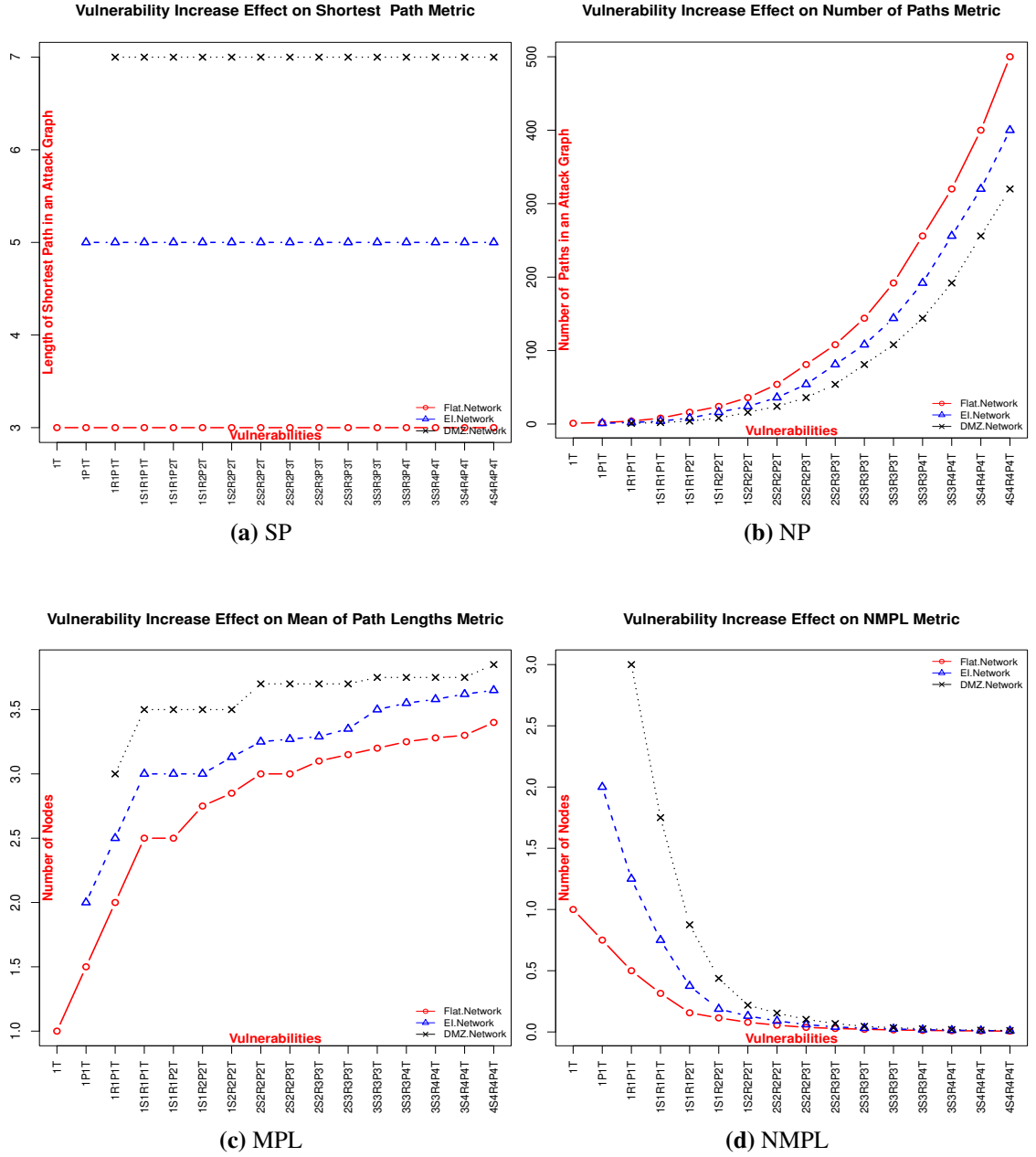


Figure 5.5: The effect of a linear increase in the number of vulnerabilities on the SP, NP, MPL, and NMPL metric under different network models.

temporal changes in the network attack surface. In case of the standard deviation of path lengths (SDPL) metric (as shown in Figure 5.6c), the variability in attack path

5.7 Results and Analysis

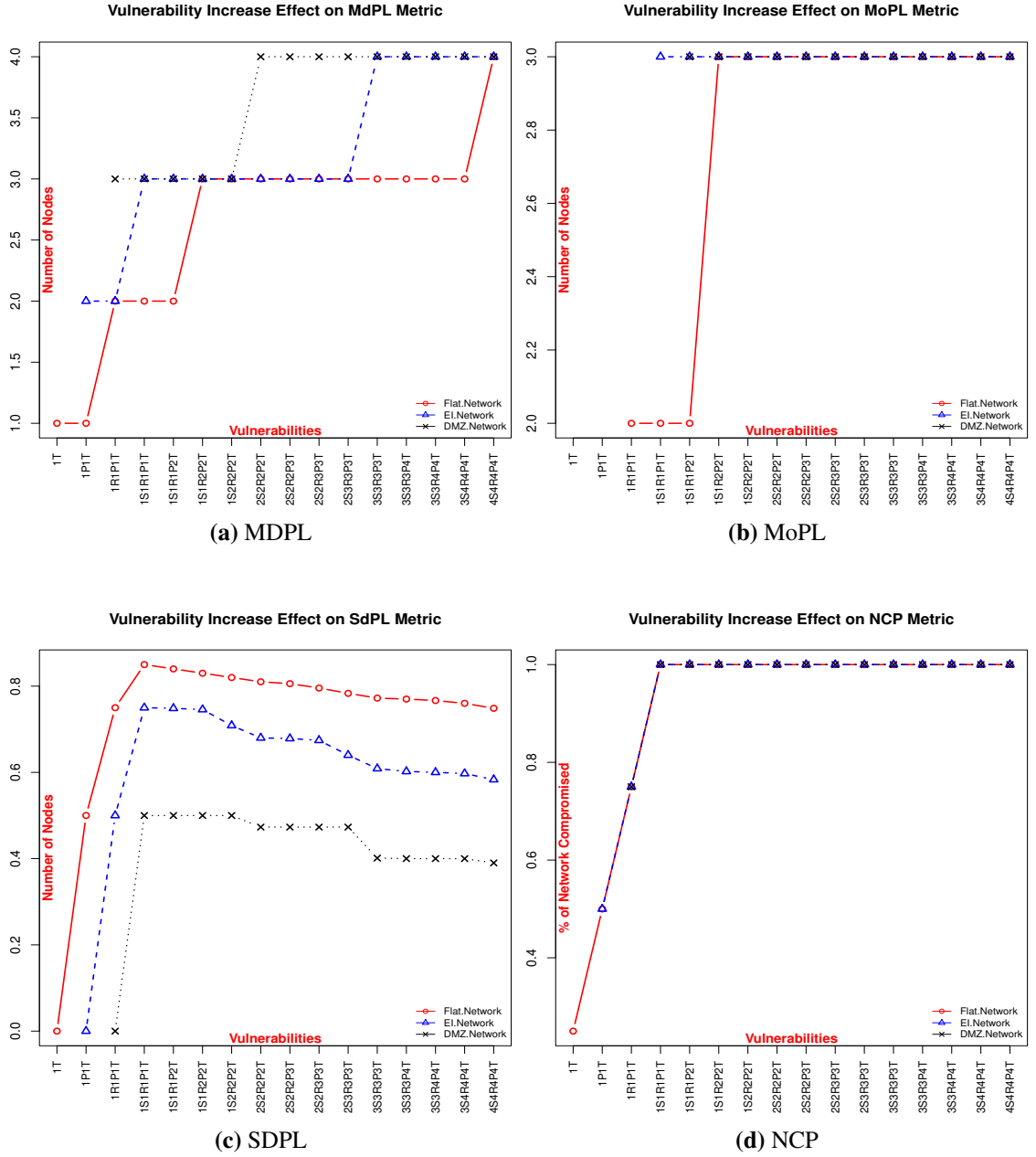


Figure 5.6: The effect of a linear increase in the number of vulnerabilities on the MDPL, MoPL, SDPL, NCP metric under different network models.

length increases more dramatically when new vulnerable host is added to the network. The variability begins to decrease once new vulnerable hosts stop joining the network.

Further, there is a decrease in the variability as there is an increase in the access control policies. Even though SDPL is sensitive to the increase in the number of vulnerabilities and also to the increase in the access control policies, the analyst should not trust this metric. This is due to the dependence of SDPL on the MPL metric. As shown in Figure 5.6d, the network compromise percentage (NCP) metric is unable to detect changes like an increase in the number of vulnerabilities and changes in the access control policies. Therefore, the NCP cannot be used to monitor the ongoing security events in the network. Finally, the weakest adversary (WA) metric (as shown in Figure 5.7) does not capture the effect of any of the above stated network (or security) events.

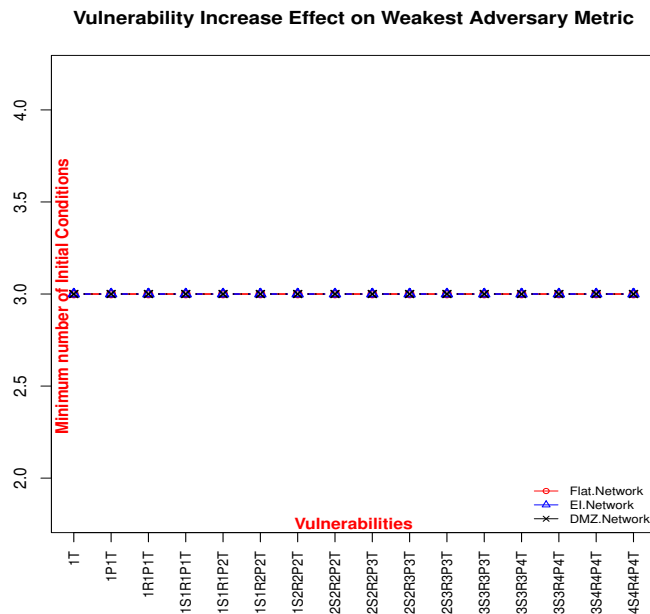


Figure 5.7: The effect of a linear increase in the number of vulnerabilities on the WA metric under different network models.

In summary, previously proposed attack graph based metrics (as discussed above) are not sensitive enough to capture the change in the network security strength (or attack surface) at a finer granularity. They do not capture the events that cause significant change in the network attack surface, prevent security analysts from understanding the root causes responsible for the change and hence the impact of security events at the micro-level. Thus, we conclude that there is a need for metrics that quantify and present the important changes and dynamics in the network. Such metrics can be useful

to security analysts in gaining better insight about network attack surface, and provide information regarding potential problems that are otherwise not so obvious even with the effective attack graph visualization. The next section (i.e. Subsection 5.7.2) discusses the results for the MCS and GED based graph distance metrics we analyzed in this study.

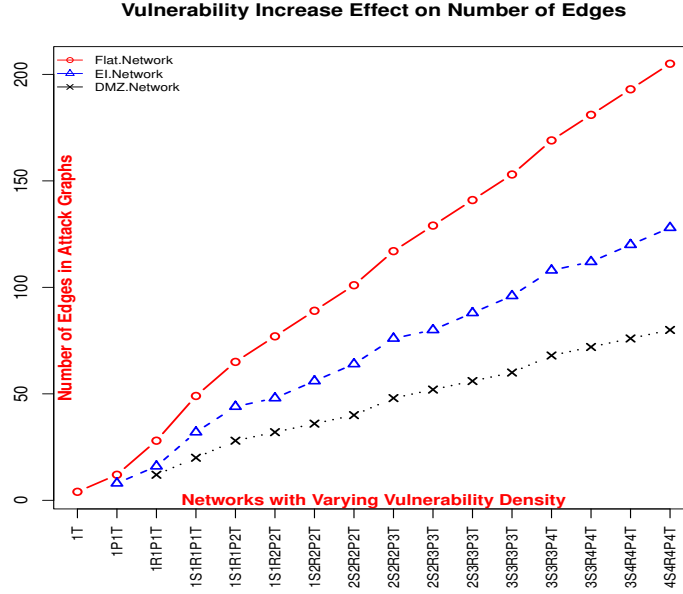


Figure 5.8: The effect of a linear increase in the number of vulnerabilities on the number of edges of the generated attack graphs under different network models.

5.7.2 Change Assessment using MCS and GED-based Graph Distance Metrics

As discussed above, the previously proposed metrics provide a rough idea about the security strength of a computer network, but at a very coarse-level. They are not good at determining changes in the network attack surface at a finer granularity. In order to bridge the gap, we have used graph distance measures such as MCS [54] and GED [55]. As seen in Section 5.5, we make use of an attack graph feature called *require edge* as a measurement variable because of their inherent capability of successfully characterizing variations (changes) in the network attack surface. Figure 5.8 shows the effect of the increase in the number of vulnerabilities on the number of graph edges for

different network models. As evident from Figure 5.8 that placement of firewall(s) in the network prevents many attacks from happening and hence results in attack graphs of reduced size. Therefore, the number of edges in the attack graphs corresponding to the flat network model (F) grows faster than the other two network models (i.e. EI and DMZ models). It shows that the attack graph feature (here, the number of edges) provides a quantitative support for the well-known network protection strategy called *defense-in-depth*.

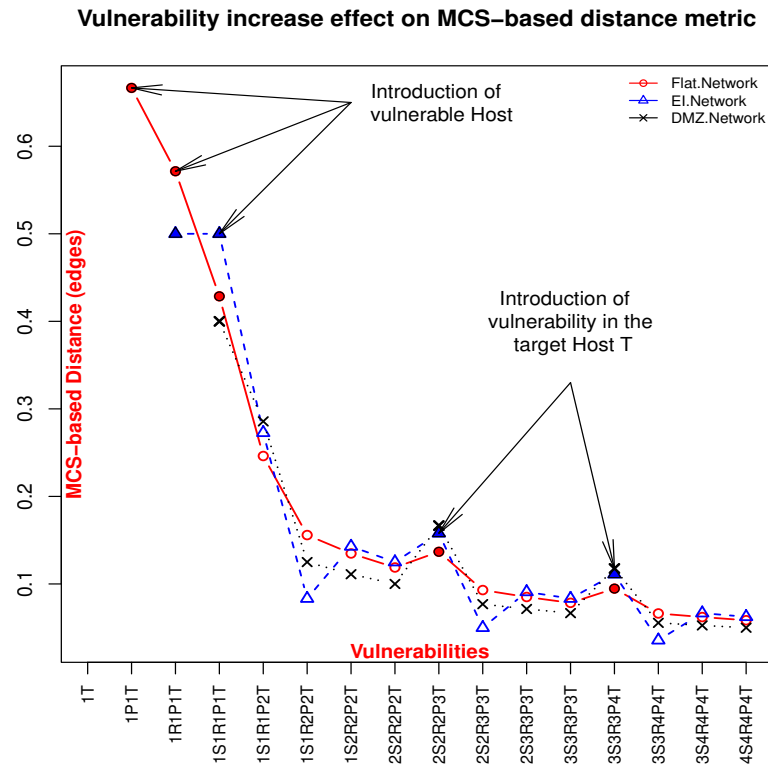


Figure 5.9: The effect of a linear increase in the number of vulnerabilities on the MCS-based metric under different network models

As evident from Figure 5.9, and 5.10, irrespective of the network model, there is a dramatic change in the MCS and GED based metrics because of the introduction of vulnerable host(s) in the network. Moreover, significant change is observed when there is an increase in the number of vulnerabilities in the *target Host T*. When the vulnerabilities in the target Host T increase, its effect on the MCS and GED based metrics is significant compared to the effect of increase in the proportional number

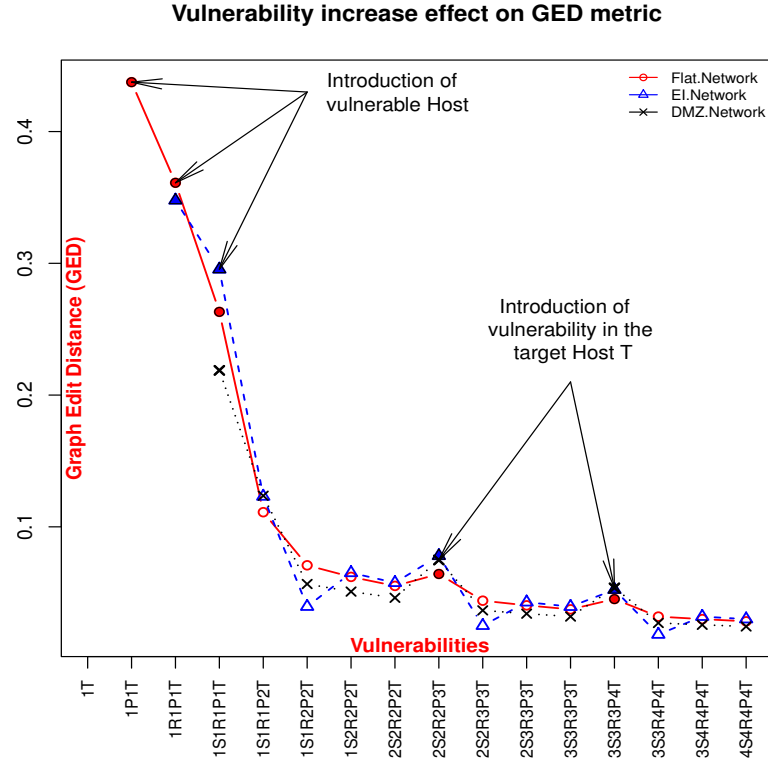


Figure 5.10: The effect of a linear increase in the number of vulnerabilities on the GED-based metric under different network models

of vulnerabilities on other hosts in the network (i.e. Hosts P , R , and S). Once new vulnerable host(s) stops joining the network, and there is an increase in the number of vulnerabilities on the target T , there will be significant change in the attack surface. This occurs due to the increased connectivity between the target T and other hosts that belong to the same *protection domain*. It increases the attacker's ability to start attacking from any other host belonging to the same protection domain and hence in turn provides more opportunities to compromise the target host T . This observation suggests that the new host that has to be added to the *protection domain* should be inspected more closely than the hosts that are already part of the protection domain. The effect of above stated events on the MCS and GED is illustrated in Figure 5.9, and 5.10. Each event is annotated and the respective attack graph pair is indicated by arrows on the plots wherever necessary.

The key observation here is that even though the number of edges in the attack

graphs corresponding to the flat network model (F) grows faster than the other two network models (as shown in Figure 5.8), all the three network models share the similar change in MCS and GED based metric results as shown in Figure 5.9, and 5.10. This is because of the computational steps proposed for both MCS and GED (as explained in Section 5.5) are roughly similar and the relative nature of their calculation methods.

Furthermore, as evident from Figure 5.9, and 5.10 the MCS and GED based metric values for EI and DMZ network models are not defined as far as the attacker can exploit a sufficient number of vulnerabilities in order to reach the target Host T . This observation suggests yet another strategy for protecting vulnerable hosts in the network. According to the strategy, if a particular Host H in a network is vulnerable (or likely to be more vulnerable) than other network hosts then access to the host H should be reduced. This can be achieved by applying the *principle of least privileges (POLP)* to the software programs or processes or services running over the host H . In practice, the hardening countermeasure such as limiting access to the host can be accomplished either by closing the ports, shutting down the vulnerable service or reconfiguring the service to increase the effort required to access the service.

5.7.3 Assessing the Effect of Non-linear Variation in the Vulnerability Density on the Network Attack Surface

We observed the effect of the linear assignment of vulnerable host(s) and also vulnerabilities on the network attack surface for all the three network models. Now it's time to evaluate the sensitivity of MCS and GED based metrics to the following network events:

- arbitrary variation in the vulnerability density,
- removal of a particular host from the network,
- patching of all vulnerabilities on a particular host.

In particular, we vary the number of vulnerabilities in the network in a non-linear fashion as shown in Figure 5.11, and 5.12. From the plots it is evident that irrespective of the network model, there is a significant change in the MCS and GED values, when new host is added to the network. A similar effect is observed when the existing host is removed from the network or all the exploitable vulnerabilities on a particular host are

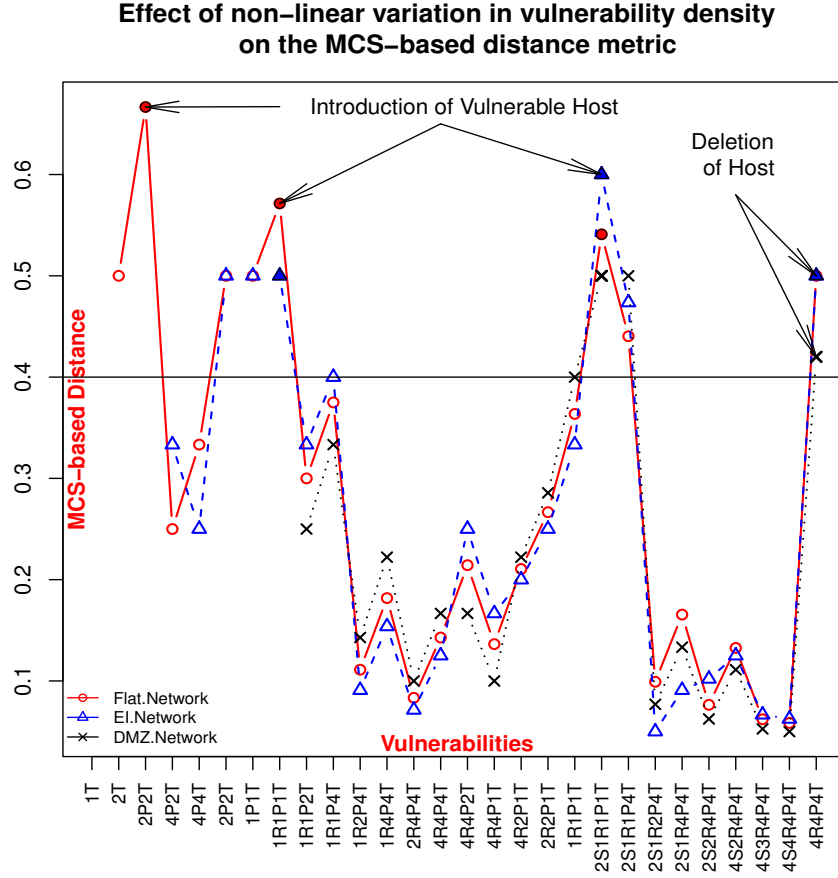


Figure 5.11: The effect of non-linear variation in the vulnerability density on the MCS-based metric under different network models

patched. This effect is clearly visible for the attack graph pair $\langle 4S4R4P4T, 4R4P4T \rangle$ in Figure 5.11 and 5.12. The *horizontal line* in each plot, specifies the manually defined threshold. The graph distance above the threshold represents there is a large change in the network attack surface. Further, there is a significant change in the attack surface because of the higher variation in the vulnerability density at the network level. Such change is in proportion (comparable) to the change caused by the addition (or removal) of a vulnerable host. Such observation is clearly visible for the attack graph pairs $\langle 4P4T, 2P2T \rangle$, $\langle 2P2T, 1P1T \rangle$ and $\langle 2S1R1P1T, 2S1R1P4T \rangle$ in Figure 5.11 and 5.12. The graph distance for these graph pairs is comparable to the distance computed for the attack graph pair $\langle 1P1T, 1R1P1T \rangle$. The effects of above stated events (i.e. An introduction/removal of vulnerable host, and higher variation in vulnerability density) on

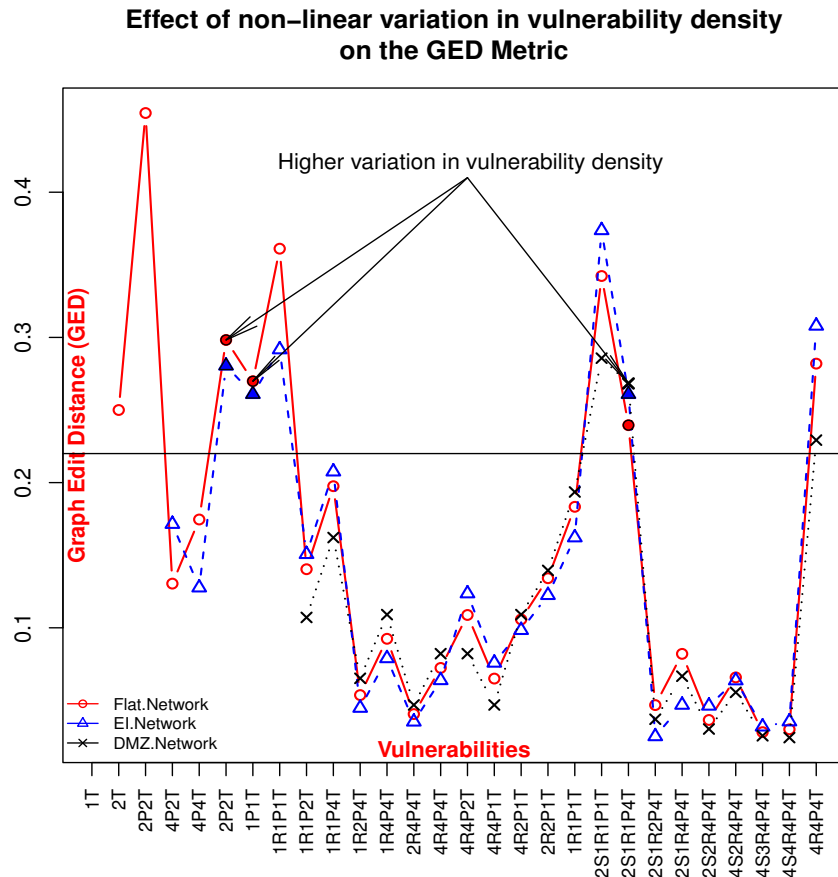


Figure 5.12: The effect of non-linear variation in the vulnerability density on the GED-based metric under different network models

the MCS and GED metrics are shown separately in Figure 5.11 and 5.12, where each event is annotated and the respective attack graph pairs are indicated by the arrows on the plots. The marked graphs correspond to the significant change in the graph distance metric (network attack surface) and coincide with the major events in the enterprise network.

For easier illustration, Figure 5.13 shows the change in the network attack surface of external-internal (*EI*) network. The effect of security events discussed above on the network attack surface is clearly depicted in Figure 5.13a and 5.13b.

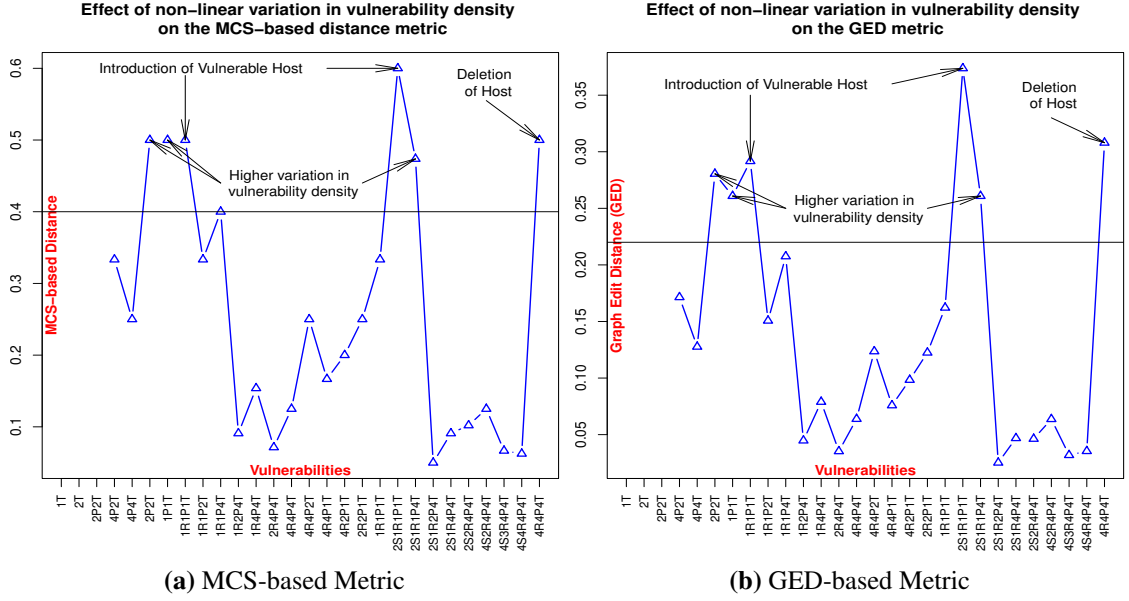


Figure 5.13: The effect of non-linear variation in the vulnerability density on the MCS and GED-based metrics for the *EI* network model.

5.7.4 Assessing the Effect of Vulnerability Variation in the Host(s) Directly Accessible to an Adversary or Belonging to the Protection Domain.

We have repeated the experiment by maintaining the same vulnerability density across the network (i.e. At the network level). It means that at any point of time, the number of vulnerabilities at host level may vary, but the total number of vulnerabilities across the network remains constant. Here our goal is to identify most dangerous hosts in the network as per their location, i.e. the subnet (internal/DMZ) they belong to. We want to identify the vulnerable hosts that cause major changes in the network attack surface and record their position. Further, our goal is to evaluate the concept of protection domain and the effect of increased connectivity among the hosts of protection domains. In particular, we want to investigate the effect of the following two events on the network attack surface, (i) *increasing vulnerabilities in the host, which is directly accessible to the adversary*, and (ii) *increasing vulnerabilities in the hosts belonging to the protection domain*.

As shown in Figure 5.14 and 5.15, the marked text alongside each point in the

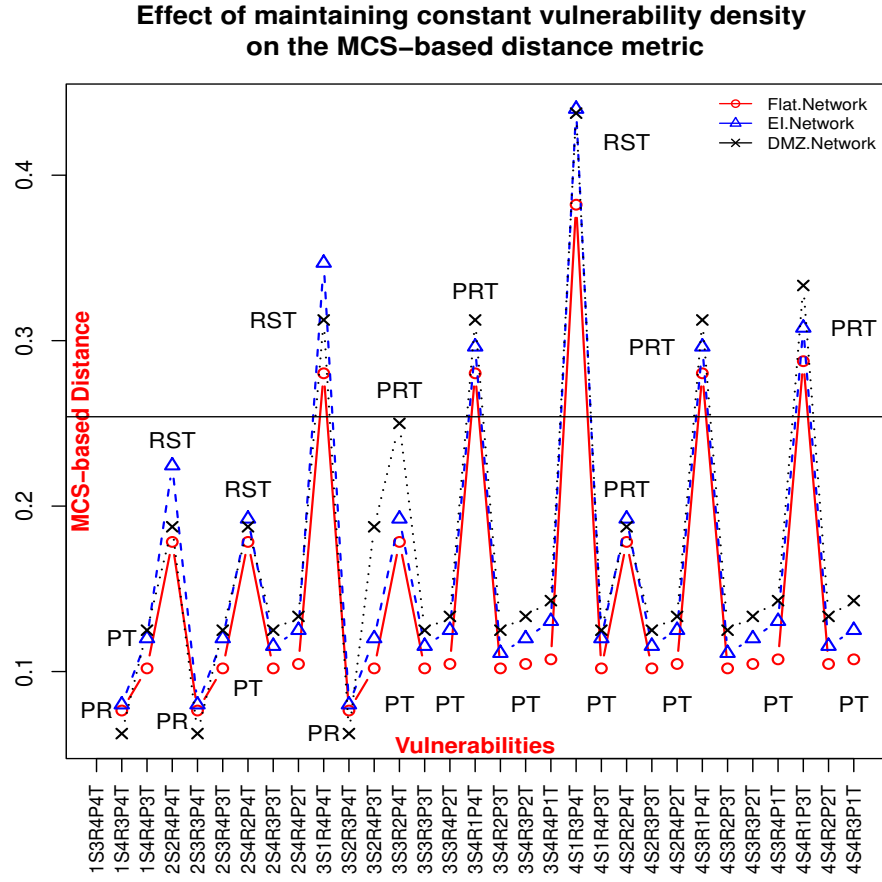


Figure 5.14: The effect of maintaining constant vulnerability density (at the network level) on the MCS-based metric under different network models

plots indicates the network hosts which are responsible for the change in the network attack surface. For example, the text *RST* indicates that the variation in the network attack surface is due to the larger change in the vulnerability density of hosts *R*, *S*, and *T* together. As evident from Figure 5.14 and 5.15 the Host *P* (in the *EI* and *DMZ* network models) which is directly accessible to an adversary has less impact on the attack surface. In other words, variation in the number of vulnerabilities at Host *P* results in a small change in the network attack surface. Whereas Hosts *R*, *S*, and *T* have comparably greater impact irrespective of the network model. In case of *EI* network, Hosts *R*, *S*, and *T* cause greater change as shown in the attack graph pair $\langle 3S4R4P1T, 4S1R3P4T \rangle$. The highest peak in the plots indicate higher variation in the network attack surface, and it is due to the high variation in the vulnerability density

Effect of maintaining constant vulnerability density on the GED metric

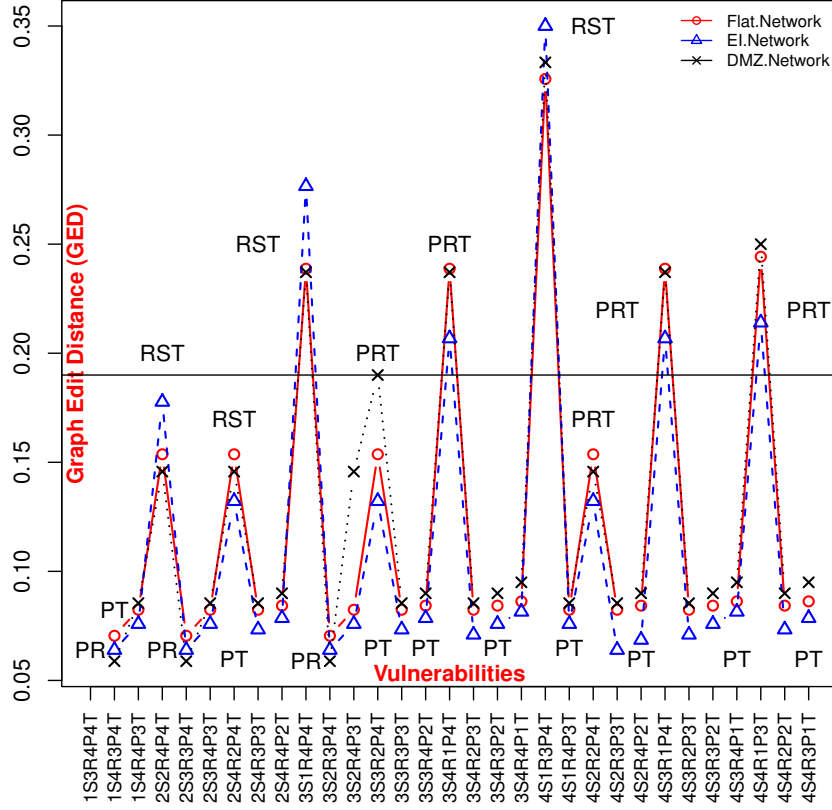


Figure 5.15: The effect of maintaining constant vulnerability density (at the network level) on the GED-based metric under different network models

of hosts R , S , and T belonging to the same protection domain. Further, significant change in attack surface results from the change in the vulnerability level of Hosts P , R , and T as shown in graph pairs $\langle 3S3R4P2T, 3S4R1P4T \rangle$, $\langle 4S2R4P2T, 4S3R1P4T \rangle$, and $\langle 4S3R4P1T, 4S4R1P3T \rangle$ of Figure 5.14 and 5.15. The hosts that belong to the internal network have greater impact on the attack surface compared to the hosts that belong to the DMZ . It is because of the increased connectivity among the hosts that belong to the same protection domain. To conclude, the vulnerable host introduction in a protection domain leads to the higher change in the network attack surface compared to the introduction of vulnerable host(s) in the DMZ . This observation suggests that the new host that wants to join a protection domain should be inspected more closely than the hosts that are already part of the protection domain.

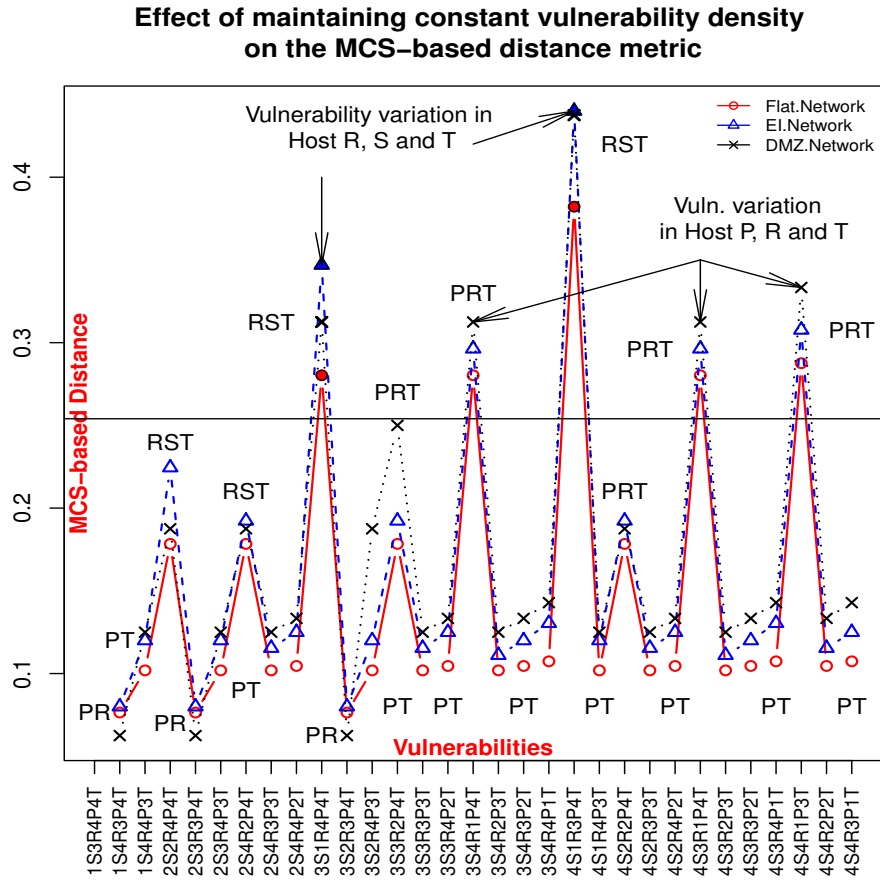


Figure 5.16: The effect of maintaining constant vulnerability density (at the network level) on the MCS-based metric under different network models. The marked text alongside each point in the plot indicates the network hosts which are responsible for the change in the network attack surface.

As shown in Figure 5.16 and 5.17, the marked attack graph pair corresponds to the significant change in the network attack surface and coincides with the above stated events in the enterprise network. Each of the events is annotated and respective attack graph pair is indicated by arrows on the plots wherever necessary. For easier illustration, Figure 5.18a and 5.18b, respectively, shows the change in the network attack surface of *EI* and *DMZ* networks. The effect of security events discussed above on the network attack surface is clearly depicted.

Effect of maintaining constant vulnerability density on the GED metric

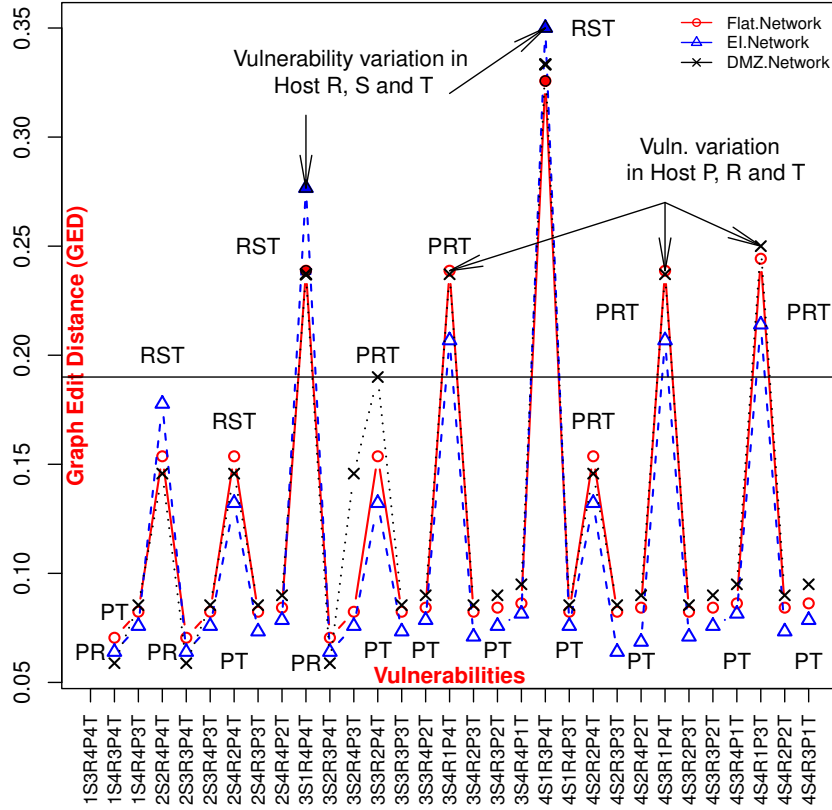


Figure 5.17: The effect of maintaining constant vulnerability density (at the network level) on the GED-based metric under different network models. The marked text alongside each point in the plot indicates the network hosts which are responsible for the change in the network attack surface.

5.7.5 Summary of the Results

Essentially, security configuration parameters, for example, (i) vulnerable service connectivity and (ii) vulnerabilities in the running services, are the two major pre-conditions vital for the successful exploitation of any remote vulnerability. These pre-conditions are called initial conditions since these are present in the network from the beginning (i.e. Prior to the vulnerability exploitation). If any one of the above initial conditions is missing, then the adversary cannot be able to exploit the vulnerability. In dynamic networks these attack surface components are changing with time. If one component (i.e. Initial condition) is kept constant (fixed) and the other is varied, then

5.7 Results and Analysis

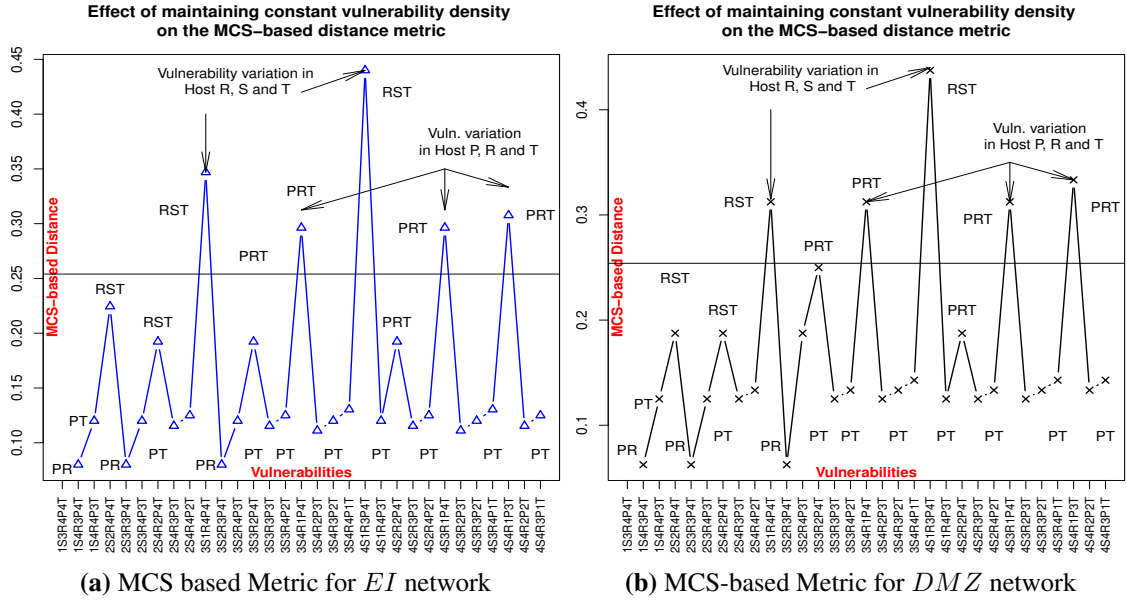


Figure 5.18: The effect of maintaining constant vulnerability density (at the network level) on the MCS-based graph distance metric under *EI* and *DMZ* network models.

we have the following four scenarios as shown in Table 5.5 (in a 2×2 matrix form):

Table 5.5: Network configuration versus vulnerabilities

		Vulnerabilities	
		Fixed	Varying
Network configuration	Fixed	Scenario 1 Fixed, fixed	Scenario 2 Fixed, varying
	Varying	Scenario 3 Varying, fixed	Scenario 4 Varying, varying

- Scenario 1 (*fixed network configuration, fixed number of vulnerabilities*): This is not a realistic situation in the case of dynamic networks.
- Scenario 2 (*fixed network configuration, varying number of vulnerabilities*): For this scenario MCS and GED based metrics weigh the host designated as a target host (here Host *T*) more than the other hosts in the network. Whenever there

is an increase in the number of vulnerabilities in the target host, its effect is greater compared to the effect of other host(s) in the same network increasing its vulnerabilities by the proportional number. Therefore, MCS and GED assist analysts in identifying the hosts which have a greater impact on the network attack surface. Further, in this scenario, analysts can find out most dangerous services and hence service connectivities, so that the network hardening efforts can be prioritized efficiently. The result of this scenario is shown in Figure 5.9 and 5.10.

- Scenario 3 (*varying network configuration, fixed number of vulnerabilities*): In this scenario, an analyst can identify the network configuration elements that cause major changes in the network attack surface. Changes in the root configuration can be of type, for example, introduction/removal of vulnerable hosts and security devices (i.e. Firewall, and gateway router), enabling/disabling of services, opening/closing of ports, change in the access control policies, etc. At a finer granularity, in this scenario, an analyst can identify the host/services/access control policies whose addition (or removal) can enable and disable the vulnerabilities. The result of this scenario is shown in Figure 5.9 and 5.10.
- Scenario 4 (*varying network configuration, varying number of vulnerabilities*): This scenario captures the combined effect of both, change in the network configuration elements and vulnerability density. Change in the network configuration enables/disables existing vulnerabilities. Even though a number of vulnerabilities increase in the network over time Δt , due to the change in the network configuration not all vulnerabilities will appear in the attack graph at time $t + \Delta t$. Simply enumerating the number of vulnerabilities does not suffice the network security management. Therefore, the analyst needs to focus on reachability in terms of vulnerable service connectivity as one of the network hardening options. The result of this scenario is shown in Figure 5.11 and 5.12.

In summary, MCS and GED-based metrics are sensitive enough to detect the introduction of new exploitable vulnerabilities, vulnerable hosts in the network, and change in the network configuration. Table 5.6 shows the sensitivity of the metrics we discussed in this Chapter to the different network and security events. Here, ✓ and ✗ marks indicate the sensitivity and insensitivity respectively of a given metric to the events. We found that the MCS and GED-based metrics are sensitive to all kinds of changes in the network attack surface. Such sensitivity analysis provides the impact

Table 5.6: Sensitivity of the attack graph based metrics to the network security principles and network/security events.

Metric	Network topology/network security factor					
	Network segregation/partitioning (defense-in-depth)	Increased connectivity in the protection domain	Principle of least privileges (POLP)	Importance of the zone (DMZ and internal)	Vulnerable host introduction/removal	Increase/decrease in the number of vulnerabilities
(1) Shortest path (SP)	✓	✗	✓	✗	✗	✗
(2) Number of paths (NP)	✓	✓	✓	✓	✓	✓
(3) Mean of path lengths (MPL)	✓	✗	✓	✗	✗	✗
(4) Normalized <i>MPL</i> (NMPL)	✓	✓	✓	✓	✓	✗
(5) Median of path lengths (MDPL)	✓	✗	✗	✗	✗	✗
(6) Mode of path lengths (MoPL)	✓	✗	✗	✗	✗	✗
(7) Standard deviation of path lengths (SDPL)	✓	✓	✓	✗	✓	✗
(8) Network compromise percentage (NCP)	✓	✗	✗	✗	✗	✗
(9) Weakest adversary (WA)	✓	✗	✗	✗	✗	✗
(10) Maximum common subgraph (MCS)	✓	✓	✓	✓	✓	✓
(11) Graph edit distance (GED)	✓	✓	✓	✓	✓	✓

of the network (security) events on the network attack surface. And hence, MCS and GED-based metrics can be used as a useful tool while monitoring temporal aspect of an enterprise network security. Using such metrics, analysts get alerted about the significant change in the attack surface, so that security events, which are responsible for the change, can be inferred. In other words, to say that these metrics facilitate detection of major security events that cause a significant change in the attack surface and hence help in an early warning system for potential intrusion. Number of paths (NP) metrics is also sensitive to all the events, but at a very coarse level. Sensitivity of *NP* to the network/security events is discussed in Subsection 5.7.1. From the temporal analysis standpoint, each network model in Figures 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15 shows typical behavior. Once the significant change in the network attack surface is identified, the analyst has to identify the external causes responsible for the change. Finally, based on the identified events, the network needs to be reviewed (reconfigured) for providing adequate security.

The use of the MCS and GED based graph distance metrics in the attack graph context complements traditional network monitoring approach for security risk mitigation. These metrics can be used as an additional security assessment tool by the security analysts. According to the expert survey and interview conducted by [168], correctness, measurability, and meaningfulness are the main quality criteria for security metrics. These quality criteria form the basis for credibility and sufficiency for the security metrics under consideration. Graph similarity metrics used in this study conform (suffice) to the defined quality criteria: correctness, measurability, and meaningfulness and hence are proved to be usable for the measurement of change in network attack surface.

5.8 Computational Complexity

In this section, we discuss the computation time of the graph similarity metrics such as MCS and GED used in our study. Measured time does not include the attack graph construction time and also the time required to derive the labeled representation of the attack graphs. This is true for all metric computation times in our experiment.

In practice, for a class of graphs with unique node labels there exist computational procedures that compute the maximum common subgraph (MCS) and graph edit distance (GED) for a pair of graphs in quadratic time with respect to the number of nodes

in the underlying graph [169]. In general, there are $\mathcal{O}(n^2)$ edges in any graph having n nodes. However, there are certain application areas where graphs are of bounded degree. For example, in case of the attack graphs the maximum number of edges incident to any node is bounded by a constant k . Therefore, the expression $\mathcal{O}(n^2)$ reduces to $\mathcal{O}(n)$.

In a logical attack graph generated for our experimental setup, there are three types of nodes, namely, (i) Initial Conditions, (ii) Exploits, and (iii) Post-conditions. The indegree and outdegree of each vertex type are shown in Table 5.7. The maximum edges incident on each vertex (node) type are bounded by a constant k (upper bound). For initial conditions (which are not post-conditions of any of the exploit) there is no single edge incident over it. For post-conditions, there may be one or more edges incident over it. Exploit node may have a number of edges incident over it, but not more than 3. In general, in a logical attack graph for any type of node the value of k is not more than 3.

We have generated an attack graph sequence \mathcal{S} for our experimental setup (described in Section 5.6). Each graph in a sequence (except the first one) is compared with immediate predecessor for similarity (or distance) using the MCS and GED based metrics. Figure 5.19 shows the plot of MCS and GED metric computation times against the number of nodes in an attack graph. As evident from Figure 5.19, the computational time is very similar and almost indistinguishable (very close) for both the metrics. Further, it shows the low computational complexity of the theoretical results, i.e. $\mathcal{O}(n^2)$.

System Configuration: *The hardware platform that we used in our experiment to measure computational times is a Windows 7 Professional with Intel (R) Core (TM), i5-3230M CPU @2, 60GHz and 8 GB of RAM. The hardware platform we used is not important and has been provided for completeness only. Here, only relative computation times that are required for the computation of MCS and GED metrics with respect to graph dimensions are important.*

Due to the unavailability of standard test graphs in the attack graph domain, the experimental evaluation of the used metrics has been performed using the synthetic graphs generated using available topology generators such as Fan-Chung algorithm [170]. We have used synthetic data sets in order to verify the computational complexity of MCS and GED based metrics. Our goal is to verify the linear dependency of the time required for computation of MCS and GED metrics. This test is used to find out whether the results obtained for experimental networks can be applied/generalized

Table 5.7: The indegree ($id(u)$) and outdegree ($od(u)$) of each vertex type in logical attack graphs

Vertex u	$id(u)$	$od(u)$
Initial condition	0	≥ 1
Exploit	≥ 2	≥ 1
Post-condition	≥ 1	≥ 1

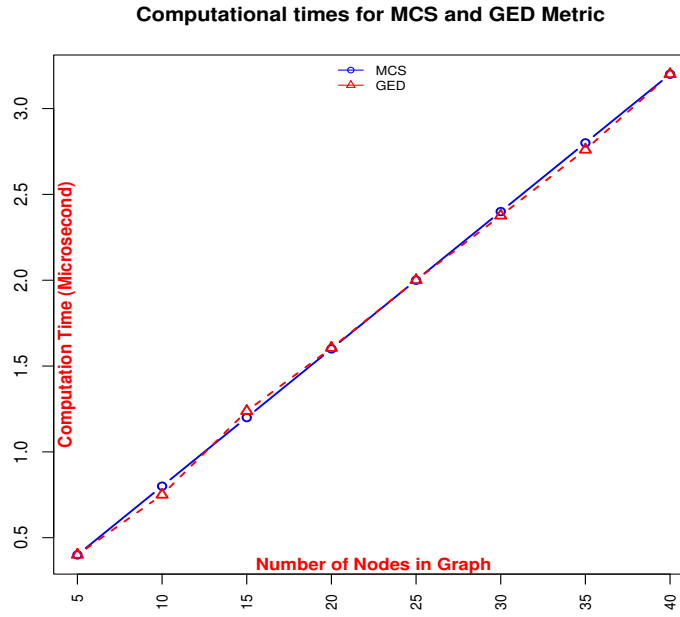


Figure 5.19: Computation time in microseconds for maximum common subgraph (MCS) and graph edit distance (GED) metrics.

to large scale implementations in real-world networks. Additionally, our goal is to verify whether MCS and GED computation times are *topology-independent*. We have produced two synthetic data sets which constitute normally-distributed random edges with average edge density of 0.02%. The data sets with 0.02% average edge density are produced to imitate the characteristics of the logical attack graphs. We have used Fan Chung algorithm [170] to create such synthetic data sets. This graph generator algorithm produces graphs where vertex degrees follow a power-law distribution. The resulting graph topology produced using this algorithm is entirely different

from those of the actual attack graphs. For each data set having an average edge density of 0.02%, we obtain a graph series (S), where each graph in a series contains 100, 1000, 3000, 5000, 7000, and 10,000 nodes. One more series S' was created as a counterpart, using the same procedure as above, for the measurements of MCS and GED computation times. The experiment is repeated for different values of average edge density, such as 0.025% and 0.03%.

For measuring the computational time taken by MCS and GED metrics, we need both the graph series, i.e. S and S' . To compute the time taken by MCS and GED, we have selected the first graph G_1 from the series S , which consists of 100 unique nodes. The graph G'_1 from the series S' with equal number of nodes (as in G_1) but few with different label representations (compared to graph G_1) is chosen. Graphs G_1 and G'_1 are equivalent in their size, but few nodes have different label representations. These two graphs G_1 and G'_1 are evaluated for *MCS* and *GED* metrics. The above procedure is repeated for the other graphs in a series S having graph sizes (i.e. nodes) 1000, 3000, 5000, 7000, and 10,000. The results of all computational time measurements are shown in Table 5.8 and 5.9. As expected, the measured computational complexity of all matching algorithms is $\mathcal{O}(n^2)$. Figure 5.20 and 5.21 illustrates this observation for MCS and GED, respectively.

Table 5.8: Computation time for the MCS-based metric.

Edge density	Computational times (in microseconds) for graphs with n vertices						
	n=50	100	1000	3000	5000	7000	10,000
0.02%	4	80	3040	27,520	89,680	194,240	418,880
0.025%	50	100	3800	34,400	112,100	242,800	523,600
0.03%	60	120	4560	41,280	134,520	291,360	628,320

As shown in Figure 5.20 and 5.21, the x-axis corresponds to the number of nodes in the graphs and the y-axis depicts the time, in seconds, to compute graph distance metric. The effect of larger edge densities on the computation time of metrics is clearly visible in Figure 5.20 and 5.21. It shows higher computational times required for graphs with larger edge densities. Such result is expected due to the metrics dependency on graph elements (features) such as nodes and edges. The computational times

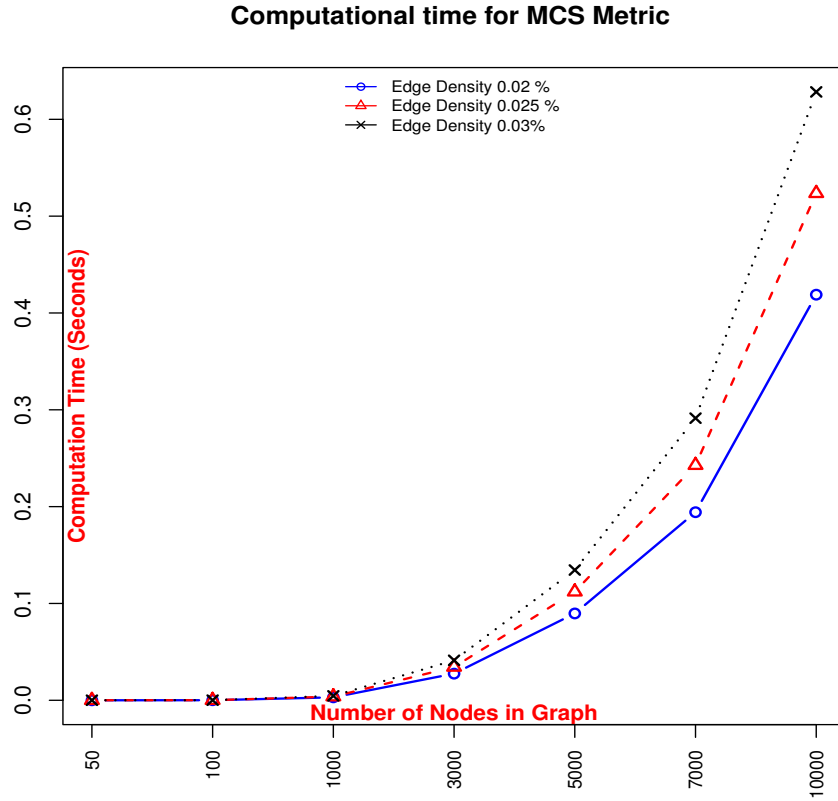


Figure 5.20: Computation time for MCS

Table 5.9: Computation time for GED-based metric.

Edge density	Computational times (in microseconds) for graphs with n vertices						
	$n=50$	100	1000	3000	5000	7000	10,000
0.02%	4	80	3200	27,920	90,720	195,680	416,720
0.025%	50	100	4000	34,900	113,400	244,600	520,900
0.03%	60	120	4800	41,880	116,080	293,520	625,080

for both MCS and GED (as observed in Figure 5.20 and 5.21) are very similar and almost indistinguishable. This is not surprising, since the computational steps proposed for MCS and GED are roughly similar. Therefore, analysts can manage with (or employ) any one of the graph distance metrics either MCS or GED, since both are equally capable of detecting variation in the attack surface. The results (i.e. Time complexity)

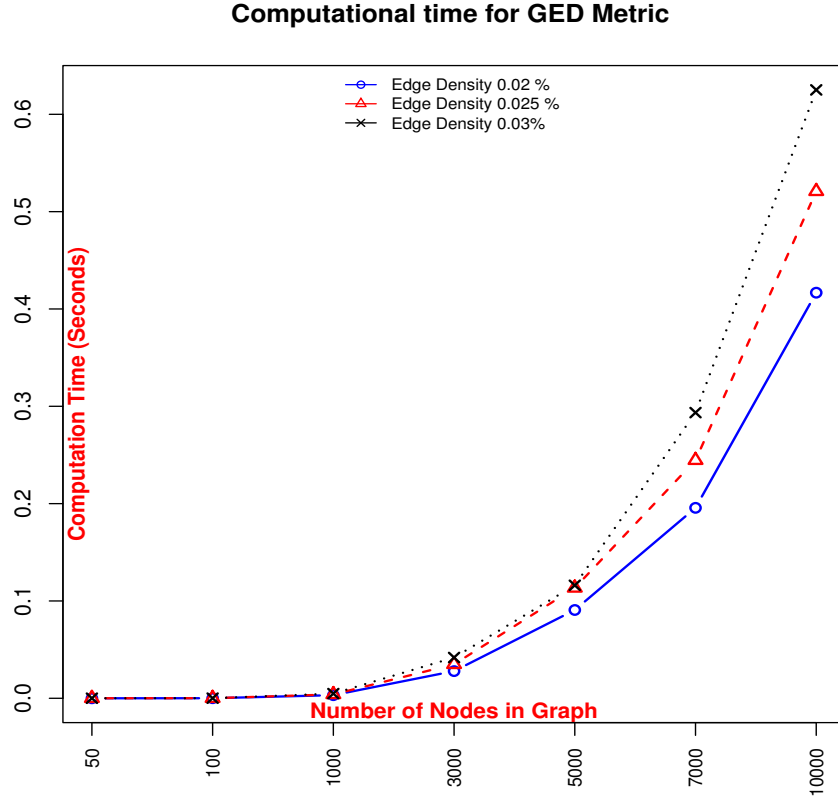


Figure 5.21: Computation time for GED

for both the metrics would be exactly the same if the numbers of edges in graphs from both the series were equal. Since the graph generation algorithm produces graphs with randomly-distributed edges, creates graphs with a specified average edge density, the actual number of edges in the graphs can vary. And hence closeness of the results verifies that MCS and GED computation times are *topology-independent*. In other words, used graph similarity metrics are neutral to the graph structure and applicable to the pair of attack graphs of any size.

5.9 Summary

In this Chapter, we have examined 11 different metrics and measured their effectiveness in capturing temporal variations in the network attack surface. We found that the graph distance metrics based on the Maximum Common Subgraph (MCS) and Graph Edit Distance (GED) are good at detecting changes in the network attack surface of

time-varying (dynamic) computer networks. We have also analyzed the sensitivity of the graph distance measures to network (or security) events such as changes in the network configuration, introduction of vulnerable hosts, variation in the vulnerability density, etc. Our experimental results show that the MCS and GED based graph distance metrics are capable of efficiently capturing the relative change in the network attack surface. Interestingly, computation of one metric is enough, as both are very similar and equally capable of detecting variation in the network attack surface.

In this section, we discuss about (i) the automation perspective of the proposed approach of change assessment, (ii) the applicability of the proposed approach to the generic networks that can be a mixture of Cyber and physical elements, (iii) finally, scope and limitations of the MCS and GED based graph distance metrics explored in this Chapter.

1. Automating the Process of Change Assessment in the Network Attack Surface

Coming to the automation aspect of our proposed approach of change assessment, automated and time-efficient generation of attack graphs is possible using the tools such as NetSPA [1], MulVAL [16], CAULDRON [28]. The generated attack graph pair $\langle G_1, G_2 \rangle$ adjacent over the sampling interval Δt needs to be processed to extract the *require edges* in each graph. We have presented an algorithm 5.1 in Section 5.4 to extract such edges. Extraction of uniquely labeled *require edges* from the consecutive attack graphs (adjacent over the time) is central to the automation of proposed change detection technique as it enables the use of graph distance metrics. Similarly, the newly introduced *require edges* in an attack graph G_2 over the sampling interval Δt can be extracted and also the shared portion of the attack graphs in terms of *require edges*. Upon computation of the graph distance metric, its value is compared with the manually defined threshold, and if above the threshold, an alert has to be generated. Such metrics alert security analyst whenever there is a significant change in the network attack surface.

2. Applicability of the Proposed Change Assessment Technique

As evident from the results discussed in Section 5.7, proposed graph distance

based approach offers a practical solution for the detection of change in the network attack surface of dynamic networks. Such approach can also be used to measure the impact of dynamics (on the security strength) of a generic network that can be a mixture of the Cyber and physical elements. Cherdantseva et al. [171] reviewed the state of the art in security risk assessment of CPS such as SCADA. Essentially, in a dynamic CPS, devices connect opportunistically and carry out general-purpose computations on behalf of the other devices. However, some of the CPS devices with potentially malicious intent can affect the integrity of the computation. Moreover, the emergence of cloud-based Cyber physical system complicates the situation because cloud computing brings out other possible avenues of Cyber attacks. The dynamics of the physical devices along with risk of Cyber attacks from potentially malicious devices poses a great challenge to the security assessor in the Cyber physical system. We argue that the concept of graph-based change detection in attack surface may be applied to the system of Cyber physical devices. Therefore, as an immediate application, one can apply our approach of change assessment in the attack surface of dynamic networks to a generic network that can be a mixture of Cyber and physical elements.

3. Advantages and Scope of the Proposed Change Assessment Technique

The change in the network attack surface is either because of the variation in the vulnerability density or changes in the network configuration or both thereof. The graph distance metrics we explored in this study can be used to generate an alert whenever there is a significant change in the attack surface. This change can be of two types: incremental or decremental. In the first case, there is an increase in the attack surface because of the increase in the number of vulnerabilities at network level, change in the network configuration or network misconfiguration, etc. On the other hand, there is a decrease in the network attack surface because of the application of security countermeasures such as vulnerability patching, service/OS hardening, re-dimensioning of the network, etc. The alert for the first type of change indicates the introduction of vulnerable hosts, increase in the number of vulnerabilities, network misconfiguration, failure of security device, loose access control policies, etc., whereas the second type of change indicates the effectiveness of the security countermeasures. So, it is a job of security

analyst to drill down the exact root causes once the graph distance metric triggers an alert. So, in order to inform the analyst clearly about the positive or negative change in the attack surface, the graph distance metrics need to be modified accordingly.

In almost all organizations, there is a common practice of logging network related activities in the form of network traffic, user activities, etc. Such logged data can be used for forensic analysis in case of a security breach (i.e. Post security incident). Likewise, generated attack graphs for a given network need to be maintained. Mere scanning and maintaining record of vulnerabilities in the network does not facilitate forensic analysis in case of network intrusion (security breach). Vulnerability reports do not provide any clue on *how the vulnerabilities are combined and exploited for incremental access*. As seen in Section 5.4, attack graph is nothing but the attack surface of the underlying computer network. It captures exploitable vulnerabilities, network misconfiguration, vulnerable service connectivities, loose access control policies, etc. The sequence of attack graphs generated over time helps in tracking the size of the network attack surface. The differential of the successive attack graphs provides clues to how the attack surface is evolving. If we have a practice of continuously generating attack graphs for the network, then such graphs will assist in figuring out the trend of the variation in the attack surface of the underlying network. Further, in case of a security incident, these attack graphs can help in detecting which attack path attacker has taken to compromise the security of the system. Such practice also reveals what sort of vulnerabilities and service connectivities was used by the attacker. If analysts combine the attack graphs with anti-forensic datasets, it will greatly enhance their capability in performing forensic activities. In order to facilitate efficient and real time analysis the graph databases such as [172], [173], etc. can be used for storing a sequence of attack graphs.

4. Limitations of the Proposed Metrics

We observed in Section 5.7 that the Maximum Common Subgraph (*MCS*) and Graph Edit Distance (*GED*) based metrics are capable of measuring the relative degree of change in the attack surface experienced by the networks over the sampling interval Δt . The limitation of such graph distance metrics is that they are relative and cannot be interpreted (or used) in an absolute sense. These metrics

make sense only when a security assessor can compare the metric values computed for the same network at the continuous times. Therefore, the metric results from the graphs at discontinuous times are incomparable since the base reference graphs in graph pairs do not really correspond to the same network. These metrics can be used to assess the change in attack surface of a given network only when Δt is small and the input graphs are adjacent in time. Unlike previously proposed attack path and non-path based metrics, these metrics cannot be used for comparing two or more different networks in terms of their security strength. This is because of the relative nature of their calculation methods.

Furthermore, the *MCS* and *GED* based metrics do not capture the *AND/OR* semantics in the attack graphs. Essentially, we do not need to capture such semantics to fulfill our purpose of assessing temporal change in the network attack surface. In the context of network security, the administrator's focus should be on the initial conditions such as exploitable technical vulnerabilities, vulnerable service connectivities, etc. Such initial conditions (i) exist in the network from the beginning (i.e. Prior to the vulnerability exploitation), (ii) can be patched independently to stop exploits from execution, and (iii) represent the network resources that an adversary can use against the network itself. Our notion of the network attack surface (Subsection 5.4.1) constitutes initial conditions. The contribution of such initial conditions (or resources) to the network attack surface is captured through the use of *require edges* (i.e. The edges between initial conditions and exploits) in a logical attack graph. As one initial condition can enable one or more number of exploits, its contribution to the attack surface can be quantified through the number of outgoing "require edges". Since, *require edge* is sensitive to the changes in the network security configuration, the resulting graph distance metrics are also sensitive to the incurred changes. As evident from the experimental results (Section 5.7) without considering the *AND/OR* semantics in an attack graph, we are able to achieve our goal and detect temporal changes in the network attack surface.

In conclusion, the *MCS* and *GED* based graph distance metrics are oblivious (unaware) of the importance of *AND* semantics between the initial conditions in the attack graphs. Thus, there is a scope for improving their performance (sensitivity) by considering the *AND* semantic. The *OR* semantic is inconse-

quential as it exists between post-conditions. That is, what happens after the attack, the aftermath of attack depends on the severity of the vulnerability(ies) and attacker's intentions. Such post-conditions (consequences) cannot be removed without removing their actual causes (i.e. initial conditions).

Additional work is needed to enrich our knowledge about the semantics of variations in the attack graphs. As *MCS* and *GED* based graph distance metrics are oblivious to the significant *AND* semantic between the initial conditions and exploits in the attack graph, additional metrics need to be developed by considering the *AND* semantic. It can be done through the generation of the Boolean expression in the form of Disjunctive Normal Form (*DNF*) for each attack graph in a graph pair $\langle G_i, G_j \rangle$ and then comparing those DNFs for their similarity or dissimilarity. As an immediate future work, we propose to find the edit distance (in terms of network hardening cost) between a pair of consecutive attack graphs in order to reduce the network attack surface. The problem of finding minimum edit distance for efficient network hardening can be formulated as an optimization problem.

Chapter 6

Depicting Temporal Variations in the Network Attack Surface

This Chapter focuses on the problem of effectively visualizing the newly introduced changes in the attack surface of dynamic networks. We have developed a change distribution matrix (CDM) based technique to identify the incremental changes in the network attack surface over the observed sampling interval. For doing this, we have reduced the problem of change detection in the network attack surface to the error-correcting graph matching (ECGM) problem. We found that the CDM based technique identifies the root causes responsible for the increase in the network attack surface in time efficient manner. Further, it identifies the newly introduced exploits and their respective enabling conditions in the dynamic systems and discerns portion of the attack graph that suffered significant change. Such identification of modification in the network attack surface and the hidden root causes in a time-efficient manner is crucial to the prevention of future attacks.

6.1 Introduction

In the management and control of network security, early detection of critical events (i.e. the events that jeopardize the security of mission-critical resources) is fundamental to the capability building like an early warning system for possible intrusion. Early detection of such incidents, for example, illegitimate access to a database (or web server) can significantly enhance organizations vigilance towards potential Cyber

attacks. Today's computer networks are dynamic and undergoing continuous evolution in terms of size and complexity. Further, because of the end users increased flexibility in installing and configuring applications, frequent changes in the network topologies, constant discovery of new vulnerabilities, misconfiguration of hardware (or software) components, etc. present-day computer networks are undergoing continuous evolution. Consequently, the network attack surface is also under constant change. Essentially, attack surface of a given network is the subset of network configurations and vulnerabilities (known vulnerabilities in particular) that an adversary can exploit to compromise the critical resources. Assessing the degree of change in the network attack surface and pinpointing the root causes responsible is of utmost importance for proactive network hardening.

6.2 Motivation

Identification of possible threats to critical enterprise resources and proactive network hardening using an attack graph is a well-established research area [134]. Kaynar [135] provides a systematic study of the various tools and techniques available for the attack graph generation. Even though several tools, for, e.g. MulVAL [21], CAULDRON [28], NetSPA [153], etc. made it possible to efficiently generate attack graphs for realistic networks, resulting graphs are very complex and incomprehensible to human eyes. Manual analysis of such complex graphs is nearly impossible and hampers the process of network hardening. In order to extract security-relevant information embedded within the attack graph, a significant number of graph analysis techniques have been proposed. The primary goal of the attack graph analysis is to predict future attacks [174], determine the optimal set of exploits, and initial conditions [22], [23] for effective network immunization, and assess the risk posed by an adversary to the enterprise critical resources [132], [133]. Further, the number of metrics proposed in [25], [123] to measure security strength of given network configuration.

While the above stated attack graph-based approaches provide network hardening solutions, they are not useful in the context of dynamic attack surface. Previously proposed attack graph-based metrics [123], [47], [48], [49], [50], [25], [124] are too coarse to be useful for network security management. They fail to capture the relative degree of change in the attack surface experienced by the computer network as it evolves over time. In other words, the two network configurations having the same

value for the attack graph-based metric(s) may differ in their attack surface. Whereas, in practice security analyst usually wants to measure the relative degree of change in the attack surface. Further, she needs to know exactly which events are responsible for the increase in the attack surface. To conclude, existing attack graph-based metrics unable to alert about compelling events, do not provide any knowledge about “where in the attack surface changes has been occurred” and “what are the root causes of such change.”

Recently, Awan et al. [145] proposed a framework for measuring temporal variance in network risk. Author’s used a system log data collected from the university campus network to validate their framework. In contrast, our primary focus here is on the temporal aspects of vulnerable service connectivities and exploitable technical vulnerabilities, which are likely to be used by an adversary. The problem of identifying the change in the attack surface, and hidden root causes has motivated our work.

6.3 Running Example

To build intuitions about the problem we intend to solve, we consider a test network, as shown in Figure 6.1. $Host_1$, $Host_2$ and $Host_3$ comprise the internal network. Here, $Host_3$ is of paramount importance as *MySQL* is the critical service running over it. Host Ha is the anonymous user (with malicious intent), and her goal is to become superuser of the $Host_3$. So, our primary concern is whether an adversary can obtain root privileges on $Host_3$. Firewalls are put in place to allow all outbound connections, but blocks inbound connection requests to $Host_0$ only. Hosts internal to the network are authorized to connect to only designated ports and hence services described by the connectivity-limiting firewall policies as shown in Table 6.1. Here, *All* indicate that source host may connect to the destination host on any port in order to have access to the services running on those ports. On the other hand, *None* specifies that source host is prohibited from having access to any of the services running on the destination host. We assume that hosts in the test network have some initial vulnerabilities, which are summarized in Table 6.2 and indexed by CVE number.

A goal-oriented logical attack graph of the test network is shown in Figure 6.2(a). Here, attack graph has two kinds of vertices, namely, exploits and conditions and it shows all the paths available to an adversary to reach and compromise the target, i.e. $Host_3$. The attack graphs, we generated are goal-oriented, finite and contain neither

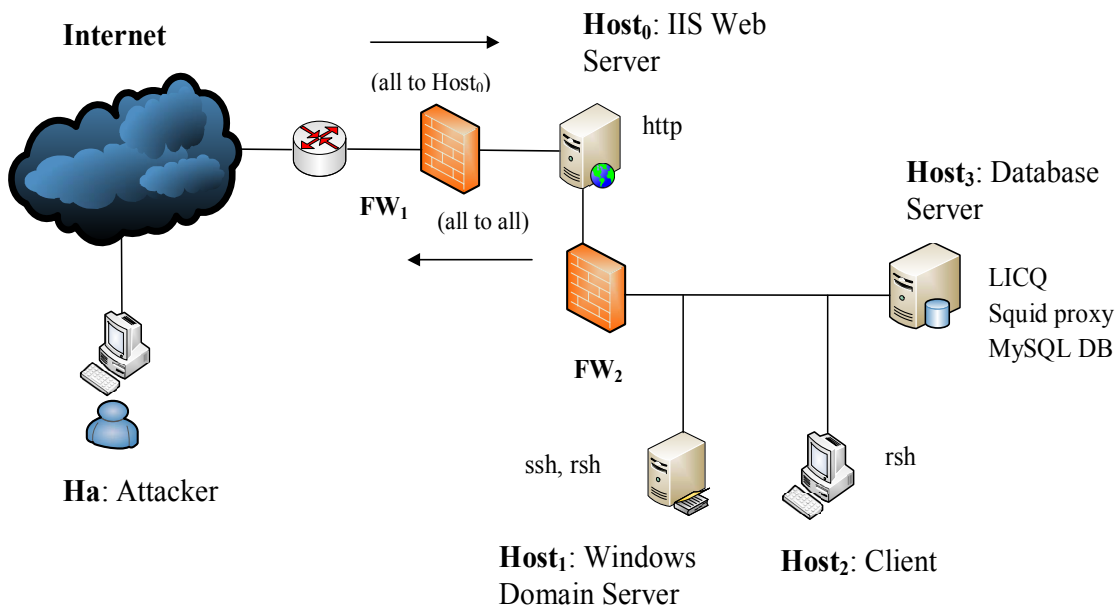


Figure 6.1: A test network

Table 6.1: Firewall rules for the test network at time t

Host	Attacker	$Host_0$	$Host_1$	$Host_2$	$Host_3$
Attacker	localhost	All	None	None	None
$Host_0$	All	localhost	All	All	Squid LICQ
$Host_1$	All	IIS	localhost	All	Squid LICQ
$Host_2$	All	IIS	All	localhost	Squid LICQ
$Host_3$	All	IIS	All	All	localhost

loops nor multiple edges. We have slightly customized MulVAL [21] so that it generates attack graphs where each node is labeled uniquely.

As the test network (Figure 6.1) is dynamic in nature, we expect certain changes in the form of change in the service connectivity, the introduction of the vulnerable software/services, the disclosure of new vulnerabilities, patching of vulnerabilities, etc. Such variations in the network configuration are highlighted in Table 6.3, and Table 6.4.

Table 6.2: Services and vulnerability description of the test network at time t

Host	Services	Ports	Vulnerabilities	CVE
$Host_0$	IIS Web Service	80	IIS Buffer Overflow	CVE-2010-2370
$Host_1$	ssh	22	ssh buffer overflow	CVE-2002-1359
	rsh	514	rsh login	CVE-1999-0180
$Host_2$	rsh	514	rsh login	CVE-1999-0180
$Host_3$	LICQ	5190	LICQ-remote-to-user	CVE-2001-0439
	Squid proxy	80	squid-port-scan	CVE-2001-1030
	MySQL DB	3306	local-setuid-bof	CVE-2006-3368

Table 6.3: Firewall rules for the test network at time $t + \Delta t$. Highlighted entries represent change in firewall policies over Δt .

Host	Attacker	$Host_0$	$Host_1$	$Host_2$	$Host_3$
Attacker	localhost	All	None	None	None
$Host_0$	All	localhost	All	Netbios-ssn	Squid LICQ
$Host_1$	All	IIS	localhost	None	Squid LICQ
$Host_2$	All	IIS	ftp rsh	localhost	None
$Host_3$	All	IIS	All	All	localhost

As there is a change in the network topology factors and security factors as well, there will be a change in the network attack surface. Goal-oriented logical attack graph for this network configuration (at time $t + \Delta t$) is shown in Figure 6.2(b). Here Δt is the arbitrary sampling interval. The decision of the Δt selection should be carefully done and it is heavily dependent on the expertise of the security assessor.

Figure 6.2 illustrates the differential visualization of attack graphs at the node level. Each *exploit* is shown by an oval, *initial condition* by a box and *postcondition* by a simple plain-text.

Quick identification of the change in the network attack surface and hidden root causes is crucial to the prevention of future attacks. The key observation is that the test

Table 6.4: Services and vulnerability description of the test network at time $t + \Delta t$. Highlighted entries represent newly introduced vulnerable services.

Host	Services	Ports	Vulnerabilities	CVE IDs
$Host_0$	IIS Web Service	80	IIS Buffer Overflow	CVE-2010-2730
	ftp	21	ftp buffer overflow	CVE-2009-3023
$Host_1$	ftp	21	ftp rhost overwrite	CVE-2008-1396
	rsh	514	rsh login	CVE-1999-0180
$Host_2$	Netbios-ssn	139	Netbios-ssn nullsession	CVE-2003-0661
	rsh	514	rsh login	CVE-1999-0180
$Host_3$	LICQ	5190	LICQ-remote-to-user	CVE-2001-0439
	Squid proxy	80	squid-port-scan	CVE-2001-1030
	MySQL DB	3306	local-setuid-bof	CVE-2006-3368

network at different instants of time (i.e. at t and $t + \Delta t$) have different attack surface. The remainder of this chapter is built upon this important observation and address the issue of measurement of change in the attack surface. Further, we are interested in pinpointing the root causes responsible for the change in attack surface.

Understanding the temporal differences between successive attack graphs generated apart in time by sampling interval Δt is usually the first important step towards finding the change in the attack surface of dynamic networks. The change detection in the attack surface, which is the primary focus of this Chapter is defined as follows:

Definition 3 (Change Detection in the network attack surface). .

Given: a sequence of goal-oriented logical attack graphs generated for the test network over the sampling interval Δt

Find:

1. the degree of change in the attack surface,
2. the top k -nodes (i.e. exploits, initial conditions) of the attack graph that contribute most to the change, as well as
3. root causes responsible for the change.

6.3 Running Example

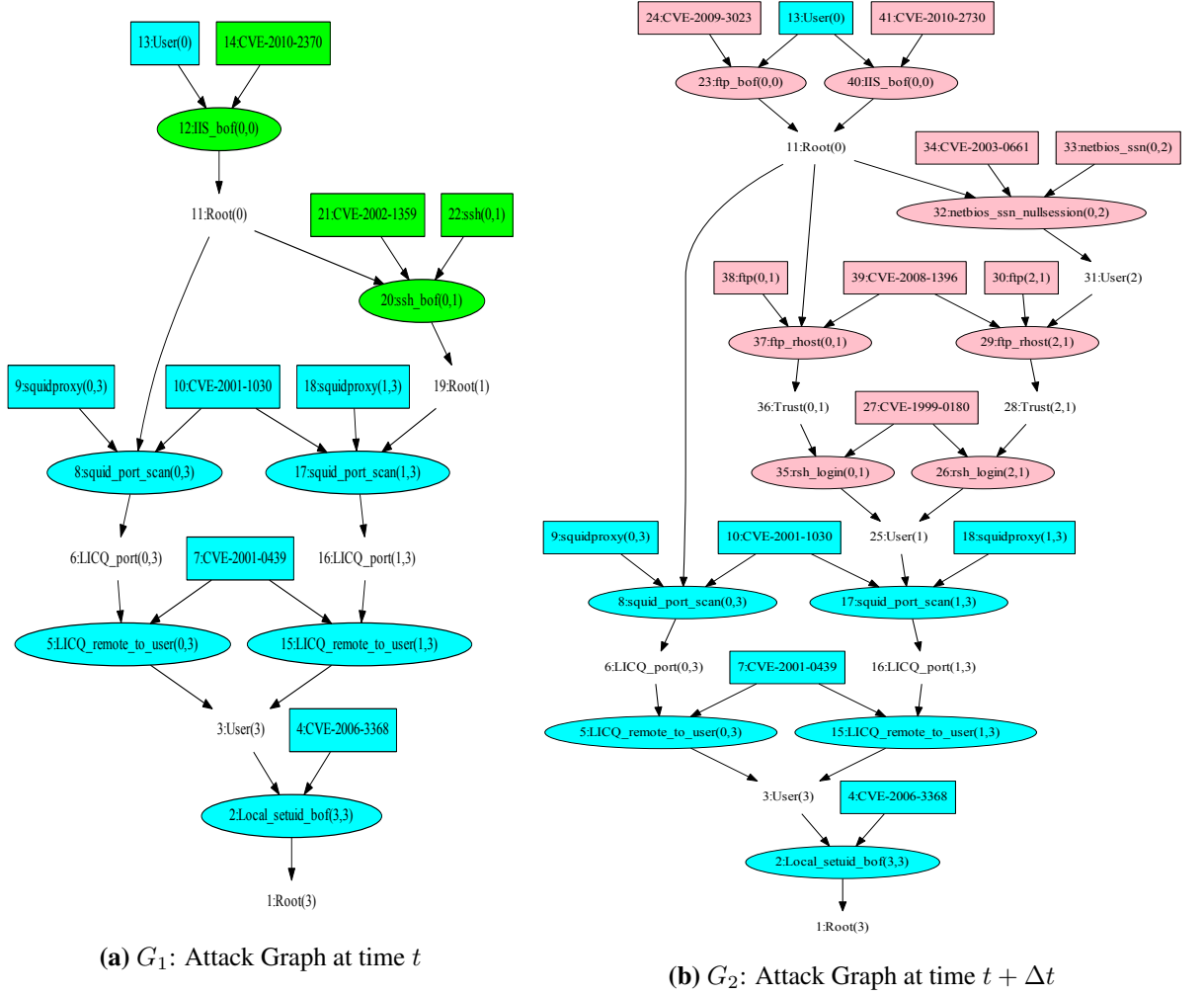


Figure 6.2: Differential visualization of attack graphs at the node level. Colored pink nodes represent newly introduced changes over Δt as appeared only in G_2 . Colored cyan nodes (in both the attack graphs G_1 and G_2) represent the portion of the attack surface which is constant over Δt . Colored green nodes appeared only in G_1 and represented the exploits and initial conditions which are removed due to the application of network hardening procedures.

6.4 Formal Model for the Network Attack Surface (NAS)

Wang et al. [30] were the first who applied the notion of attack surface to the entire computer network. Their focus is on interfaces like remotely exploitable services. According to Cybenko et al. [71], NAS consists of vulnerable network configurations and existing exploitable vulnerabilities. Our work borrows the idea of network attack surface proposed by Cybenko et al. [71].

Mostly, vulnerable service connectivities and existing well-known vulnerabilities act as the prerequisite for the incremental exploitation of remote vulnerabilities. According to [141], service connectivity is one of the critical network characteristics which is imperative to the measurement of the network security risk. Therefore, informally, vulnerable software/services and service connectivities between vulnerable hosts are the system resources used by an adversary against the network itself. In practice, not all service connectivities, and vulnerabilities contribute equally during the network intrusion, and hence we decide to focus on only those likely to be used by an adversary. Similar to [71], our notion of the network attack surface (NAS) comprises a subset of network resources that are likely to be utilized by an adversary. We have used the same notion of the NAS formalized in Chapter 5.

6.4.1 Model to Capture the Network Attack Surface

Essentially, logical attack graph depicts exploits and their respective enabling conditions such as vulnerable service connectivities, the presence of remotely exploitable vulnerabilities, etc. A directed edge from the security condition to an exploit represent *require edge*, whereas an edge from the exploit to a security condition indicates a *imply edge*. So, the notion of attack surface for an entire computer network is ideally captured by the attack graph, particularly in terms of *require edges*.

We represent the attack surface of network configuration, i.e. $NAS(Conf(\mathcal{N}))$, by a logical attack graph $G(\mathcal{E} \cup \mathcal{J}, R_r \cup R_i)$ defined by Wang et al. [56]. Here, \mathcal{E} is the set of exploits and \mathcal{J} is the set of conditions in the logical attack graph. Therefore, we can assess the variation in the attack surface of a given computer network by computing the two successive attack graphs adjacent over time. In doing so, we can pinpoint important changes and dynamics, which are responsible for the change in the NAS .

6.4.2 Dealing with Variations in the NAS

In practice, change in network configuration is due to the enumeration of various actions taken by an administrator as a part of network maintenance and security hardening procedure. A partial list of such activities/events is compiled as follows:

- Installation/removal of network devices, software, and services, etc.
- Change in access control policies
- Misconfiguration of software and hardware devices
- Vulnerability patching
- Disclosure of new vulnerabilities

Let M be the enumeration of all the above events. After applying the action set M , the initial configuration $Conf(N)$ changes. We represent the resulting configuration as $Conf(N) \oplus M$ and resulting network attack surface as $NAS(Conf(N \oplus M))$. Such change in NAS is successfully captured by an attack graph generated at time $t + \Delta t$.

The occurrence of any of the above events can lead to significant change in the network attack surface. The attack graph is the ideal way to represent NAS . Changes in NAS can be incremental or decremental and hence the size of an attack graph generated at time $t + \Delta t$. The enumerated list of possible changes that can happen in the attack graph at time t (because of above-stated events) is as follows:

- Addition of a new exploits to the attack graph and hence corresponding pre-conditions and implied post-condition. Security events such as disclosure of new vulnerabilities, installation of vulnerable software/services, enabling of vulnerable service connectivities, etc. lead to the increase in network attack surface.
- Deletion of some of the exploit nodes and their respective pre and post-conditions. Events like vulnerability patching, change in access control policies, disabling or removal of software/services lead to the invalidation of the enabling pre-conditions of exploits.
- Since patching vulnerability in a software/service may introduce new vulnerabilities; one exploit node could be replaced by one or more exploits with the same or different post-condition.

Note that, during the sampling interval Δt , some of the vulnerabilities are stable. The root causes of such vulnerabilities are already known to the administrator.

6.5 Assessing Change in the Network Attack Surface

In this section, we present a method to assess, and pinpoint changes in the network attack surface based on error-correcting graph matching (ECGM), which has been studied extensively in pattern recognition [54], [55], [175], [176], [177]. Since the attack graphs successfully capture the network attack surface (as shown in Section 6.4), the difference between two successive attack graphs is indeed representative of the change in the attack surface.

We are particularly interested in three problems. First, how much change has occurred in the attack surface over Δt as a result of network dynamics? Second, what are the newly introduced changes in the attack surface? Third, what are the root causes? The first can be answered by applying the concept of error-correcting graph matching (ECGM) [54], [175] for a pair of successive attack graphs $\langle G_i, G_j \rangle$ generated over a sampling interval Δt . Whereas, the latter two can be answered by the application of change distribution matrix (CDM).

To facilitate the later discussion, we give a brief overview of ECGM. Supplementary details can be found in the rich literature on ECGM in [54], [55], [175], [176], [177].

6.5.1 Error-correcting Graph Matching (ECGM)

Prior to the discussion of the concept of ECGM, an attack graph representation of the NAS must be defined. Only the issues associated with dynamic changes in the NAS are considered here.

Definition 4. A labelled attack graph G is a three-tuple $G = (V, E, \rho)$, where

- V is a finite set of vertices, i.e. $V = e \cup c$
- $E \subseteq V \times V$ is a set of Edges, i.e. $E = R_r \cup R_i$
- $\rho : V \rightarrow L_V$ is a function assigning labels to the vertices

In this definition L_V is the set of symbolic labels uniquely identifying each node in the attack graph G .

Two graphs believed to be same if there exists a graph isomorphism between them [162]. In addition to the detection of the graph/subgraph isomorphism, in order to cope with distortion in the input graphs a concept of ECGM has been proposed in the

literature. Similarity measures such as- maximum common subgraph [54] and graph edit distance [55], etc. also proposed to deal with ECGM. Showbridge et al. [143] used ECGM to monitor the performance of telecommunication networks. Whereas, Ning et al. [163] applied error-correcting graph/subgraph isomorphism for learning attack strategies. We focus on graph edit distance to study the application of ECGM in detecting the change in the network attack surface of dynamic networks.

In general, ECGM measures the difference between the two input graphs by determining the least cost sequence of edit operations required to transform one graph into the other. Such graph edit distance based method considers a set of edit operations like- insertion, deletion, substitution of nodes and edges. Each of the above edit operations has an associated cost that is defined by the edit cost function. Therefore, the graph edit distance determines the similarity/distance between the two graphs in terms of the least cost sequence of the edit operations that convert one graph into the other.

To successfully use ECGM for determining the change in the network attack surface, we need to answer the question: what are the possible edit operations on an attack graph. Before deciding on the possible edit operations over the attack graph, we need to understand the attack graph construct/parameters that are sensitive to the changes in the network attack surface.

As our goal is to detect changes in the network attack surface in terms of newly introduced vulnerabilities and associated initial conditions, we only consider an edge deletion operation to fulfill our purpose. In reality, initial conditions (which are not postcondition of any of the exploit) are present in the network from the beginning and are under the control of security analysts. An example of such initial conditions includes vulnerable service connectivity, a vulnerability in software/services, user/service privileges, etc. The analyst can disable one (or more) initial conditions to prevent an exploit from execution. In other words, an analyst has to remove *require edge* between initial conditions and an exploit. Since our goal is the detection of a change in the attack surface and then reduction through network hardening, we only consider the edit operation in terms of deletion of *require edges*. The algorithm 6.1 detects such newly introduced *require edges* in the attack graph G_2 generated at time $t + \Delta t$.

Definition 5. Given the attack graph $G_2 = (V_2, E_2, \rho)$ generated at time $t + \Delta t$, define an edit operation δ on G_2 as follows:

Algorithm 6.1 *findEdges*: find newly introduced *require edges* in an attack graph G_2

Input:

$\langle G_1, G_2 \rangle \rightarrow$ a pair of goal-oriented logical attack graphs generated over the sampling interval Δt

Output:

$NewInit \subset V_2 \setminus V_1 \rightarrow$ set of newly introduced initial conditions

$NewRequireEdge \subset E_2 \setminus E_1 \rightarrow$ set of newly introduced require edges (i.e. the edges between initial conditions and exploits).

```

1:  $\langle V_1, E_1 \rangle \leftarrow G_1$ 
2:  $\langle V_2, E_2 \rangle \leftarrow G_2$ 
3:  $G_3 \langle V_3, E_3 \rangle \leftarrow G_2 \setminus G_1$ 
4: if  $(E_2 \setminus E_1 = \phi)$  then
5:   Print: “no new vulnerability introduced into the network over  $\Delta t$ ”
6: else
7:   for all  $u \in V_3$  do
8:     if  $\left( (indegree(u) = 0) \wedge (outdegree(u) \geq 1) \right)$  then
9:        $NewInit \leftarrow u$ 
10:    end if
11:  end for
12:  for all  $u \in NewInit$  do
13:    if  $(u = i)$  for one or more  $(i, j) \in E_3$  then
14:       $NewRequireEdge \leftarrow (i, j)$ 
15:    end if
16:  end for
17: end if

```

- $(e \rightarrow \$)$: deleting the newly introduced require edge e from G_2 . This represents removing an edge (dependency) between the initial condition and an exploit.

Here ρ is a function assigning unique labels to the vertices.

From the hardening perspective, analysts are interested only in newly introduced vulnerabilities and configuration changes that lead to the increase in attack surface. Therefore, unlike [143], we have applied the ECGM procedure to the attack graph pair $\langle mcs(G_1, G_2), G_2 \rangle$. Here $mcs(G_1, G_2)$ represents the portion of the attack graph G_1

(or attack surface) which is invariant over time Δt . The persistence of $mcs(G_1, G_2)$ over Δt is due to the constraints like- unavailability of vulnerability patches, limited hardening budget, loss of service availability due to service interruption during patch time, etc.

In practice, some of the vulnerabilities in the network at time t (i.e. in G_1) qualifies for removal/patching. While rest of the vulnerabilities cannot be patched immediately due to various constraints and hence the decision of fixing such vulnerabilities is hindered. All (or few) of such unpatched vulnerabilities in G_1 will appear straight away in G_2 at time $t + \Delta t$. We assume, the causes of such unpatched vulnerabilities are already known to the analysts. Therefore, they are interested only in newly introduced vulnerabilities or configuration changes over Δt that lead to the increase in attack surface. So, the analyst's goal is to identify all the newly introduced vulnerabilities, respective enabling conditions, and then patch/fix of them so that graph G_2 will reduce to $mcs(G_1, G_2)$. In case, if patches for the vulnerabilities, which are steady over time Δt are available, then the analyst can patch them also.

Definition 6. Given an attack graph $G_2 = (V_2, E_2, \rho)$ at time $t + \Delta t$ and a sequence of edit operation $\Delta = (\delta_1, \delta_2, \dots, \delta_k), k \geq 1$, the edited attack graph $\Delta(G)$ becomes $\Delta(G) = \delta_k(\dots \delta_2(\delta_1(G_2)) \dots) = mcs(G_1, G_2) = (V_\delta, E_\delta, \rho)$ where:

- $V_\delta \subseteq V_2$
- $E_\delta \subseteq E_2$
- $\rho_\delta(x) = \rho(x), \forall x \in V_\delta$

Note that the removal of each newly introduced initial condition (i.e. removal of require edge δ_i) is assigned a cost $C(\delta_i)$, then the total cost associated with the sequence of edit operations Δ is

$$C(\Delta) = \sum_{i=1}^k C(\delta_i) \quad (6.1)$$

Definition 7. Given two attack graphs $G_1 = (V_1, E_1, \rho)$, $G_2 = (V_2, E_2, \rho)$, their maximum common subgraph $mcs(G_1, G_2)$ and a sequence of edit operations Δ on G_2 such that G_2 can be transformed into $mcs(G_1, G_2)$, the distance $d(mcs(G_1, G_2), G_2)$ between G_2 and $mcs(G_1, G_2)$ is the sum of edit costs associated with the edit sequence Δ , i.e. $d(mcs(G_1, G_2), G_2) = C(\Delta)$

In practice, an exploit can be prevented from execution by disabling any one of the initial enabling conditions. But, here we consider the removal of all the *require edges* of all the newly introduced exploits in G_2 . It is because our goal is to assess the degree of change in the attack surface and pinpoint the root causes as well.

If the cost associated with the removal of each of the newly introduced *require edge* is 1, then the edit sequence cost becomes the difference between the total number of *require edges* in attack graph G_2 and all the *require edges* that are in common to both G_1 and G_2 . More formally:

Definition 8. *Let the attack graph for the enterprise network operating at time t is $G_1 = (V_1, E_1, \rho)$, and let $G_2 = (V_2, E_2, \rho)$ be the attack graph for the same network at time t' , where $t' = t + \Delta t$. The distance $d(mcs(G_1, G_2), G_2)$ is given by:*

$$d(mcs(G_1, G_2), G_2) = |\text{require edges}(G_2)| - |\text{require edges}(mcs(G_1, G_2))| \quad (6.2)$$

where the cost function for edit operation δ is

$$C(\delta) = \begin{cases} 1 & ; \delta = (e \rightarrow \$) \\ 0 & ; \text{otherwise} \end{cases}$$

Clearly, the edit distance d , as a measure of change in network attack surface, increases with increasing degree of change experienced by the network over time Δt . The edit distance d is bounded below by $d = 0$ when there is no change in the attack surface, and above by $d = |\text{require edges}(G_2)|$, when the network attack surface is completely different.

6.5.2 Pinpointing the Root Causes

Since each network event (discussed in Section 6.4.2) can cause changes in the attack surface, our next objective is to pinpoint such events. For the attack graph pair $\langle G_1, G_2 \rangle$ shown in Figure 6.2, the edit distance d has indicated significantly different attack surface. As discussed in Section 6.5.1, the portion of the attack graph (or attack surface), i.e. $mcs(G_1, G_2)$ is invariant over Δt . Of special interest is the distribution of change in the network attack surface during the transition from G_1 to G_2 . For easier illustration, the attack graph pair $\langle G_1, G_2 \rangle$ obtained for the example network is shown in Figure 6.3.

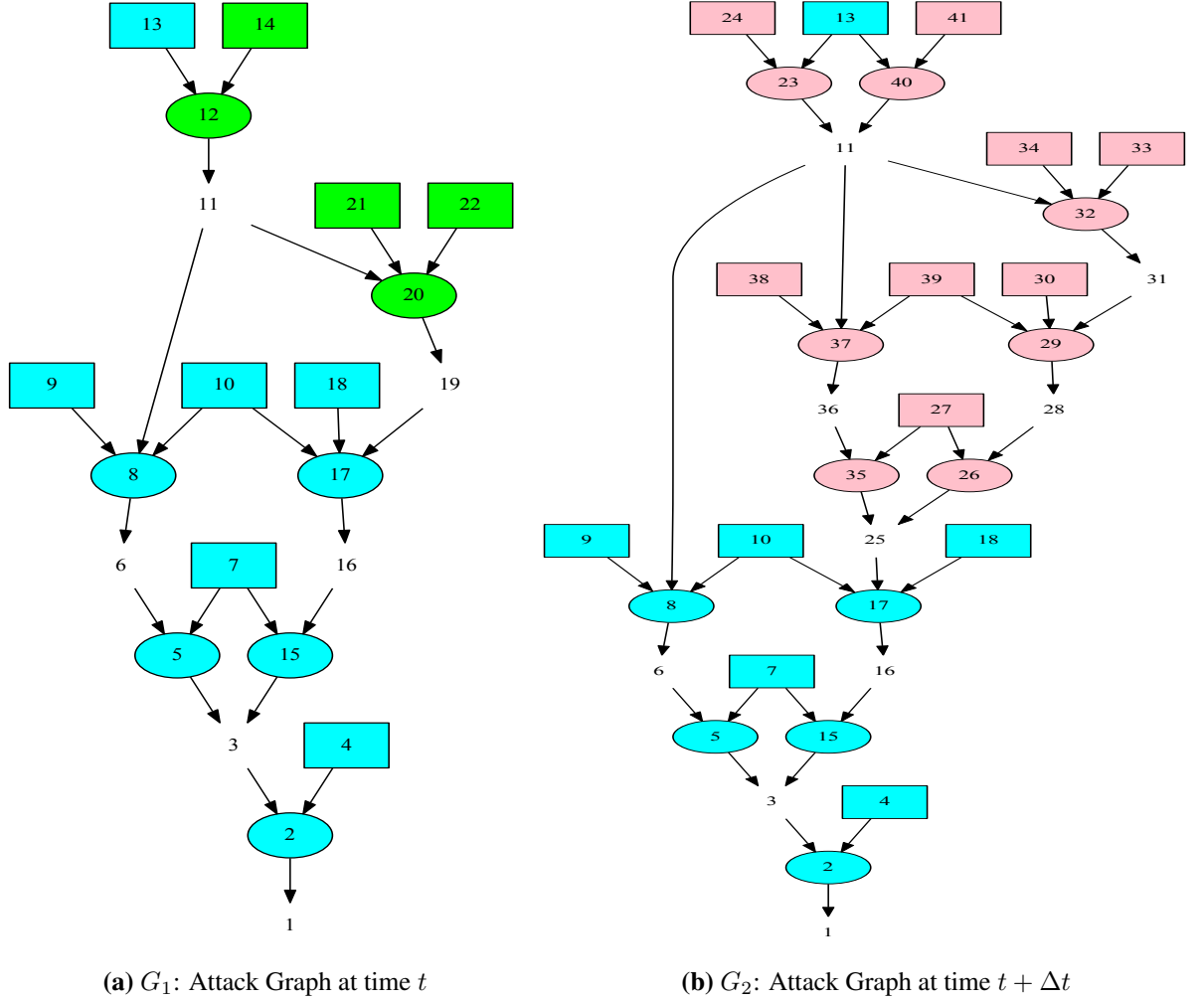


Figure 6.3: Simplified attack graphs for the test network at time t and $t + \Delta t$.

In order to portray the newly introduced changes in G_2 over Δt , we present a change distribution matrix C , as shown in Figure 6.4. It shows newly introduced changes in the network attack surface, which is not so obvious even with the effective attack graph visualization. The rows and columns of C represent initial conditions and exploits in the attack graph G_2 , respectively. The existence of *require edge* deleted from G_2 is represented in the matrix $C = [c_{ij}]$ by the corresponding row column entry $c_{ij} = 1$. Here, i and j are the particular initial condition and exploit of the deleted *require edge*. C with non-zero entry $c_{ij} = (-1)$ represents the set of *require edges* that are stable over time Δt . Whereas, an entry $c_{ij} = 0$ record edges that are not present in

6.5 Assessing Change in the Network Attack Surface

G_2 . Essentially, C is a sparse matrix because of the unidirectional edges in the logical attack graph.

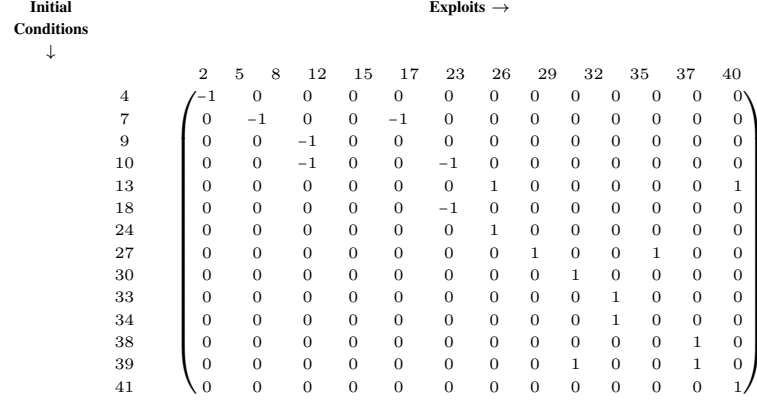


Figure 6.4: Change distribution matrix C

Definition 9. The change distribution matrix $C = [c_{ij}]$ is $|V_p| \times |V_q|$ matrix that describes the variations in the network attack surface in terms of newly introduced exploits and associated initial conditions, where

- $V_p, V_q \subset V_2$ and

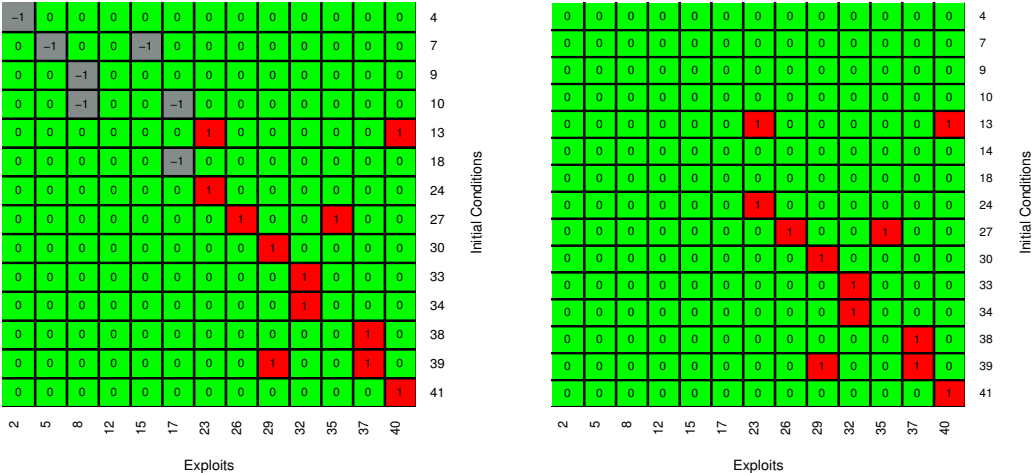
$$c_{ij} = \begin{cases} 1 & ; e(i, j) \in R_{r'}, R_{r'} \subset R_r \subset E_2 \\ -1 & ; e(i, j) \in R_{r''}, R_{r''} \subset R_r \subset E_2 \\ 0 & ; \text{otherwise} \end{cases}$$

In this definition, the matrix $C = [c_{ij}]_{p \times q}$ is a set of p initial conditions and q exploits arranged in a rectangular array of p rows and q columns. $R_{r'}$ is the set of newly introduced *require edges* between initial conditions and exploits in an attack graph G_2 . $R_{r'}$ does not constitute *require edges* which starts from the postcondition. Whereas, $R_{r''}$ represents the set of *require edges* that are stable over Δt .

The heat map corresponding to the matrix C is shown in Figure 6.6(a). It provides a quick overview of the security configuration of the managed network. In other words, it provides a concise summary of the impact of network configuration changes on the attack surface. As shown in the heat map (Figure 6.6(a)), each newly introduced change is represented by a single matrix element, as opposed to a drawn line in an attack graph G_2 (Figure 6.3b). Graph vertices for, e.g. initial conditions and exploits are

[illegible]

Figure 6.5: Reduced change distribution matrix C_F



a: Unclustered change distribution matrix C

b: Unclustered change distribution matrix C_R

Figure 6.6: Change distribution matrices: Original (C) and Reduced (C_R)

implicitly represented as matrix rows and columns, respectively. Since security analyst is interested only in newly introduced changes, we reduced the original matrix C to the matrix C_R as shown in Figure 6.5. Such reduced matrix C_R and corresponding heat map in Figure 6.6 (b) shows the succinct representation of the newly introduced

changes in the network attack surface.

The rows and columns of matrix C_R could be placed in any order, without affecting the “require relationship” between initial conditions and exploits. But the ordering that captures similarly connected attack graph elements is clearly desirable. In particular, we seek ordering that tends to cluster non-zero matrix elements. It will allow us to treat such groups of similarly connected elements as a single unit as we analyze matrix C_R . We, therefore, proposed to apply Euclidean distance-based clustering technique that reorders the CDM so that the block of similarly connected attack graph elements emerge.

Clustering of rows and columns reveals security relevant information, making important features apparent for network hardening. Clustering re-arranges rows and columns of matrix C_R to form homogeneous groups. In this way, patterns of similar “require relationship” in G_2 are clear, and groups can be considered as a single unit. For the clustered matrix, we retain the ordering induced by clustering, so that patterns in the attack graph structure are still apparent. The employed clustering technique is fully automatic, is free of the parameter, and having quadratic time complexity in the size of change distribution matrix C_R .

6.5.3 Observations

Careful analysis of the clustered change distribution matrix C_R (Figure 6.7) reveals the newly introduced changes in the network attack surface. Security analysts make system hardening decisions based on the top initial conditions that contribute most to the attack surface. This Section discusses the results obtained from the test network shown in Figure 6.1. To extract security relevant information (in the form of newest and largest problems), we have performed Euclidean distance based clustering operations over C_R as follows:

1. Column-wise Clustering

Figure 6.7(b) shows the matrix C_R after column clustering. The essence of such clustering is that it provides the knowledge of the initial conditions that enables two or more exploits. Initial conditions for, e.g. 27, and 39 are responsible for the successful execution of two exploits each. If any one of the two vulnerabilities, i.e. *CVE* – 1999 – 0180 and *CVE* – 2008 – 1396 is patched, then the majority of the attack paths can be removed and hence attack opportunities available to

6.5 Assessing Change in the Network Attack Surface

an adversary. Knowledge of such initial conditions that covers a large number of exploits can be used for efficient network hardening in a resource constrained environment.

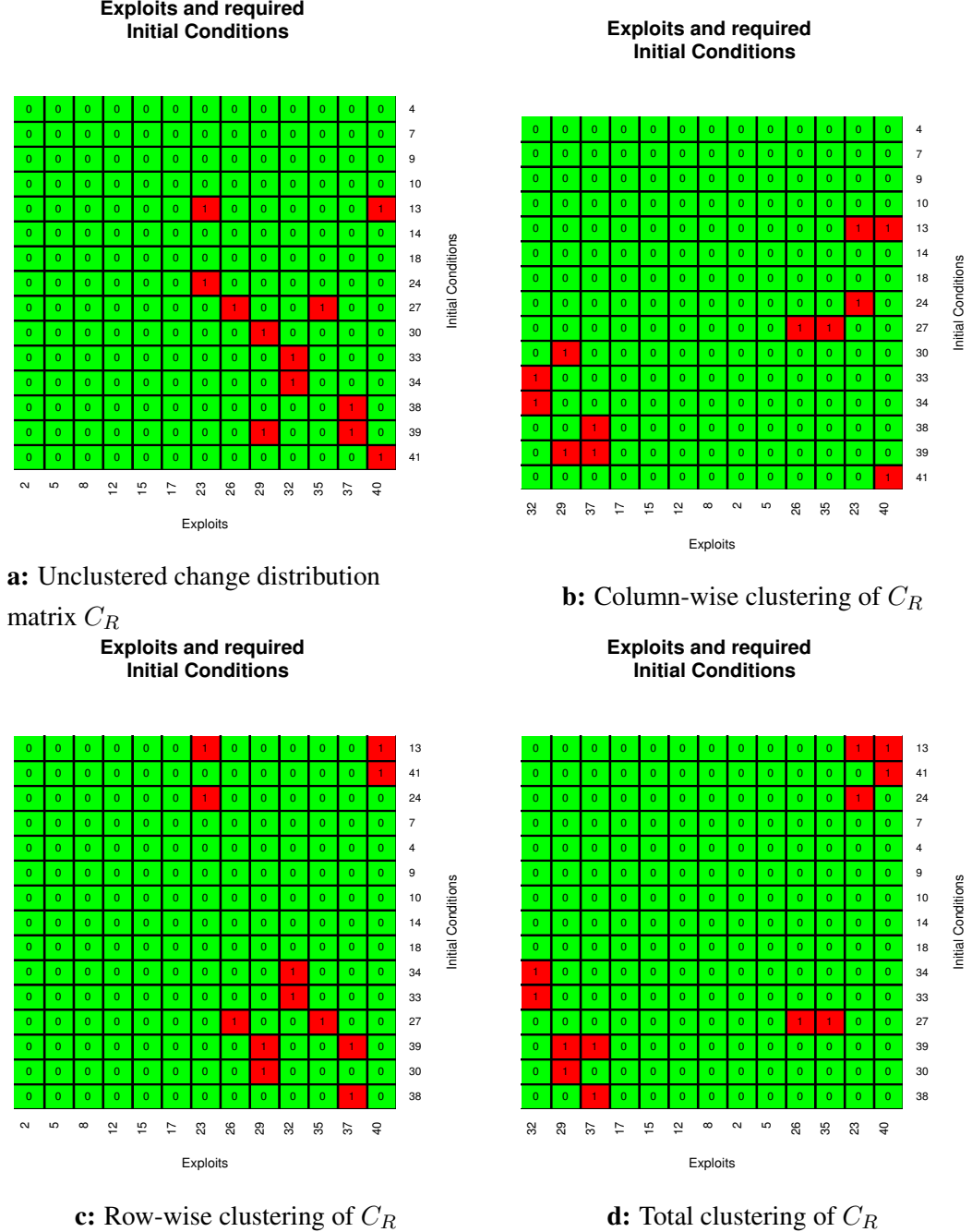


Figure 6.7: Clustering operations over change distribution matrix C_R

2. Row-wise Clustering

Figure 6.7(c) shows the matrix C_R after row clustering. The goal is to distinguish between stable and newly introduced exploits. From Figure 6.7(c), it is clear that exploit 26, and 35 are stable over time. In attack graph G_1 , these exploits are dormant (i.e. even though vulnerabilities were there, they were not exploitable due to the unavailability of one or more initial condition). Due to the change in the network configuration and the introduction of new vulnerabilities over Δt , these exploits are activated and become the part of the attack graph G_2 . Stable exploits can be easily identified as there is only one *require edge* for them. Whereas, as shown in Figure 6.7(c), exploits 23, 29, 32, 37, and 40 are newly introduced exploits. Proactive detection of such newly introduced exploits alerts, security assessor about the possible intrusion.

3. Total Clustering

Finally, we have performed a total clustering over C_R as shown in Figure 6.7(d). It can be conducted row-wise clustering followed by a column-wise or vice versa. The intuition behind such clustering is that it provides the knowledge about the portion of the attack surface that suffered maximum change. It shows top k nodes both initial conditions and exploits that have maximum impact on the attack surface. In this way, we can cluster the nodes, i.e. initial conditions and exploits of attack graph G_2 for recognizing nodes that suffered maximum change separately to those nodes that are relatively stable over Δt . As the vulnerability remediation is resource and time consuming, it is crucial to identify the newest and largest problems and remediate those first [178].

We have shown how our proposed change detection technique can succinctly summarize major changes in the network attack surface resulting from changes in the network (or host) configuration. In other words, our change detection technique helps security analysts in doing what-if analysis of planned changes and impact of real-time changes in the network configuration. Using CDM-based technique, the network topology factors and security factors which have maximum impact on the network attack surface can be detected and prioritized for efficient network hardening.

Our implementation of change distribution matrix uses Matlab sparse matrices for internal computations and visualization as well. Essentially, the computational complexity of operations over the sparse matrices is proportional to the number of nonzero

elements in the matrix (in our case number of newly introduced *require edges*). Our change distribution technique is fully automatic, uses quadratic memory in the order of vertices to store the C_R matrix. As an alternative to the CDM, one can use the adjacency list. The adjacency list keeps all the newly introduced required edges and use no space to record the edges that are not present in G_2 . Therefore, there is a trade-off (in both space and time) between CDM and adjacency list, depending on the graph sparseness and the required clustering operations [147].

The rows and columns of CDM could be placed in any order. Note that the vertices, i.e. initial conditions and exploits must also need to be ordered using the same permutation performed in C_R in order to correlate the correct vertices in G_2 with those indexed by the row and column vertex identifiers used in matrix C_R . Since rows and columns in C_R are likely to be ordered differently, it complicates the cross-referencing of the row, column indices in matrix C_R , to the corresponding vertices in the attack graph G_2 . The optimal node labeling mechanism for the generated attack graphs can reduce the cost of matrix permutation while ordering initial conditions and exploits for making change apparent.

6.6 Experiments

In this Section, we present some numerical results for experimental verification. We performed a series of experiments to study the ECGM-based change detection technique. Our primary objective is to analyze the sensitivity of the proposed ECGM-based metric, i.e. d to the variations in the network attack surface. Further, our goal is to analyze the performance and feasibility of our change detection technique by simulation results. Measured time does not include the attack graph construction time and also the time required to derive the labeled representation of the logical attack graphs.

In order to illustrate the use of ECGM for measuring the change in the NAS , we have generated sample directed graphs and subjected it to random variations to show an increase in graph edit distance d (Definition 8) with increasing degree of change in the network. We have randomly generated arbitrary directed graph G containing 100 edges where the average edge density (or degree of connectivity) is not more than 3. In practice, the maximum edges incident on each node type in a logical attack graphs are bounded by a constant k as shown in Table 5.7. Graphs with average edge density 3 were generated to imitate the characteristics of the attack graphs.

We model the dynamic behavior of digraph G by randomly inserting and deleting vertices to form the modified graph G' , subject to an input probability that vertices should be changed. Figure 6.8 show the effect of a change in the graph topology on the edit distance d . Here the input probability of change is the independent variable being manipulated during the experiment, and the graph edit distance is the dependent variable being recorded. From Figure 6.8, it is clearly visible that there is an increase in the edit distance with an increase in the change in the graph topology from G to G' . Therefore, we state that the proposed ECGM based edit distance (d) is sensitive to the structural differences between a pair of input attack graphs and hence to the change in the network attack surface. Each experiment was conducted several times with 99% confidence intervals plotted using error bars.

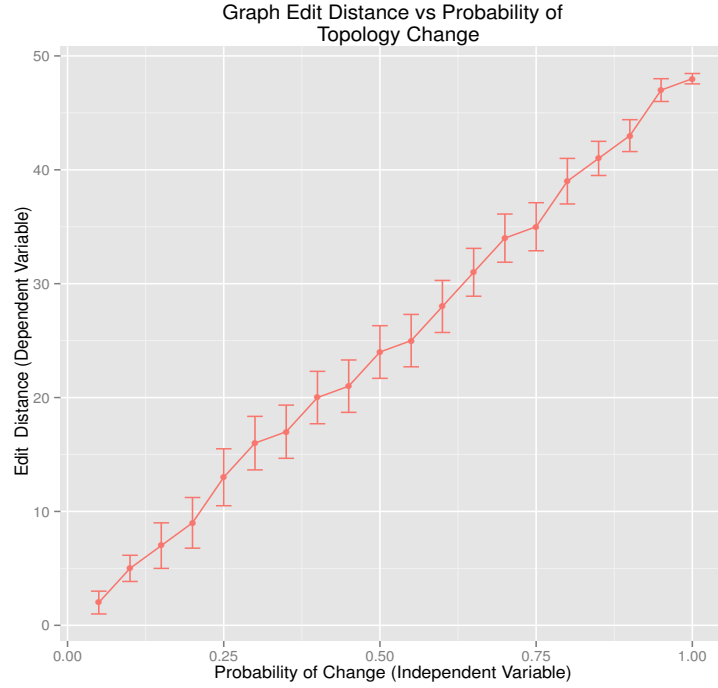


Figure 6.8: Graph edit distance vs. probability of graph topology change

Next, we discuss the computation time required for the ECGM-based technique; we used in our study.

Because of the unavailability of benchmark data sets in the attack graph domain, we used synthetic data sets in order to verify the computational complexity of the proposed ECGM based technique. The test is dedicated to verifying whether the results obtained

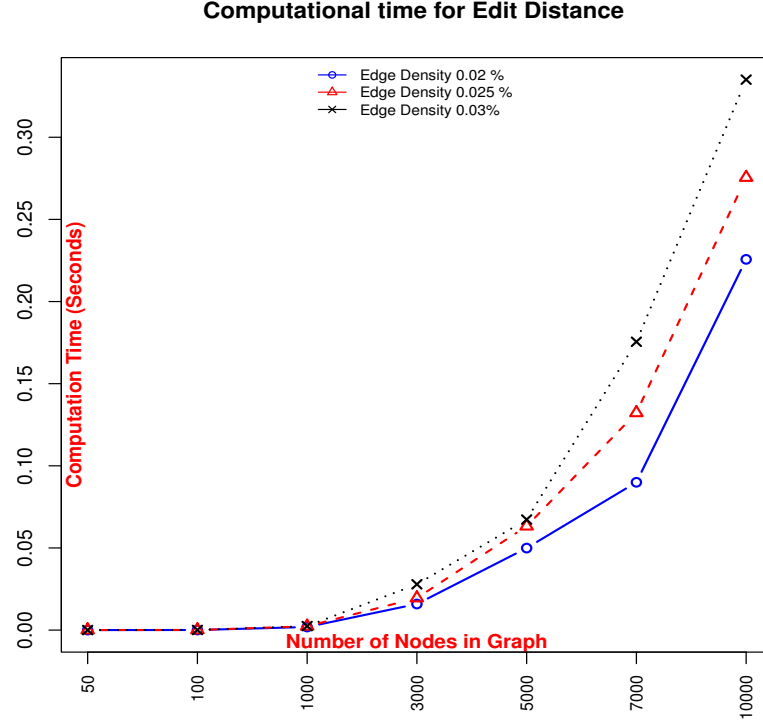


Figure 6.9: Computation time in seconds for ECGM-based edit distance under different edge densities

for a small hypothetical network can be generalized to large enterprise networks. We have generated two synthetic data sets which constitute normally-distributed random edges with average edge density of 0.02%. In particular, we have used Fan Chung algorithm [170] to create such data sets. For each data set having an average edge density 0.02%, we obtain a graph series (\mathcal{S}), where each graph in a series contains 50, 100, 1000, 3000, 5000, 7000, and 10,000 nodes. One more series \mathcal{S}' is created as a counterpart subjected to random variations (with input probability of change: 0.05%) to show an increase in the edit distance d , using the same procedure as above, for measurements of ECGM computation times. The experiment is repeated for different values of average edge density, such as 0.025% and 0.03%.

For measuring the computational time taken by ECGM based edit distance, we need both the graph series \mathcal{S} and \mathcal{S}' . We have selected the first graph G_1 from \mathcal{S} , which consists of 50 unique nodes. The graph G'_1 from \mathcal{S}' with equal number of nodes (as in G_1) but few with different label representation is chosen. These two graphs G_1 and G'_1 are evaluated for edit distance. The above procedure is repeated for the other

graphs in a series \mathcal{S} . As depicted in Figure 6.9, the effect of larger edge densities on the computation time of edit distance is clearly visible. It shows higher computational times required for graphs with higher edge densities.

6.7 Summary

In this Chapter, we have presented a differential attack graph-based change detection technique to assess changes in the attack surface of dynamic computer networks. An error-correcting graph matching (ECGM) based similarity measure is employed to identify the degree of change in the attack surface. The difference between two successive attack graphs helps in finding the root causes responsible for the change. Further, we have introduced a change distribution matrix CDM based technique in the context of vulnerability-centric attack graphs, so that the portion (or region) of the attack graph that suffered change can be inferred. Such CDM based technique can help make attack graph more understandable and useful. We have explored the viability and efficacy of our proposed method and showed that our technique is capable of assessing change in the attack surface at the level of initial conditions and hence can be used in practice for network hardening.

Chapter 7

Conclusions and Future Work

The aim of this thesis has been to provide the attack graph-based methods and metrics for network security hardening. The work in this thesis is partitioned into three topics, namely, risk assessment of network vulnerabilities, software/services diversification to address the threats of zero-day attack, and the change assessment in the attack surface of dynamic networks.

In the preceding Chapters, we have proposed attack graph-based methods and metrics for:

1. Estimating the security risk posed by each exploitable vulnerability in a given network,
2. Determining the diversification level of each attack path in a resource graph (zero-day attack graph) and thereby detect and diversify the repeated services along the attack paths for making network more robust against the zero-day attacks,
3. Assessing temporal change in the attack surface of dynamic network, and
4. Identify the newly introduced changes in the network attack surface that are not so obvious even with the effective attack graph visualization.

Here, we summarize our main contributions and discuss interesting directions of future research.

7.1 Summary of Contributions

- The third Chapter has been concerned with a systematic risk assessment of exploitable vulnerabilities in a network aiming to identify the top vulnerabilities to prioritize vulnerability remediation activities or countermeasures. The presented proximity-based vulnerability risk assessment approach builds on an exploit-dependency attack graph [56] taking into account various network risk conditions that affect the success of an adversary. The most distinctive feature of this approach is that it considers the repetition of exploits/vulnerabilities along the attack paths and the risk of the neighboring vulnerabilities while assessing the security risks posed by the vulnerabilities. Based on the proposed comprehensive measure called IRCR, an administrator can accurately determine top vulnerabilities and prioritize vulnerability remediation activities accordingly. Furthermore, the efficacy and applicability of IRCR is validated through case studies and it is observed that the IRCR is complementary with state-of-the-art vulnerability risk scoring techniques (for, e.g. P [26], R [27]) on synthetic networks.
- In the fourth Chapter, the utilization of opportunistic diversity [161] existing among the software systems for increasing the network robustness against zero-day attacks has been considered. In particular, a technique for quantifying the diversification level of each attack path in a resource graph (zero-day attack graph) is proposed. Further, we have proposed an algorithm for the identification and diversification of the repeated services along the attack paths to increase the system robustness against the zero-day attacks. In order to analyze the robustness of networks against zero-day attacks, each installed product (i.e. software/services/resources) is assumed to have at least one zero-day vulnerability. A resource graph [30], generated for a given network, is utilized as a network security model. The intra-path and inter-path (i.e. uniqueness (u) and overlap (o)) diversity measurement metrics are proposed to assess the current diversification level of each attack path in a resource graph.
- The fifth Chapter has been concerned with an assessment of the temporal variation in the attack surface of dynamic networks. In particular, the utilization of classical graph distance metrics based on the Maximum Common Subgraph (MCS), and Graph Edit Distance (GED) for quantifying the distance between a

pair of successive attack graphs has been considered. We have used a *logical attack graph* [16], [21] as a security model to capture the attack surface of the underlying network. Since the attack graphs are capable of successfully capturing the network attack surface, the distance between a pair of successive attack graphs (generated over the observed sampling interval) indicates the change in the network attack surface. We found that the MCS and GED based graph distance metrics are competitive with state-of-the-art attack graph-based metrics on heterogeneous network models. The MCS and GED based graph distance metrics successfully capture the temporal variation in the attack surface and also generate an alert about the security events which are responsible for such change.

- In the sixth Chapter, a technique for identifying the newly introduced changes in the attack surface of dynamic networks is proposed. In particular, we have developed a change distribution matrix (CDM) based method to discern the newly introduced changes in the network attack surface. For doing this, we have reduced the problem of change detection in the network attack surface to the error-correcting graph matching (ECGM) problem. We found that the CDM based technique identifies the root causes responsible for variations in the network attack surface in time efficient manner. Further, it identifies the newly introduced exploits and their respective enabling conditions in the dynamic systems and discerns portion of the attack graph that suffered significant change. Such identification of modification in the network attack surface and hidden root causes in a time-efficient manner is crucial to the prevention of future attacks.

7.2 Future Directions

- There are many directions for extending the work presented in Chapter 3. First of all, we propose to investigate the reduction in attacker's work factor (vulnerability resistance) because of the repetition of vulnerabilities along the attack paths. To the best of our knowledge, there is no study on how much reduction in attacker's work factor happens when the attacker exploits the same vulnerability second time. Such decrease in the work factor/attackers effort could be different for different classes of vulnerabilities. If it is true, then the reduction in attackers effort is a function of vulnerability types and their repetitiveness. Further,

we want to improve the IRCR metric itself, and use it for investigating specific aspects of network security.

- In Chapter 4, we have proposed a technique for software/services diversification along the attack paths to eliminate the problem caused by misplaced diversity. Proposed method assumed that an enterprise has enough resources to sustain adequate diversification. We further assumed that configuration space is adequate/sufficient, i.e. the number of functionally equivalent software alternatives are already available in sufficient numbers or quantities. Since all operational networks have constraints and resource limitations, our future work consists of cost-controlled network diversification.
- Additional work is needed to enrich our knowledge about the semantics of variations in the attack graphs. As MCS and GED based graph distance metrics are oblivious (unaware) of the significant *AND* semantic (structural property) between the initial conditions in the attack graph; appropriate metrics need to be developed by consideration of the *AND* semantic. It can be done through the generation of the Boolean expression in the form of Disjunctive Normal Form (*DNF*) for each attack graph in a graph pair $\langle G_i, G_j \rangle$ and then comparing those *DNF*'s for their similarity or dissimilarity. As an immediate future work, we propose to find the edit distance (in terms of network hardening cost) between a pair of successive attack graphs in order to minimize the network attack surface. The problem of finding minimum edit distance for efficient network hardening can be formulated as an optimization problem.

References

- [1] MICHAEL LYLE ARTZ. **NetSPA: a Network Security Planning Architecture**. PhD thesis, Massachusetts Institute of Technology (MIT), USA, 2002.
- [2] FIRST. **The Forum of Incident Response and Security Teams, Common Vulnerability Scoring System v3.0: Specification Document**. June, 2015.
- [3] CLARK WEISSMAN. **Handbook for the Computer Security Certification of Trusted Systems, Chapter 10: Security Penetration Testing Guidelines**. Technical report, Center for Secure Information Technology, Naval Research Laboratory (NRL), USA, October 1993.
- [4] RICHARD P. LIPPMANN AND KYLE W. INGOLS. **An Annotated Review of Past Papers on Attack Graphs**. Project Report ESC-TR-2005-054. MIT Lincoln Laboratory, Massachusetts, USA, March 2005.
- [5] BRUCE SCHNEIER. **Attack trees: Modeling security threats**. *Dr. Dobbs Journal*, 7(3):243–254, December 1999.
- [6] MARVIN RAUSAND AND ARNLJOT HØYLAND. **System reliability theory: models, statistical methods, and applications**, 396. John Wiley & Sons, 2004.
- [7] VIRGINIA NUNES LEAL FRANQUEIRA. **Finding Multi-step Attacks in Computer Networks Using Heuristic Search and Mobile Ambients**. PhD thesis, University of Twente, the Netherlands, November 13, 2009.
- [8] ROBERT SCHUPPENIES, CHRISTOPH MEINEL, AND FENG CHENG. **Automatic extraction of vulnerability information for attack graphs**. *Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam*, 2009.

- [9] 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 *Computer Security - ESORICS 2014*, **8713** of *Lecture Notes in Computer Science*, pages 494–511. Springer International Publishing, 2014.
- [10] LIONEL GILES. **The art of war, Volume 1**. Library of Alexandria, 1961.
- [11] INFOSEC RESEARCH COUNCIL (IRC). **“Hard Problems List”**. Technical report, Department of Homeland Security (DHS) Cyber Security Research and Development Center, 2005.
- [12] ANDREW JAQUITH. **Security metrics**. Pearson Education, 2007.
- [13] NADYA BARTOL, BRIAN BATES, KAREN MERCEDES GOERTZEL, AND THEODORE WINOGRAD. **Measuring cyber security and information assurance: a state-of-the-art report**. *Information Assurance Technology Analysis Center IATAC*, 2009.
- [14] MATTHEW A. BISHOP. **The Art and Science of Computer Security**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [15] PAUL AMMANN. **Scalable, graph-based network vulnerability analysis**. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 217–224. ACM Press, 2002.
- [16] XINMING OU AND WAYNE F. BOYER. **A Scalable approach to Attack Graph Generation**. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, pages 336–345. ACM Press, 2006.
- [17] SUSHIL JAJODIA AND STEVEN NOEL. **Topological Vulnerability Analysis: A Powerful New Approach for Network Attack Prevention, Detection, and Response**. In *Proceedings of Algorithms, Architectures, and Information System Security, Indian Statistical Institute Platinum Jubilee Series*, pages 285–305, 2009.
- [18] NIRNAY GHOSH AND S.K. GHOSH. **A planner-based approach to generate and analyze minimal attack graph**. *Applied Intelligence*, **36**(2):369–390, 2012.

- [19] OLEG SHEYNER, JOSHUA HAINES, SOMESH JHA, RICHARD LIPPMANN, AND JEANNETTE M. WING. **Automated generation and analysis of attack graphs.** In *Proceedings of the IEEE Symposium on Security and Privacy, SP'02*, pages 273–284, Washington, DC, USA, 2002.
- [20] OLEG MIKHAIL SHEYNER. **Scenario graphs and attack graphs.** PhD thesis, Carnegie Mellon University, 2004.
- [21] 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, SSYM'05*, pages 8–16, Berkeley, CA, USA, 2005. USENIX Association.
- [22] 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.
- [23] ARKADEEP KUNDU AND SOUMYA K. GHOSH. **A multi-objective search strategy to select optimal network hardening measures.** *International Journal of Decision Support Systems*, **1**(1):130–148, 2015.
- [24] LINGYU WANG, ANYI LIU, AND SUSHIL JAJODIA. **An Efficient and Unified Approach to Correlating, Hypothesizing, and Predicting Intrusion Alerts.** In *Proceedings of the 10th European Conference on Research in Computer Security, ESORICS'05*, pages 247–266, 2005.
- [25] 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, 2012.
- [26] LINGYU WANG, TANIA ISLAM, TAO LONG, ANOOP SINGHAL, AND SUSHIL JAJODIA. **An Attack Graph-Based Probabilistic Security Metric.** In *Proceedings of the 22Nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 283–296, Berlin, Heidelberg, 2008. Springer-Verlag.
- [27] LINGYU WANG, ANOOP SINGHAL, AND SUSHIL JAJODIA. **Measuring the Overall Security of Network Configurations Using Attack Graphs.** In STEVE

- BARKER AND GAIL-JOON AHN, editors, *Data and Applications Security XXI*, **4602** of *LNCS*, pages 98–112. Springer Berlin Heidelberg, 2007.
- [28] SUSHIL JAJODIA. **Topological Analysis of Network Attack Vulnerability**. In *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, ASIACCS '07, 2007.
- [29] 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, pages 109–118, New York, NY, USA, 2004. ACM.
- [30] LINGYU WANG, SUSHIL JAJODIA, ANOOP SINGHAL, PENG SU 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.
- [31] QIXU LIU AND YUQING ZHANG. **VRSS: A new system for rating and scoring vulnerabilities**. *Computer Communications*, **34**(3):264 – 273, 2011.
- [32] FENG ZHAO, HEQING HUANG, HAI JIN, AND QIN ZHANG. **A hybrid ranking approach to estimate vulnerability for dynamic attacks**. *Computers & Mathematics with Applications*, **62**(12):4308 – 4321, 2011.
- [33] CANDACE SUH-LEE AND JUYEON JO. **Quantifying security risk by measuring network risk conditions**. In *Proceedings of the 14th International Conference on Computer and Information Science (ICIS), 2015 IEEE/ACIS*, pages 9–14, June 2015.
- [34] US-CERT. **United States Computer Emergency Response Team**. <https://www.us-cert.gov/>, Online.
- [35] SANS. <http://www.sans.org/newsletters/cva/>, Online.
- [36] X-FORCE. **X-Force frequently asked questions. Available from:**. <http://www-935.ibm.com/services/us/iss/xforce/faqs.html>, Online.
- [37] NVD. **National Vulnerability Database**. <https://nvd.nist.gov/>, Online.

REFERENCES

- [38] VUPEN-SECURITY. <http://www.vupen.com/english/>, Online.
- [39] SECUNIA. http://secunia.com/about_secunia_advisories/, Online.
- [40] MICROSOFT. <http://www.microsoft.com/technet/>, Online.
- [41] SYMANTEC. **Symantec Security Response, Threat severity assessment. Available from:** <http://www.symantec.com/avcenter/threat.severity.html>, Online.
- [42] PETER MELL, KAREN SCARFONE, AND SASHA ROMANOSKY. **Common Vulnerability Scoring System.** *IEEE Security Privacy*, 4(6):85–89, Nov 2006.
- [43] PETER MELL, KAREN SCARFONE, AND SASHA ROMANOSKY. **A Complete Guide to the Common Vulnerability Scoring System Version 2.0.** June, 2007.
- [44] CWSS. **Common Weakness Scoring System.** <https://cwe.mitre.org/cwss/>, 2016.
- [45] FENG CHEN, DEHUI LIU, YI ZHANG, AND JINSHU SU. **A Scalable Approach to Analyzing Network Security using Compact Attack Graphs.** *Journal of Networks*, 5(5), 2010.
- [46] LINGYU WANG, SUSHIL JAJODIA, ANOOP SINGHAL, AND STEVEN NOEL. **k-Zero Day Safety: Measuring the Security Risk of Networks against Unknown Attacks.** In DIMITRIS GRITZALIS, BART PRENEEL, AND MARIANTHI THEOHARIDOU, editors, *Computer Security ESORICS 2010*, 6345 of *Lecture Notes in Computer Science*, pages 573–587. Springer Berlin Heidelberg, 2010.
- [47] CYNTHIA PHILLIPS AND LAURA PAINTON SWILER. **A Graph-based System for Network-vulnerability Analysis.** In *Proceedings of the Workshop on New Security Paradigms*, NSPW’98, pages 71–79, New York, NY, USA, 1998. ACM.
- [48] R. ORTALO, Y. DESWARTE, AND M. KAANICHE. **Experimenting with quantitative evaluation tools for monitoring operational security.** *IEEE Transactions on Software Engineering*, 25(5):633–650, September 1999.

- [49] WEI LI AND R.B. VAUGHN. **Cluster Security Research Involving the Modeling of Network Exploitations Using Exploitation Graphs.** In *6th IEEE Int. Symp. on Cluster Computing and the Grid, CCGRID'06*, **2**, pages 26–36, May 2006.
- [50] 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*, pages 31–38, New York, NY, USA, 2006. ACM.
- [51] ISO/IEC27005. **Information technology - Security Techniques - Information security risk management. ISO/IEC 27005**, 2011.
- [52] ENISA. <https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory>.
- [53] ENISA. **Priorities for Research on Current and Emerging Network Technologies.** http://www.eu-mesh.eu/files/publications/procent_report_ENISA.pdf.
- [54] HORST BUNKE AND KIM SHEARER. **A Graph Distance Metric Based on the Maximal Common Subgraph.** *Pattern Recogn. Lett.*, **19**(3-4):255–259, March 1998.
- [55] B.T. MESSMER AND H. BUNKE. **A new algorithm for error-tolerant subgraph isomorphism detection.** *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(5):493–504, May 1998.
- [56] LINGYU WANG, STEVEN NOEL, AND SUSHIL JAJODIA. **Minimum-cost network hardening using attack graphs.** *Computer Communications*, **29**(18):3812 – 3824, 2006.
- [57] MICKI KRAUSE NOZAKI AND HAROLD F TIPTON. **Information Security Management Handbook**, **5**. CRC Press, 2016.
- [58] CVE. **Common Vulnerabilities and Exposures.** <https://cve.mitre.org/>.

- [59] RICHARD LIPPMANN, SETH WEBSTER, AND DOUGLAS STETSON. **The Effect of Identifying Vulnerabilities and Patching Software on the Utility of Network Intrusion Detection**, pages 307–326. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.
- [60] EXPLOITDB. **Offensive Securitys Exploit Database Archive**. <https://www.exploit-db.com/>.
- [61] SPAFFORD E. KRSUL, I. AND M. TRIPUNITRA. **Computer Vulnerability Analysis**. Technical report, COAST Publications, West Lafayette, IN, April 1997.
- [62] UMESH SHANKAR, KUNAL TALWAR, JEFFREY S. FOSTER, AND DAVID WAGNER. **Detecting Format String Vulnerabilities with Type Qualifiers**. In *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*, SSYM’01, Berkeley, CA, USA, 2001. USENIX Association.
- [63] CAIDA. **CODE-RED WORMS: A GLOBAL THREAT, CAIDA Analysis of Code-Red**. <https://www.caida.org/research/security/code-red/>, September 2016.
- [64] NIMBDA. **The Nimda Worm: An Overview, System Administration, Networking and Security Institute (SANS)**. <https://www.sans.org/reading-room/whitepapers/malicious/nimda-worm-overview-95>, September 2016.
- [65] SIV HILDE HOUMB, V NUNES LEAL FRANQUEIRA, AND ERLEND A ENGUM. **Estimating impact and frequency of risks to safety and mission critical systems using CVSS**. 2008.
- [66] JOHN D FERNANDEZ AND ANDRES E FERNANDEZ. **SCADA systems: vulnerabilities and remediation**. *Journal of Computing Sciences in Colleges*, 20(4):160–168, 2005.
- [67] CPNI. **UK Centre for the Protection of the National Infrastructure (CPNI), Process control and SCADA security, Good practice guide (version 2)**, 2008.
- [68] ROSS ANDERSON AND TYLER MOORE. **Information Security Economics – and Beyond**. Springer Berlin Heidelberg, 2007.

- [69] BRUCE SCHNEIER. **Information security and externalities.** *ENISA Quarterly*, 2(4):3–4, 2007.
- [70] R. ANDERSON. **Why Information Security is Hard-An Economic Perspective.** In *Proceedings of the 17th Annual Computer Security Applications Conference, ACSAC '01*, pages 358–, Washington, DC, USA, 2001. IEEE Computer Society.
- [71] GEORGE CYBENKO, SUSHIL JAJODIA, MICHAEL P. WELLMAN, AND PENG LIU. **Adversarial and Uncertain Reasoning for Adaptive Cyber Defense: Building the Scientific Foundation.** In ATUL PRAKASH AND RUDRAPATNA SHYAMASUNDAR, editors, *Information Systems Security*, 8880 of *Lecture Notes in Computer Science*, pages 1–8. Springer International Publishing, 2014.
- [72] KENNETH P. BIRMAN AND FRED B. SCHNEIDER. **The Monoculture Risk Put into Context.** *IEEE Security & Privacy*, 7(1):14–17, Jan 2009.
- [73] NICOLAS FALLIERE, LIAM O MURCHU, AND CHIEN ERIC. **W32.Stuxnet Dossier, Symantec Security Response.** https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf, 2011.
- [74] STEFAN FREI, MARTIN MAY, ULRICH FIEDLER, AND BERNHARD PLATTNER. **Large-scale vulnerability analysis.** In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, pages 131–138. ACM, 2006.
- [75] ERIC RESCORLA. **Security Holes... Who Cares?** In *Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12, SSYM'03*, pages 6–6, Berkeley, CA, USA, 2003. USENIX Association.
- [76] GARY MCGRAW. **Testing for Security During Development: Why we should scrap penetrate-and-patch.** In *Computer Assurance, 1997. COMPASS'97. Are We Making Progress Towards Computer Assurance? Proceedings of the 12th Annual Conference on*, pages 117–119. IEEE, 1997.
- [77] WH BAKER, A HYLENDER, C DAVID PAMULA, J PORTER, AND C SPITLER. **M,” 2011 data breach investigations report,”.** *Verizon RISK Team*, Available:

REFERENCES

- www.verizonbusiness.com/resources/reports/rp_databreach-investigationsreport-2011_en_xg.pdf*, pages 1–72, 2011.
- [78] BRUCE SCHNEIER. **Secrets & Lies: Digital Security in a Networked World.** *INTERNATIONAL HYDROGRAPHIC REVIEW*, 2(1):103–104, 2001.
- [79] IBM. **Cyber Security Intelligence Index. Essential reports on today's security landscape.** <http://www-03.ibm.com/security/data-breach/cyber-security-index.html?ce=ISM0484&ct=SWG&cmp=IBMSocial&cm=h&cr=Security&ccy=US>, 2014.
- [80] VERIZON. **2013 Data Breach Investigations Reports.** <http://www.verizonenterprise.com/verizon-insights-lab/dbir/>, 2013.
- [81] TCCR AGREEMENT. **Common criteria for information technology security evaluation part 1: Introduction and general model july 2009 revision 3 final foreword.** *NIST*, 49:93, 2009.
- [82] ETSI-TS-102-165-1. **Part 1: Method and proforma for Threat, Risk, Vulnerability Analysis.**
- [83] GREG HOGLUND AND GARY MCGRAW. **Exploiting software: How to break code.** Pearson Education India, 2004.
- [84] ANDREW MOORE, ROBERT ELLISON, AND RICHARD LINGER. **Attack Modeling for Information Security and Survivability.** Technical Report CMU/SEI-2001-TN-001, Software Engg. Institute, Carnegie Mellon University, Pittsburgh, PA, 2001.
- [85] BRUCE SCHNEIER. **Attack Trees.** <https://www.schneier.com/paper-attacktrees-ddj-ft.html>.
- [86] BUGTRAQ. **Data Breach Investigations Reports.** <http://bugtraq-team.com/>, 2013.
- [87] WILLIAM A. ARBAUGH, WILLIAM L. FITHEN, AND JOHN MCHUGH. **Windows of Vulnerability: A Case Study Analysis.** *Computer*, 33(12):52–59, December 2000.

- [88] NESSUS. <http://www.tenable.com/products/nessus>.
- [89] GFILANGUARD. <http://www.gfi.com>.
- [90] RETINA. <http://www.amtsoft.com/retina/>.
- [91] HANNES HOLM, TEODOR SOMMESTAD, JONAS ALMROTH, AND MATS PERS-
SON. **A quantitative evaluation of vulnerability scanning**. *Information Manage-
ment & Computer Security*, **19**:231 – 247, 2011.
- [92] QIXU LIU, YUQING ZHANG, YING KONG, AND QIANRU WU. **Improving
VRSS-based Vulnerability Prioritization Using Analytic Hierarchy Process**.
J. Syst. Softw., **85**(8):1699–1708, August 2012.
- [93] LUCA ALLODI AND FABIO MASSACCI. **A Preliminary Analysis of Vulnera-
bility Scores for Attacks in Wild: The Ekits and Sym Datasets**. In *Proceedings
of the 2012 ACM Workshop on Building Analysis Datasets and Gathering Expe-
rience Returns for Security*, BADGERS '12, pages 17–24, New York, NY, USA,
2012. ACM.
- [94] LUCA ALLODI, WOOHYUN SHIM, AND FABIO MASSACCI. **Quantitative As-
sessment of Risk Reduction with Cybercrime Black Market Monitoring**. In
Proceedings of the 2013 IEEE Security and Privacy Workshops, SPW '13, pages
165–172, Washington, DC, USA, 2013. IEEE Computer Society.
- [95] HANNES HOLM, MATHIAS EKSTEDT, AND DENNIS ANDERSSON. **Empirical
Analysis of System-Level Vulnerability Metrics through Actual Attacks**. *IEEE
Transactions on Dependable and Secure Computing*, **9**(6):825–837, Nov 2012.
- [96] PETE HERZOG. **Open-source security testing methodology manual**. *Institute
for Security and Open Methodologies (ISECOM)*, 2003.
- [97] DANIEL GEER AND JOHN HARTHORNE. **Penetration testing: A duet**. In *Com-
puter Security Applications Conference, 2002. Proceedings. 18th Annual*, pages
185–195. IEEE, 2002.
- [98] JOHN WACK, MILES TRACY, AND MURUGIAH SOUPPAYA. **Guideline on net-
work security testing**. *Nist special publication*, **800**:42, 2003.

REFERENCES

- [99] LIWEN HE AND NIKOLAI BODE. **Network Penetration Testing**, pages 3–12. Springer London, London, 2006.
- [100] SECURITYFOCUS. <http://www.securityfocus.com/>.
- [101] EDSGER WYBE DIJKSTRA. **A discipline of programming, Volume 1**. prentice-hall Englewood Cliffs, 1976.
- [102] S CRAIG. **A taxonomy of information systems audits, assessments and reviews**, 2007.
- [103] CLAUDE BERGE AND EDWARD MINIEKA. **Graphs and hypergraphs, Volume 7**. North-Holland publishing company Amsterdam, 1973.
- [104] MICHAEL STAMATELATOS, HOMAYOON DEZFULI, GEORGE APOSTOLAKIS, CHESTER EVERLINE, SERGIO GUARRO, DONOVAN MATHIAS, ALI MOSLEH, TODD PAULOS, DAVID RIHA, CURTIS SMITH, ET AL. **Probabilistic risk assessment procedures guide for NASA managers and practitioners**. 2011.
- [105] GUY HELMER, JOHNNY WONG, MARK SLAGELL, VASANT HONAVAR, LES MILLER, AND ROBYN LUTZ. **A software fault tree approach to requirements analysis of an intrusion detection system**. *Requirements Engineering*, 7(4):207–220, 2002.
- [106] MICHAEL STAMATELATOS, WILLIAM VESELY, JOANNE DUGAN, JOSEPH FRAGOLA, JOSEPH MINARICK, AND JAN RAILSBACK. **Fault tree handbook with aerospace applications**, 2002.
- [107] WILLIAM E VESELY, FRANCINE F GOLDBERG, NORMAN H ROBERTS, AND DAVID F HAASL. **Fault tree handbook**. Technical report, DTIC Document, 1981.
- [108] STEFANO BISTARELLI, FABIO FIORAVANTI, AND PAMELA PERETTI. **Defense trees for economic evaluation of security investments**. In *First International Conference on Availability, Reliability and Security (ARES'06)*, pages 8–pp. IEEE, 2006.
- [109] SECURELTREE. **Amenaza Technologies**. <http://www.amenaza.com/>.

- [110] NESCOR. **Analysis of Selected Electric Sector High Risk Failure Scenarios, Version 1.0, Technical Working Group 1, Electric Power Research Institute (EPRI), USA.** <http://smartgrid.epri.com/NESCOR.aspx>, September 2013.
- [111] SUMEET JAUHAR, BINBIN CHEN, WILLIAM G. TEMPLE, XINSHU DONG, ZBIGNIEW T. KALBARCZYK, WILLIAM H. SANDERS, AND DAVID M. NICOL. **Model-Based Cybersecurity Assessment with NESCOR Smart Grid Failure Scenarios.** In *PRDC*, 2015.
- [112] NANCY R MEAD AND TED STEHNEY. **Security quality requirements engineering (SQUARE) methodology, Volume 30.** ACM, 2005.
- [113] DAN GORDON, TED STEHNEY, NEHA WATTAS, AND EURGENE YU. **System Quality Requirements Engineering (SQUARE): Case Study on Asset Management System, Phase II.** Technical report, DTIC Document, 2005.
- [114] KYLE INGOLS, RICHARD LIPPMANN, AND KEITH PIWOWARSKI. **Practical Attack Graph Generation for Network Defense.** In *22nd Annual Computer Security Applications Conference, ACSAC'06*, pages 121–130, Dec 2006.
- [115] RICHARD P LIPPMANN, KYLE W INGOLS, CHRIS SCOTT, KEITH PIWOWARSKI, KENDRA KRATKIEWICZ, MICHAEL ARTZ, AND ROBERT CUNNINGHAM. **Evaluating and strengthening enterprise network security using attack graphs.** *Lexington, Massachusetts October*, 2005.
- [116] PAUL AMMANN, JOSEPH PAMULA, RONALD RITCHEY, AND J DES STREET. **A host-based approach to network attack chaining analysis.** In *21st Annual Computer Security Applications Conference (ACSAC'05)*, pages 10–pp. IEEE, 2005.
- [117] SOMESH JHA, OLEG SHEYNER, AND JEANNETTE WING. **Two Formal Analysis of Attack Graphs.** In *Proceedings of the 15th IEEE Workshop on Computer Security Foundations, CSFW '02*, pages 49–57, Washington, DC, USA, 2002. IEEE Computer Society.

- [118] NIRNAY GHOSH AND S.K. GHOSH. **An Approach for Security Assessment of Network Configurations Using Attack Graph.** In *1st International Conference on Networks and Communications, NETCOM'09.*, pages 283–288, Dec 2009.
- [119] HANNES HOLM AND KHALID KHAN AFRIDI. **An expert-based investigation of the Common Vulnerability Scoring System.** *Computers & Security*, **53**:18 – 30, 2015.
- [120] HANNES HOLM. **Baltic Cyber Shield: Research from a Red Team versus Blue Team Exercise**, 2012.
- [121] XINMING OU AND ANOOP SINGHAL. **Quantitative Security Risk Assessment of Enterprise Networks**, Springer-Verlag New York. 2011.
- [122] ZHANG LUFENG, TANG HONG, CUI YIMING, AND ZHANG JIANBO. **Network Security Evaluation through Attack Graph Generation**, World Academy of Science, Engineering and Technology, 2009.
- [123] STEVEN NOEL AND SUSHIL JAJODIA. **Metrics suite for network attack graph analytics.** In *CISR'14*, pages 5–8, 2014.
- [124] 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 *Proceedings of Military Communications Conference, MILCOM 2006. IEEE*, pages 1–10, Oct 2006.
- [125] NWOKEDI C. IDIKA. **Characterizing and Aggregating Attack Graph-based Security Metrics.** PhD thesis, Center for Education and Research Information Assurance and Security, Purdue University, West Lafayette, IN 47907-2086, August 2010.
- [126] MELANIE TUPPER AND A. NUR ZINCIR-HEYWOOD. **VEA-bility Security Metric: A Network Security Analysis Tool.** In *Proceedings of the 2008 Third International Conference on Availability, Reliability and Security, ARES '08*, pages 950–957, Washington, DC, USA, 2008. IEEE Computer Society.

- [127] MARJAN KERAMATI, HASSAN ASGHARIAN, AND AHMAD AKBARI. **Cost-aware network immunization framework for intrusion prevention.** In *IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE)*, pages 639–644, Dec 2011.
- [128] MARJAN KERAMATI AND AHMAD AKBARI. **An attack graph based metric for security evaluation of computer networks.** In *Telecommunications (IST), 2012 Sixth International Symposium on*, pages 1094–1098, Nov 2012.
- [129] MOHAMMED ALHOMIDI AND REED MARTIN. **A Genetic Algorithm Approach for the Most Likely Attack Path Problem.** In *Eighth International Conference on Availability, Reliability and Security (ARES)*, pages 360–366, Sept 2013.
- [130] MOHAMMED ALHOMIDI AND MARTIN REED. **Finding the minimum cut set in attack graphs using genetic algorithms.** In *International Conference on Computer Applications Technology (ICCAT), 2013*, pages 1–6, Jan 2013.
- [131] B. YIGIT, G. GR, AND F. ALAGZ. **Cost-Aware Network Hardening with Limited Budget Using Compact Attack Graphs.** In *2014 IEEE Military Communications Conference*, pages 152–157, Oct 2014.
- [132] P. BHATTACHARYA AND S. K. GHOSH. **Analytical framework for measuring network security using exploit dependency graph.** *IET Information Security*, 6(4):264–270, Dec 2012.
- [133] FANGFANG DAI, YING HU, KANGFENG ZHENG, AND BIN WU. **Exploring risk flow attack graph for security risk assessment.** *IET Information Security*, 9(6):344–353, 2015.
- [134] VIVEK SHANDILYA, CHRIS B SIMMONS, AND SAJJAN SHIVA. **Use of attack graphs in security systems.** *Journal of Computer Networks and Communications*, 2014, 2014.
- [135] KEREM KAYNAR. **A taxonomy for attack graph generation and usage in network security.** *Journal of Information Security and Applications*, 29:27 – 56, 2016.

- [136] MICHAEL HOWARD. **Fending off future attacks by reducing your attack surface.** <https://msdn.microsoft.com/en-us/security/securecode/columns/default.aspx?pull=/library/en-us/dncode/html/secure02132003.asp>.
- [137] MICHAEL HOWARD, JON PINCUS, AND JEANNETTE WING. **Measuring Relative Attack Surfaces.** In D.T. LEE, S.P. SHIEH, AND J.D. TYGAR, editors, *Computer Security in the 21st Century*, pages 109–137. Springer US, 2005.
- [138] PRATYUSA K. MANADHATA AND JEANNETTE M. WING. **Measuring a System’s Attack Surface.** In *Tech. Report CMU-CS-04-102*, January 2004.
- [139] PRATYUSA K. MANADHATA AND JEANNETTE M. WING. **An Attack Surface Metric.** *IEEE Trans. on Software Engineering*, **37**(3):371–386, May 2011.
- [140] KUN SUN AND SUSHIL JAJODIA. **Protecting Enterprise Networks Through Attack Surface Expansion.** In *Proceedings of the 2014 Workshop on Cyber Security Analytics, Intelligence and Automation, SafeConfig ’14*, pages 29–32, New York, NY, USA, 2014. ACM.
- [141] JENNIFER A. COWLEY, FRANK L. GREITZER, AND BRONWYN WOODS. **Effect of network infrastructure factors on information system risk judgments.** *Computers & Security*, **52**:142 – 158, 2015.
- [142] HORST BUNKE, PETER J. DICKINSON, MIRO KRAETZL, AND WALTER D. WALLIS. **A Graph-Theoretic Approach to Enterprise Network Dynamics (Progress in Computer Science and Applied Logic (PCS)).** Birkhauser, 2006.
- [143] P. SHOWBRIDGE, M. KRAETZL, AND D. RAY. **Detection of abnormal change in dynamic networks.** In *Proceedings of Information, Decision and Control, IDC’99.*, pages 557–562, 1999.
- [144] QI LIAO AND AARON STRIEGEL. **Intelligent network management using graph differential anomaly visualization.** In *2012 IEEE Network Operations and Management Symposium*, pages 1008–1014, April 2012.
- [145] MALIK SHAHZAD KALEEM AWAN, PETE BURNAP, AND OMER RANA. **Identifying cyber risk hotspots: A framework for measuring temporal variance in computer network risk.** *Computers & Security*, **57**:31–46, 2016.

- [146] LAURA P. SWILER, CYNTHIA PHILLIPS, DAVID ELLIS, AND STEFAN CHAKERIAN. **Computer-attack graph generation tool**. In *Proceedings of the DARPA Information Survivability Conference amp; Exposition II, 2001. DISCEX '01*, **2**, pages 307–321, 2001.
- [147] STEVEN NOEL AND SUSHIL JAJODIA. **Understanding complex network attack graphs through clustered adjacency matrices**. In *In Proceedings of Computer Security Applications Conference, 21st Annual*, pages 10 pp.–169, Dec 2005.
- [148] VAIBHAV MEHTA, CONSTANTINOS BARTZIS, HAIFENG ZHU, EDMUND CLARKE, AND JEANNETTE WING. **Ranking Attack Graphs**. In DIEGO ZAMBONI AND CHRISTOPHER KRUEGEL, editors, *Proceedings of Recent Advances in Intrusion Detection*, **4219** of *Lecture Notes in Computer Science*, pages 127–144. Springer Berlin Heidelberg, 2006.
- [149] REGINALD E. SAWILLA AND XINMING OU. **Identifying Critical Attack Assets in Dependency Attack Graphs**. In SUSHIL JAJODIA AND JAVIER LOPEZ, editors, *Proceedings of Computer Security - ESORICS 2008*, **5283** of *Lecture Notes in Computer Science*, pages 18–34. Springer Berlin Heidelberg, 2008.
- [150] LEEVAR WILLIAMS, RICHARD LIPPMANN, AND KYLE INGOLS. **An Interactive Attack Graph Cascade and Reachability Display**. In JOHN R. GOODALL, GREGORY CONTI, AND KWAN-LIU MA, editors, *Proceedings of VizSEC 2007, Mathematics and Visualization*, pages 221–236. Springer Berlin Heidelberg, 2008.
- [151] JOHN HOMER, ASHOK VARIKUTI, XINMING OU, AND MILES A. MCQUEEN. **Improving Attack Graph Visualization Through Data Reduction and Attack Grouping**. In *Proceedings of the 5th International Workshop on Visualization for Computer Security, VizSec '08*, pages 68–79, Berlin, Heidelberg, 2008. Springer-Verlag.
- [152] DIPTIKALYAN SAHA. **Extending Logical Attack Graphs for Efficient Vulnerability Analysis**. In *Proc. of the 15th ACM Conference on Computer and Communications Security, CCS '08*, pages 63–74, New York, NY, USA, 2008. ACM.

- [153] KYLE INGOLS, MATTHEW CHU, RICHARD LIPPMANN, SETH WEBSTER, AND STEPHEN BOYER. **Modeling Modern Network Attacks and Countermeasures Using Attack Graphs**. In *Proceedings of Computer Security Applications Conference, 2009. ACSAC '09. Annual*, pages 117–126, Dec 2009.
- [154] STEVEN NOEL, SUSHIL JAJODIA, B. O'BERRY, AND M. JACOBS. **Efficient minimum-cost network hardening via exploit dependency graphs**. In *Proceedings of the 19th Annual Computer Security Applications Conference*, pages 86–95, Dec 2003.
- [155] 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, June 2012.
- [156] ARKADEEP KUNDU, NIRNAY GHOSH, ISHAN CHOKSHI, AND SOUMYA K. GHOSH. **Analysis of attack graph-based metrics for quantification of network security**. In *2012 Annual IEEE India Conference (INDICON)*, pages 530–535, Dec 2012.
- [157] 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, 2014.
- [158] JÖRG PREUSS, STEVEN M. FURNELL, AND MARIA PAPADAKI. **Considering the potential of criminal profiling to combat hacking**. *Journal in Computer Virology*, **3**(2):135–141, 2007.
- [159] NIRNAY GHOSH AND S.K. GHOSH. **An Approach for Security Assessment of Network Configurations Using Attack Graph**. *Proceedings of the International Conference on Networks & Communications*, pages 283–288, 2009.
- [160] DANIEL BORBOR, LINGYU WANG, SUSHIL JAJODIA, AND ANOOP" SINGHAL. **Diversifying Network Services Under Cost Constraints for Better Resilience Against Unknown Attacks**, pages 295–312. Springer International Publishing, Cham, 2016.

- [161] MIGUEL GARCIA, ALYSSON BESSANI, ILIR GASHI, NUNO NEVES, AND RAFAEL OBELHEIRO. **OS Diversity for Intrusion Tolerance: Myth or Reality?** In *Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks*, DSN '11, pages 383–394, Washington, DC, USA, 2011. IEEE Computer Society.
- [162] JOHN ADRIAN BONDY. **Graph Theory With Applications**. Elsevier Science Ltd., Oxford, UK, UK, 1976.
- [163] PENG NING AND DINGBANG XU. **Learning Attack Strategies from Intrusion Alerts**. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 200–209, New York, NY, USA, 2003. ACM.
- [164] VLADIMIR KVASNICKA AND JIRI POSPICHAL. **Fast Evaluation of Chemical Distance by Tabu Search Algorithm**. *Journal of Chemical Information and Computer Sciences*, **34**(5):1109–1112, 1994.
- [165] SÉBASTIEN SORLIN AND CHRISTINE SOLNON. **Reactive Tabu Search for Measuring Graph Similarity**, pages 172–182. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- [166] JOHN W. RAYMOND, ELEANOR J. GARDINER, AND PETER WILLETT. **RASCAL: Calculation of Graph Similarity using Maximum Common Edge Subgraphs**. *The Computer Journal*, **45**(6):631–644, 2002.
- [167] DANAI KOUTRA, JOSHUA T. VOGELSTEIN, AND CHRISTOS FALOUTSOS. **DELTA CON: A Principled Massive-Graph Similarity Function**. *CoRR*, abs/1304.4657, 2013.
- [168] REIJO M. SAVOLA. **Quality of security metrics and measurements**. *Computers & Security*, **37**:78 – 90, 2013.
- [169] PETER J. DICKINSON, HORST BUNKE, AREK DADEJ, AND MIRO KRAETZL. **Matching graphs with unique node labels**. *Pattern Analysis and Applications*, **7**(3):243–254, 2004.

- [170] FAN CHUNG AND LINYUAN LU. **Connected Components in Random Graphs with Given Expected Degree Sequences.** *Annals of Combinatorics*, **6**(2):125–145, 2002.
- [171] YULIA CHERDANTSEVA, PETE BURNAP, ANDREW BLYTH, PETER EDEN, KEVIN JONES, HUGH SOULSBY, AND KRISTAN STODDART. **A review of cyber security risk assessment methods for SCADA systems.** *Computers & Security*, **56**:1–27, 2016.
- [172] GRAPHDB. <http://graphdb.net/>.
- [173] NEO4J. <http://neo4j.com/>.
- [174] MOHAMMAD GHASEMI GOL, ABBAS GHAEMI-BAFGHI, AND HASSAN TAK-ABI. **A comprehensive approach for network attack forecasting.** *Computers & Security*, **58**:83 – 105, 2016.
- [175] B. MESSMER. **Efficient Graph Matching Algorithms for Preprocessed Model Graphs**, 1996.
- [176] BRUNO T. MESSMER, HORST BUNKE, AND IEEE COMPUTER SOCIETY. **Efficient subgraph isomorphism detection: a decomposition approach.** *IEEE Trans. Knowledge Data Eng.*, pages 307–323, 2000.
- [177] B.T. MESSMER AND H. BUNKE. **A decision tree approach to graph and sub-graph isomorphism detection.** *Pattern Recognition*, **32**(12):1979 – 1998, 1999.
- [178] HANNES HOLM. **Performance of automated network vulnerability scanning at remediating security issues.** *Computers & Security*, **31**(2):164 – 175, 2012.

List of Publications

- [1] GHANSHYAM S. BOPCHE AND BABU M. MEHTRE. **Attack graph generation, visualization and analysis: Issues and challenges.** Jaime Lloret Mauri, Sabu M. Thampi, Danda B. Rawat, Di Jin (Eds.): *Proceedings of the 2nd International Symposium on Security in Computing and Communications, SSCC'2014*, Springer Berlin Heidelberg; Volume 467 of Communications in Computer and Information Science, pages 379-390, 2014.
- [2] GHANSHYAM S. BOPCHE AND BABU M. MEHTRE. **Exploiting domination in attack graph for enterprise network hardening.** Jemal H. Abawajy, Sougata Mukherjea, Sabu M. Thampi, Antonio Ruiz-Martnez (Eds.): *Proceedings of the 3rd International Symposium on Security in Computing and Communications, SSCC'2015*, Springer Berlin Heidelberg; Volume 536 of Communications in Computer and Information Science, pages 342-353, 2015.
- [3] GHANSHYAM S. BOPCHE AND BABU M. MEHTRE. **Exploiting curse of diversity for improved network security.** *Proceedings of the 4th International Conference on Advances in Computing, Communications and Informatics, ICACCI-2015*, pages 1975-1981, 2015.
- [4] GHANSHYAM S. BOPCHE AND BABU M. MEHTRE. **Change-point detection in enterprise attack surface for network hardening.** Atulya Nagar, Durga Prasad Mohapatra, Nabendu Chaki (Eds.): *Proceedings of the 3rd International Conference on Advanced Computing, Networking and Informatics, ICACNI-2016*, Springer India; Volume 43 of Smart Innovation, Systems and Technologies; pages 475-485, 2016.
- [5] GHANSHYAM S. BOPCHE AND BABU M. MEHTRE. **Extending attack graph-based metrics for enterprise network security management.** Atulya

- Nagar, Durga Prasad Mohapatra, Nabendu Chaki (Eds.): *Proceedings of the 3rd International Conference on Advanced Computing, Networking and Informatics, ICACNI-2016*, Springer India; Volume 44 of Smart Innovation, Systems and Technologies; pages 315-325, 2016.
- [6] GHANSHYAM S. BOPCHE AND BABU M. MEHTRE. **Graph Similarity Metrics for Assessing Temporal Changes in Attack Surface of Dynamic Networks.** *Computers & Security*, Elsevier, Volume 64, pages 16-43, January 2017, DOI: 10.1016/j.cose.2016.09.010.
- [7] GHANSHYAM S. BOPCHE AND BABU M. MEHTRE. **Differential Attack Graph-based Approach for Change Detection and Network Hardening.** *IET Information Security*, (Submitted on April 27, 2016): Journal Paper Under Review.
- [8] GHANSHYAM S. BOPCHE, BABU M. MEHTRE, MEERJA A. JABBAR, AND BULUSU L. DEEKSHATULU. **A Proximity-based Measure for Quantifying the Security Risk of Vulnerabilities.** *Journal of Computer Virology and Hacking Techniques*, (Revised Version Submitted on December 10, 2016): Journal Paper Under Review.

Appendix A

Annexure

Table A.1: Fact sheet for publication [1]

Title	Attack graph generation, visualization and analysis: Issues and challenges
Authors	Ghanshyam S. Bopche, and Babu M. Mehtre
Publication	In Proceedings of the 2 nd International Symposium on Security in Computing and Communications, SSCC'2014, pages 379-390
ISBN/ISSN	ISBN 978-3-662-44966-0
DOI	10.1007/978 – 3 – 662 – 44966 – 0_37
Status	Published
Publisher	Springer Berlin Heidelberg
Publication Type	SSCC Workshop Proceedings, Volume 467 of Communications in Computer and Information Science

Attack Graph Generation, Visualization and Analysis: Issues and Challenges

Ghanshyam S. Bopche, Babu M. Mehtre

[Buy chapter](#)
\$29.95 / €24.95 / £19.95 *

[Buy eBook](#)
\$79.99 / €63.06 / £52.99*

[Get Access](#)

* Final gross prices may vary according to local VAT.



Chapter Metrics

Readers	1
Downloads	624

Provided by Bookmetrix

Reference tools

[Export citation](#)

[Add to Papers](#)

Other actions

- [About this Book](#)
- [Reprints and Permissions](#)

Abstract

In the current scenario, even the well-administered enterprise networks are extremely susceptible to sophisticated multi-stage cyber attacks. These attacks combine multiple network vulnerabilities and use causal relationship between them in order to get incremental access to enterprise critical resources. Detection of such multi-stage attacks is beyond the capability of present day vulnerability scanners. These correlated "multi-host, multi-stage" attacks are potentially much more harmful than the single point/ isolated attacks. Security researchers have proposed an Attack Graph-based approach to detect such correlated attack scenarios. Attack graph is a security analysis tool used extensively in a networked environment to automate the process of evaluating network's susceptibility to "multi-host, multi-stage" attacks. In the last decade, a lot of research has been done in the area of attack graph- generation, visualization and analysis. Despite significant progress, still there are issues and challenges before the security community that needs to be addressed. In this paper, we have tried to identify current issues and important avenues of research in the area of attack graph generation, visualization and analysis.

Figure A.1: Publication [1]

Table A.2: Fact sheet for publication [2]

Title	Exploiting Domination in Attack Graph for Enterprise Network Hardening
Authors	Ghanshyam S. Bopche, and Babu M. Mehtre
Publication	In Proceedings of the 3 rd International Symposium on Security in Computing and Communications, SSCC'2015, pages 342-353
ISBN/ISSN	ISBN 978-3-319-22915-7
DOI	10.1007/978-3-319-22915-7_32
Status	Published
Publisher	Springer International Publishing
Publication Type	SSCC Workshop Proceedings, Volume 536 of Communications in Computer and Information Science

Security in Computing and Communications
Volume 536 of the series Communications in Computer and Information Science pp 342-353
Date: 08 August 2015

Exploiting Domination in Attack Graph for Enterprise Network Hardening

Ghanshyam S. Bopche  , Babu M. Mehtre

Buy chapter

\$29.95 / €24.95 / £19.95 *

Buy eBook

\$89.00 / €67.82 / £58.99*

 Get Access

* Final gross prices may vary according to local VAT.



Chapter Metrics

 Citations	1
 Readers	2
 Downloads	544

Provided by Bookmetrix

Reference tools

Export citation 

Add to Papers 

Other actions

[» About this Book !\[\]\(deec3bd7ddc76f736b74a4f9de028b5f_img.jpg\)](#)

[» Reprints and Permissions !\[\]\(f5c9d1670e99070850d6a31606c69e25_img.jpg\)](#)

Abstract

Attack graph proved to be a tool of great value to an administrator while analyzing security vulnerabilities in a networked environment. It shows all possible attack scenarios in an enterprise network. Even though attack graphs are generated efficiently, the size and complexity of the graphs prevent an administrator from fully understanding the information portrayed. While an administrator will quickly perceive the possible attack scenario, it is typically tough to know what vulnerabilities are vital to the success of an adversary. An administrator has to identify such vulnerabilities and associated/enabling preconditions, which really matters in preventing an adversary from successfully compromising the enterprise network. Extraction of such meaningful information aid administrator in efficiently allocating scarce security resources. In this paper, we have applied a well known concept of domination in directed graphs to the exploit-dependency attack graph generated for a synthetic network. The *minimal dominating set (MDS)* computed over the generated attack graph gives us the set of initial preconditions that covers all the exploits in the attack graph. We model the problem of computing *MDS* as a *set cover problem (SCP)*. We have presented a small case study to

Figure A.2: Publication [2]

Table A.3: Fact sheet for publication [3]

Title	Exploiting curse of diversity for improved network security
Authors	Ghanshyam S. Bopche, and Babu M. Mehtre
Publication	Proceedings of the 4 th International Conference on Advances in Computing, Communications and Informatics, ICACCI'2015 pages 1975-1981
ISBN/ISSN	978-1-4799-8790-0
DOI	10.1109/ICACCI.2015.7275907
Status	Published
Publisher	IEEE
Publication Type	Conference Proceedings

2	v Ghanshyam S. Bopche ; v Babu M. Mehtre						View All Authors
Author(s)							
Abstract	Authors	Figures	References	Citations	Keywords	Metrics	Media

Abstract:

Higher species diversity in biological systems increases the robustness of the system against the spread of disease or infection. However, computers are remarkably less diverse. Such lack of diversity poses serious risks to the today's homogeneous computer networks. An adversary learns with the initial compromises and then applies the learned knowledge to compromise subsequent systems with less effort and time. An exploit engineered to take advantage of a particular vulnerability could be leveraged on many other systems to multiply the effect of an attack. The existence of the same vulnerability on multiple systems in an enterprise network greatly benefits the adversary because she can gain incremental access to enterprise resources with relative ease. In this paper, we have proposed a metric to identify all the attack paths that are not fairly/truly diversified. Our goal is to identify all the attack paths in an enterprise network in which one or more vulnerabilities that could be exploited more than once. Additionally, our goal is to identify what are all those vulnerabilities and what are the affected software's/services? Based on the proposed heuristics, identical and vulnerable services were identified and diversified by functionally equivalent alternatives in such a way that adversary requires an independent effort (i.e. additional or new effort) for exploiting each vulnerability along every attack path. We have presented a small case study to demonstrate the efficacy and applicability of the proposed metric and proposed an algorithm for diversifying attack paths for making enterprise network more robust against 0-day attacks. Initial results show that our approach is capable of identifying identical and vulnerable software/applications/services that need to be diversified for increased network security.

Published in: [Advances in Computing, Communications and Informatics \(ICACCI\), 2015 International Conference on](#)

Figure A.3: Publication [3]

Table A.4: Fact sheet for publication [4]

Title	Change-point detection in enterprise attack surface for network hardening
Authors	Ghanshyam S. Bopche, and Babu M. Mehtre
Publication	In Proceedings of 3 rd International Conference on Advanced Computing, Networking and Informatics: ICACNI 2015, Volume 2 pages 475-485
ISBN/ISSN	ISBN 978-81-322-2529-4
DOI	10.1007/978-81-322-2529-4_33
Status	Published
Publisher	Springer India
Publication Type	Conference Proceedings, Volume 43 of the series Smart Innovation, Systems and Technologies

Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics
Volume 43 of the series Smart Innovation, Systems and Technologies pp 475-485

Date: 08 October 2015

Change-Point Detection in Enterprise Attack Surface for Network Hardening

Ghanshyam S. Bopche  , Babu M. Mehtre

Buy chapter

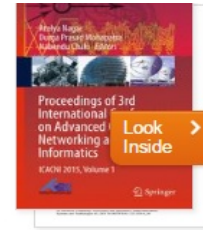
\$29.95 / €24.95 / £19.95 *

Buy eBook


\$179.00 / €142.79 / £122.00*




* Final gross prices may vary according to local VAT.



Chapter Metrics

 Readers 2


 Downloads 296

Provided by Bookmetrix

Abstract

Applications of change-point detection typically originate from the perspective of enterprise network security and network monitoring. With the ever-increasing size and complexity of enterprise networks and application portfolios, network attack surface keeps changing. This change in an attack surface is detected by identifying increase or decrease in the number of vulnerabilities at network level. Vulnerabilities when exploited successfully, either provide an entry point to an adversary into the enterprise network or can be used as a milestone for staging multi-stage attacks. In this paper, we have proposed an approach for change-point detection in an enterprise network attack surface. In this approach, a sequence of static attack graphs are generated for dynamic (time varying) enterprise network, and successive graphs in a sequence are compared for their dissimilarity for change-point detection. We have presented a small case study to demonstrate the efficacy and applicability of the proposed approach in capturing a change in network attack surface. Initial results show that our approach is capable of capturing the newly introduced vulnerabilities into the network and is able to differentiate these vulnerabilities from existing network-based data.

Reference tools

Export citation 

Add to Papers 

Other actions

 About this Book 

 Reprints and Permissions 

Figure A.4: Publication [4]

Table A.5: Fact sheet for publication [5]

Title	Extending Attack Graph-Based Metrics for Enterprise Network Security Management
Authors	Ghanshyam S. Bopche, and Babu M. Mehtre
Publication	In Proceedings of 3 rd International Conference on Advanced Computing, Networking and Informatics: ICACNI 2015, Volume 1 pages 475-485
ISBN/ISSN	ISBN 978-81-322-2538-6
DOI	10.1007/978-81-322-2538-6_49
Status	Published
Publisher	Springer India
Publication Type	Conference Proceedings, Volume 43 of the series Smart Innovation, Systems and Technologies

Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics
Volume 44 of the series Smart Innovation, Systems and Technologies pp 315-325

Date: 03 September 2015

Extending Attack Graph-Based Metrics for Enterprise Network Security Management

Ghanshyam S. Bopche , Babu M. Mehtre

Buy chapter

\$29.95 / €24.95 / £19.95 *

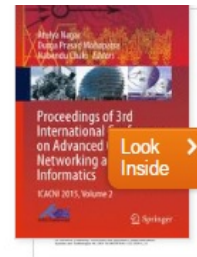


Get Access




* Final gross prices may vary according to local VAT.

Abstract

Measurement of enterprise network security is a long standing challenge to the research community. However, practical security metrics are vital for securing enterprise networks. With the constant change in the size and complexity of enterprise networks, and application portfolios as well, network attack surface keeps changing and hence monitoring of security performance is increasingly difficult and challenging problem. Existing attack graph-based security metrics are inefficient in capturing change in the network attack surface. In this paper, we have explored the possible use of graph-based distance metrics for capturing the change in the security level of dynamically evolving enterprise networks. We used classical graph similarity measures such as *Maximum Common Subgraph (MCS)*, and *Graph Edit Distance (GED)* as an indicator of change in the enterprise network security. Our experimental results shows that graph similarity measures are efficient and capable of capturing changing network attack surface in dynamic (i.e. time varying) enterprise networks.



Chapter Metrics

	Citations	1
	Readers	3
	Downloads	298

Provided by Bookmetrix

Reference tools

Export citation

Add to Papers 

Other actions

- [» About this Book !\[\]\(49cafc1b4ac9c36b24a666d112dd1bdd_img.jpg\)](#)
- [» Reprints and Permissions !\[\]\(0ff44b1c51a0cf0a3e3adb3d5834a98e_img.jpg\)](#)

Figure A.5: Publication [5]

Table A.6: Fact sheet for publication [6]

Title	Graph Similarity Metrics for Assessing Temporal Changes in Attack Surface of Dynamic Networks
Authors	Ghanshyam S. Bopche, and Babu M. Mehtre
Publication	Computers & Security, Volume 64, pages 16-43, January 2017
ISBN/ISSN	ISSN 0167-4048
DOI	10.1016/j.cose.2016.09.010
Status	Published
Publisher	Elsevier
Publication Type	Journal



Figure A.6: Publication [6] decision letter



Figure A.7: Publication [6]