Advanced Attack Tree Based Intrusion Detection

by

Jie Wang

A Doctoral Thesis

Submitted in partial fulfilment
of the requirements for the award of

Doctor of Philosophy
of
Loughborough University

22th February 2012

Research Student Office, Academic Registry
Loughborough University, Leicestershire, LE11 3TU, UK
Switchboard: +44 (0)1509 263171 Fax: +44 (0)1509 223938

Certificate of Originality

. . . .

Jie Wang

22th February 2012

Abstract

Computer network systems are constantly under attack or have to deal with attack attempts. The first step in any network's ability to fight against intrusive attacks is to be able to detect intrusions when they are occurring. Intrusion Detection Systems (IDS) are therefore vital in any kind of network, just as antivirus is a vital part of a computer system. With the increasing computer network intrusion sophistication and complexity, most of the victim systems are compromised by sophisticated multi-step attacks. In order to provide advanced intrusion detection capability against the multi-step attacks, it makes sense to adopt a rigorous and generalising view to tackling intrusion attacks. One direction towards achieving this goal is via modelling and consequently, modelling based detection.

An IDS is required that has good quality of detection capability, not only to be able to detect higher-level attacks and describe the state of ongoing multi-step attacks, but also to be able to determine the achievement of high-level attack detection even if any of the modelled low-level attacks are missed by the detector, because no alert being generated may represent that the corresponding low-level attack is either not being conducted by the adversary or being conducted by the adversary but evades the detection.

This thesis presents an attack tree based intrusion detection to detect multi-step attacks. An advanced attack tree modelling technique, Attack Detection Tree, is proposed to model the multi-step attacks and facilitate intrusion detection. In addition, the notion of Quality of Detectability is proposed to describe the ongoing states of both intrusion and intrusion detection. Moreover, a detection uncertainty assessment mechanism is proposed to apply the measured evidence to deal with the uncertainty issues during the assessment process to determine the achievement of high-level attacks even if any modelled low-level incidents may be missing.

Acknowledgements

At the end of an amazing experience to pursue a Ph.D, there is always somebody to thank. First and foremost, I want to express my deepest gratitude to my main supervisor, Dr. Raphael C.W. Phan, for his tireless efforts to teach me the knowledge and skills I need to become a real researcher and to learn how to expand breadth of knowledge without sacrificing depth. I am also grateful to my co-supervisor Professor David J. Parish who gave me many suggestions and encouragement to improve my research. Raphael and David's willingness to go beyond their duty to provide me with research directions, valuable guidance and encouragement throughout my study has made this Ph.D possible.

I am extremely grateful to my parents, and also my fiancee, Miss Yuan Cao, and her parents, without whom I would not have been able to pursue my degree successfully.

In addition, I want to thank all of my colleagues in the High Speed Networks group for their support and for sharing their knowledge, especially, Dr. John N. Whitley, for his comments and suggestions that help me improve the quality of my research.

Then, I would like to thank all my family members and friends, especially, Ms. Lihong Wang, Mr. Kun Liu, Dr. Lin Guan, Dr. Chong Fu, Dr. Xunli Fan, Dr. Lee Booi Lim, Dr. Jin Fan, Mr. Istvan Szabo, for supporting me through all the years of my study and life in the U.K..

Finally, I would specifically like to thank my thesis examiners, Dr. Robert Edwards and Dr. Ali Al-Sherbaz, who had provided me useful feedback and contributed to improve the quality of the final version of this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

IDS	Intrusion Detection System	1
ADtT	Attack Detection Tree	2
QoD	Quality of Detectability	2
DUA	Detection Uncertainty Assessment	2
UPAT	Unified Parametrisable Attack Tree	3
ATIDS	Attack Tree based IDS	4
LAN	Local Area Network	7
DDoS	Distributed Denial-of-Service	7
NSTAC	National Security Telecommunications Advisory Committee	7
SID	Signature based Intrusion Detection	8
AID	Anomaly based Intrusion Detection	8
SBIDS	Signature Based IDS	9
ABIDS	Anomaly Based IDS	9
NID	Network based Intrusion Detection	9
HID	Host based Intrusion Detection	9
NBIDS	Network Based IDS	10
HBIDS	Host Based IDS	10
DIDS	Distributed IDS	11
TP	True Positive	15
FP	False Positive	15
TN	True Negative	15
FN	False Negative	15
DR	Detection Rate	15
FAR	False Alarm Rate	15
ROC	Receiver Operating Characteristic	17
IDOC	Intrusion Detection Operating Characteristic	17
PPV	Positive Predictive Value	17
IPS	Intrusion Prevention System	20
NBIPS	Network Based Intrusion Prevention System	20
HBIPS	Host Based Intrusion Prevention System	20
GUI	Graphical User Interface	31

TVA	Topological Vulnerability Analysis	31
CAT	Conventional Attack Tree	37
AAT	Augmented Attack Tree	38
DT	Defence Tree	38
ACT	Attack Countermeasure Tree	38
ART	Attack-Response Tree	38
ADTree	Attack-Defense Tree	38
PT	Protection Tree	38
MPAT	Multi-Parameter Attack Tree	39
SAT	Serial Attack Tree	39
UML	Unified Modeling Language	39
EAT	Enhanced Attack Tree	40
NFETA	Nondeterministic Finite Enhanced Tree Automaton	40
OBC	Order Based Connector	40
PBOC	Priority Based Order Connector	40
PAND	Priority-AND	41
TBOC	Time Based Order Connector	41
PN	Parent Node	43
CN	Child Node	43
TBC	Threshold Based Connector	42
K/N	K-out-of-N	42
TTL	Time-To-Live	46
AL	Attack Level	46
AP	Attack Probability	46
ASP	Attack Success Probability	46
MC	Monetary Cost	46
I	Impact	47
R	Risk	47
SV	System Vulnerability	47
NC	Network Configuration	47
SC	System Configuration	47
APr	Access Privilege	47
CVSS	Common Vulnerability Scoring System	47
UPAT	Unified Parametrizable Attack Tree	3
SbD	Steps before Detection	62
StG	Steps to Goal	63
PtG	Progress to Goal	63
TtD	Time to Detect	64
TtCS	Time to Current Subgoal	65

TtG	Time to Goal	65
NAT	Number of Alert Types	66
MAN	Mean Alert Number	66
MAS	Mean Alert Severity	66
D-S	Dempster-Shafer	67
bpa	basic probability assignment	68
IDMEF	Intrusion Detection Message Exchange Format	69
DW	Detection Window	79
NAN	Not A Number	93

List of Symbols

$\mathcal{R}$	Representation algorithm	12
$\mathcal{S}$	Data structure algorithm	12
$\mathcal{C}$	Classification algorithm	12
$\mathbf{D}$	Data source	13
$\sum$	Data states	13
$\mathbf{F}$	Data characteristics	13
$\mathbf{K}$	Knowledge profile	13
$\mathcal{P}$	Profiling algorithm	13
$\|.\|$	Metric frequency	15
E	Elementary node	48
O	Observed node	48
$\vee$	Disjunctive dependency relationship	49
$\wedge$	Conjunctive dependency relationship	49
r	Individual resistance of an exploit	50
R	Cumulative resistance of an exploit	50
e	Exploit	50
c	Condition	50
$p(e)$	Individual score on exploit	50
$p(c)$	Individual score on condition	50
$P(e)$	Cumulative score on exploit	50
$P(c)$	Cumulative score on condition	50
$\mathcal{N}$	Node	53
$\mathcal{E}$	Edge	53
u	Parent Node	53
v	Child Node	53
$\mathcal{C}$	Connector	53
$\mathcal{A}$	Attribute	53
eParam	Unified edge parameter	54
cParam	Unified connector parameter	54
nParam	Unified node attribute parameter	54
$f_u^{\mathsf{connector}}$	Attribute assignment function	57

R_u	Attribute on node u	57
$\mathcal{R}_{\mathcal{E}}$	Set of all attributes	57
v_w	The weakest node	58
I	The system which parent node has one child node	58
II	The system which parent node has multi child nodes	58
M_L	Logical steps based QoD metrics	62
M_T	Time based QoD metrics	62
M_A	Alert based QoD metrics	62
A_c	Complex attack	62
m	Number of attack steps	68
$\{a_i\}_{i=1}^m$	Sequence of attack steps	62
P	Attack path	62
n_l	Leaf node	62
n_r	Root node	62
n_s	Any node between n_l and n_r	62
P'	Attack path with achieved attack steps	62
d	Index of the currently detected node	62
$\langle \ldots \rangle$	Sequence	62
P''	Attack path with unachieved attack steps	63
α	The number of actual achieved subgoals	63
β	The theoretical least effort	63
k	Number of child node	63
$\{t_i\}_{i=0}^m$	Sequence of time	64
t_0	Initial time step	64
t_1	Time step of the adversary achieved first attack step	64
t_m	Time step of the adversary achieved last attack step	64
t_d	Time step of the adversary currently achieved attack step	64
T_{min}	Expected minimal time	65
q	Number of modelled attack incident types	66
t	Number of actual detected attack incident types	66
$\{I^+{}_j\}_{j=1}^t$	Actual detected attack incident types	66
$\{N_j\}_{j=1}^t$	Number of actual detected incidents of each attack incident type	66
AS_i	Alert severity level	66
AS_1	Bottom alert severity level	66
AS_r	Top alert severity level	66
Θ	Frame of discernment	68
m	Mass function	68
V	Focal element	68
ϕ	Empty set	68

$m(V)$	V's basic probability number	68
$\bigoplus$	Orthogonal summation	68
f_1	Alert amount functions	69
A_i	Number of generated alerts	69
γ	Threshold	69
f_2	Alert severity functions	70
P_{r0}	Alert severity of an incident	70
δ	The number of severity levels	70
I_i	Index of the detected incident	70
j	Index of the focal element	70
P_{IDS}	Detection precision of the intrusion detector	71
V_1	First proposed focal element	71
V_2	Second proposed focal element	71
V_3	Third proposed focal element	71
$m^{I_i}(V_1)$	Belief value of the first proposed focal element on incident	71
$m^{I_i}(V_2)$	Belief value of the second proposed focal element on incident	71
$m^{I_i}(V_3)$	Belief value of the third proposed focal element on incident	71
Bel^{f1}	Belief value set for the alert amount membership functions	73
Bel^{f2}	Belief value set for the alert severity membership functions	73
$\mathcal{D}$	Decomposition tuple	77
$\mathcal{M}$	Metrics on the tree	77
$\mathcal{F}$	Assessment factors	77
$Label$	Name of the exploit associated with each edge	77
$SIG_{u,v}$	Attack signature of an atomic attack	77
$\mathfrak{T}$	Time interval	80
T_s	Starting time of time interval	80
T_e	Ending time of time interval	80
$\mathbb{S}$	Window size	80
Δ	Sliding move step	80
$\cup$	Incident connector in atomic attack signature	97

List of Publications

Conference Publications

- J. Wang, R.C.W. Phan, J.N. Whitley and D.J. Parish. Augmented Attack Tree Modelling of SQL Injection Attacks. In *Proceedings of The 2nd IEEE International Conference on Information Management and Engineering (ICIME 2010)*, vol.6, pages 182-186. Chengdu, China. 2010.
- J. Wang, R.C.W. Phan, J.N. Whitley and D.J. Parish. Augmented Attack Tree Modeling of Distributed Denial of Services and Tree Based Attack Detection Method. In *Proceedings of The 10th IEEE International Conference on Computer and Information Technology (CIT2010)*, pages 1009-1014. Bradford, UK. 2010.
- J. Wang, R.C.W. Phan, J.N. Whitley and D.J. Parish. Quality of Detectability (QoD) and QoD-aware AAT-based Attack Detection. In *Proceedings of The 5th International Conference for Internet Technology and Secured Transactions (ICITST-2010)*, pages 152-157. London, UK. 2010.
- J. Wang, R.C.W. Phan, J.N. Whitley and D.J. Parish. DDoS Attacks Traffic and Flash Crowds Traffic Simulation with a Hardware Test Center Platform. In *Proceedings of IEEE World Congress on Internet Securitys (WorldCIS2011)*, pages 29-34. London, UK. 2011.

Journal Publications

- J. Wang, J.N. Whitley, R.C.W. Phan and D.J. Parish. Unified Parametrizable Attack Tree. *International Journal for Information Security Research*, vol.1(1/2), pages 20-26, 2011.
- J.N. Whitley, R.C.W. Phan, J. Wang and D.J. Parish. Attribution of Attack Trees. *Computers & Electrical Engineering*, vol.37(4), pages 624-628, 2011.

- J. Wang, J.N. Whitley, R.C.W. Phan and D.J. Parish. Advanced DDoS Attacks Traffic Simulation with a Test Center Platform. *International Journal for Information Security Research*, vol.1(4), pages 169-177, 2011.

Chapter 1

Introduction

With the advances of computer network systems, it is crucial to ensure the security properties such as *confidentiality*, *integrity* and *availability* [12, 73] in computer networks. However, since the computer network system becomes more complex [76], the security properties are threatened by the increasing computer network intrusions, not only the intrusion number but also the intrusion sophistication. The sophisticated intrusions are conducted by applying multiple intrusion tools or approaches in order to achieve an attack goal. Typically, they are composed of multiple attacks as steps and even each step may further be composed of multiple low-level actions. Both malicious adversaries and network security researchers are continually investigating new ways for launching or detecting multi-step attacks. The former mainly concentrates on the system compromise by exploiting various vulnerabilities with such multi-step attacks. While the latter focuses on the system protection by modelling and detection to mitigate such multi-step attacks.

Intrusion detection remains a vital task for current as well as emerging networks as the threat of adversarial intrusions is ever present, often irrespective of the underlying technology or architecture. Since the number and types of intrusion methods increase over time, it makes sense to adopt a more rigorous and generalizing view to tackling those intrusion threats. One direction towards achieving this goal is via modelling. The purpose of modelling is not only to provide details on how each sophisticated attack has been carried out, but also to facilitate the attack detection process itself.

In order to carry out the modelling against the sophisticated attacks, there is a need to identify any detailed information, for example, the attack step, the connections between attack steps. While it is good to be able to detect low-level attacks as these are fundamental to any ultimate attack, it is also necessary for an Intrusion Detection System (IDS) to detect and generate high-level attack alerts because otherwise it may be prone to (low-level) alert flooding and false alerts [68]. Thus, one major challenge is how to model the whole operation of multi-step attacks to assist the detection process. Once we obtain the model, the next key challenge is how to conduct the intrusion detection process according to the generated model.

Within the intrusion detection on any real networks, it is possible that any low-level attacks may evade the detection from IDS. Therefore, an additional challenge is how we could deal with any missed or unachieved low-level attack detection when performing the high-level attack detection. Moreover, the last challenge is that how the system administrator could quickly know the progress of the ongoing multi-step attack detection process based on the generated attack model and the conducted intrusion detection process.

1.1 Research Contributions

In this thesis, the proposed advanced attack tree based intrusion detection is based on two main research fields: attack tree modelling and intrusion detection. Thus, the research contributions can be classified into two groups: (1) contributions to the *attack tree modelling* research; and (2) contributions to the *intrusion detection* research.

Our research contributions and their linkages with the relevance to the related research field are illustrated in Figure 1.1. As shown in Figure 1.1, our research and our contributions are drawn as a block diagram within the big box. An inner rectangle represents a major contribution, while an inner olive represents a minor contribution. Several minor contributions contribute the achievement of a major contribution. In addition, a cloud outside of the box represents an independent research field. Each of our minor contribution relates to a research field. The top left clouds are about the *attack tree modelling* research, while the bottom clouds are about the *intrusion detection* research. These research fields will be described in the background part as the following Chapter 2 and Chapter 3. Our research contributions will be presented after Chapter 3.

Our first major contribution is the *Attack Detection Tree (ADtT)* and concerns the multi-step attack modelling. It is shown as the middle inner rectangle with "**ADtT (Advanced attack tree)**". To the best of our knowledge, it is the first advanced attack tree specialised for intrusion detection and with uncertainty assessment.

The tree based *Quality of Detectability (QoD) metrics* (that is, the first left bottom olive inside the big box) and *Detection Uncertainty Assessment (DUA)* (that is, the middle bottom olive inside the big box) are specialised for *Attack Detection Tree*. The first mechanism is able to conduct the realtime monitoring

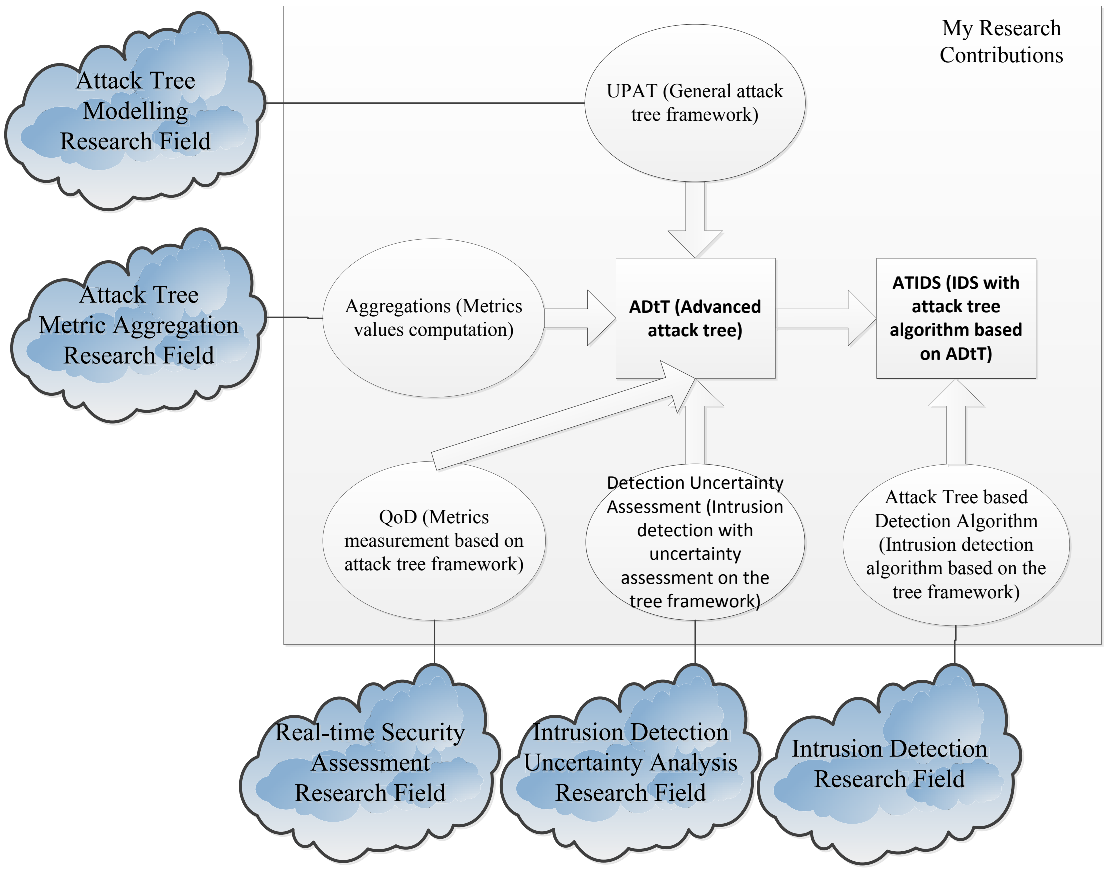

My Research Contributions
Attack Tree Modelling Research Field
UPAT (General attack tree framework)
Attack Tree Metric Aggregation Research Field
Aggregations (Metrics values computation)
ADtT (Advanced attack tree)
ATIDS (IDS with attack tree algorithm based on ADtT)
QoD (Metrics measurement based on attack tree framework)
Detection Uncertainty Assessment (Intrusion detection with uncertainty assessment on the tree framework)
Attack Tree based Detection Algorithm (Intrusion detection algorithm based on the tree framework)
Real-time Security Assessment Research Field
Intrusion Detection Uncertainty Analysis Research Field
Intrusion Detection Research Field

Figure 1.1: Research Contributions and their Relevance

on the ongoing intrusion detection progress based on the tree framework for the system administrator. So, the system administrator can quickly know the status of the intrusion detection progress. The second mechanism is able to conduct the evidence assessment to deal with the detection uncertainty issues during the tree based intrusion detection process.

Besides the aforementioned QoD and DUA, two further minor research contributions: (1) the *Unified Parametrisable Attack Tree (UPAT)* (that is, the first top left olive inside the big box), and (2) the *attack resistance metrics aggregations* (that is, the second top left olive inside the big box) are also contribute to our first major contribution. Both of them provide generic but essential theoretical work on attack tree research. The first one modelles the attack tree into the generic framework and defines three configurable parameters to set the possible attack tree extensions. While the second one provides the attack resistance aggregation approaches in the attack tree and analyses the weakest links of security systems.

Our second major contribution is about the *attack tree based intrusion detection*, which is shown as the first right inner rectangle with "**ATIDS (IDS with attack tree algorithm based on ADtT)**", and concerns in particular the high-level attack detection instead of the low-level detection on any single malicious network packet. We propose two different attack tree based intrusion detection algorithms (that is, the right bottom olive inside the big box), called *Augmented Attack Tree based intrusion detection* and *Attack Detection Tree based intrusion detection*. The first one is able to detect the high-level attacks according to the Augmented Attack Tree. This is a crucial result, because being able to detect the modelled high-level attacks not only demonstrates the applicability of this approach, but also provides the prototype for the second more advanced one. The second detection algorithm is based on our first major contribution. More precisely, our second intrusion detection algorithm conducts the additional *Quality of Detectability measurement* and the *detection uncertainty assessment* to assist the high-level attack detection. The experimental results show that both the *sequential based approach* and the *combination based approach* can deal with the detection uncertainty issue, but the combination one has better detection performance with more high-level attacks detection.

1.2 Research Assumptions

The main research scope of this research is to propose an intrusion detection approach based on the modelled attack tree. However, there are some research issues outside the research scope of this research. Therefore, we define three assumptions used in this research to focus on the research scope.

Assumption 1: *The applied low-level detector and the proposed intrusion detection system are vulnerability free.*

Any software systems may have vulnerabilities. If any of the vulnerabilities been exploited, the adversary may maliciously control the compromised software system and cause any unpredictable damage. Our applied low-level detector Snort and our developed **A**ttack **T**ree based **IDS** (ATIDS) are software systems. It is possible that both of them may have vulnerabilities due to poor design or implementation. Once the adversary compromises any applied software, ATIDS may conduct the misbehavior, such as, stop the detection process, evade any particular intrusions.

Therefore, there is a need for us to assume that Snort and ATIDS are vulnerability free. Thus, the attack traffic from the adversary can neither compromise ATIDS nor intentionally evade the detection.

Assumption 2: *The generated attack detection tree represents the modelled multi-step attacks precisely.*

Though there are some free and commercial tools to generate attack tree model, no existing tool can assist us to construct our proposed attack detection tree against any multi-step attacks. Thus, we model the attack detection tree manually by analysing the data and documentations about the applied attack data sets. Because design and development of an automatic attack detection tree generation tool does not fall in the scope of our research, we assume that our manually generated attack detection tree can precisely represent the modelled multi-step attacks.

Assumption 3: *The adversary can conduct a high-level attack step within a short time period.*

The way to generate the multi-step attack is not our research goal. Thus, we adopt the attack traffic generation within our experiments by replaying the attack traffic captures, which are provided by the third party. In order to facilitate the conducted detection experiments, the corresponding traffic capture of each attack step is replayed based on the speed of the replaying software. Therefore, we assume that the adversary can conduct a high-level attack step within a short time period.

1.3 Thesis Outline

The remainder of this thesis is organized as follows.

Chapter 2 provides a background of intrusion detection including preliminary concepts and current state of the subject.

Chapter 3 reviews the relevant literature on attack graph and attack tree modelling approaches as the foundation of this research.

Chapter 4 describes our *Unified Parametrisable Attack Tree* to provide the scope to various attack tree extensions. In addition, this chapter presents our proposed *attack resistance aggregation* approaches within the attack tree.

Chapter 5 describes our proposed *Attack Detection Tree* with two additional mechanisms: the *Quality of Detectability* mechanism and the *detection uncertainty assessment.*

Chapter 6 presents our two proposed tree based intrusion detection mechanisms based on the original Augmented Attack Tree and Attack Detection Tree.

Chapter 7 presents the experimental setup and the achieved results.

Finally, Chapter 8 concludes the thesis and presents some future research that can extend our work.

Parts of this research have been published in international journals and conference proceedings to demonstrate the novelty of this research. To date, six papers have been published including two journal articles. An additional journal paper has been accepted and in press. Moreover, another journal paper will be submitted soon on the outcome of the attack detection tree based intrusion detection.

The model of Unified Parametrisable Attack Tree described in Chapter 4 has been published in part in [117], while the fourth section "Attack Resistance Aggregation" has been published in [124]. In Chapter 5, the first described "Quality of Detectability Metrics" has been published in part in [116]. In Chapter 6, the second section "Augmented Attack Tree Based Intrusion Detection" has been published in [115].

Chapter 2

Intrusion Detection

Intrusions are defined to be "*unauthorised uses, misuses, or abuses of computer systems by either internal authorized users or external adversaries*" [76]. According to the National Security Telecommunications Advisory Committee (NSTAC) of U.S. Government [84], *intrusion detection* is "*the process of identifying that an intrusion has been attempted, is occurring, or has occurred*".

Since Denning [28] methodised and systemised intrusion detection into IDS, IDS had attracted numerous research in the last two decades and became one of the important computer network infrastructure components. Figure 2.1 illustrates a typical abstracted network topology with IDS. The left Internet cloud communicates with the right local area network (LAN) cloud through an IDS, which is located on the physical link between two network clouds. Note that it is possible to place IDS either outside or inside the firewall of LAN according to the usage. The Internet is usually the root source of intrusion, whereas the LAN is usually the ultimate target of intrusion. Take a coordinated attack, Distributed Denial-of-Service (DDoS) attack, for example, the adversary-compromised secondary victims (also known as bots) consist of the botnet, which may be a part of Internet cloud, generate and transmit the DDoS traffic into the primary victim server in the LAN cloud through IDS to congest the LAN and overload the primary victim server. As another example, a Web application attack, such as SQL injection attack injects the malicious SQL statement into the victim Web server within the LAN cloud to bypass the authorization and obtain confidential information. Besides the network physical link located IDS, it is also possible to install and employ IDS on any hosts with LAN to monitor the intrusion to the corresponding hosts. Hence, the appropriate IDS can detect the aforementioned computer network attacks and generate alarms to warn the network administrator of the LAN.

This chapter reviews various intrusion detection related research from the basic foundation to the additional augmentations. Section 2.1 describes the classification

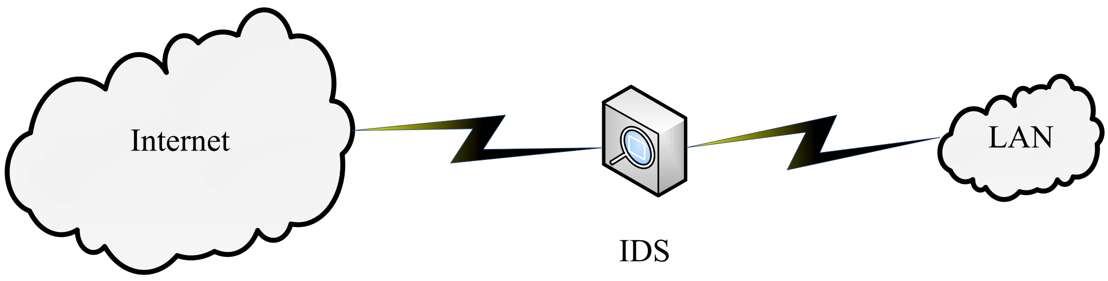


Figure 2.1: Abstracted Network Topology of Intrusion Detection System

of intrusion detection based on detection principles and data types. Section 2.2 introduces the typical research of intrusion detection. Then, the research difficulties and existing limitations are described in Section 2.3. Intrusion detection extensions like intrusion prevention and intrusion response are discussed in Section 2.4. Section 2.5 describes the intrusion analysis including situation awareness and uncertainty analysis of intrusion detection. Finally, Section 2.6 summaries this chapter.

2.1 Classification of Intrusion Detection

2.1.1 Classification Based on Detection Principles

Currently known detection techniques can be categorized into two main principles: *signature based intrusion detection* (SID) and *anomaly based intrusion detection* (AID). Note that *signature based intrusion detection* is also known as *misuse intrusion detection*, *knowledge based intrusion detection* or *rule based intrusion detection*. Likewise, *anomaly based intrusion detection* is also known as *behaviour based intrusion detection*.

- *Signature based intrusion detection.* The principle of SID is that the intrusion detection mechanism contains a number of known attacks description or 'signatures' in the internal database. The stored signatures can be immediately recognized when the intrusion pattern is matched.

  The main advantage of SID is that it concentrates on audit data analysis with less false alerts generation [47, 50]. However, the significant drawback of SID is that it only detects the known intrusion patterns. Hence, SID cannot identify novel intrusions which lack defined signatures. In addition, SID suffers the bottleneck problem of signature updating [62], since the signatures only can be defined and updated once any intrusions been achieved by the adversary and been identified by the security researchers.

The intrusion detection system which applies the SID principle is known as Signature Based IDS (SBIDS). The most well-known and widely-used SBIDS in intrusion detection research field appears to be Snort [2, 94], which examines the security of each single network packet according to signatures (rules).

- *Anomaly based intrusion detection.* The principle of AID is that the intrusion detection mechanism detects any anomalous activities that run differently from the acceptable normal profiles. AID relies on the models of the intended normal behaviour of the computer network system and legitimate user. The typical AID approach is characterise the normal network and user behaviors with statistical profiles. AID interprets deviations from normal behaviour as evidence of malicious behaviour [28]. Hence, the differences between the normal behaviour and the anomaly behaviour can be presented quantitatively and qualitatively [51].

  The key advantage of AID is that it can detect not only AID known intrusions but also AID unknown intrusions. The main limitation of AID is that it suffers from the difficulty of building robust models [47]. In order to provide the detection precision, the professional expertise is typically needed for the security analyst to tune the detection threshold of the statistical files [111]. However, the imprecise detection leads the large number of false alarms generation. These numerous alarms sending from IDS to the system administrator are known as the alarm flooding, which may cause IDS to be unusable and decrease the alarm sensitivity presented to the system administrator [50].

  The intrusion detection system which applies the AID principle is known as Anomaly Based IDS (ABIDS). The widely-applied ABIDSs in intrusion detection research field appear to be PHAD [69], PAYL [118] and Bro [3].

2.1.2 Classification Based on Type of Data

The data analysed by intrusion detection are generally classified into two categories: the network traffic, and the log information (including both the operating system's audit trails and the application's logging information [73]). Accordingly, there are two general groups of intrusion detection mechanisms on computer network systems, namely, *network based intrusion detection* (NID) and *host based intrusion detection* (HID).

- *Network based intrusion detection.* NID is normally applied at the gateway of a network to monitor the real-time traffic which are transmitting across the network. NID examines the raw network packet and identifies the intrusion patterns from the packet header and the packet payload. The intrusion patterns can be as simple as an attempt to access a specific port or as complex as sequences of operations directed at multiple hosts over an arbitrary period of time.

NID has the following three strengths [8]: (1) NID monitors the network traffic for multiple hosts within local network at the same time; (2) NID correlates attacks against multiple hosts; (3) NID does not affect host performance.

However, NID has the following three major limitations [73]: (1) NID lacks the capability to analyse the encrypted network traffic; (2) NID faces challenges to exhaustively capture and examine the network traffic since the traffic bandwidth is continually increasing; and (3) NID conducts the detections based on the directly extracted traffic information without high reliability as the traffic information may be masqueraded or insufficient.

The intrusion detection system which applies NID is known as Network Based IDS (NBIDS). The widely-used NBIDS in the intrusion detection research field is Snort [2].

- *Host based intrusion detection.* HID is typically applied at the single host within computer network systems to monitor the events which are generated by both applications and users on the host. HID examines the host operating system audit data from host logs, especially the system log, event log and security log. When any of these log files change, HID makes a comparison between the new log entry and attack pattern to detect the intrusion.

  HID has the following three advantages [8]: (1) HID detects attacks that do not involve the network; (2) HID can analyse what an application is doing; (3) HID does not require additional hardware.

  Nevertheless, HID suffers from the following two main limitations [73]: (1) HID decreases the system performance due to the system resource and time consumed to capture the event; and (2) HID examines the intrusion activities on the single host, which means multiple detectors are required in the network to achieve efficient monitoring.

  The intrusion detection system which applies HID is known as Host Based IDS (HBIDS). HBIDS started in the early 1980s when networks were not prevalent, comprehensive and heterogeneous as today. However, in the current intrusion detection research field, there are only very limited research that still focus on HID (for example, [109, 130]) in comparison with NID.

In addition, the development of HBIDS has not been as successful as for NBIDS in the security industry [73].

2.1.3 Distributed Intrusion Detection

Irrespective of the categories described in Section 2.1.1 and Section 2.1.2, an IDS could be such that the detections of the intrusion are performed on a number of detectors proportional on the network that is being monitored. This is called a Distributed IDS (DIDS) [10, 123].

2.2 Typical Research of Intrusion Detection

2.2.1 Detection Techniques

In terms of detection accuracy, the ideal intrusion detection can instantaneously detect all of the intrusions with 100% detection rate and 0% false rate. Unfortunately, there are several practical problems, which will be presented in Section 2.3 in this chapter, that handicapped the satisfaction of this ideal detection accuracy. In order to get close to this goal, the typical computer network intrusion detection research focus on the proposal and implementation of new detection techniques to detect either known or novel attacks with high detection rate and low false rate [23, 37].

Many of the proposed detection techniques (for example, Bayesian network [42, 50], data mining [57, 58, 59], neural network [102]) are summarised in the literature [70, 86]. There are also several recently proposed detection techniques (for example, self-organizing map [14], support vector machines [15], maximum entropy estimation [38]). Note that normally the applied detection techniques are falling under the aforementioned main principles and classification.

2.2.2 Fundamental Theoretical Research

Besides the aforementioned typical intrusion detection research that focus on detection techniques, there are few but substantial research [23, 37, 62] that examine the fundamental theory of intrusion detection (system) to fill in the gap between practice and theory.

- *Modelling of Intrusion Detection Process.* The theoretical formalisation of intrusion detection process has been given in [23]. The modelled intrusion detection is defined as a set of algorithms, including: *representation algorithm*

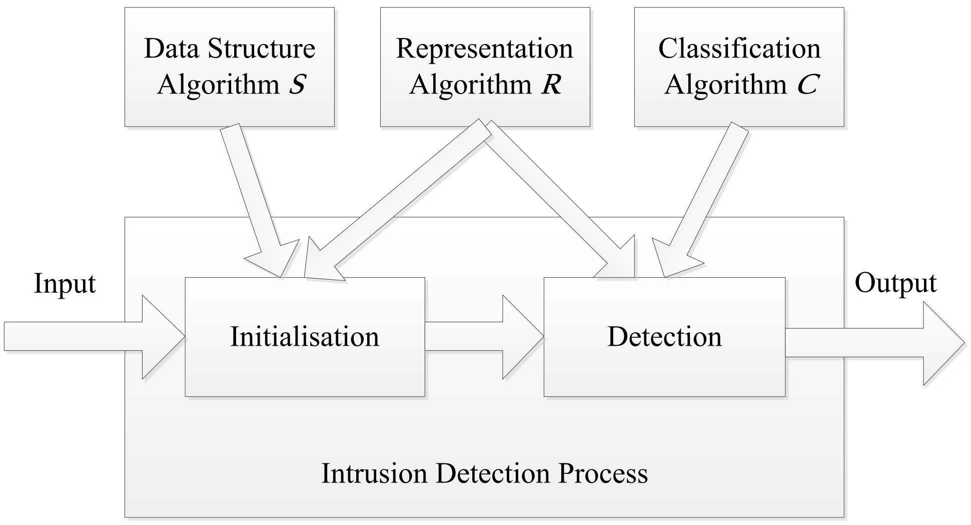
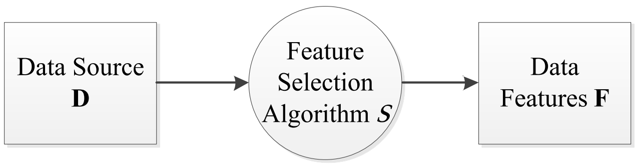

Data Structure Algorithm S
Representation Algorithm R
Classification Algorithm C
Input
Initialisation
Detection
Output
Intrusion Detection Process

Figure 2.2: Modelling of an Intrusion Detection Process

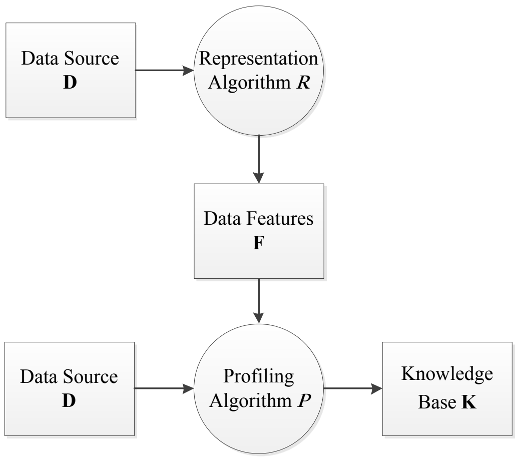
$\mathcal{R}$, *data structure algorithm* $\mathcal{S}$ and *classification algorithm* $\mathcal{C}$. The *representation algorithm* $\mathcal{R}$ contains any functions relating to how to represent data within the detection process, such as data filtering, feature selection, formatting, etc. The *data structure algorithm* $\mathcal{S}$ contains any functions relating to how to manipulate data during the detection process, such as data collection, aggregation, knowledge creation, etc. The *classification algorithm* $\mathcal{C}$ includes any detection principles.

Figure 2.2 illustrates the general intrusion detection process with the provided intrusion detection modelling from [23]. The constructed model divides the implementation of the intrusion detection into two phases: *initialization* and *detection*, which are shown as two bottom solid line boxes. Meanwhile the three modelled algorithms are represented as the top boxes. In the typical intrusion detection process, the detector analyses and extracts the relevant information from the coming traffic into the appropriate format, then, the detector classifies the generated data into either the normal or malicious. Therefore, $\mathcal{S}$ runs in the initialization phase and $\mathcal{C}$ applies in the detection phase, while $\mathcal{R}$ assists the process in both $\mathcal{S}$ and $\mathcal{C}$. Specifically, in the initialization phase, $\mathcal{S}$ uses $\mathcal{R}$ to process the input; in the detection phase, $\mathcal{C}$ uses $\mathcal{R}$ to process towards the detection output.

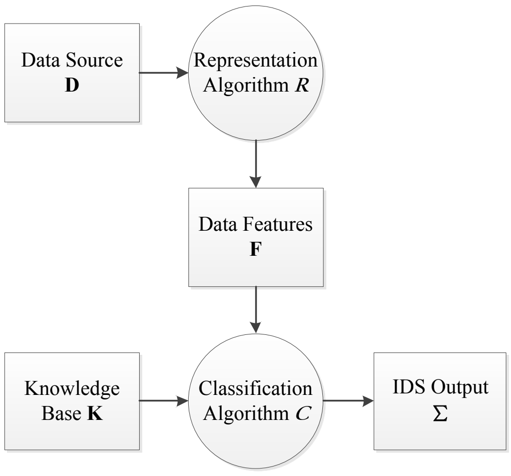
- *Modelling of Intrusion Detection System.* The theoretical model of IDS has been presented in [37] as an eight-tuple ($\mathbf{D}$, $\sum$, $\mathbf{F}$, $\mathbf{K}$, $\mathcal{S}$, $\mathcal{R}$, $\mathcal{P}$, $\mathcal{C}$). This IDS modelling considers three different procedures of the IDS from the system development till the system deployment, namely *feature selection procedure*; *training procedure*; and *detection procedure*. Figure 2.3 illustrates the modelled IDS procedures.



(a) Feature Selection Procedure



(b) Profiling Procedure



(c) Detection Procedure

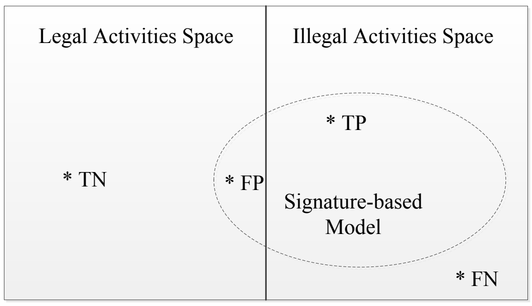
Figure 2.3: Procedure for the Modelling of Intrusion Detection Systems

In the eight-tuple, the first four elements represent a set of data structures, while, the last four elements represent a set of algorithms. For the four data structure elements, **D** represents the data source (for example, network traffic, system log) for IDS examination; $\sum$ represents a finite set of data states indicating whether the data unit is normal or anomalous; **F** represents the data characteristics (for example, network protocol, port number); and **K** represents the knowledge profile (for example, signatures, rules, statistical profiles). For the four algorithms, $\mathcal{S}$ represents the feature selection algorithm to either manually or automatically select and generate the features of the data during the developing process of IDS; $\mathcal{R}$ represents the data reduction and representation algorithm to automatically map the data source **D** to data features **F**; $\mathcal{P}$ represents the profiling algorithm to generate the profile knowledge base **K**; $\mathcal{C}$ represents the classification algorithm to map the features of given data to normal or anomalous data state.

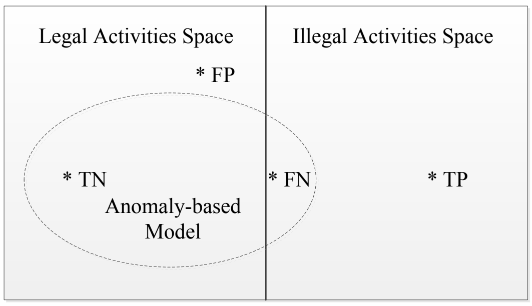
Figure 2.3(a) shows the modelling of the feature selection procedure. The typical feature selection procedure extracts the relevant data characteristics from **D** and generates **F** by applying $\mathcal{S}$. Figure 2.3(b) displays the modelling of the profiling procedure. The typical training procedure generates **K** from **D** by applying $\mathcal{R}$ and $\mathcal{P}$. The modelling of the detection procedure is shown in Figure 2.3(c). The typical detection procedure is to extract **F** from **D** by first $\mathcal{R}$, then, $\mathcal{C}$ conducts the detection and outputs $\sum$ according to **K**.

- The detection principles of intrusion detection have been examined in [62] to analyse the intrusion detection problems of model inaccuracy and model incompleteness as well as lack of the distinguishabilily in the features utilised.

  Figure 2.4 [62, 73] illustrates the theoretical activity space models for both *signature based intrusion detection* and *anomaly based intrusion detection*. It is clear that SID consists of illegal activities modelling as Figure 2.4(a), whereas AID consists of legal activities modelling as Figure 2.4(b).



(a) Signature-based Model



(b) Anomaly-based Model

Figure 2.4: Theoretical Activity Spaces Models

According to Figure 2.4, the intrusion detection faced challenges are *completeness* (the intrusion detection can detect all illegal activities) and *accuracy* (the intrusion detection can only detect illegal activities) in both SID and AID [62, 73]. The correctly detected activities within either *signature based model* or *anomaly based model*, that is, the detected illegal activity in *signature based model* and the detected legal activity in *anomaly based model* are known as *True Positive* (TP). While, the incorrectly detected activities within either *signature based model* or *anomaly based model*, that is, the detected legal activity in *signature based model* and the undetected illegal activity in *anomaly based model* are known as *false detection*. In SID, the incompleteness leads to FN, whereas inaccuracy leads to *False Positive* (FP). In AID, the incompleteness leads to FP, whereas inaccuracy leads to FN.

2.2.3 Intrusion Detection Evaluation

The utilisation of good evaluation approaches is crucial to attain an accurate and reliable evaluation of an IDS's detection capabilities and performance. From the view of intrusion detection evaluation, the IDS can be typically modelled as a black box, which receives data (for example, network packets, log file entries) from a certain source and has to determine if the input is an intrusion or not [8]. Generally, there are two main methods to evaluate intrusion detection performance:

(1) *Metrics Based Evaluation*; and (2) *Graphical Based Analysis*.

- *Metrics Based Evaluation.* Metrics based evaluation mechanism is fundamental but essential in evaluating an IDS. It assesses how many intrusions are detected correctly and how many are not. Note that, the ground truth, which is the reality of the examined data with the intrusion information, must be known in order to conduct the evaluation. Table 2.1 shows the classification of such evaluation metrics. Normally, the class of generated alarms is regarded as the *positive* class, and the class without alarm generation is regarded as the *negative* class. The definitions of common evaluation metrics [34] are given as follows.

Table 2.1: Attack Detection Classifications

	Attack Data	Normal Data
Alarm Generated	True Positive (TP)	False Positive (FP)
No Alarm Generated	False Negative (FN)	True Negative (TN)

Definition 1 ***True Positive.*** *A True Positive (TP) is an instance of the positive class which is labelled as positive. A TP indicates a piece of attack data which is correctly detected by the IDS.*

Definition 2 ***False Positive.*** *False Positive (FP) is an instance of the negative class which is labelled as positive. A FP indicates a piece of normal data which is wrongly labelled as an attack by the IDS.*

Definition 3 ***True Negative.*** *True Negative (TN) is an instance of the negative class which is labelled as negative. A TN indicates a piece of normal data which is correctly labelled as normal by the IDS.*

Definition 4 ***False Negative.*** *False Negative (FN) is an instance of the positive class which is labelled as negative. A FN indicates a piece of attack data which is not detected by the IDS and consequently evades it.*

Based on these four common metrics, an IDS can be evaluated in terms of *Detection Rate* (DR) [62], also called *True Positive Rate* or *Recall*; *False Alarm Rate* (FAR) [62], also called *False Positive Rate*; *Precision* [34]; and *Accuracy* [34]. Note that *True Positive Rate* and *False Positive Rate* are different from the TP and FP common metrics. "$|.|$" denotes the frequency of the metrics, for example, |TP| represents the number of TP been classified within detection process.

DR is defined as

$$\text{DR} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|} \tag{2.1}$$

and measures the ratio between the number of correctly classified attacks (detected alarms) and the total number of attacks.

FAR is defined as

$$\text{FAR} = \frac{|\text{FP}|}{|\text{TP}| + |\text{FP}|} \tag{2.2}$$

and measures the ratio between the number of incorrectly classified normal data and the total number of alarms.

Precision is defined as

$$\begin{aligned} \text{Precision} &= 1 - FAR \\ &= \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|} \end{aligned} \tag{2.3}$$

and measures the ratio between the number of correctly classified normal data and the total number of alarms.

Accuracy is defined as

$$\text{Accuracy} = \frac{|\text{TP}| + |\text{TN}|}{|\text{TP}| + |\text{FP}| + |\text{TN}| + |\text{FN}|} \tag{2.4}$$

and measures the ratio between the total correctly classified data, which includes both the attack data and normal data, against the summation of the *positive* class and the *negative* class.

In addition to the metrics mentioned above, there are other evaluation metrics used to investigate the intrusion detection's detection performance. Some of the metrics (for example, F-Score [105] in Equation (2.5)) can be directly measured by applying the four common metrics. Some metrics need extra information (for example, Bayesian detection rate [9] requires the *Base Rate value*) to assist the evaluation. While, some metrics require different information instead of the common metrics, for example *intrusion detection capability* [36], which is simply the ratio of the mutual information between IDS input and output to the entropy of the input. Furthermore, some metrics are proposed to measure other aspects of intrusion detection, for example, *fuzzy comprehensive evaluation based entropy weight coefficient* [106] considers fuzzy logic to examine IDS in functionality, expansibility, practicability and security aspects by measuring the truth degree of each aspect; *expected cost* [35] assesses the monetary investment of an IDS in a given IT security infrastructure.

$$\text{F-Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} \tag{2.5}$$

- *Graphical Based Analysis*. This offers the graphical evaluation capability to intrusion detection according to the plotted curves. Instead of the detection accuracy evaluation, the evaluation curves (for example, Receiver Operating Characteristic Curves (ROC) [34, 35], Intrusion Detection Operating Characteristic (IDOC) [20]) focus on the provision of additional evaluation tasks by applying some of the aforementioned metrics.

  ROC represents the detection probability at a given false alarm rate. By illustrating multiple ROC curves, it is easy to identify the total detection performance of intrusion detection approaches when curves do not cross. However, the limitation of ROC is that it cannot make comparison if the curves cross.

  Since some of the metrics (like Bayesian detection rate) require priori information to measure the performance with uncertainties such as the likelihood of an attack and the operational costs of intrusion detection, IDOC is proposed to represent detection probability at a Bayesian detection rate, also known as Positive Predictive Value (PPV). Hence, IDOC can evaluate intrusion detection with the model of adversary information.

2.3 Problems of Intrusion Detection

An ideal intrusion detection should be free of false alerts and thus detect attacks with 100% precision. It should avoid excessive response times and have minimal computational cost [127]. Though intrusion detection being globally studied for more than two decades since Denning's intrusion detection prototype [28], there are unsolved difficulties and limitations that handicap the achievement of perfect intrusion detection.

2.3.1 Difficulties

Many novel intrusion detection techniques have been proposed and employed for better performance in terms of high detection rate and low false alarm rate, which are closer to the ideal conditions, in the experimental environments. However, no perfect intrusion detection practically exists due to the following known difficulties.

- *Rapid augmentation of computer network systems.* Current computer networks are becoming more comprehensive, as various emerging digital devices (for example, laptop, iPad, smart phone) are connected into a heterogeneous system. Thus, it is difficult for a single IDS to precisely detect all kinds of known vulnerabilities and known intrusions (as well as deviations of any intrusions) for different IDS-deployed digital devices. Any imprecisely developed detection signatures or statistical profiles will lead to inaccurate detection with FP and FN.

- *Rapid emergence of new vulnerabilities and intrusions.* Within the heterogeneous networks, there may exist unknown vulnerabilities on either recent devices or old products. Vendor unidentified security flaws, which are new vulnerabilities and intrusions (that is, zero-day attack, which compromises publicly unreported vulnerabilities that are exploited by the adversary.), may exist on the latest computer network products within either hardware drivers or application software. Normally, the IDS updates the intrusion knowledge after adversaries compromise or security analysts identify the flaws. Thus, intrusion detection lacks the capability to detect new vulnerabilities and unknown intrusions leading to inaccurate detection with FN.

2.3.2 Limitations

The limitations of intrusion detection could be solved by the researchers compared with the difficulties of intrusion detection. Usually, the limitations are weaknesses of intrusion detection mechanisms or deployed systems. Some of the known limitations are as follows.

- *Low-level detection with independent intrusion reporting* [79, 81]. No matter what detection principle has been applied and what audit data has been examined in intrusion detection, the deployed intrusion detection mechanism or IDS normally detects the intrusion entirely from *low-level raw information* (for example, raw network traffic, host log file record). Though low-level information contains the potential implications for intrusion possibilities, it is difficult for the intrusion detector to directly report the high-level alarm abstraction (that is, the system has been compromised). Typically, IDS implements the detection by ignoring the context of the overall computer network system and its most critical resources, like the actual network configuration and vulnerabilities. Take Snort for example, it only examines every single raw network packet's header and payload information to determine the intrusion.

In addition, the current computer network intrusions are not only increasing in numbers, but have also become more sophisticated. Instead of the single step intrusion, the adversary proposes a detailed and systematic plan before conduct the intrusion, then, compromises the system by implementing multi-step intrusions logically. However, current intrusion detection lacks the capability to distinguish the hidden logic and relation (that is, causal relation, which represents that the achievement of one intrusion step causes the occurrence of another intrusion step) from the various detected low-level intrusions.

Hence, an additional problem is *independent intrusion reporting.* As one low-level detected intrusion causes one alarm, IDS may generate repetitive alarms within one specified period, since the adversary may continually implement the same type of intrusion for information gathering or other intrusion purposes. For example, DDoS attacks contain huge numbers of malicious packets within attack traffic. Snort raises alarms against every malicious packet, though most of the generated alarms may be quite redundant. Also, Snort generated alarms can not tell the precise progress of multi-step intrusions. Meanwhile, the system administrator is overwhelmed by the flood of redundant alarms.

- *Self-existed vulnerabilities on implemented IDS.* Since IDS is usually a set of computer programs, there are some possibly existed security flaws or vulnerabilities on the IDS due to the poor system design and implementation. Hence, if an adversary can somehow prevent or terminate an IDS by compromising the explored vulnerabilities, the whole IDS-monitored computer network system is left without protection.

2.4 Extensions of Intrusion Detection

Since intrusion detection (system) only deals with discovery of intrusions on computer networks, there are several additional intrusion detection based extensions.

2.4.1 Intrusion Prevention

The process of intrusion detection stops with generating alarms and relies on *manual* responses by the administrator. The resulting delays between the detection and response may range from minutes to months. Therefore, there is a need to respond to intrusion automatically.

The system that applies the intrusion prevention mechanism is known as an Intrusion Prevention System (IPS). An IPS provides the intrusion detection capabilities additionally reacting in real-time to stop or prevent undesired access, malicious content and inappropriate transactions [16]. IPS can also be divided into network based IPS (NBIPS) and host based IPS (HBIPS) according to the audit data type [16, 125, 128]. NBIPS operates inside the network to monitor all network traffic for malicious code or attacks. The main prevention functions are traffic blockage of potential intrusion and traffic termination of current intrusion [7]. When an intrusion has been detected, NBIPS drops the malicious traffic while continually allowing legal traffic to pass. HBIPS operates on the individual host where system and user actions are inspected for intrusions.

2.4.2 Intrusion Response

Intrusion response is a further extension to intrusion detection and intrusion prevention. Intrusion response provides the response functions with further analysis to minimize impact from the intrusion [7]. The response process can be typically classified into two categories: *active response* and *passive response*. According to [7], an *active response* fights against an intrusion in order to minimize the impact from intrusion to a victim, whereas a *passive response* informed the occurrence of an intrusion to other parties and relies upon them to take further action. Active response can be further subdivided into a *proactive response*, which controls the potential intrusion before it happens, and a *reactive response*, which reports and responses after an intrusion.

2.5 Intrusion Analysis

Intrusion analysis refers to the analysis of intrusion detection and forensics to identify attack traces from possibly large number of audit data [85]. In this thesis, the intrusion analysis has only been applied for the intrusion detection aspect. Typically, intrusion detection concentrates on the detection processing of any adversary intended and executed intrusions. However, there is a need to analyse the intrusion detection for the additional information beyond the low-level detection before and during, even after the detection process. The intrusion analysis process can be classified into two main categories according to the time of analysis: (1) *intrusion threat and situation analysis*, which take place before the end of intrusion detection; and (2) *intrusion detection uncertainty analysis*, which is conducted during intrusion detection process.

2.5.1 Intrusion Threat and Situation Analysis

Intrusion threat analysis and intrusion situation analysis investigate any additional intrusion information (for example, adversary's prerequisite intrusion conditions, intrusion progress within the multi-step intrusion) before and during the intrusion detection process.

- *Intrusion threat analysis.* Intrusion threat analysis mainly investigates the preconditions of intrusions by taking the standpoint of the adversary before intrusion detection takes place. In order to implement a successful intrusion, the adversary usually needs to achieve the following three aspects [39]: (1) *capability* (the ability to execute an intrusion); (2) *opportunity* (the environment permits the intrusion to happen); and (3) *intension* (the adversary must have a reason for the intrusion).

  An adversary's *capability* depends not only on the types of intrusions that he/she can implement, but also the ability to execute these intrusions efficiently [30]. The sophisticated adversary, who has high level of intrusion capability, usually has a detailed intrusion plan before the intrusion and adapted methodology to treat problems identified during the intrusion process, even tricking the intrusion countermeasures (for example, botnet usage, malicious intrusion packet masquerade) and cleaning the intrusion trials to prevent the potential forensic analysis. An adversary's *opportunity* implies the vulnerable and exposed entities on the computer network system that can be accessed given the current progress of attack [30]. The *opportunity* information contains the physical connection between the explored vulnerabilities within entities of the network system. These compromised entities sometimes are simply the stepping stones to the ultimate victim or intrusion goals. An adversary's *intentions* are the desired effects that the adversary wants to achieve [87]. Some typical intrusion goals are the satisfaction of curiosity, confidential information theft and hostile intrusion.

Intrusion threat analysis may partially provide intrusion detection with the adversary's *opportunity* information. However, as per the previously mentioned limitations of existing intrusion detection in Section 2.3.2, the detection mechanisms usually concentrate on low-level detection of the adversary launched intrusions. They seldom combine the potential *opportunity* information into the detection process. Besides the *opportunity* information, it is extremely difficult for intrusion detection to identify *capability* and *intention* information due to both *capability* and *intention* cannot be objectively measured or identified.

- *Intrusion situation analysis.* Intrusion situation analysis analyses the progress of intrusion from the point of detection during the intrusion detection process. This analysis usually needs the assistance of the priori constructed model, which constructs the formal model of intrusion situation measurement and supports the general process of using and analysing intrusion alerts collected from intrusion detection detectors. Attack graph [6, 40, 41, 43, 99, 103] and attack tree [25, 56, 107, 108] are main stream graph-based modelling approaches that provide the security analysis to computer network systems. Either attack graph or attack tree illustrates possible multi-stage intrusions for the targeted network, typically by presenting the logical causality relations between intrusion steps and physical configuration settings. More information about attack graph and attack tree will be described in Chapter 3.

  Through the run time intrusion situation analysis against the model, the derived information from on-going situation may be additionally used to determine appropriate actions for proactive defense [39]. In addition, the casual relations between intrusion steps allow pinpointing where the current intrusion situation is located within the sophisticated multi-step attack process.

  The generation of network security situation [33] can be divided into three steps: (1) to obtain attack patterns from the sample data set by applying knowledge discovery methods; (2) to transform the measured patterns into the correlation rules; and (3) to conduct the network security situation assessment with real-time data set. However, there are several main difficulties [33] that handicap the implementation of intrusion situation analysis: (1) there are lots of detection uncertainties because the false detection rate may be high from the applied IDS; (2) the relations between the alerts, which are generated from the large scale network attacks, are complex and difficult to be determined.

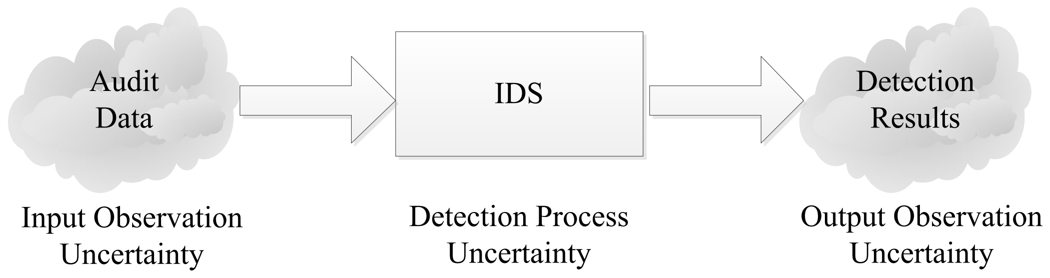

Audit Data
IDS
Detection Results
Input Observation Uncertainty
Detection Process Uncertainty
Output Observation Uncertainty

Figure 2.5: Uncertainties within Intrusion Detection

Therefore, one could conclude that the modelling of intrusion opportunity information and the multi-step intrusion with hidden logical relation can enable the analysis function before and during the intrusion detection process to overcome the limitation of low-level detection.

2.5.2 Intrusion Detection Uncertainty Analysis

Uncertainty is an innate feature of intrusion analysis due to the limited information obtained by system security monitoring tools (that is, IDS, firewall, system logs) [85]. Within the scope of intrusion detection, the possible uncertainty analysis deals with any uncertain intrusion logical relations and uncertain intrusion information, such as, trustworthiness of the detection result and trustworthiness of the intrusion source, during the intrusion detection process.

Figure 2.5 shows three uncertainties during the whole intrusion detection process: *input observation uncertainty*, *detection process uncertainty*, and *output observation uncertainty*.

- *Input observation uncertainty.* This uncertainty issue is related to the observed input data by the detector. Normally, the intrusion detector simply extracts the relevant information from the requested fields of the observed audit data. However, the professional adversary may fake the audit data. The typical process is to masquerade the source IP address of malicious network packet. In addition, they remotely control the compromised bots to conduct further intrusions on the primary victim without revealing his/her real location.

  Thus, the typical input observation uncertainty means the uncertainty of whether the incoming audit data is generated by the claimed source or not, where the adversaries are, and what intrusion options he/she has made to carrying out the intrusion [61].

- *Detection process uncertainty.* This uncertainty represents uncertainty issues during the detection process by the IDS. There are two possible uncertainties:

(1) *classification uncertainty*; and (2) *alert correlation uncertainty.*

  * *Classification uncertainty* is related to how the IDS can ascertain that the observed symptoms can correctly lead to the alarm generation. IDS typically conducts the detection process by matching the captured or partial predefined signatures or statistical profiles. Thus, the obtained information has varying levels of certainty to raise the alarm. In Snort, a rule defines a set of malicious information as a specified attack signature. Some rules have specific information (for example, specified port number in port field, unique malicious packet payload content in content field) on every rule field, while, some rules have less specific information (for example, any in IP address field, any in port field) instead of fixed malicious information on some rule fields. But, it is possible for a single malicious packet to be matched with several different Snort rules. In that case, it is difficult to ascertain that the generated Snort alerts are correct.
  * *Alert correlation uncertainty.* In the alert correlation process [24, 74, 105, 112, 113], the uncertainty is related to how to explicitly model or determine the certainty level in the correlation process [85]. The typical scenario of alert correlation uncertainty is within DIDS. Since the multiple intrusion detectors may generate different detection results on the same piece of audit data, the question is how the analyser can deal with different detection results and fuse them into the a single output. Dempster-Shafer theory [21, 123] has been applied to deal with this issue.

- *Output observation uncertainty.* This uncertainty can be regarded as IDS detection performance analysis in terms of *detection accuracy.* Once the detector obtains any results, it lacks an automatic capability to verify the generated results as being a true positive alarm. It is difficult even for the human security expert to identify the detection accuracy without the ground truth of the examined traffic.

2.6 Summary

This chapter has described the background of intrusion detection, which is a mechanism to identify unauthorized use, misuse and abuse of computer network systems. Firstly, the intrusion detection classifications was described according to different detection principles and different data types. Next, the typical intrusion detection research were presented including detection techniques, fundamental theoretical research and the evaluation approaches as the concrete research results in recent years. Then, the difficulties and the limitations to intrusion detection research were discussed to motivate why intrusion detection is a challenging but still attractive research area. After that, intrusion prevention and intrusion response were briefly described as subsequent stages after the intrusion detection. Finally, the existing intrusion analysis approaches were identified and reviewed.

The following chapter reviews attack graph and attack tree modelling techniques as our research's second background part.

Chapter 3

Attack Modelling Approaches

This chapter reviews the recent attack modelling research. As we need to build an attack model to facilitate the corresponding intrusion detection process, it is necessary for us to propose a specialised attack modelling approach by identifying and extending the current and known attack modelling approaches. In addition, it is essential for us to examine the known node connectors, node metrics and aggregations for the relevant modelling and computation. Moreover, since the attack graph modelling technique has been applied to conduct any intrusion detection related research already, we additionally investigate the attack graph modelling technique. The presented information in this chapter provides the foundation for our attack tree modelling research. Thus, we may propose our attack tree modelling technique by applying or extending the selected node connectors, node metrics and aggregations to generate attack tree model for the intrusion detection in the following chapters of this thesis.

In the multi-step intrusion process, the adversary usually uses a compromised victim machine as a stepping stone to launch another exploit on a new victim machine and then, repeats the process until an ultimate goal has been achieved. This style is normally known as a chain style [6]. Attack chaining is an iterative process that supports automated vulnerability analysis by enumerating vulnerability interactions [26].

In order to modelling these multi-step attacks, security researchers often organise the chains of exploits into graphs or trees by applying attack graph modelling techniques or attack tree modelling techniques. Although the precise definitions of attack graph or attack tree vary by the researcher, it is useful to think of an attack tree as a structure in which each possible attack chain ends in either a leaf node or a root node that satisfies the intrusion goal, and an attack graph as a consolidation of the attack tree in which some or all common states are merged [6].

The attack graph modelling technique has usually been applied to model and analyse the given computer network system security with system information (for example, specific vulnerabilities on various hosts, connectivity between hosts, adversary access privileges on various hosts) in conjunction with the intrusion process. While attack tree modelling technique, which is a specialised attack graph with a tree structure, typically has been applied to model and analyse system security without additional hybrid system information.

An attack path in either attack graph or attack tree represents a series of intrusions (also known as *atomic attacks*) corresponding to a sequence of state transitions culminating in the adversary achieving his/her goal. The entire attack graph or attack tree is thus a representation of all the known ways that the adversary can succeed to compromise the target victim [99]. Hence, the resulting set of all possible attack paths is a predictive attack roadmap [82].

In this chapter, Section 3.1 and Section 3.2 describe the attack graph and the attack tree with the following aspects: representations, generation approaches, and the combination with intrusion detection, respectively. Section 3.3 investigates the node connector extensions besides the conventional conjunctive and disjunctive. Section 3.4 classifies and describes the metrics and attributes on the modelled graph/tree nodes. Then, the possible metric aggregation approaches and the attack resistance aggregation are represented in Section 3.5. Finally, the summary of this chapter is given in Section 3.6.

3.1 Attack Graph Modelling Approach

The attack graph modelling technique provides capabilities for the network security researcher or the system analyst to model, analyse and assess the security among any particularly given networks. Based on the constructed graph model, both the analyst already known and the analyst unexplored attack information may be explicitly illustrated. This attack information usually includes the relationship between network components and potential attacks and also their consequences in a particular context. Such a context typically considers the physical or logical connectivity between network devices, vulnerabilities on each single network device. Thus, attack graphs allow the network security researcher or the system analyst to assess the true vulnerability of critical network resource, and to understand how vulnerabilities in individual network components contribute to the overall vulnerability [77, 121].

Since attack graphs try to model various possible intrusion paths on the targeted system, the constructed attack graph is termed as *exhaustive* if it covers all possible attacks, and is termed as *succinct* if it contains only those network states from which the adversary can achieve his/her intrusion goal [99].

3.1.1 Representations of Attack Graph

There are many different attack graph literature and many representations [43, 89, 99, 119, 120, 121]. However, the core idea of attack graph [40] remains the same: an attack graph shows the ways an adversary could compromise a network or host. In addition, the fundamental elements of an attack graph remain the same: the node usually represents the possible state and the edge usually represents the possible state transition. Several typical representations of attack graph are now discussed.

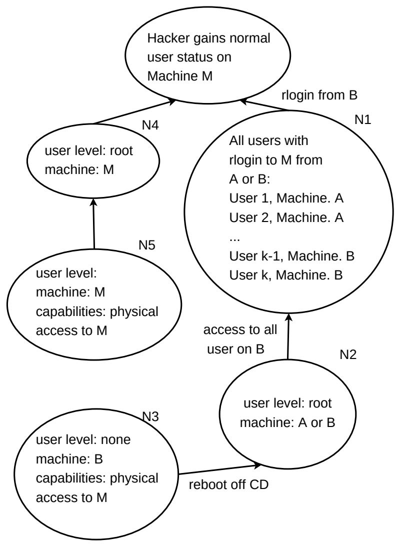
Common Attack Graph Representations

The usual attack graphs [43, 89, 99] apply a geometrical shape to represent the node. Within the node, there is various information, such as, intrusion information to describe node state, which can be filled in according the modelling requirements.

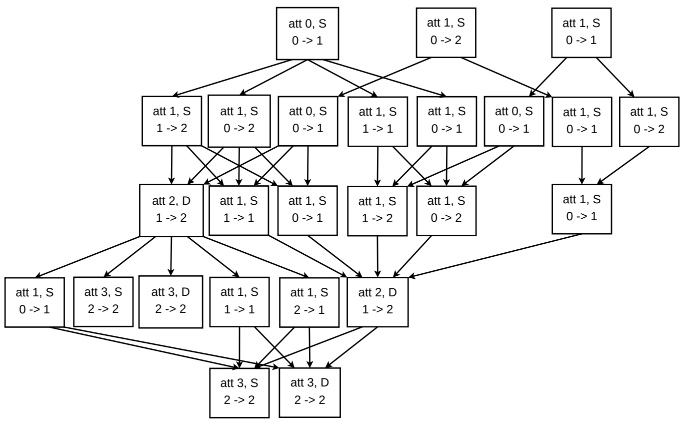
The node in the attack graph [89] represents the state as the combination of a physical host, the corresponding user access level, and effects of the attack so far. An edge is a change of state caused by a single malicious attack or legitimate action taken by the adversary, or a single action taken by an unwitting victim from a piece of malicious code. Figure 3.1(a) illustrates a sample attack graph diagram from [89]. The top node is the ultimate intrusion goal on the primary victim machine. While the two connecting branches are the attack paths from another two machines which are physically linking to the primary victim machine. Each node contains any of the following node attributes information: user level as the possible user privilege; host machine ID; vulnerability that represents the changes to the original configuration caused by the intrusion; capability that represents the physically possible intrusion behaviors.

The node in attack graph [43, 99] contains the state information with several attack information (that is, pre-defined attack ID, the detectable flag of IDS, and attack source and target hosts). An edge corresponds to an atomic attack whose preconditions are satisfied in the source node and postconditions in the destination node. Figure 3.1(b) shows the generated example attack graph from [99]. The first left node on the top row "*att* 0" represents the researcher defined attack index number, which corresponds to the intrusion step to be attempted next; "*S*" indicates the detection capability of IDS to the corresponding attack, which is detectable or stealthy; "0 −> 1" represents the intrusion source "0" to the intrusion destination "1", where "0" and "1" are defined host numbers.

Though Figure 3.1(a) and Figure 3.1(b) look dissimilar (that is, node shape (circle in Figure 3.1(a) and square in Figure 3.1(b)), graph structure (Figure 3.1(a) had modelled the graph from the standpoint of host with physical connection,


Hacker gains normal user status on Machine M
rlogin from B
N4
N1
user level: root machine: M
All users with rlogin to M from A or B: User 1, Machine. A User 2, Machine. A ... User k-1, Machine. B User k, Machine. B
N5
user level: machine: M capabilities: physical access to M
access to all user on B
N2
user level: root machine: A or B
N3
user level: none machine: B capabilities: physical access to M
reboot off CD

(a) Attack Graph Example [89]



(b) Attack Graph Example [99]

Figure 3.1: Two examples of a Common Attack Graph Representations

whereas Figure 3.1(b) had built the graph from the standpoint of attack process), node attribute types), both of them have applied the single node type to represent the state in addition with the defined node attributes.

Jajodia's Attack Graph Representation

Besides the above mentioned representations with geometrical shapes, Jajodia, who is the world leading attack graph researcher, had proposed his unique attack graph representation approach [119, 120, 121] compared with the common approaches [43, 89, 99].

In Jajodia's attack graphs [119, 120, 121], the system states have been defined as *conditions* in plaintext instead of any geometrical notations. Additionally, the commonly represented edge transition splits into two edge sections by an additional oval in the middle. The ovals are shown as *exploits*, which are adversary utilised vulnerabilities between connected host machines. Moreover, the split directed edge represents the relation and connects *conditions* with *exploits*. An edge from a *condition* to an *exploit* denotes the required relation, which means the *exploit* cannot be executed unless the *condition* is satisfied; whereas an edge from an *exploit* to a *condition* denotes the implication relation, which means that the adversary executes the *exploit* to achieve the *condition*.

Figure 3.2 illustrates an attack graph example from [119]. The top node $user(0)$ indicates the root node and the bottom node $root(2)$ indicates the leaf node. Compared with the aforementioned attack graphs, the *condition* node within this representation provides less state information. The only depicted information are

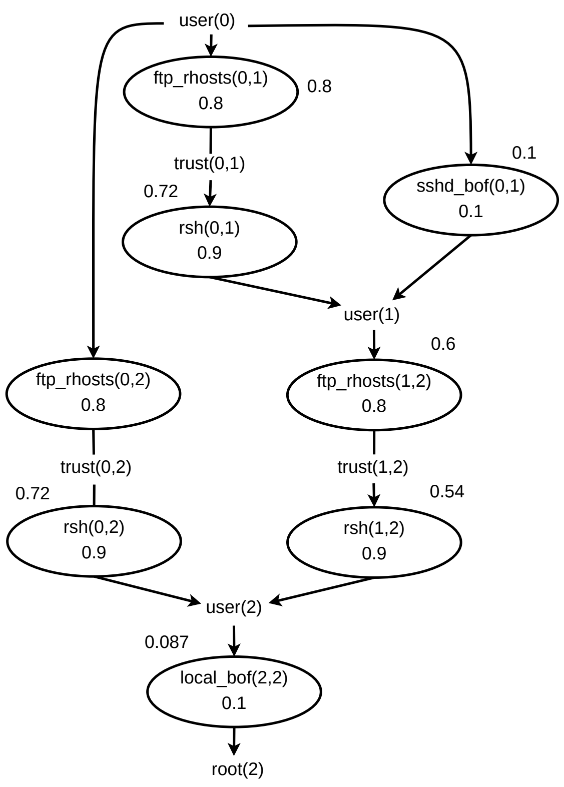


Figure 3.2: An example of Jajodia's Attack Graph Representation

the user level at any particular machine (for example $user(0)$ represents the user level at host machine 0, $root(2)$ represents the root level at host machine 2) and trust relationship between two physical hosts (for example $trust(0,1)$ represents the established trust relationship from host machine 0 to host machine 1). However, the additional *exploit* oval on the state transition shows the vulnerability between two host machines (for example $rsh(0,1)$ represents a remote shell login from host machine 0 to host machine 1). Furthermore, the probability attributes are assigned to the *condition* and the *exploit* to analyse the likelihood of successful intrusion from the adversary.

Generally, no matter what kind of attack graph representation had been proposed or applied, the core idea of attack graphs is essentially the same, that is, for the graphical illustrations of how an adversary can compromise a network or host step by step.

3.1.2 Generation of Attack Graph

Specialised Attack Graph Generation

The old fashioned attack graph generation method was that security experts (like Red Team [100]) manually drew the graph, which is tedious, error-prone and impractical for large network [99]. In recent years, there have been several advanced attack graph generation approaches [41, 89, 99, 100, 103] been proposed to generate attack graph automatically. The automated generation process can be mainly classified into following two steps: (1) model the target network; and (2) produce an attack graph for the target network. In the former step, the analyst identifies and collects the fundamental information of a targeted network (for example, network topology, host vulnerability) as the input source for the automated attack graph generation. In the latter step, the developed graph generator processes the input source and displays the constructed graph on a graphical user interface (GUI).

Since the incomplete information or the wrong information may lead to inaccurate description of the target network, the fundamental information identification and collection is the critical process in attack graph generation. In order to build a complete and correct attack graph, the analyst tries to collect various information from the network. The common information contains the following aspects: *network information*, *host information*, *vulnerability* and *attack library*. Different researchers also have their own extra focuse (*adversary attack capability* (novice or expert) [89], *IDS detection capability* (detectable attack or stealthy attack) [99, 100], *adversary privilege level on host* (none, user, root) [99, 100]). *Network information* usually includes the physical network topology, the network connectivity between hosts and ports, whereas *host information* usually includes the host type, operating system version of host, enabled service on the host. *Vulnerability* lists the measured vulnerabilities on host. *Attack library* stores the knowledge of known attacks. The abstracted attack graph generation process is illustrated as Figure 3.3.

The way to obtain this input information is another problem that affects the quality of attack graph generation. As the analyst may manually specify all of the necessary information, there is no guarantee about the correctness and completeness during the manually analysis, especially with large and comprehensive networks. One direction to address this problem is to apply an automatical information gathering tools, such as Nessus vulnerability scanner [80, 82]. With the assistance of Nessus, which generates the vulnerability report for each host and the host connectivity in network, the Topological Vulnerability Analysis (TVA) tool [41] can automatically generate attack graphs for the large network. Similarly, Network Security and Planning Architecture [65] provides the attack graph generation capability with up to 50,000 hosts with the assistance of Nessus. Besides Nessus, more scanning tools (like Nmap [22]) obtain the detailed system information about host and vulnerabilities, then integrate the scan outputs from the multiple scanners into the merged information with the unified format to fa-

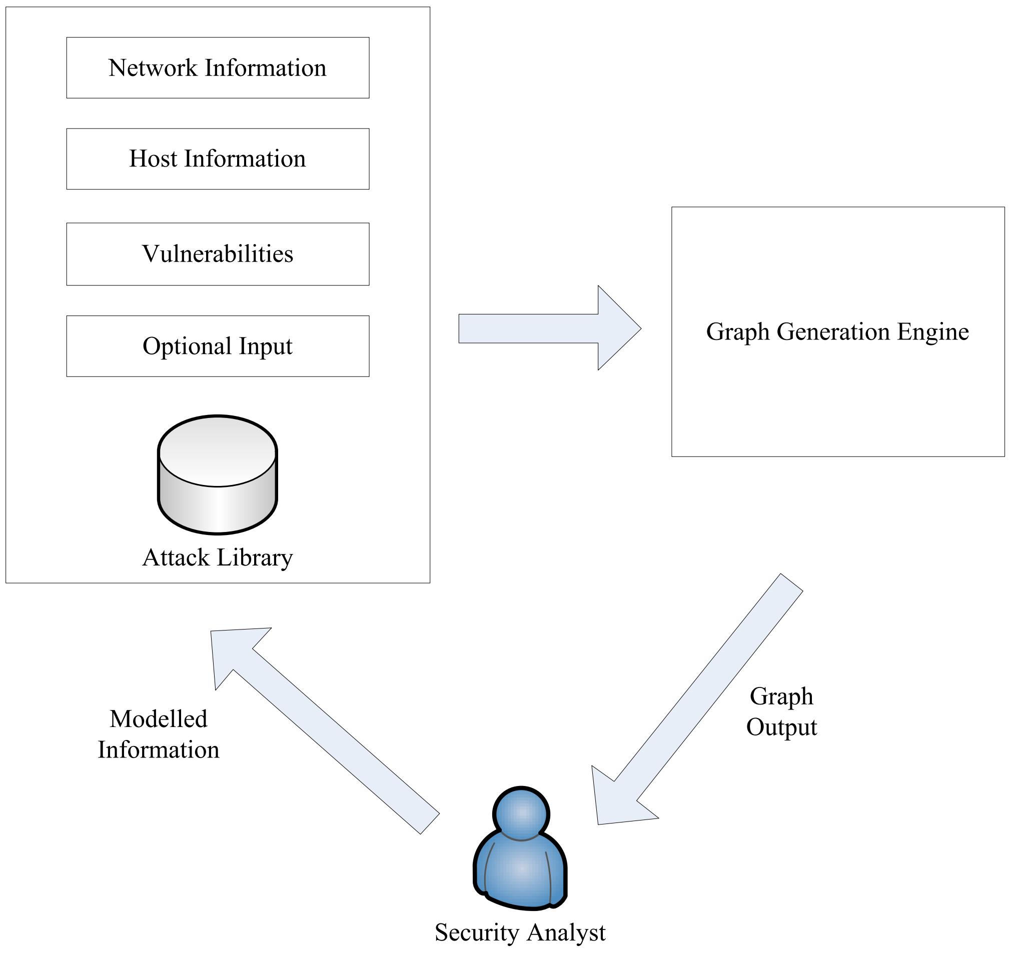


Figure 3.3: Abstracted Attack Graph Generation Process

cilitate the attack graph generation.

In terms of automatically gathering information, *vulnerability analysis* [6, 41, 65, 89, 104] is one of the foundations of attack graph modelling. This topic is beyond the scope of my research.

Other Attack Graph Generations

Besides the above specialised attack graph generation approaches and tools, some security researchers have applied other methods (for example, data mining [60, 63]) to generate the attach graph. However, since the goals of attack graph generation may be different, the differences between graph representations do occur.

In the data mining approach [60, 63], instead of vulnerability analysis against the targeted network, the researchers target on the generation and the probability computation of attack scenarios in the attack graph, and apply the constructed attack graph to predict the next attack step. Through the data mining approach, the researchers mine the associations between the attack classes according to the off-line IDS analysis of captured dataset, which is the only attack graph generation input source. Hence, in contrast with the generated attack graphs by the specialised aforementioned attack graph tools, the representation of generated nodes lack detailed host information and network information.

3.1.3 Attack Graph with Intrusion Detection (System)

The main goal of attack graphs is to analyse the security of a target network, whereas the main goal of intrusion detection is to identify the intrusion by the adversary to a target network. Since intrusion detection by IDS typically explores the attacks according to the known vulnerability signatures or attack rules from the vulnerability analysis, thus, attack graph can serve as the basis of intrusion detection for the analyst and IDS.

Refer to IDS, two following principles [43] have been pointed out about IDS based on attack graph: (1) the generated alerts from an IDS can match individual alerts to attack edges in the graph; and (2) the successive matched alerts to individual paths in the attack graphs dramatically increases the likelihood that the network is under attack. Hence, attack graph allows the IDS to predict attacker goals, aggregate alarms to reduce the volume of alert information to be analysed, and reduce the false alarm rates [43].

Fundamental Intrusion Detection Analysis

A generated attack graph describes all likely attacks and possible attack paths against the targeted network. As mentioned in the above paragraphs, the attack graph has been generated based on network and host information together with the vulnerabilities in the network or hosts. Thus, the attack graph analyses the vulnerabilities and potential intrusions by the analyst for intrusion detection. Usually, the attack graph assists the analysis of intrusion detection to examine the intrusion behaviour from the adversary within a given network, instead of providing the foundation for detection process from IDS on the given network.

Security analysts [43, 99] use attack graphs for intrusion detection analysis, which has targeted on the generation of all attack paths and demonstration of how the adversary can compromise the target without detection by the IDS. According to the provided network example, an attack graph model with finite states has been constructed. A state in the constructed model represents the state of the system between atomic attacks, whereas a state transition corresponds to a single atomic attack by the adversary. The main analysis steps including: (1) *State*

Identification; (2) *Connectivity Determination*; and (3) *Transition Analysis.*

The *State Identification* step identifies the finite states of the model. Each state contains the following three components: the fact of network; the privilege of intruder; and the detection capability of IDS. For the fact of network, the analyst determines host services, host vulnerabilities, connectivity and remote login trust relation between hosts. For the privilege of intruder, the analyst determines the level of privilege that the intruder has on each host. For the detection capability of IDS, the analyst determines the stealthy or detectable attack for IDS. The *Connectivity Determination* step determines the connection availability and trust relation between hosts. The *Transition Analysis* step models the atomic attack with the four following fields: adversary preconditions; network preconditions; intruder effects; and network effects. Referring to the adversary's threat assessment, the first two conditions define the adversary's opportunity, whereas the third condition represents the adversary's intention.

In the constructed attack graph, the node contains four elements: attack type; source host; target host; and attack strain indicator. The root nodes represent the initial intrusion states from the adversary's source host. While the leaf nodes represent the achieved intrusion states on the compromised victim host. Any path in the graph from a root node to a leaf node illustrates a sequence of atomic attacks that the intruder can employ to achieve the ultimate goal.

Advanced Implementation for Intrusion Detection

Besides the intrusion detection analysis of the targeted network, the attack graph had been applied to assist other intrusion detection related research.

In [82], the attack graph assists the optimal placement of IDS on the network to cover all critical paths with minimal cost. In [95], the attack graph has been integrated into the IDS management system to improve the alert and correlation quality. In [60, 63], the attack graph assists the analyst to predict the next attack step from the adversary.

Intrusion Uncertainty Analysis in Attack Graph

In the typical security analysis with attack graph modelling technique, the logical relations are usually regarded as *deterministic*: the successful intrusions by adversary will certainly happen in their worst forms as long as all the prerequisites are satisfied, and no intrusions will happen if such conditions do not hold [61]. However, such logical causalities encoded in a deterministic attack graph do not precisely describe the process of real-time security events and nor do that exactly identify the nature of real-time security events. Usually, the implemented cyber intrusions are not 100% guaranteed to exactly obey the modelled intrusion paths on the attack graph due to the incompleteness of attack graph construction with missed known vulnerabilities or any unknown vulnerabilities compromised by the adversary. Thus, there may be any uncertainties from the nature of exploits [61].

Incorporating probabilistic behaviour into attack graphs [43] may assist to address the uncertainty of attack graph. The Bayesian network had been suggested to encode the uncertain nature into the conditional probability table on graph nodes [61]. However, since it is difficult to examine the ground truths of real attack traces, the unaddressed problem is how to set and maintain the statistical parameters in the practical security analysis.

3.2 Attack Tree Modelling Approach

The attack tree research area has seen relatively moderate developed in the past 10 years since Schneier's work [97]. Therefore, there is still much potential research to forge attack tree into new discoveries and developments [32]. In the computer network security, the attack tree modelling technique was informally introduced to model intrusions against computer network systems in [97]. Generally, the attack tree provides a formal, methodical way to describe the security of a system [97]. From the view of intrusion by the adversary, an attack tree shows the progression of exploits to the top-level goal [26]. An alternative formalisation [71] is given in a denotational semantical way to study not only the equivalence of attack tree transformation, but also the attribute and the projection of an attack tree. Generally, the attack tree modelling technique has a more generic and structured manner [71, 110]. Construction of an attack tree depends on the expertise of the analyst. Any designing errors result in a flawed attack tree that can lead to incorrect analysis.

3.2.1 Differences Between Attack Tree and Attack Graph

Though both attack tree and attack graph are graph-based security modelling and analysis techniques, and attack tree can be seen as a special case of attack graph with tree like structure, there are several main differences in terms of *root-leaf relation*, *standpoint of modelling*, and *functional application*.

- Root-leaf relation. The root-leaf relation describes the relationships between the root and the leaf in terms of quantity by borrowing the concept of entity relations from relational database.

In a typical attack graph without tree-like structure, there are usually multi-root nodes and multi-leaf nodes. Any of the root nodes within the attack graph usually have more than one attack paths which are leading to different leaf nodes. It is possible to conclude that one root node has $1 : n$ relation to multi-leaf nodes. Hence, an attack graph typically has $n : n$ relations between multi-root nodes and multi-leaf nodes.

In a typical attack tree, there is only one single root node and multi-leaf nodes, whereas all of the leaf nodes have their own attack path with the root node. Thus, the attack tree typically has $1 : n$ relation between the single root node and multi-leaf nodes.

- Standpoint of modelling. Attack graph and attack tree have significant modelling differences in terms of standpoint. In attack graph, the modelling is conducted by considering the possible intrusions, and especially with host and system information. Some attack graphs [89] apply the physical machine connectivity as the fundamental structure. Other attack graphs [43, 99, 119, 120, 121] embed the communication between source and destination as node attribute. However, attack tree ignores any additional system information by purely concentrating on the modelling of intrusion techniques and the process.

- Functional utilisations. The main functionalities of both attack graph and attack tree modelling techniques are model and analyse the security of a given computer network system.

  In terms of intrusion detection, the attack graph is more suitable to implement the intrusion analysis of the security vulnerabilities on the given network before the intrusion detection process. The main reason is because the generated attack graph is often unique and only representable according to the current configuration of the given network. Even if the generated graph can facilitate the intrusion detection, it is workable specific to that given network.

  However, although attack tree has similar modelling and analysing functions as attack graph, attack tree is not only available to conduct the intrusion analysis before the intrusion detection process, but can also assist the intrusion detection process as we will show later in this thesis. As the generated attack tree model is generic by focusing on the conducted intrusion techniques from the adversary, the constructed attack tree can be applied to represent multi network environments. Therefore, intrusion detection can execute the high-level detection against the multi-step intrusion in different

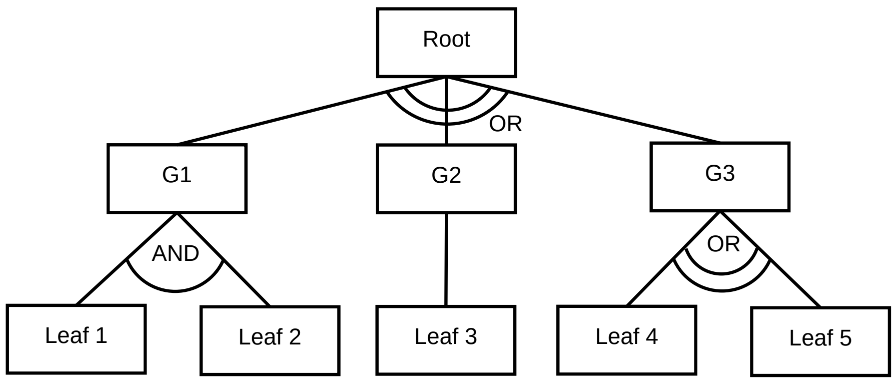


Figure 3.4: An example Attack Tree

networks by applying the attack tree model as the intrusion roadmap.

Besides the aforementioned functional disadvantage for intrusion detection, the attack graph modelling technique has several additional limitations compared with the attack tree technique. Since the generated network attack graphs can be both large and exhibit very dense connectivity, it is a serious challenge for humans to understand [78]. In addition, it is a complex and time-consuming modelling process to generate and analyse attack graphs if there are many hosts, even with automatic generation approaches [66].

3.2.2 Representation of Attack Tree

In the attack tree, the tree structure is utilised to represent attacks against a system, with the root node representing the ultimate attack goal (for example, bypass authentication, obtain confidential, system compromise) and the branches representing the ways to achieve the goal. Two connection types, OR and AND, are used to connect multiple child nodes with their parent node. OR means that the goal can be reached if any one of the subgoals is reached, whereas AND means that the goal can be reached only if all subgoals are reached. Some attributes (for example, descriptive intrusion information [6], boolean value [97]) can be assigned on the node. In this thesis, we will term such a basic attack tree as conventional attack tree (CAT) to differentiate it from further extensions. Figure 3.4 displays an example conventional attack tree. The example abstracted CAT has 5 leaf nodes, 3 intermediate subgoals and 1 root node, but, does not a particular computer network intrusion and assigning any node attributes.

3.2.3 Extensions of Attack Tree

As one of the critical security analysis and modelling tools, CAT has been built on to produce extensions in order to provide flexibility and adaptability with more precise information. Generally, these extensions can be classified into three main categories: (1) *functional extension*; (2) *computational extension*; and (3) *hybrid*. The main purpose of the first category is to enlarge the modelling capability and augment the modelling functionality with extra information for CAT. The key goal of the second category is to compute and measure the relevant attack decision making from the view of the adversary. In the last category, CAT had been combined with other modelling methods together as a hybrid method. Part of our discussion here has appeared in [117].

Functional Extensions

The functional extensions proposed in the literature for the forensic analysis and the intrusion countermeasure analysis.

Augmented Attack Tree (AAT) [91, 92] modelling technique was originally proposed for the computer attack forensic purpose with top-down investigation. It extends CAT with extra attack modelling capability on the tree edges by associating the tree branches with a sequence of malicious operations that could have been implemented in the attack.

Besides CAT and AAT, there are several extensions specialised to provide corresponding countermeasures against attacks into the modelled tree structure in order to protect the system. Defence Tree (DT) [13] provides a set of attack countermeasures on each leaf node. The provided countermeasures represent the possible threat mitigation of the specific vulnerability scenario. Attack Countermeasure Tree (ACT) [96] provides a similar attack countermeasure mechanism as Defence Tree does, but ACT offers countermeasures at every tree nodes instead of only the leaf nodes. There are three distinct types of nodes in ACT: attack event node, detection event node and mitigation event node. Attack-Response Tree (ART) [131] defines and analyses the possible vulnerabilities to compromise a system and the possible response actions against attacks. In ART, every leaf node represents a specific vulnerability exploitation attempt by the attacker, while the root node represents the security property. The consequence nodes in ART are tagged by the response box that represents countermeasure actions. Attack-Defense Tree (ADTree) [49] describes the attack actions an attacker can take to compromise the system and the defense actions that a defender can employ to protect the system. A node in ADTree could be an attack node or a defense node. Each node may also have one additional child node representing a countermeasure. The node of Protection Tree (PT) [31] contains four metrics: probability, cost, impact and risk. With the run-time obtained metrics value, PT ensures the limited resources are consumed to achieve the highest probability to stop an attack successfully.

Computational Extensions

Most attack tree research applies the essential modelling nature of attack tree to model different security scenario, and additionally with the aforementioned intrusion countermeasures to analyse the security of system. Few pay attention to the comprehensive computation studies for the whole generated attack tree analysis, but simply apply conjunctive and disjunctive decompositions or any possible metrics aggregation approaches, which will be discussed in Section 3.5.

The substantial computational based extensions for the entire tree analysis are given in [18, 44, 45, 46]. Generally, the modelled attack tree investigates the possible attack option combinations from the view of the adversary to study the adversary's decision making process in order to predict and execute the most profitable intrusion. The analysis process particularly considers the cost-and-gain relation in terms of the monetary cost of intrusion, the monetary gain of intrusion and the monetary penalty once caught, and further with any probabilities like the likelihood of successful intrusion, the probability of being detected and the probability of being caught.

The basic Multi-Parameter Attack Tree (MPAT) computation [18] has introduced the idea of game-theoretic modelling associated with the aforementioned parameters. Then, the proposed basic computation had been expanded by providing estimated parameter values [44] and exact parameter values [45]. Moreover, the Serial Attack Tree (SAT) model [46] extends the classic parallel attack tree model with the temporal order of the elementary attacks. The main advantage of serial attack tree model is to provide the flexibility to model the adversary's behaviour more accurately and reality, for example, the skipping of elementary intrusions. In addition, SAT computes the better expected outcomes of the adversary.

Hybrid

In order to enrich the attack modelling capability, some additional modelling techniques have been combined with the attack tree into a more comprehensive modelling mechanism.

Misuse cases, derived from Unified Modelling Language (UML) use cases, describe the actions that may harm the system. In [110], misuse cases are applied together with the attack tree and linked to model the detailed security activity and analyse the system security requirements to achieve threat mitigation during the software development lifecycle.

3.2.4 Attack Tree and Intrusion Detection

Besides the modelling capability, attack tree modelling technique had been applied to assist the process of intrusion detection in both indirect and direct forms.

In the former, the intrusion detection related analyses are conducted to provide intrusion information and detection information to facilitate the intrusion detection process. Usually, the attack tree modelling technique generates the attack pattern or attack process information (for example, automatic attack pattern generation [126], causal attack plan [90], and threat damage evaluation [93]) for intrusion detection.

In the latter form, the constructed attack tree provides the intrusion detection roadmap, which enables the detector to detect the high-level intrusions according to the modelled attack tree. However, to the best of our knowledge, there is little work that applies the attack tree technique to directly implement the intrusion detection process based on the constructed framework of an attack tree. [19] is the only work relating to that direction. In [19], enhanced attack tree (EAT) had been utilised to perform the intrusion detection of complex attacks for 802.11 WLAN with the Nondeterministic Finite Enhanced Tree Automaton (NFETA) technique and tree automaton to achieve the tree structure update.

3.3 Node Connectors

In the attack tree modelling technique, the nodes are usually connected with connecters based on logical relations such as conjunctive (also known as AND) and disjunctive (also known as OR). Besides the typical AND and OR connectors, there are some advanced node connectors. We summarise and classify these advanced node connectors into the following two categories: *Order Based Connector* and *Threshold Based Connector*, respectively.

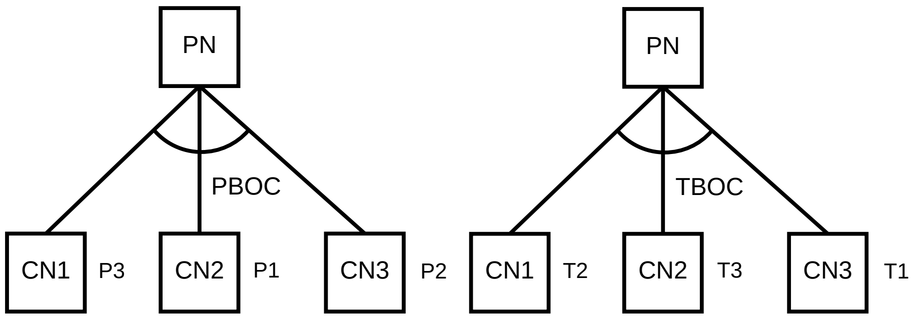
3.3.1 Order Based Connectors

The Order Based Connectors (OBC) are subdivided into *Priority Based Order Connector* and *Time Based Order Connector* according to the specified sequences. Usually, the order based connectors are implemented based on the conjunctive decomposition.

- *Priority Based Order Connector* (PBOC). PBOC represents that the parent node can be achieved only if all the child nodes are accomplished in a priority order, from the highest priority one to the lowest priority one.

Figure 3.5(a) shows a sample PBOC connector with one parent node PN (attack goal) and three child nodes (sub-goals) CN1, CN2 and CN3. The corresponding priority level on these child nodes are P3 (the lowest priority), P1 (the highest priority) and P2 (the medium priority), respectively. Since the priority levels are P1 → P2 → P3, the child nodes must be accomplished in the sequence of CN2 → CN3 → CN1 by following exactly that priority levels. Priority-AND (PAND) [17, 48] obeys this kind of order.

Note that, it is possible that multiple child nodes are attributed with the same priority level. In that case, it is suggested to apply the first come, first achieve principle on those equal-priority nodes.



(a) Priority Based Order Connector (b) Time Based Order Connector

Figure 3.5: Sample Attack Trees with Order based Connectors

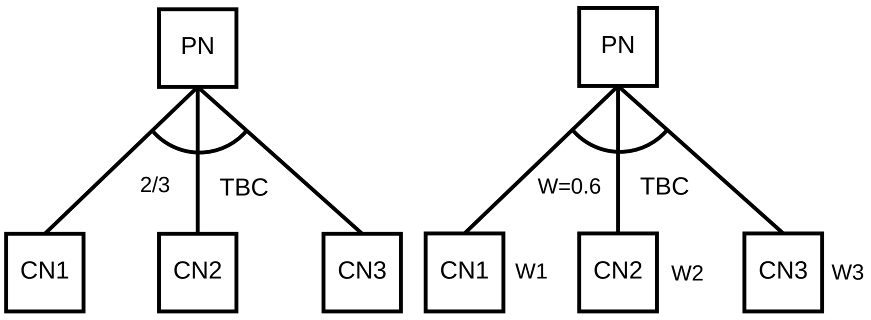
- *Time Based Order Connector* (TBOC). TBOC represents that the parent node can be achieved only if all the child nodes are accomplished in a pre-defined time sequence manner.

  Figure 3.5(b) illustrates a sample TBOC connector with one parent node PN (attack goal) and three child nodes (sub-goals) CN1, CN2 and CN3. The corresponding timed achievement sequence on three child nodes are T2 (timed second achievement), T3 (timed last achievement) and T1 (timed first achievement), respectively. According to the time order T1 → T2 → T3, the child nodes are accomplished with the CN3 → CN1 → CN2 sequence. Ordered-AND (O-AND) [19] and Sequence Enforcing (SEQ) [48] are known extended connectors applying this time based order mechanism.

Practically, POBC and TOBC share some similarities since both of them follow the defined sequences. In POBC, the priority sequence requests that the highest priority level child node be achieved first, whereas the lowest priority level child node be achieved last. It is possible to view that the timed first achieved child node has the top priority, whereas the timed last achieved child node has the least priority as in TOBC. Therefore, it is possible to conclude that POBC and TOBC are similar connectors but focus on different views.

3.3.2 Threshold Based Connectors

Threshold Based Connector (TBC) determines the number of accomplished child nodes for the successful accomplishment of the parent node according to the predefined threshold. However, TBC concentrates on the threshold satisfaction of the number of achieved child nodes without the determination on which nodes need to be achieved. The threshold can be further classified into *amount based threshold* and *weight based threshold.*



(a) Amount Based Threshold (b) Weight Based Threshold

Figure 3.6: Sample Attack Trees with Threshold based Connectors

Table 3.1: Truth Table of Sample Attack Trees with Threshold Based Connector

Child1	**Child2**	**Child3**	Amount Based Threshold		Weight Based Threshold	
			T_A	**Parent**	T_W	**Parent**
F	F	F	2	F	0.6	F
F	F	T	2	F	0.6	F
F	T	F	2	F	0.6	F
T	F	F	2	F	0.6	F
F	T	T	2	T	0.6	T
T	F	T	2	T	0.6	T
T	T	F	2	T	0.6	F
T	T	T	2	T	0.6	T

- *Amount based threshold.* Amount based threshold makes the connector determine the minimum quantity of achieved child nodes to accomplish the parent node. The *amount based threshold* connector is also known as K-out-of-N (K/N) [48]. Figure 3.6(a) displays a sample attack tree with three child nodes (CN1, CN2 and CN3) and one parent node (PN), where the node decomposition is a 2-out-of-3 threshold connector. In this model, the amount of child nodes is 3 and the threshold value is 2. Typically, the parent node can be achieved if more than 2 child nodes have been accomplished. The truth table of the sampled 2-out-of-3 threshold based connector is given in Table 3.1 with three child node columns ($Child1$, $Child2$ and $Child3$), one threshold value column T_A and one parent node column $Parent$ within the *Amount Based Threshold* column set. The symbol **T** indicates the achieved node, whereas the symbol **F** indicates the unachieved node.

Note that K/N has two special cases. K/N is equivalent to OR decomposition if the defined threshold value is one. Meanwhile, K/N is equivalent to AND decomposition if the defined threshold value equals to the amount of all child nodes.

- *Weight based threshold.* Weight based threshold determines the probabilistic threshold as the number of child nodes that need be satisfied for the parent node accomplishment. For this weight based connector, weight components W_i (i is the index of child nodes) must satisfy the following two conditions: (1) $0 \leq \mathrm{W}_i \leq 1$ and (2)$\sum_{i=1}^{n} \mathrm{W}_i = 1$, where n is the total number of child nodes.

  As each child node had been individually assigned one weight probability, the number of child nodes is determined by accumulating the achieved child nodes' weights together. Once the accumulated value exceeds the threshold, the number of accomplished child nodes is determined.

  Figure 3.6(b) illustrates a WBC connected sample attack tree with one parent node (PN) and three child nodes (CN1, CN2 and CN3). Each child node has been assigned with their own weight, W_1 of CN1 is 0.2, W_2 of CN2 is 0.3 and W_3 of CN3 is 0.5. The threshold value is set as 0.6. The *Weight Based Threshold* column set with T_W and $Parent$ on Table 3.1 shows the truth values of this sample.

3.4 Node Metrics

Metrics are a good way to assess the security risk of a computer network. In the attack graph and attack tree modelling techniques, metrics have been widely applied to measure the real-time security status and the on-going intrusion process of the node. In order to appropriately define metrics in an attack graph and an attack tree, five principles [121] have been suggested as follows: (1) *the metric value assignment should be based on specific, unambiguous interpretations rather* *than abstract and meaningless rules*; (2) *the metrics should take into consideration all the information that may be relevant to its potential application*; (3) *the metrics should at the same time leave to users the decisions that cannot be automated with unambiguous rules*; (4) *measuring hosts as collections of vulnerabilities, instead of as the combination of hardware/software configurations*; (5) *the outcome of the metrics should enable its application to make an immediate decision.*

No matter what principles have been suggested, generally, there are two main approaches to apply metrics in attack graph and attack tree: (1) a global metric concludes the general security for the overview; (2) multiple metrics elaborate the relevant aspects for the detail.

The first group assesses the risk by the only metric directly. The *Attack Likelihood* [83] metric quantifies the risk probability with the initial uncertain input values by the consistent computing based on the node of the attack graph. The *k-Zero Day Safety* [122] metric assesses the network security risk against the unknown attacks by counting the number of needed unknown vulnerabilities on the attack graph to compromise a network asset. *Attack Probability* [91] is defined as the probability to measure how far the adversary has progressed towards achieving the ultimate goal in terms of the *least effort* along the attack path in AAT by counting the number of achieved nodes as well.

In the second group, the obtained metrics represent the particularly defined information, and, these metrics need to be further processed by extra aggregation or fusion manipulations, for example the IDS alert-based metrics were fused by Dempster-Shafer theory to assess the risk of intrusion scenarios [75].

In the attack graph and the attack tree, most of the proposed node metrics can be summarised into two groups: (1) the generically defined metrics and (2) the specifically defined metrics.

3.4.1 Generically Defined Node Metrics

The generically defined metrics are generic without the explicit representations to any specified aspects. They are usually applied to examine the overall security of the system. There are two main kinds of metrics to assist the analysis execution, namely, *attack resistance metric* and *attack probability metric.*

Attack Resistance Metrics

Attack resistance metrics on attack graph or attack tree have abstractly been defined as conditions to represent different concepts and issues to achieve the attack and quantify potential attacks [120]. Attack resistance metrics are intuitive properties derived from common sense. In addition, they assess and compare the security of different network configurations according to the constructed attack graph and attack tree.

Attack resistance metrics have been applied to study the metric aggregation [120], which will be discussed in Section 3.5, and [124].

Attack Probability Metrics

Attack probability metrics on attack graph or attack tree have been defined to represent the likelihood that all the required conditions are accomplished to achieve the parent node, and the probability of any node transitions (intrusions) being executed.

Attack probability metrics have been applied to investigate the probability aggregation to determine the probability of the root node achievement [119]. The aggregation is described in Section 3.5.

3.4.2 Specifically Defined Node Metrics

The explicitly defined node metrics have specified any particular meanings for each metric. With the specific state information within attack graph or attack tree, these node metrics are usually known as node *attributes*.

Attack graph and attack tree can not only capture the steps of attack on the constructed attack roadmap, but also represent and calculate risks, cost or weighting [101]. In CAT, Schneier [97] had defined that node metrics can be assigned with either boolean or continuous values. The raw value of those explicitly defined node metrics are assigned at the leaf nodes. The metrics of upper nodes are obtained through the propagation of pre-defined mathematical calculation. With the assistance of metrics, attack tree provides the ability to describe the full complexity of the attacker's decision-making process. Besides those original node attributes, there are several extra node metrics applied in attack tree modelling. In the terms of *usage*, we classify those attributes into two main categories, *attack metrics* and *victim system metrics*. Meanwhile, the attack metrics are subdivided into two sub-categories, *attack accomplishment metrics* and *attack evaluation metrics*. The general metrics in each group are stated as follows.

Attack Accomplishment Metrics

Attack accomplishment metrics examine the real-time attack procedure information on the current node. They describe the temporal dependencies between attack graph/tree nodes, expiration of an attack, and the attack success probability.

- *Time-To-Live* (TTL). TTL [19] defines the lifetime for attack actions at attack tree nodes. With the regulation of TTL, the adversary needs to finalize the attack within the TTL specified period. If more than TTL time has elapsed since the node accomplished, the intrusion must be expired and be implemented again.

- *Attack Level* (AL) and *Attack Probability* (AP). AL and AP [19] define the level and the percentage of whole constructed attack paths on the attack roadmap when any particular nodes have been accomplished. As attack graph/tree usually models the multi-step attack process, AL and AP assist in the measurement of the distance between the current node and the ultimate root node by examining the latest accomplished node. AL calculation assumes each attack action has equal difficulty and weight, whereas AP calculation defines each attack action with different difficulty and weight according to the statistical analysis. Through the statistical analysis, the actions generated more often are assumed as easier to be accomplished. From the standpoint of security analyst, AL and AP reveal the on-going progress against the predefined intrusion roadmap to any particular ultimate goals, and both AL and AP can be used to establish an early warning system against intrusions.

- *Attack Success Probability* (ASP). ASP [18, 31, 44, 45] indicates the likelihood to successfully execute an attack action and accomplish one particular node. From the standpoint of the adversary [44, 45], ASP represents the intrusion capability and the confidence of the adversary in terms of probability.

Attack Evaluation Metrics

Attack evaluation metrics are associated with each of the nodes in the attack tree to analyse and evaluate attacks.

- *Monetary Cost* (MC). MC [18, 31, 44, 114] indicates the money cost of man-hour to achieve the subgoal during the whole attack procedure. Besides the basic cost to conduct the intrusion, monetary based penalty [44, 45] had been considered to pay the fine once the intrusion had been caught by the defender.

- *Impact* (I). I [31] indicates how serious the damage is, that is caused by the intrusion to the system. The impact has been defined as the numeric set $I \in [1,2, \ldots, 10]$ [31] to represent the harmful level of the intrusion to the system.

- *Risk* (R). R [31] indicates how risky the intrusion is. The exploitability, the dependency and the potential damage [11] have been utilised to assess the risk. The exploitability is associated to each vulnerability to measure the likelihood that the vulnerability may be successfully used against the security of the system. The dependency is identified between the enabling vulnerabilities of the former and the latter attacks. The damage potential measures the ability to damage as the number of the affected users times the average number of days the affected service is unavailable.

Victim System Metrics

Victim system metrics describe the attack information relating to the victim system on the attack tree node. It is possible that different properties of computer network system may affect different ways for an adversary to compromise the victim [29].

Several network properties such as *System Vulnerability* (SV), *Network Configuration* (NC), *System Configuration* (SC) and *Access Privilege* (APr) have been applied as the attack tree node attributes. SVs are already reported vulnerabilities from some well-known security databases (like Common Vulnerability Scoring System (CVSS)). NC is the related network information (for example, open port, unsafe firewall configuration). SC may include any information about data accessibility, unsafe default configuration, or read-write permission in file structures. AP includes user account, guest account and root account.

3.5 Metric Aggregation Approaches

Relevant to the computational aspects on the attack graph and attack tree, the aggregation approaches provide the relevant mechanisms to aggregate metric values from different nodes on the attack graph and attack tree.

Compared with the node connectors of attack tree, the metric aggregater shares similar background principle, as both of them fuse the provided multiple elementary values into the single one.

3.5.1 Possible Aggregaters

The fundamental possible approaches include Min, Max, $Multiply$, $Weighted-sum$ and $Average$ [88], which are some common logical computations. In order to formalize these aggregation approaches, the following definitions are given to assist the sematic representations of metric aggregations by borrowing the ideas from [88] and [92].

Definition 5 ***Elementary Node***. *An elementary node E is an attack graph/tree node, which attributes with n different metrics* M_i (i $\in$ [*1, 2, …, n*]) *to signify one set of arguments to the aggregater.*

Definition 6 ***Observed Node***. *An observed node O is an attack graph/tree node, which is under the processing of aggregation with at least two elementary nodes Es.*

Definition 7 ***Aggregation***. *An aggregation is a process to combine the same kind of metric values* M_{E_i} *from multiple* E_i*s* ($i \in$ [*1, 2, …, n*]) *into O as* M_O *with the selected aggregater.*

The formalisations of fundamental aggregation approaches are stated as follows.

- **Min**. The Min aggregater provides the function to select the single minimal metric value from the number of n nodes as the aggregation result.

$$M_O = \min(M_{E_1}, M_{E_2}, \ldots, M_{E_n}) \tag{3.1}$$

- **Max**. The Max aggregater provides the function to select the single maximal metric value from the number of n nodes as the aggregation result.

$$M_O = \max(M_{E_1}, M_{E_2}, \ldots, M_{E_n}) \tag{3.2}$$

- **Addition**. The *Addition* aggregater provides the function to aggregate the same type of metrics from all number of n nodes by summing each single metric on all n nodes.

$$M_O = M_{E_1} + M_{E_2} + \ldots + M_{E_n} \tag{3.3}$$

- **Multiplication**. The *Multiplication* aggregater provides the function to aggregate the same type of metrics from all number of n nodes by multiplying each single metric on all n nodes.

$$M_O = M_{E_1} \times M_{E_2} \times \ldots \times M_{E_n} \tag{3.4}$$

- **Weighted-Sum**. The $Weighted-Sum$ aggregater provides the function to aggregate the same type of metrics from all number of n nodes with different node weight value into the single one. Note that the summation of weight on all Es equals to 1.

$$M_O = \sum_{i=1}^{n}(w_i \times M_{E_i}), where \sum_{i=1}^{n} w_i = 1 \tag{3.5}$$

- **Average**. The *Average* aggregater provides the function to aggregate the same type of metrics from all number of n nodes into the single mean one.

$$M_O = \frac{M_{E_1} + M_{E_2} + \ldots + M_{E_n}}{n} \tag{3.6}$$

3.5.2 Cumulative Aggregations

Besides the aforementioned fundamental aggregaters, there are several advanced aggregation approaches [119, 120] particularly proposed based on attack graph by cumulating either the real number or the probability from the attack graph elements.

Attack Resistance Aggregation

In the attack graph [120], the attack resistance has been presented to assess the security of network configurations, especially on the graph exploits (nodes E and O). The resistance of an attack is interpreted as the effort that an adversary requires to put in until the target is compromised. In order to aggregate the attack resistance metrics, two types of aggregaters, denoted as $\vee$ and $\wedge$, represent disjunctive and conjunctive dependency relationships between exploits with the ideas of series and parallel circuits theory.

Generally, there are two possible scenarios how node relates with another: (1) the parent node has only one child node; and (2) the parent node has more than two child nodes. For the first scenario, the aggregation principle is based on the series circuits theory by adding each attack resistance metric together in sequence.

For the second, the aggregation principle is based on the parallel circuits theory by adding the reciprocals of the attack resistance metrics.

r represents the individual resistance of an exploit, whereas R represents the cumulative resistance of an exploit from the corresponding exploits. The main task of R in aggregating attack resistance metrics is to aggregate the individual resistance r on both E and O by applying either $\vee$ or $\wedge$. The cumulative resistance of each attack goal provides a quantitative measure as how likely that attack goal can be achieved, or equivalently, how vulnerable the corresponding resource is under a given network configuration.

The aggregater $\wedge$ is simply the summation of metric values. Equation (3.7) represents the aggregation of conjunctive with both E and O. The aggregater $\vee$ represents the reciprocal of the sum of the reciprocal of individual metric values. Equation (3.8) illustrates the disjunctive aggregation with E and O.

$$R_O(r_{E_1} \wedge r_{E_2} \wedge \ldots \wedge r_{E_n}) = \sum_{i=1}^{n} r_{E_i} + r_O \tag{3.7}$$

$$R_O(r_{E_1} \vee r_{E_2} \vee \ldots \vee r_{E_n}) = \frac{1}{r_{E_1}} + \frac{1}{r_{E_2}} + \ldots + \frac{1}{r_{E_n}} + r_O \tag{3.8}$$

Attack Probabilistic Aggregation

The cumulative attack probabilistic score of a given goal condition thus indicates the likelihood that a corresponding resource will be compromised during an attack, or equivalently, among all adversaries attacking the given network over a given time period, the average fraction of adversaries who will successfully compromise the resource [119].

Two probabilities have been associated with exploit e and condition c: individual score $p(e)$ and $p(c)$; and cumulative score $P(e)$ and $P(c)$. The individual score $p(e)$ stands for the intrinsic likelihood of an exploit e being executed, given that all the conditions required for executing e in the given attack graph are already satisfied. The cumulative scores $P(e)$ and $P(c)$ measure the overall likelihood that an adversary can successfully either reach and execute the exploit e or satisfy the condition c in the given attack graph.

In addition, the cumulative attack probabilistic aggregation takes into account the causal relationships between exploits e and conditions c in the attack graph [119]. R_r indicates the relation which represents condition c causing exploit e, whereas R_i indicates the relation which represents exploit e causing condition c. Equation (3.9) and Equation (3.10) show the cumulative probabilistic aggregations on exploit e and condition c in the attack graph.

$$P(e) = p(e) \times \prod_{c \in R_r(e)} P(c) \tag{3.9}$$

$$P(c) = \begin{cases} p(c) & \text{if } R_i(c) = \emptyset \\ p(e_1) + p(e_2) - p(e_1) \times p(e_2) & \text{if } R_i(c) \neq \emptyset \end{cases} \tag{3.10}$$

3.6 Summary

In this chapter, the states of the art in attack graph and attack tree modelling techniques are summarised. The attack graph has seen a fair amount of research but it had been limited for the security analysis on large network systems due to the computational complexity. The attack tree, which is a subset of attack graph, has been developed with the moderate amount on the theoretically security analysis, but lacks of practical applications or concrete results. Then, the known node connectors, node metrics, possible metric aggregation approaches and their extensions have been investigated and discussed.

With many different extensions having been proposed based on attack trees, it is important to summarise these extensions into a unified attack tree framework and locate the potential extension points for a new attack tree in Chapter 5. In the next chapter, the Unified Parametrisable Attack Tree and the attack resistance metrics aggregation within attack tree are presented.

Chapter 4

Unified Parametrisable Attack Tree

This chapter presents the unification of different ways in which parameters of an attack tree may be extended, notably: *node attribute*, *edge augmentation* and *connector type*. In formally describing the elements of the tree, a better overview of the relationship between the tree elements can be appreciated.

This generic attack tree structure is termed as *Unified Parametrisable Attack Tree* (UPAT). With UPAT, any of the CAT based extensions (such as tree structure extensions and computational extensions) can generally fall in the scope of parameter settings.

The remainder of this chapter is written as follows. Firstly, section 4.1 presents the formalisation of the conventional attack tree with elementary components. Then, Section 4.2 describes the definitions of unified parameters for conventional attack tree in order to extend it into a unified parametrisable attack tree formalized in Section 4.3. Next, Section 4.4 presents the attack resistance aggregation in attack tree. Finally, the summary of this chapter is given in Section 4.5.

4.1 Formalisation of Conventional Attack Tree

Although the attack tree has been studied for over a decade, few results formally propose a solid and complete attack tree formalisation with a theoretical foundation analysis. Although aimed to extend Schneier's informally described attack tree [97], Mauw and Oostdijk's proposed attack tree formalisation [71] is unable to represent the essence of a conventional attack tree.

The formalisation [71] defines a parent node connecting to a multi-set of child nodes viewed as a *bundle*, which ignores the internal structure within the attack tree. But in CAT, it is important not only to identify the conjunctive and disjunctive connections between multiple nodes, but also to determine the amount of connected child nodes to each parent node. In addition, the formalisation [71] defines that one child node can connect to more than one parent node causing directed cycles, which is more analogues to attack graphs rather than attack trees.

In order to provide a general and complete attack tree formalisation, we propose a formalisation to represent CAT with all of the possible attack tree components, namely, *nodes*, *edges*, *connectors* and *attributes*.

Definition 8 ***Conventional Attack Tree.*** *A Conventional Attack Tree (CAT) is a rooted tree denoted by* $\mathsf{AT} = \langle \mathcal{N}, \mathcal{E}, \mathcal{C}, \mathcal{A} \rangle$*, where*

- $\mathcal{N}$ *is the set of nodes in the tree representing the different states of partial compromise or sub-goals that an adversary needs to move through in order to fully compromise a system, such that for the two subsets* $\mathsf{leafNodes} \subset \mathcal{N}$ *and* $\mathsf{internalNodes} \subset \mathcal{N}$*, we have*
  - $\mathsf{leafNodes} \cup \mathsf{internalNodes} = \mathcal{N}$ *and,*
  - $\mathsf{leafNodes} \cap \mathsf{internalNodes} = \emptyset$ *and,*
  - $v_0 \in \mathcal{N}$ *is the root and represents the ultimate intrusion goal of adversary.*
- $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ *is the set of edges in the attack tree. An edge* $\langle u, v \rangle \in \mathcal{E}$ *represents the state transition (emergent) from a child node* v *(and incident) to a parent node* u*, where* $u, v \in \mathcal{N}$*.*
- $\mathcal{C}$ *is a set of tuples of the form* $\langle u, \mathsf{nodeConnector} \rangle$ *where*
  - $u \in \mathsf{internalNodes}$ *and,*
  - $\mathsf{nodeConnector} \in \{\mathsf{AND}, \mathsf{OR}\}$*, where the definitions are given in Definition 9 and Definition 10.*
- $\mathcal{A}$ *is a set of tuples of the form* $\langle u, \mathsf{nodeAttribute} \rangle$ *where*
  - $u \in \mathcal{N}$ *and,*
  - $\mathsf{nodeAttribute} \in \{\mathsf{booleanvalueAttribute}, \mathsf{continuousvalueAttribute}\}$*, where the definitions are given in Definition 11 and Definition 12.*

Definition 9 AND ***Connector.*** *A node* $u \in \mathsf{internalNodes}$ *has an* AND *connector if all linked child nodes* v_i *incident to the parent node are connected by the* AND *connector, or there is exactly one edge incident to the node.*

Definition 10 OR ***Connector.*** *A node $u \in$* internalNodes *has an* OR *connector if all linked child nodes v_i incident to the parent node are connected by the* OR *connector.*

Definition 11 booleanvalueAttribute ***Attribute.*** *A node $u \in$* internalNodes *has a* booleanvalueAttribute *attribute if it is a boolean value.*

Definition 12 continuousvalueAttribute ***Attribute.*** *A node $u \in$* internalNodes *has a* continuousvalueAttribute *attribute if it is a continuous value.*

4.2 Unified Parameters

As already described in Section 3.2.3, Section 3.3 and Section 3.4, it is known that most attack tree extensions focus on the structural expansions such as adding advanced connectors into the attack tree, or appending additional attributes on nodes within the attack tree. Hence, three parameters are defined to generally represent the types of extension parameters, namely, edge, connector and attribute, respectively, as the unified attack tree inputs.

Past literature [91, 92] has considered edge extensions to the conventional attack tree. For this purpose, we have the following definitions.

Definition 13 ***Edge Parameter.*** eParam *is assigned a boolean value, i.e.* eParam $\in \{0, 1\}$ *to determine whether to augment the edge within the attack tree.* eParam $= 0$ *denotes that the edge is devoid of augmentation, while* eParam $= 1$ *denotes that the edge is augmented with tuples $\langle Label, SIG_{u,v} \rangle$ as per Definition 16.*

The *connector parameter* cParam denotes different types of connectors including both classic AND and OR decompositions or any advanced connectors (that is, order based connector, threshold based connector) on any particular nodes within the attack tree.

Definition 14 ***Connector Parameter.*** cParam *is assigned with an integer value, i.e.* cParam $\in \{1, 2, \ldots, p\}$ *where: each integer denotes an index to a particular connector type, and p is the maximum index number of connectors to select a particular connector on one parent node. The possible connectors may be the ones aforementioned in Section 3.3.*

The *node attribute parameter* nParam indicates possible attributes assigned to any particular nodes within the attack tree.

Definition 15 ***Node Attribute Parameter.*** nParam *is assigned with the tuples $\langle n, Attr_{ID} \rangle$, where: n denotes the number of desired node attributes and $Attr_{ID}$ denotes the index of a particular attribute type. The possible attributes may be the metrics aforementioned in Section 3.4.*

4.3 Formalisation of Unified Parametrisable Attack Tree

The main purpose of UPAT is to serve as a general framework for attack tree based extensions. As modern computer network intrusions increase in the sophistication, UPAT could capture tree extensions to CAT that split an attack into multiple fractions as denoted *incidents* [91, 92]. In addition, the formalisation of UPAT can be expressed by expanding the formalisation of CAT with three parameters eParam, cParam and nParam on the corresponding tree components.

Definition 16 ***Unified Parametrisable Attack Tree.*** *A unified parametrisable attack tree (UPAT) is a rooted labelled tree given by* $\mathsf{UPAT} = \langle \mathcal{N}, \mathcal{E}, \mathcal{C}, \mathcal{A} \rangle$*, where*

- $\mathcal{N}$ is a finite set of nodes in the tree representing the different states of partial compromise or sub-goals that an attacker needs to move through in order to fully compromise a system. $\nu \in \mathcal{N}$ is a special node, distinguished from others, that forms the root of the tree. It represents the ultimate goal of the attacker, namely system compromise. The set $\mathcal{N}$ can be partitioned into two subsets, $\mathsf{leafNodes}$ and $\mathsf{internalNodes}$, such that

  ∗ $\mathsf{leafNodes} \bigcup \mathsf{internalNodes} {=} \mathcal{N}$ and,

  ∗ $\mathsf{leafNodes} \bigcap \mathsf{internalNodes} {=} \emptyset$ and,

  ∗ $\nu \in \mathsf{internalNodes}$.

- $\mathcal{E}$ is the set of edges in the attack tree. An edge $\langle u, v, \mathsf{eParam} \rangle \in \mathcal{E}$ defines an atomic attack, as per Definition 17, and represents the state transition (emergent) from a child node v (and incident) to a parent node u, where $u, v \in \mathcal{N}$. While, eParam determines the edge augmentation as defined in Definition 13.

- $\mathcal{C}$ is a set of node connector tuples of the form $\langle u, \mathsf{cParam}, \mathsf{nodeConnector} \rangle$ such that

  ∗ u $\in \mathsf{internalNodes}$ and,

  ∗ cParam, as Definition 14, determines the particular connector choice of one node, and,

  ∗ $\mathsf{nodeConnector} \in$ {OBC, TBC, OR}, where OBC and TBC are given in Definitions 19 and 20.

- $\mathcal{A}$ is a set of node attribute tuples of the form $\langle u, \mathsf{nParam}, \mathsf{nodeAttribute}\rangle$, where
    - ∗ u $\in \mathcal{N}$ and,
    - ∗ nParam, as Definition 15, determines the node attributes information, and,
    - ∗ $\mathsf{nodeAttribute} \in \{\mathsf{booleanvalueAttribute}, \mathsf{continuousvalueAttribute}\}$.

Definition 17 ***Atomic Attack.*** *An atomic attack is a combination of n* ($n \geq 1$) *incidents (incident*$_1$*, incident*$_2$*, …, incident*$_n$*) with a particular order. The occurrence of an atomic attack contributes towards the state transition in the attack tree.*

Definition 18 ***Incident.*** *An incident is a basic benign or malicious action performed by the adversary. The occurrence of a single incident may not contribute to an attack but taken together leads to an atomic attack.*

Definition 19 ***Order Based Connector.*** *Given a node u of a unified parametrisable attack tree such that* $u \in \mathsf{internalNodes}$*, the node has an Order Based Connector (OBC) if all edges incident to the node are connected by the AND operation but with either the priority based order or the time based order.*

Definition 20 ***Threshold Based Connector.*** *Given a node u of a unified parametrisable attack tree such that* $u \in \mathsf{internalNodes}$*, the node is a Threshold Based Connector (TBC) if all edges incident to the node are connected by the AND operation and the parent node is reached by either the amount based threshold or the weight based threshold.*

4.4 Attack Resistance Attribution

Attack trees have been applied to model diverse security systems in different settings [19, 29, 45, 72, 129]. In a parallel direction, work has also been dedicated to enriching the attack tree technique itself. Some notable works consider: different node connectors; node attribution; multi-attribute (or parameter) nodes; and augmented edges [17, 18, 19, 44, 45, 71, 92, 107].

In this section, we revisit the notion of attack tree attribution, that is, how explicit attribute values of child nodes are aggregated to form the attribute of the parent node, and present a novel attribution approach. We use this attribution within the context of analysing the weakest links of security systems; and thereby demonstrate how the weakest link may not necessarily always be so, instead it depends on the existence of other stronger links.

4.4.1 Attribution

The attribution at a parent node u with a $\mathsf{connector} \in \{\mathsf{AND}, \mathsf{OR}\}$ is the function $f_u^{\mathsf{connector}} : \mathcal{N}^n \to \mathcal{R}_{\mathcal{E}}$ taking as input all the n child nodes $v_1, \ldots, v_n \in \mathcal{N}$ of u and outputting an effective attribute value $R_u \in \mathcal{R}_{\mathcal{E}}$.

The specific instantiations of this attribution operation is parametrised by the type of combining operation of the parent node.

Definition 21 ***Attribution for AND Connector Node.*** *For some n child nodes $v_1, \ldots, v_n$ of parent node u, we define the attribution operation for the AND connector parent node u as the mathematical summation of the attributes R_i (for $i = 1 \ldots n$) of all child nodes. More precisely, the output of this attribution is the effective resistance of these n child nodes that will lead to the parent node u, and is defined as*

$$f_u^{\mathsf{AND}}(v_1, \ldots, v_n) = R_u^{\mathsf{AND}} = \sum_{i=1}^{n} R_i. \tag{4.1}$$

Definition 22 ***Attribution for OR Connector Node.*** *For some n child nodes $v_1, \ldots, v_n$ of parent node u, we define the attribution operation for the OR connector parent node u to be*

$$f_u^{\mathsf{OR}}(v_1, \ldots, v_n) = R_u^{\mathsf{OR}} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n}}. \tag{4.2}$$

Equation (4.1) and Equation (4.2) are inspired from the field of electrical circuit analysis for effective electrical resistance across series and parallel circuits respectively. For AND connector, the adversary needs to fulfill all child nodes before achieving the parent node, hence the overall resistance of the system up to that point of abstraction (represented by the parent node) is the summation of individual resistances of each child node. For OR connector, the adversary only needs to fulfill any child node in order to achieve the parent node.

So it is clear that the AND and OR attributions as defined in Equation (4.1) and Equation (4.2) capture the 'all' and 'any' requirements that an AND and/or OR connector dictates; that is, for any n and some parent node u with n child nodes v_i of resistance R_i ($i = 1 \ldots n$), it is always the case that $f_u^{\mathsf{AND}}(v_1, \ldots, v_n) > f_u^{\mathsf{OR}}(v_1, \ldots, v_n)$.

It turns out that these two attribution equations intuitively capture an adversary's view of his attack options in terms of ease (or resistance) with which to achieve the goal represented by the parent node.

To see why Equation (4.2) intuitively models the adversary's options in this context and why existence of options affects the overall effective resistance of a system to attacks, we revisit the weakest link principle with regards to a system's strength against attacks.

4.4.2 Case Study: Analysis of the Weakest Link

For a security system composed of n components, each can be seen as a link within a security chain that represents the overall resistance of the entire system against attacks. Then the overall resistance of the system and its reliance on the individual security of each component can be described by an attack tree having a parent node whose attribute represents the overall resistance of the system, while its n children nodes each represents one of the links connected together via an OR connector. This models the fact that a system is only as strong as its weakest link.

In more detail, for some n child nodes $v_1, \dots, v_n$ of an OR connector parent u, the node v_w for $w \in \{1, \dots, n\}$ is said to be the weakest node if $R_w = \min(\mathcal{R}_{\mathcal{E}})$ where $\mathcal{R}_{\mathcal{E}} = \{R_1, \dots, R_n\}$ denotes the set of n resistances corresponding to the n nodes. Clearly, the weakest node is considered the weakest link to the security of the parent node because if the adversary decides to target this node v_w, this would lead to achieving the parent node with the least resistance, in line with the weakest link principle.

Going further, Equation (4.2) can be put to good use to analyse the influence that the weakest link and other stronger sibling links have on the overall resistance of the system. More precisely, consider two systems I and II with parent node u, and where in system I the parent node u has one child node v_w with resistance R_w while in system II the parent node u is an OR connector node that has $n+1$ child nodes $v_1, v_2, \dots, v_n, v_w$ where $R_i \in \{R_1, R_2, \dots, R_n\} > R_w$, for example, v_w is the weakest link. The easiest path for the adversary would be via the weakest child node v_w, thus it appears in both systems that the adversary in this case meets with the same resistance if he decides to attack the weakest link. This is evidently demonstrated by conventional attribution methods for the OR connector for example [97], which is simply the $\min(\cdot)$ function.

Nonetheless, effectively the resistance of the systems against a generic adversary, without presuming on how the adversary behaves, can be computed based on our attribution definitions above as

$$R_u^{\mathsf{I}} = R_w, \tag{4.3}$$

$$R_u^{\mathsf{II}} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n} + \frac{1}{R_w}}. \tag{4.4}$$

We then have the following result.

Lemma: The Effect of Being There. *For a parent node u with $n+1$ child nodes $v_1, \ldots, v_n, v_w$, the existence of the child nodes $v_1, \ldots, v_n$ with resistance $R_i > R_w$ stronger than that of the weakest child node $v_w \notin \{v_1, \ldots, v_n\}$ decreases the effective attributed resistance of the parent node u.*

Proof. Rewrite Equation (4.4) as

$$\begin{aligned} R_u^{\mathsf{II}} &= \frac{1}{\frac{\Pi_{i\neq 1}R_i + \Pi_{i\neq 2}R_i + \cdots + \Pi_{i\neq n}R_i + \Pi_1^n R_i}{R_w \cdot \Pi_1^n R_i}} \\ &= \frac{R_w \cdot \Pi_1^n R_i}{\Pi_{i\neq 1}R_i + \Pi_{i\neq 2}R_i + \cdots + \Pi_{i\neq n}R_i + \Pi_1^n R_i} \end{aligned}$$

where $\Pi_1^n R_i$ denotes the product of all R_i for $i = 1 \ldots n$ and $\Pi_{i\neq j} R_i$ denotes the product of all R_i for $i = 1 \ldots n$ except for $i = j$. Rewrite Equation (4.3) as

$$\begin{aligned} R_u^{\mathsf{I}} &= \frac{R_w}{1} \\ &= \frac{R_w \cdot (\Pi_{i\neq 1}R_i + \Pi_{i\neq 2}R_i + \cdots + \Pi_{i\neq n}R_i + \Pi_1^n R_i)}{\Pi_{i\neq 1}R_i + \Pi_{i\neq 2}R_i + \cdots + \Pi_{i\neq n}R_i + \Pi_1^n R_i} \end{aligned}$$

Comparing the above two equations, it is clear that $R_u^{\mathsf{II}} < R_u^{\mathsf{I}}$. □

For an adversary considering which system to attack, the above lemma justifies that system II is a better choice because although all child nodes other than v_w have stronger resistance than v_w, the fact that they exist actually decreases the effective resistance of system II because there is now more than one way to break the system, even if the ways other than the easiest (that is, v_w) do have higher resistance.

From the perspective of a system administrator, the above result is counter-intuitive in the sense that the effect of the weakest link v_w cannot be strengthened by ensuring that all other nodes have a stronger resistance than v_w, and this even if the node v_w exists in both systems as the weakest link individually. In fact in the case of system II, the effective resistance is weakened to a value even below that of v_w. In contrast, such a result cannot be sufficiently captured by conventional attribution techniques for the OR connector.

4.4.3 Concluding Remarks

In this section, we considered the notion of attack trees with node attributes; each such node attribute denotes the resistance of a system against an attack action represented by that node. We proposed a novel attribution approach and applied this within the context of analysing the weakest links of security systems. This helps to cast the relevance of the weakest link in interaction with other stronger links to show how their presence impacts the effective security of the entire system.

4.5 Summary

The essential reason to the proposed UPAT is to provide a general framework for any attack tree based extensions. At first, the formalisation of CAT had been described by adapting notions from AAT. Then, several unified parameters were proposed to represent the extendable elements on the attack tree. Next, the formalisation of UPAT was presented by embedding the unified parameters into CAT's formalisation. Finally, the attack resistance attributions of attack tree with the weakest link analysis are presented.

In the next chapter, a new attack tree will be proposed based on UPAT to provide the relevant mechanisms to facilitate the intrusion detection process.

Chapter 5

Advanced Attack Detection Tree

Although the attack tree has been widely applied to model and analyse system security in terms of vulnerabilities and countermeasures, while, the attack tree had been additionally utilised to conduct the intrusion detection analysis as the early discussion of this thesis in Section 3.2.4, no previous contribution had been provided to execute intrusion detection based on a modelled intrusion road map in conjunction with the real-time intrusion threat analysis on high-level detection (that is, one aggregated intrusion alarm generated after the multiple relevant malicious network packets been identified), and the detection uncertainty analysis (that is, the high-level detection process to deal with the uncertainty and the ignorance when an intrusion detector generates a false detection result).

This chapter presents the proposed mechanisms, which contribute to the functionalities of advanced attack detection tree. First, Section 5.1 proposes three categories of Quality of Detectability (QoD) metrics. All of these QoD metrics provide the guarantees of service on the modelled attack tree structure with respect to the ability of an intrusion detector to detect service-disrupting events caused by malicious attacks. Then, Section 5.2 describes how the Dempster-Shafer evidence theory based detection uncertainty analysis mechanism can be used to examine any uncertainty or ignorance caused during the intrusion detection process. Section 5.3 gives the formalisation of Advanced Attack Detection Tree. Finally, Section 5.4 describes the summary of this chapter.

5.1 Quality of Detectability Metrics

In views that the hidden logics and relations, for example, the causal relation (one attack step enables another), the temporal relation (one attack step happens before another), the spatial relation (one attack step relates to another in the network topology) [67], usually exist in complex attacks involving multiple steps between atomic attacks. It is crucial to consider some way of keeping track of the state of an attack and its corresponding state of intrusion detection. For this purpose, we propose to consider Quality of Detectability (QoD) metrics.

This section describes three QoD metrics groups: *logical steps based QoD metrics M_L*, *time based QoD metrics M_T*, and *alert based QoD metrics M_A*. To appreciate the detection capability of an IDS protected computer network system that leverages on the attack tree, it is possible to view the intrusion event occurrence logically or along the time axis, denoted as *logical steps based QoD metrics* and *time based QoD metrics.* Both of these are in terms of the sequence of attack steps between the start of the attack, the actual detection and the end goal of the attack, which are represented by the nodes of the tree. To investigate the nature of the raw alerts generated by the intrusion detector, *alert based QoD metrics* examine the attack detector's output during the specific atomic attack detection process in terms of the alert type number, the total counted alert number and the alert threat level.

5.1.1 Logical Steps Based QoD Metrics

The main concept behind *logical steps based QoD metrics M_L* is to provide the logical threat level information of the protected target, which is currently facing the multi-step attack according to the modelled attack tree. It considers the attack tree node level as the logical step between nodes. The higher value of node level, the closer the adversary is to achieve the ultimate goal.

A multi-step attack A_c is represented as a sequence of m attack steps $\{a_i\}_{i=1}^{m}$, the culmination of the last attack step, that is, a_m denotes the achievement of the attack end goal. In this sense, each attack sequence can be projected into the corresponding attack path $P = \langle n_l, \ldots, n_s, \ldots, n_r \rangle$ traversed upwards from a leaf node to the root node via any particular subgoal nodes, represented simply by the terminal nodes (that is, leaf node n_l, root node n_r), and any nodes n_s in between. Typically, an attack step is embedding between the corresponding child node and parent node. Therefore, the number of required attack steps from start to culmination of attack is simply $|P| - 1$, where $| \cdot |$ denotes the number of elements in a sequence $\langle \ldots \rangle$. With this in place, the logical steps based QoD metrics M_L can be defined as follows.

- *Steps before Detection* (SbD): SbD is defined as the number of attack steps already achieved on path P' ($P' = \langle n_l, \ldots, n_d \rangle$) between the leaf node n_l and the currently detected node n_d (for $d \in \{1, 2, \ldots, m\}$ as the index of the currently detected node). The computation of SbD is give as,

$$\mathsf{SbD} = |P'| - 1 \tag{5.1}$$

SbD logically measures how fast an intrusion detector can discover something is wrong once an attack sequence is launched, in terms of how many attack steps have gone past before the detection occurs. The higher value of SbD, the more risk of the adversary to compromise the victim system.

- *Steps to Goal* (StG): StG is defined as the number of attack steps not yet achieved on path P'' between n_d and n_r ($P'' = \langle n_d, \ldots, n_r \rangle$). The computation of SbD is give as,

$$\mathsf{StG} = |P''| - 1 \tag{5.2}$$

Rather than measuring the effectiveness of an intrusion detector, this is more of a metric to indicate after an attack detection, how many more steps have yet to be performed by the adversary before the attack succeeds. The less value of StG represents the closer to achieve the ultimate goal of intrusion.

- *Progress to Goal* (PtG): PtG [91] is defined as the probability to measure how far the adversary has progressed towards achieving the ultimate goal in terms of the *least effort* along the attack path in the constructed attack tree. The computation of PtG is given as

$$\mathsf{PtG} = \frac{\alpha}{\beta} \tag{5.3}$$

where, α denotes the number of actually achieved subgoals moving towards the ultimate goal, and β denotes the theoretical least effort to achieve the ultimate goal.

In order to automatically obtain the value of β, it is defined that a node u has k child nodes $\{v_i\}_{i=0}^{k}$. The least effort of a child node v_i is denoted as β_{v_i}. The computation approaches [91] of least effort β are stated as follows: (1) if u is a leaf node of the modelled attack tree, $u \in$ leafNodes, then, β=0; (2) if u is an interior node of the modelled attack tree, $u \in$ internalNodes and has an conjunctive decomposition, then, the computation approach is as Equation (5.4); (3) if u is an interior node of the modelled attack tree, u $\in$ internalNodes and has a disjunctive decomposition, then, the computation approach is as Equation (5.5).

$$\beta = \sum_{i=0}^{k}(\beta_{v_i}) + k \tag{5.4}$$

$$\beta = \min(\beta_{v_i}) + 1 \tag{5.5}$$

5.1.2 Time Based QoD Metrics

Alternatively to viewing the logical steps based QoD metrics along the logical sequences, analogous metrics can be defined to work along the time axis of the attack steps (represented as edges within the modelled attack tree). Let $\{t_i\}_{i=0}^{m}$ denote the time sequence corresponding to the attack sequence with the discrete time steps, thus, t_0 denotes the initial time step that the adversary started the attack actions, t_1 denotes the time step that the adversary achieved the first attack subgoal, t_m denotes the time step at the end goal achievement. Thus, t_d (for $d \in \{1, 2, \ldots, m\}$) denotes the time that an attack step $a_i \in \{a_i\}_{i=1}^{m}$ is detected by the applied IDS if it can detect the attack step instantaneously. Note that in terms of traversing the program of the attack through the constructed attack tree, a_0 would correspond to a leaf node n_l and a_1 is its parent node, etc. The time based QoD metrics M_T are defined as follows.

- *Time to Detect* (TtD): TtD measures how fast an intrusion detector can discover something is wrong once an attack sequence is launched. It is defined as the difference between t_0 and t_d:

  $$\mathsf{TtD} = t_d - t_0 \tag{5.6}$$

  However, practically, it is not possible to determine t_0 from the view of protection (neither the system administrator nor the intrusion detector can identify when an adversary has started an attack). Instead, it can be only measured when an attack activity has been implemented or achieved, that is, an edge has been traversed, and therefore detected.

  W.l.o.g and for simplicity of discussion, the minimum time factor T_{min} is defined as the assumed time cost for each attack step in the attack scenario corresponding to an attack tree edge that is emergent from a leaf node. Hence, it is able to compute t_0 once t_1 is determined by rolling back T_{min}, that is, $t_0 = t_1 - T_{min}$.

- *Time to Current Subgoal* (TtCS): TtCS measures how long an intrusion detector had taken to detect the next consecutive attack step after the achievement of last attack step. TtCS is defined as the difference between t_{d-1} and t_d:

$$\mathsf{TtCS} = t_d - t_{d-1} \tag{5.7}$$

  It implicitly represents the adversary's attack skill. Typically, a skilless adversary may spend quite a long time to conduct the attack, while a skillful adversary may execute the attack quickly. On the other hand, the sophisticated adversary may wait a long period to implement the consecutive intrusion steps or slow down the intrusion process to avoid the intrusion exposure.

- *Time to Goal* (TtG): TtG is defined as the difference between t_d and t_m as Equation (5.8). Rather than measure the effectiveness of an intrusion detector, it is the metric indicating that, after an attack detection, how much more minimal time is left before the ultimate attack goal would be compromised.

$$\mathsf{TtG} = t_m - t_d \tag{5.8}$$

  Similarly, it is also not possible to determine t_m (neither system administrator nor IDS detector can identify when an adversary is expected to complete an attack). Instead, it can be only determined by calculating the assumed expected minimal time T_{min} in each step with the number of unachieved steps to goal StG as Equation (5.9), if all T_{min}s have the same value for all steps.

$$t_m = T_{min} \times \mathsf{StG} \tag{5.9}$$

  Alternatively, it is possible that different steps may have their own unique expected minimal time T_{min} (for example, T_{min} may be short for a single SQL injection attack, but T_{min} may be quite long for a DDoS attack). Thus, t_m can be computed by summing all T_{min}s of the unachieved attack steps from step $d+1$ to step m on the attack path (that is, steps to goal StG):

$$t_m = \sum_{i=d+1}^{m} T_{min_i} \tag{5.10}$$

5.1.3 Alert Based QoD Metrics

Besides the tree structure based QoD metrics, an additional group of QoD metrics is to investigate the possible detected atomic attack information by intrusion detection detector. By providing the possible and essential attack information within QoD mechanism, it implicitly indicates or predicts the next possibly intrusion step and the on-going threat from the adversary.

According to Definition 17 and Definition 18 in Section 4.3, an atomic attack a_i is a combination of q types of modelled attack incidents $\{I_j\}_{j=1}^{q}$.

However, the amount of actual detected attack incident types during one attack step process may be different from the modelled number of incident types, because the audited data may contain other kinds of un-modelled incident types due to noise, or, the adversary may be attempting to achieve the subgoal by applying different attack techniques. Let t denotes the number of actual attack incident types, and $\{{I^+}_j\}_{j=1}^{t}$ denote the actual detected attack incident types. Let $\{N_j\}_{j=1}^{t}$ denote the actual detected incident number corresponding to each attack incident type $\{{I^+}_j\}_{j=1}^{t}$. The metrics in this category are proposed based on metrics in [75] and examine the statistical information about the generated alerts of the low-level incidents. The defined M_A metrics are then as follows.

- *Number of Alert Types* (NAT): NAT is defined as the number of actual IDS detected incident types within an attack step detection between node n_d and n_{d-1}. The measured NAT means that the attack is in progress to compromise the subgoal. In addition, NAT provides the basis for the following two metrics.

$$\mathsf{NAT} = t = |\{{I^+}_j\}_{j=1}^{t}| \tag{5.11}$$

- *Mean Alert Number* (MAN): MAN is defined in Equation (5.12) as the average alerts amount of the actual detected incident types within one attack step detection. The larger value of measured MAN, the more likely that the adversary is attempting intrusion, and more possibly DoS/DDoS attack.

$$\mathsf{MAN} = \frac{\sum_{j=1}^{\mathsf{NAT}} N_j}{\mathsf{NAT}} \tag{5.12}$$

- *Mean Alert Severity* (MAS): MAS is defined as the average alert severity risk level of the actual IDS detected incident types within one attack step detection. The alert severity AS_i (for i $\in \{1, 2, \ldots, r\}$) denotes the severity level, where, AS_1 denotes the bottom severity level, AS_r denotes the top severity level. Note that the value of a single alert severity risk level AS may be obtained from the low-level detector directly, for example three "sig_priority" values ("1" indicates highest risk, whereas "3" indicates least risk) in the "signature" table of the Snort database. The higher (lower) the MAS level, the less(more) risk of the intrusion incident.

$$\mathsf{MAS} = \frac{\sum_{j=1}^{\mathsf{NAT}} AS_i \times N_j}{\mathsf{NAT}}, i \in \{1, 2, \ldots, r\} \tag{5.13}$$

5.2 Detection Uncertainty Analysis

Most intrusion detector and attack tree modelling research treat alerts and symptoms with certainty without doubting the truth behind the alert detectors. In intrusion detection, the intrusion is considered detected if the identified symptoms completely match the predefined signature or statistics information. While in attack tree modelling, the event denoted by an edge, is considered satisfied if the prerequisites denoted by incidents forming the edge, are explicitly achieved. Very little work appear to consider the *uncertainty* analysis.

Though there are many potential uncertainty issues, as previously discussed in Section 2.5.2, our proposed detection uncertainty analysis considers the uncertainty of the generated intrusion alerts and the tree edge achievement process. In order to propose the detection uncertainty analysis within the attack tree, Dempster-Shafer evidence theory (D-S theory) [98] will be applied to deal with the uncertainty due to the following reasons: (1) it reflects uncertainty or a lack of complete information; and (2) Dempster's Rule of Combination gives a numerical procedure to fuse multiple uncertain data as evidence together. Note that D-S theory has been widely applied in the DIDS related research to fuse the multiple intrusion alerts from the high-level IDS detectors [21, 123].

In this detection uncertainty analysis mechanism, the main idea is to analyse how the missed intrusion alerts or the false alerts can effect the modelled attack tree edge transition based on incidents achievement. Since the intrusion detector monitors the real-time network traffic to examine any malicious attack packets, it is possible to obtain statistical information denoted as the *assessment factors* from alerts generated by the intrusion detector. The *assessment factors* can be measured by applying the membership functions, which will be discussed in Section 5.2.2. Once a set of assessment factor values have been generated, the next step is to conduct the *basic probability assignment* process to determine the belief on each claimed focal element for one particular observer. Section 5.2.3 explains the corresponding process in detail. Finally, the Dempster's Rule of Combination will be applied to fuse the belief values from multiple observers with two different assessment approaches as discussed in Section 5.2.4 and Section 5.2.5.

5.2.1 Foundation of Dempster-Shafer Evidence Theory

Dempster-Shafer evidence theory allows the explicit representation of ignorance and uncertainty. Precisely, it is not only *"a theory of evidence because it deals with weights of evidence and with numerical degrees of support based on evidence"*, but also *"a theory of probable reasoning because it focuses on the fundamental operation of probable reasoning: the combination of evidence"* [98].

The *Frame of Discernment* Θ in D-S theory is a finite hypothesis space that consists of mutually exclusive propositions for which the information sources can provide evidence. All possible subsets V of Θ are also called as *focal elements*.

Equation (5.14) is called a *basic probability assignment* (*bpa*) or mass function m whenever it satisfies conditions in Equation (5.15) and Equation (5.16). Equation (5.15) reflects the fact that no belief ought to be committed to ϕ, where ϕ is an empty set. While Equation (5.16) reflects the convention that one's total belief has measure one. The value of $m(V)$ is called V's *basic probability number*, and it measures the belief committed to V.

$$m : 2^{\Theta} \to [0,1] \tag{5.14}$$

$$m(\phi) = 0 \tag{5.15}$$

$$\sum_{V \subseteq \Theta} m(V) = 1 \tag{5.16}$$

Dempster's Rule of Combination provides a data fusion approach that can combine difference pieces of evidence from different observers together to obtain a joint support contribution and can reduce uncertainties simultaneously. The general rule is given by the combined mass function as Equation (5.17), where the combination operator $\bigoplus$ is called *orthogonal summation*. If there are more than two sets of belief values that need to be fused, the combination process is expressed as Equation (5.18). Equation (5.19) shows two equivalent computational approaches based on the different intersection between the observers' focal elements.

$$m = m_1 \bigoplus m_2 \tag{5.17}$$

$$m = m_1 \bigoplus m_2 \bigoplus \cdots \bigoplus m_n \tag{5.18}$$

$$\begin{aligned} m(V) &= \frac{\sum_{\bigcap V_j = V} \prod_{1 \le q \le n} m_q(V_j)}{1 - \sum_{\bigcap V_j = \phi} \prod_{1 \le q \le n} m_q(V_j)} \\ &= \frac{\sum_{\bigcap V_j = V} \prod_{1 \le q \le n} m_q(V_j)}{\sum_{\bigcap V_j \ne \phi} \prod_{1 \le q \le n} m_q(V_j)} \end{aligned} \tag{5.19}$$

5.2.2 Membership Functions

For our detection uncertainty analysis, three focal elements are defined to determine the risk of the achievement of the corresponding attack incident. These three focal elements are: *not risk* (V_1), *risk* (V_2) and *uncertain* (V_3).

All of the elementary membership functions examine the related aspects on modelled attack tree edge according to the intrusion detector generated alert information. The generated alert information normally contains the detected attack information (for example, source information, target information) as requested by the Intrusion Detection Message Exchange Format (IDMEF) [27] and the corresponding statistical information, yet it is difficult to distinguish the evidence from the genuine and direct source of attack. One of main reasons is that the adversary may masquerade the content of malicious packet and intrusion detector therefore identifies the attack symptoms without the capability to judge between the genuine or fake malicious packets. Possible measures [75] to overcome this are to count the number of alerts and retrieve the severity of each alert.

- *Alert Amount.* The alert amount functions f_1 examine the total number of generated alerts to each single incident type during an edge detection process. The obtained value represents not only the attack strength but also the attack confidence. The more the alerts in an edge detection process, the more likely the intrusion and the achievement of the edge.

  The alert amount A_i represents the quantity of generated alerts on incident type I_i, where i is the index of attack incident type on the modelled attack tree edge. Three possible thresholds γ_1, γ_2 and γ_3 indicate the border of *no risk*, the border between *no risk* and *risk*, and the border of *risk*. Thus, there are four corresponding ranges as $[0, \gamma_1)$, $[\gamma_1, \gamma_2]$, (γ_2, γ_3) and $[\gamma_3, \infty]$ to represent *no risk*, *may be no risk*, *may be risk* and *risk*.

  Therefore, the assessment factor on *no risk* is measured by considering γ_2 as Equation (5.20), while the assessment factor on *risk* is measured by considering γ_1 and γ_3 as Equation (5.21).

$$f_{11} = \begin{cases} \frac{\gamma_2 - A_i}{\gamma_2} & \text{if } A_i \leq \gamma_2 \\ 0 & \text{if } A_i > \gamma_2 \end{cases} \tag{5.20}$$

$$f_{12} = \begin{cases} 0 & \text{if } \gamma_1 > A_i \\ \frac{A_i - \gamma_1}{\gamma_3 - \gamma_1} & \text{if } \gamma_1 < A_i \leq \gamma_3 \\ 1 & \text{if } \gamma_3 < A_i \end{cases} \tag{5.21}$$

- *Alert Severity.* The alert severity functions f_2 of an incident type determine how serious is the impact of the attack incident. The low-level intrusion detector normally predefines the severity level. Thus, the alert severity P_{r0} of an incident type I_i can be directly obtained from most low-level intrusion detectors, such as Snort.

  δ represents the number of severity levels according to the intrusion detector's pre-defined signature database. In Snort, the "*sig_priority*" field within "*signature*" table defines three attack severity levels for each corresponding alert to the detected attack. *sig_priority* $= 1$ represents the top severity level for the detected attack; *sig_priority* $= 2$ indicates the medium; while *sig_priority* $= 3$ indicates the lowest. Hence, set $\delta = 3$ and $P_{r0} = 4 -$ *sig_priority* for each Snort detected attack.

  Equation (5.22) expresses the computation process to measure the assessment factor about *no risk*. Equation (5.23) shows the computation process to measure the assessment factor about *risk*.

$$f_{21} = \begin{cases} \frac{\delta - P_{r0}}{\delta} & \text{if } P_{r0} \leq \delta \\ 0 & \text{if } P_{r0} > \delta \end{cases} \tag{5.22}$$

$$f_{22} = \begin{cases} \frac{P_{r0}}{\delta} & \text{if } P_{r0} \leq \delta \\ 1 & \text{if } P_{r0} > \delta \end{cases} \tag{5.23}$$

5.2.3 Basic Probability Assignment

According to the above membership functions, the basic probability assignment can be conducted by generating a set of belief values $m_q^{Ii}(V_j)$ on the claimed focal elements (*not risk*, *risk* and *uncertain*) for each incident type. Note that q indicates the index of the above two membership functions, I_i represents the index of the detected incident type and V_j only represents the j^{th} focal element without the *uncertain* one. Equation (5.24) and Equation (5.25) provide the way to compute each focal element's belief value.

$$m_q^{I_i}(V_j) = \frac{f_{qj}}{\sum_{j=1}^{2} f_{qj} + 1 - P_{IDS}} \tag{5.24}$$

$$m_q^{I_i}(V_3) = 1 - \sum_{j=1}^{2} m_q^{I_i}(V_j) \tag{5.25}$$

where, q = 1,2; j = 1,2; P_{IDS} is the general detection precision of the intrusion detector. $1 - P_{IDS}$ represents intrusion detector's incorrect detection rate, which is one of the main uncertainty issues. In addition, V_1 denotes the first focal element as *not risk*, V_2 denotes the second element as *risk*, while V_3 denotes the third focal element as *uncertain*.

Once a set of belief values have been generated, these belief values can be further fused into $m^{I_i}(V_1)$, $m^{I_i}(V_2)$ and $m^{I_i}(V_3)$ based on Dempster's Rule of Combination as Equation (5.19).

5.2.4 Proposed Sequential Based Assessment Approach

This subsection describes our proposed *sequential assessment approach*. In the *sequential based approach*, the two membership functions are selected as the only observer set. Figure 5.1 depicts the abstracted assessment approach on a modelled edge. The edge between child node CN and parent node PN has n kinds of incidents I_i (i ∈ {1,2,. . .,n}). For each incident, the corresponding belief values of all three claimed focal elements *not risk*, *risk* and *uncertain* are represented as $m^{I_i}(V_1)$, $m^{I_i}(V_2)$ and $m^{I_i}(V_3)$, respectively.

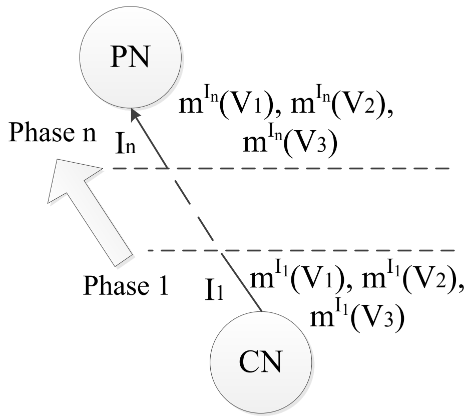

PN
$m^{I_n}(V_1)$, $m^{I_n}(V_2)$, $m^{I_n}(V_3)$
Phase n
I_n
Phase 1
I_1
$m^{I_1}(V_1)$, $m^{I_1}(V_2)$, $m^{I_1}(V_3)$
CN

Figure 5.1: Process of Sequential Based Approach

Sequential based approach examines the detection uncertainty by investigating the belief values of each single modelled attack incident type within the tree edge

Algorithm 5.2.1: SEQUENTIAL BASED ASSESSMENT(Input)

```
{Definitions:
  n: Number of Modelled Incidents within one Edge
  $Ii$: $i^{th}$ Incident Type I within one Edge
  q: Index of Membership Function
  Focal Elements: $V_1$ as Not Risk, $V_2$ as Risk, $V_3$ as Uncertain
  Belief Values of Incident Type $Ii$: $m^{Ii}(V_1)$, $m^{Ii}(V_2)$, $m^{Ii}(V_3)$}
{Input: Real-time obtained Alert Amount $A_i$ and Alert Severity $P_{r0}$}
{Output: Edge Transition with Success or Failure}
BEGIN
FOR $(i = 1; i \leq n; i++)$
  IF $(q == 1)$ //First Membership Function to Examine Alert Amount
    $m_q^{Ii}(V_1) = f_{q1}(A_i)$, $m_q^{Ii}(V_2) = f_{q2}(A_i)$
    $m_q^{Ii}(V_3) = 1 - m_q^{Ii}(V_1) - m_q^{Ii}(V_2)$
  END IF
  IF $(q == 2)$ //Second Membership Function to Examine Alert Severity
    $m_q^{Ii}(V_1) = f_{q1}(P_{r0})$, $m_q^{Ii}(V_2) = f_{q2}(P_{r0})$
    $m_q^{Ii}(V_3) = 1 - m_q^{Ii}(V_1) - m_q^{Ii}(V_2)$
  END IF
  $m^{Ii}(V_1)$ = DSCombination($m_1^{Ii}(V_1)$, $m_2^{Ii}(V_1)$)
  $m^{Ii}(V_2)$ = DSCombination($m_1^{Ii}(V_2)$, $m_2^{Ii}(V_2)$)
  $m^{Ii}(V_3)$ = DSCombination($m_1^{Ii}(V_3)$, $m_2^{Ii}(V_3)$)
  IF $(m^{Ii}(V_2) > m^{Ii}(V_1) + m^{Ii}(V_3))$
    IF (i < n)
      Check Next Incident Type in Sequence
    ELSE
      Current Edge Achieved and Check Next Edge in Sequence
      BREAK
    END IF
  ELSE
    Keep Check Current Incident Type Till Been Achieved
  END IF
END FOR
END
```

Figure 5.2: Pseudocode of Sequential Based Approach

according to the modelled bottom-up sequence. The main purpose of this approach is to certain that each incident type within edge has been achieved by the adversary according to the detected evidence from the intrusion detector without any ignorance.

By applying two membership functions as two independent observers to provide probability belief evidence, the main analysis principle is to take the provided belief evidence and examine whether the detected incident alerts are trustworthy or not for each particular incident type. Each process of the incident type is regarded as a separate phase. Figure 5.1 illustrates the abstracted process from the first modelled incident type I_1 to the last modelled incident type I_n. Note that the horizontal dash line distinguishes the connective incidents as the consecutive phases.

There are two main steps in each phase. The first main step obtains the belief values of the incident. Once the values of assessment factors have been measured by applying the membership functions (that is, Equation (5.20), Equation (5.21), Equation (5.22) and Equation (5.23)) based on intrusion detector's real-time detection results, two sets of belief values (that is, $Bel^{f1} = \{m_1(V_1), m_1(V_2)$ and $m_1(V_3)\}$; $Bel^{f_2} = \{m_2(V_1), m_2(V_2)$ and $m_2(V_3)\}$) to focal elements are generated for both membership functions f_1 and f_2. Then, the essential process is to fuse the two set of belief values into one set ($m(V_1)$, $m(V_2)$ and $m(V_3)$) by applying the *D-S's Rule of Combination* as Equation (5.19).

The second key step is to compare $m(V_2)$ against the summation of $m(V_1)$ and $m(V_3)$. Once the conducted analysis generates the appropriate results (that is, $m(V_2) > m(V_1) + m(V_3)$), the analysis determines the current incident had been achieved and goes up to examine the next type of modelled attack incident along the edge. Because the measured evidence proves that the detected incident is risky enough as the intrusion. Otherwise, the checking process remains at the current incident. When all of the modelled attack incidents have been achieved, then, the whole current edge is considered to be achieved. Consequently, the sequential based analysis on that edge is completed. Figure 5.2 shows the pseudocode for this sequential based approach.

The strength of the *sequential based approach* is that every modelled incident has been proved by the evidence, considering the uncertainty of the alerts. However, the possible limitation is that the edge cannot be deemed to be achieved if any incident along that edge is not achieved.

5.2.5 Proposed Combination Based Assessment Approach

Combination based approach examines the detection uncertainty by investigating the fused belief values from all modelled attack incident types as a bundle. The main purpose of this approach is to determine if the whole edge has been achieved by the adversary according to the provided evidence from the intrusion detector, even if the intrusion detector generates any wrong detection results (for example, miss any malicious network packets without alert generation, mark any normal network packets with alert generation) due to ignorance or uncertainty.

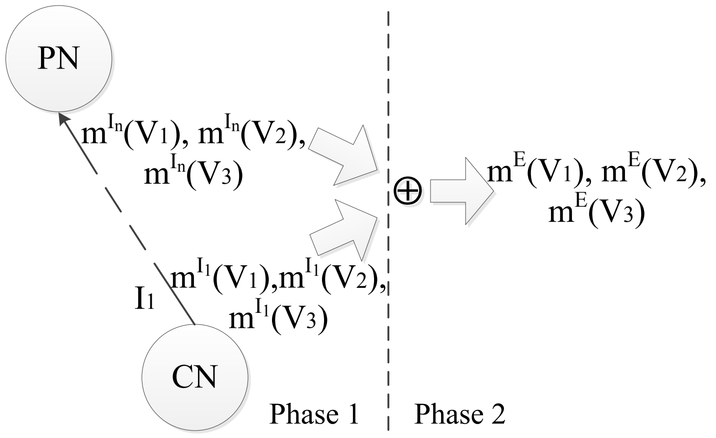

PN
$m^{I_n}(V_1)$, $m^{I_n}(V_2)$, $m^{I_n}(V_3)$
$\oplus$
$m^E(V_1)$, $m^E(V_2)$, $m^E(V_3)$
I_1
$m^{I_1}(V_1)$,$m^{I_1}(V_2)$, $m^{I_1}(V_3)$
CN
Phase 1
Phase 2

Figure 5.3: Process of Combination Based Approach

In the *combination based approach*, the two membership functions are selected as the first observer set, then, all of the modelled incidents within one edge are selected as the second observer set. Compared with the previous *sequential based approach*, the difference is the selection of observers. Beyond the membership functions as the first observer set, all of the modelled incident types are additionally taken as the second observer set.

There are two main phases in the processing of this approach. The former phase is to measure the belief values of each incident type by applying the membership functions as the observer. While the latter phase is to fuse multiple belief value sets into the ultimate set by applying the incident types as the observer. Figure 5.3 displays the abstracted overview of the *combination based approach* on one sampled attack tree edge with all of the incident types from I_1 to I_n. Note that a vertical dash line had been plotted to separate the two phases.

In Phase 1, by conducting the same process as the first step of each phase in the *sequential based approach*, we firstly generate n sets of belief values Bel^{Ii} = $\{m^{Ii}(V_1)$, $m^{Ii}(V_2)$, $m^{Ii}(V_3)\}$, where i = 1, 2, ..., n, for all incident types by applying D-S combination according to evidence provided by the first observer.

Algorithm 5.2.2: COMBINATION BASED ASSESSMENT(Input)

```
{Definitions:
   n: Number of Modelled Incidents within one Edge
   $Ii$: $i^{th}$ Incident Type I within one Edge
   q: Index Number of Membership Function
   E: Targeted Edge
   Focal Elements: $V_1$ as Not Risk, $V_2$ as Risk, $V_3$ as Uncertain
   Belief Values of Particular Incident Type: $m^{Ii}(V_1)$, $m^{Ii}(V_2)$, $m^{Ii}(V_3)$}
{Input: Real-time obtained Alert Amount $A_i$ and Alert Severity $P_{r0}$}
{Output: Edge Transition with Success or Failure}
BEGIN
//Phase 1 Process
FOR $(i = 1; i <= n; i++)$
   IF $(q == 1)$ //First Membership Function to Examine Alert Amount
      $m_q^{Ii}(V_1) = f_{q1}(A_i)$, $m_q^{Ii}(V_2) = f_{q2}(A_i)$
      $m_q^{Ii}(V_3) = 1 - m_q^{Ii}(V_1) - m_q^{Ii}(V_2)$
   END IF
   IF $(q == 2)$ //Second Membership Function to Examine Alert Severity
      $m_q^{Ii}(V_1) = f_{q1}(P_{r0})$, $m_q^{Ii}(V_2) = f_{q2}(P_{r0})$
      $m_q^{Ii}(V_3) = 1 - m_q^{Ii}(V_1) - m_q^{Ii}(V_2)$
   END IF
   $m^{Ii}(V_1) = \text{DSCombination}(m_1^{Ii}(V_1), m_2^{Ii}(V_1))$
   $m^{Ii}(V_2) = \text{DSCombination}(m_1^{Ii}(V_2), m_2^{Ii}(V_2))$
   $m^{Ii}(V_3) = \text{DSCombination}(m_1^{Ii}(V_3), m_2^{Ii}(V_3))$
   $Bel^{Ii} = \{m^{Ii}(V_1), m^{Ii}(V_2), m^{Ii}(V_3)\}$
END FOR
//Phase 2 Process
$Bel^E = Bel^{I1} \bigoplus Bel^{I2} \bigoplus \ldots \bigoplus Bel^{In}$
IF $(m^E(V_2) > m^E(V_1) + m^E(V_3))$
   Current Edge Achieved and Check Next Edge in Sequence
ELSE
   Keep Check Current Edge Till Been Achieved
END IF
END
```

Figure 5.4: Pseudocode of Combination Based Approach

Then, in Phase 2, the Phase 1 generated belief values are taken as another set of evidence and further fused by applying D-S rule of combination as Equation (5.19) by selecting all incident types as the second observer set. Note that the combination rule may be repeatedly applied since there could be more than two incidents types. The combination process is shown as Equation (5.18). Figure 5.4 illustrates the pseudocode of the combination based analysis with two-phase computation.

The limitation of the *combination based approach* is that it does not require that every modelled incident type had been achieved according to the evidence. Nevertheless, the significant advantage is to conduct the edge-based intrusion detection analysis that considers ignorance and uncertainty in the alerts.

5.3 Formalisation of Attack Detection Tree

This section presents the formalisation of Attack Detection Tree. Attack Detection Tree (ADtT) [1] is proposed to model the attack detection progress and additionally satisfy the quality of detectability mechanism and the detection uncertainty analysis by extending the notations of AAT [91, 92]. The formalisation of AAT is given in Appendix A. The formalisation of ADtT is given as follows:

Definition 23 ***Attack Detection Tree.*** *An attack detection tree is a node-labelled rooted tree given by ADtT=$\langle \mathcal{N}, \mathcal{E}, \mathcal{D}, \mathcal{M}, \mathcal{F}, Label, SIG_{u,v} \rangle$, where*

- *$\mathcal{N}$ is a finite set of nodes in the tree representing the different states of partial compromise or sub-goals that an adversary needs to move through in order to fully compromise a system. $r \in \mathcal{N}$ is a special node, distinguished from others, that forms the root of the tree. It represents the system compromise as the ultimate goal of the adversary. The set $\mathcal{N}$ can be partitioned into two subsets,* leafNodes *and* internalNodes*, such that*
  - $\mathsf{leafNodes} \bigcup \mathsf{internalNodes} = \mathcal{N}$,
  - $\mathsf{leafNodes} \bigcap \mathsf{internalNodes} = \emptyset$,
  - $r \in \mathsf{internalNodes}$.
- *$\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ constitutes the set of edges in the attack detection tree. An edge $\langle u,v \rangle \in \mathcal{E}$ defines an atomic attack, as per Definition 26, and represents the state transition from a child node v to a parent node u, for $u, v \in \mathcal{N}$. The edge $\langle u, v \rangle$ is said to be emergent from v and proceeding to u.*

[1]Note that the abbreviation "ADtT" is used to represent Attack Detection Tree instead of "ADT" to avoid the confusion with the acronym for Attack Defence Tree in [49].

- *$\mathcal{D}$ is a set of tuples of the form* $\langle v, \textsf{decomposition}\rangle$, *such that*
  - $*$ $v \in$ internalNodes,
  - $*$ decomposition $\in$ *[O-AND-decomposition, OR-decomposition], where O-AND-decomposition and OR-decomposition are given in Definition 24 and Definition 25 below.*
- *$\mathcal{M}$ is a finite set of metrics on the tree nodes and tree edges representing Quality of Detectability of an attack detector. More details on the types of metrics considered have been presented in Section 5.1. The set $\mathcal{M}$ partitioned into two sections,* edgeMetrics *and* nodeMetrics, *such that*
  - $*$ edgeMetrics $\bigcup$ nodeMetrics $= \mathcal{M}$,
  - $*$ edgeMetrics $\bigcap$ nodeMetrics $= \emptyset$.
- *$\mathcal{F}$ is a finite set of assessment factors to an incident, as per Definition 27, on the tree edges representing Detection Uncertainty Analysis mechanism to an attack detector. More details on the detection uncertainty analysis considered have been presented in Section 5.2.2. The set $\mathcal{F}$ can be partitioned into two subsets,* amountFactor *and* severityFactor, *such that*
  - $*$ amountFactor $\bigcup$ severityFactor $= \mathcal{F}$,
  - $*$ amountFactor $\bigcap$ severityFactor $= \emptyset$.
- *Label is the name of the exploit associated with each edge.*
- *$SIG_{u,v}$ is an attack signature of an atomic attack for attack detection purpose which is defined in Definition 28 below.*

Definition 24 ***O-AND Decomposition.*** *Given a node v of an attack detection tree such that $v \in$ internal_nodes, the node is an O-AND-decomposition if all edges incident to the node are connected by the AND operation but with a particular sequence order.*

Definition 25 ***OR Decomposition.*** *Given a node v of an attack detection tree such that $v \in$ internal_nodes, the node is an OR-decomposition if all edges incident to the node are connected by the OR operation.*

Definition 26 ***Atomic Attack.*** *An atomic attack A is a combination of n incidents (I_1, I_2, $\ldots$, I_n) with a particular order. The occurrence of an atomic attack contributes towards the state transition in the attack tree.*

Definition 27 ***Incident.*** *An incident I is a basic benign or malicious action performed by the adversary or the corresponding response by the target victim. The occurrence of a single incident does not contribute an attack but taken together may lead to an atomic attack.*

Definition 28 ***Attack Signature.*** *An attack signature $SIG_{u,v}$ of an atomic attack is a sequence of sub-signatures combination for each single incident (SIG_{I_1}, SIG_{I_2}, ..., SIG_{I_n}) in the atomic attack.*

5.4 Summary

This chapter has described two proposed mechanisms: the *Quality of Detectability* mechanism and the *detection uncertainty assessment* mechanism. The goal of the first mechanism is to measure and show the on-going intrusion state and intrusion detection progress based on the tree structure and the generated low-level alerts. While the task of the second mechanism is to measure the evidence from the obtained low-level alerts information to identify the achievement of the modelled atomic attack. Two relevant processes are proposed: the *sequential based assessment* and the *combination based assessment*. The former conducts the assessment on every single modelled incident according to the measured evidence, whereas the latter conducts the assessment by fusing the measured evidence together from all modelled incidents. By applying the notion of AAT, ADtT is proposed and formalized by additionally augmenting the QoD mechanism and the detection uncertainty assessment mechanism into the tree framework.

In the next chapter, the intrusion detection approaches will be proposed based on the original AAT and our proposed ADtT.

Chapter 6

Attack Tree Based Intrusion Detection

This chapter describes two different detection mechanisms based on two advanced attack trees: Augmented Attack Tree (AAT), and Advanced Attack Detection Tree (ADtT), respectively. The AAT-based intrusion detection conducts the bottom-up high-level detection process based on the modelled attack tree by applying the original AAT modelling technique [91, 92]. Then, taking the AAT-based intrusion detection mechanism as the foundation, ADtT-based intrusion detection mechanism further implements the Quality of Detectability mechanism and investigates the detection uncertainty from the lower-level intrusion detector to guarantee the quality of detection results, despite there being uncertainty in low-level alerts.

The remainder of this chapter is organized as follows. Section 6.1 describes the proposed time window scheme for the detection mechanisms. Then, two intrusion detection mechanisms based on AAT and ADtT are described in Section 6.2 and Section 6.3, respectively. Finally, the summary of this chapter is given in Section 6.4.

6.1 Sliding Based Detection Window

The detection algorithms are implemented with respect to sliding window based time intervals, denoted Detection Window (DW). The main purpose of DW is to provide an expected detection time range to facilitate the implementations of our detection algorithms, not only to provide the high-level detection, but also to identify hidden relations of intrusions within time bound. Within each DW, the algorithms examine the detected low-level alert information to ascertain if any high-level alerts would be triggered. The possible time unit of DW is set in terms of seconds.

There are two possible scenarios with this DW scheme. The first time scheme provides the overlapping part between two consecutive DWs, whereas the second time scheme provides the non-overlap between two consecutive DWs. The main advantage of the former is to investigate the hidden relations between the currently detected alerts in current DW and the partial history from the previous DW. When carrying over from the current DW to a subsequent one, the algorithm should continue along the modelled tree path from the detected edges in the current DW to the next one so that this historic information is also carried forward to the next DW. Meanwhile, the latter concentrates on the detection within current DW without any information provision from the previous DW.

The formalisation of sliding based detection window scheme is as follows:

Definition 29 ***Detection Window.*** *A sliding based detection window is given by* $DW = \langle \mathfrak{T}, \mathbb{S}, \Delta \rangle$*, where*

- $\mathfrak{T}$ *is the time interval* $[T_s, T_e]$*, where* T_s *and* T_e *are the starting time and the ending time of one desired time interval.*
- $\mathbb{S}$ *is the window size:*
  - $\mathbb{S} = T_e - T_s + 1$
- Δ *is the sliding move step:*
  - $[T_e + \Delta, T_s + \Delta]$

For example, a detection window sets as DW=$\langle [T_s, T_e]$, 5 ,2$\rangle$ means the window size is 5 seconds and the move step is 2 seconds. If the first DW is [11:26:01, 11:26:05], then, the second DW is [11:26:03, 11:26:07], the third DW is [11:26:05, 11:26:09], etc. For another example, DW=$\langle [T_s, T_e]$, 1 ,1$\rangle$ represents the window size is 1 second and the move step is 1 second. Thus, this defined DW works as the normal second tick.

Note that the size of DW plays an important role in the detection mechanism, not only in terms of detection performance, but also to the potential logical relations between the atomic attacks that constitute multi-step attacks. In practice, both of these points can be achieved by slicing the detection in time into overlapping windows, such that atomic attacks detected in different time windows can be considered in unison to avoid mis-detecting multi-step attacks that span more than one time window.

Lemma: DW Overlap. *Two DWs could overlap only if* $\mathbb{S} > \Delta$*.*

Proof. For any pair of distinct DWs, with starting and ending times respectively denoted as $[T_{s1}, T_{e1}]$, $[T_{s2}, T_{e2}]$, $T_{s1} \neq T_{s2}$ and $T_{e1} \neq T_{e2}$, we consider all possible scenarios: (1) $\mathbb{S} = \Delta$; (2) $\mathbb{S} < \Delta$; (3) $\mathbb{S} > \Delta$. Observe that w.l.o.g. it must be that $T_{s2} = T_{s1} + k\Delta$ and $T_{e2} = T_{e1} + k\Delta$ for some integer k. For the DWs to have any chance of overlapping, it suffices to consider the smallest integer multiple, $k = 1$. In scenario (1), for $k = 1$, we have $T_{s2} = T_{s1} + \Delta = T_{s1} + \mathbb{S} = T_{e1} + 1$, so there can be no overlap. In scenario (2), we have $(T_{s2} = T_{s1} + \Delta) > (T_{e1} = T_{s1} + \mathbb{S} - 1)$ since $\mathbb{S} < \Delta$, so in fact some gap of at least $T_{s2} - T_{e1} = \Delta - \mathbb{S}$ exists between two DWs. In scenario (3), we could have, for example for $k = 1$, the situation where $(T_{s2} = T_{s1} + \Delta) \leq (T_{e1} = T_{s1} + \mathbb{S} - 1)$ since $\mathbb{S} > \Delta$, so the DWs could overlap. $\square$

6.2 Augmented Attack Tree Based Intrusion Detection

The AAT based intrusion detection is proposed according to the original AAT [92]. The corresponding AAT formalisation is given in Appendix A. Generally, this proposed detection mechanism is a bottom-up approach by detecting the atomic attacks from the initial leaf nodes towards the root node of the modelled attack tree. For the detection of every atomic attack, the detection mechanism inspects the specified low-level alerts from the low-level intrusion detector to match the predefined atomic attack signature within multiple successive DWs. This detection algorithm notably considers the overlapping time windows mechanism to determine the potential relations between the consecutive intrusions within the defined time range.

Algorithm 6.2.1 within Figure 6.1 shows the pseudocode of proposed AAT based intrusion detection mechanism. The top part, which is from Line 2 to Line 7, provides some basic information of algorithm. While, the main body, which from Line 8 to the last line, provides the algorithm. The part between Line 2 and Line 5 is the relevant definitions relating to AAT formalisation. Line 6 states the algorithm input, which is the real-time network traffic. Line 7 states the algorithm output, which is the high-level detection alert on each atomic attack.

The proposed algorithm works as follows. It initializes the detection at one of the bottom edges (whose child nodes are leaf nodes) by defining all the bottom edges as a set of *SelectedEdges*. The *SelectedEdges* are all potential atomic attacks, and either of them could be achieved in the next step. Meanwhile, the low-level intrusion detector commences to examine the coming traffic and generates any alerts once any malicious packets have been identified. Once both of the initialization steps have been achieved, the algorithm identifies the DW information to

Algorithm 6.2.1: AAT-BASED INTRUSION DETECTION(Input)

```
{Definitions:
  Nodes (currentChild, currentParent, leaf, root)
  Edges (currentEdge, parentEdge, siblingEdge)
  Signatures (compoundAlert, compoundSig)
  Decompositions (AND,OR)}
{Input: Real-time Network Traffic}
{Output: High-Level Intrusion Alert}
BEGIN
 Select All Bottom Edges as SelectedEdges
 WHILE (INTRUSION_DETECTOR_IS_DETECTING) DO
    Determine the Current DW
    FOR currentEdge ∈ SelectedEdges
       Retrieve compoundSig of currentEdge
       Generate compoundAlert in Current DW
       IF sigMatch(compoundAlert, compoundSig) THEN
          Record compoundSig, edgeID and DW
          Retrieve currentParent.Decomposition
          selectNextAppropriateEdge(currentParent.Decomposition)
          IF currentParent.Decomposition == OR THEN
             Select parentEdge
          END IF
          IF currentParent.Decomposition == AND THEN
             IF All siblingEdge been Detected THEN
                Select parentEdge
             ELSE
                Select siblingEdge
             END IF
          END IF
          break
       END IF
    END FOR
    IF rootMatch(currentParent, Root) THEN
         Generate High-Level Alert
    END IF
    Update SelectedEdges
    DW Move Forward
 END WHILE
END
```

Figure 6.1: Pseudocode of AAT Based Intrusion Detection Algorithm

set the specified time range for current intrusion detection process as its first mission, as Line 11. Then, from Line 12, the algorithm retrieves each bottom edge from the *SelectedEdges*, and for each, the algorithm retrieves the corresponding signature information (Line 13). In addition, the algorithm generates the compound alert (*compoundAlert*) in the specified DW (Line 14). Then, in Line 15, the algorithm compares between the generated compound alert (*compoundAlert*) and the retrieved edge signature (*compoundSig*). If they match, the algorithm records the related information, *compoundSig*, *edgeID* and DW information, as Line 16.

Then, the algorithm identifies the subsequent edge that should be checked next. To do so, the algorithm retrieves the decomposition of the current edge's parent node from Line 17 to Line 28. If the decomposition is OR, the algorithm sets the current parent node as the next child node and retrieves the corresponding edge that has that node as a child node. If the decomposition is AND, the algorithm selects the sibling edge, which has the same parent node. If all sibling edges have been detected, the algorithm would proceed to select the parent edge.

Next key step is to check if the current parent node of the newly selected edge is the ultimate root node and generates a final high-level alert if it is, as the part between Line 32 and Line 34. The algorithm is then ready to start from the beginning, in preparation for the next DW to detect new ultimate attacks, as Line 36. Part of this preparation involves updating the *SelectedEdges* set to include edges in the path following the recently detected edges, which should be checked next. The algorithm adds the newly selected edge into *SelectedEdges*, due to that edge may be one of the next attack steps by the adversary; also, it removes the just achieved non-bottom edge from *SelectedEdges*, because that attack step had already been achieved. Aside from bottom edges, the remaining edges within *SelectedEdges* should also be checked in the next DW to ensure continuity between DWs.

6.3 Attack Detection Tree Based Intrusion Detection

The ADtT based intrusion detection is proposed according to our formalisation of ADtT in Section 5.3. Generally, this detection mechanism adds the detection uncertainty analysis mechanism and the QoD mechanism into the existed AAT based intrusion detection algorithm.

Figure 6.2 and Figure 6.3 display the pseudocode of ADtT based intrusion detection algorithm. The whole pseudocode had been separated into two parts due to the limited page length. This algorithm is also a tree-based bottom-up detection mechanism. It starts with the same detection process as AAT-based intrusion detection algorithm does. In each edge's detection process, the ADtT based algorithm further conducts the detection uncertainty analysis, the QoD mechanism measurement, the metrics aggregation.

Once *compoundSig* and *compoundAlert* have been generated in each DW, the algorithm deals with the signature matching process.

If *compoundSig* is fully matching with *compoundAlert*, the algorithm records the related information, for example, *compoundSig*, *edgeID* and DW information. Then, the next essential step is to implement the detection uncertainty analysis with the real-time measured assessment factors as Line 19, for example, alert amount, alert severity, based on either the *sequential based assessment* or the *parallel based assessment*. If the measured result is failure (the part from Line 20 to Line 22), the D-S evidence theory based assessment claims that the detection result is failed. Thus, the algorithm stops the following additional analysis for this DW and prepares another detection on the next coming DW. However, if the assessment result is success, then, the algorithm continually implements the QoD metrics measurement (as Line 23) and the metrics aggregation (as Line 24). For Line 23, the algorithm measures the detectability quality metrics from the logical, time and alert statistics aspects. For Line 24, the algorithm computes the aggregated values of the parent node from the multiple child nodes.

If *compoundSig* is only partially matching with *compoundAlert*, the algorithm records the relevant detected incidents information. Then, the algorithm updates *compoundSig* of that edge by marking the already matched incident alerts. Hence, it records the current detected process. The undetected incident types within *compoundSig* will be examined in the following coming DWs. The part between Line 39 and Line 42 in Figure 6.3 shows the corresponding process.

Then, the algorithm identifies the subsequent edge that should be checked next, according to the modelled attack detection tree. To do so, the algorithm retrieves the decomposition of the current edge's parent node. If the decomposition is OR, the algorithm sets the current parent node as the next child node and retrieves the corresponding edge. If the decomposition is O-AND, the algorithm selects the next sibling edge (sharing the same parent node) following some defined order; if all sibling edges have been detected in that order, the algorithm would proceed to select the parent edge. The part between Line 27 and Line 36 in Figure 6.2 shows the corresponding process. The rest of process is same as Algorithm 6.2.1.

Algorithm 6.3.1: ADtT-based Intrusion Detection(Input)

```
01 {Definitions:
02    Nodes (currentChild, currentParent, leaf, root)
03    Edges (currentEdge, parentEdge, siblingEdge)
04    Decompositions (O-AND,OR)
05    Metrics (resistanceMetrics, QoDMetrics)
06    Assessment Factors (alertAmount, alertSeverity)
07    Signatures (compoundAlert, compoundSig)}
08 {Input: Real-time Network Traffic}
09 {Output: High-Level Intrusion Alert and QoD Values}
10 BEGIN
11  Select All Bottom Edges as SelectedEdges
12  WHILE (INTRUSION_DETECTOR_IS_DETECTING) DO
13      Determine the Current DW
14      FOR currentEdge ∈ SelectedEdges
15         Retrieve compoundSig of currentEdge
16         Generate compoundAlert in Current DW
17         IF sigFullMatch(compoundAlert, compoundSig) THEN
18            Record compoundSig, edgeID and DW
19            detectionUncertaintyAnalysis(alertAmount, alertSeverity) on currentEdge
20            IF detectionUncertainAnalysis obtains Failure THEN
21               break
22            END IF
23            QoDMeasurement(QoDMetrics) on currentEdge
24            metricsAggregation(resistanceMetrics) on currentParent
25            Retrieve currentParent.Decomposition
26            selectNextAppropriateEdge(currentParent.Decomposition)
27            IF currentParent.Decomposition == OR THEN
28               Select parentEdge
29            END IF
30            IF currentParent.Decomposition == O-AND THEN
31               IF All siblingEdge been Detected THEN
32                   Select parentEdge
33               ELSE
34                   Select siblingEdge in Order
35               END IF
36            END IF
37            break
38         ELSE
to be continued
```

Figure 6.2: Pseudocode of ADtT Based Intrusion Detection Algorithm (Part 1)

Algorithm 6.3.2: CONTINUE OF ALGORITHM 6.3.1(Input)

```
            IF sigPartMatch($compoundAlert$, $compoundSig$) THEN
                Record detected incidents information
                Update $compoundSig$ of currentEdge
            END IF
        END IF
    END FOR
    IF rootMatch($currentParent$, $Root$) THEN
        Generate High-Level Alert
    END IF
    Update $SelectedEdges$
    DW Move Forward
END WHILE
END
```

Figure 6.3: Pseudocode of ADtT Based Intrusion Detection Algorithm (Part 2)

6.4 Summary

This chapter has stated two attack tree based intrusion detection mechanisms based on AAT and ADtT, respectively. Both of the proposed detection mechanisms apply detection window based on the sliding window scheme. The AAT based mechanism provides the prototype as a bottom-up tree based detection algorithm. Then, the additional QoD metrics measurement and the detection uncertainty assessment mechanisms are added into the detection algorithm as the ADtT based mechanism.

In the next chapter, we will set up the experimental testbed and conduct experiments to evaluate our proposed attack tree based intrusion detection mechanisms.

Chapter 7

Experiment and Analysis

This chapter presents and evaluates the experimental results on our proposed attack tree based intrusion detection mechanisms. First, Section 7.1 describes the design of the attack tree based intrusion detection system. Second, the initialization processes of the experiment are stated in Section 7.2. Then, the experimental results of AAT based intrusion detection mechanism and ADtT based intrusion detection mechanism are presented in Section 7.3. Next, Section 7.4 presents our discussions on the detection uncertainty assessment with different value settings. Finally, Section 7.5 summarises this chapter.

7.1 Design of Attack Tree Based Intrusion Detection System

This section describes the architecture and database structure for the proposed intrusion detection system, named as **A**ttack **T**ree based **IDS** (ATIDS).

7.1.1 Architecture of ATIDS

Figure 7.1 illustrates the architecture of the proposed ATIDS. In the protected network, ATIDS monitors the incoming and outgoing network traffic on the edge device with the appropriate configuration, for example, port mirroring in router. There are three key components in ATIDS: (1) *Detector Module*, which functions as the low-level detector to generate low-level alerts for the high-level attack tree based detection process; (2) *Detection Module*, which conducts the high-level intrusion detection and the advanced intrusion analysis mechanisms; and (3) *Support Dababase Module*, which stores all the relevant data, such as: the generated low-level alert from the *Detector Module*; and the obtained high-level attack data from the *Detection Module*.

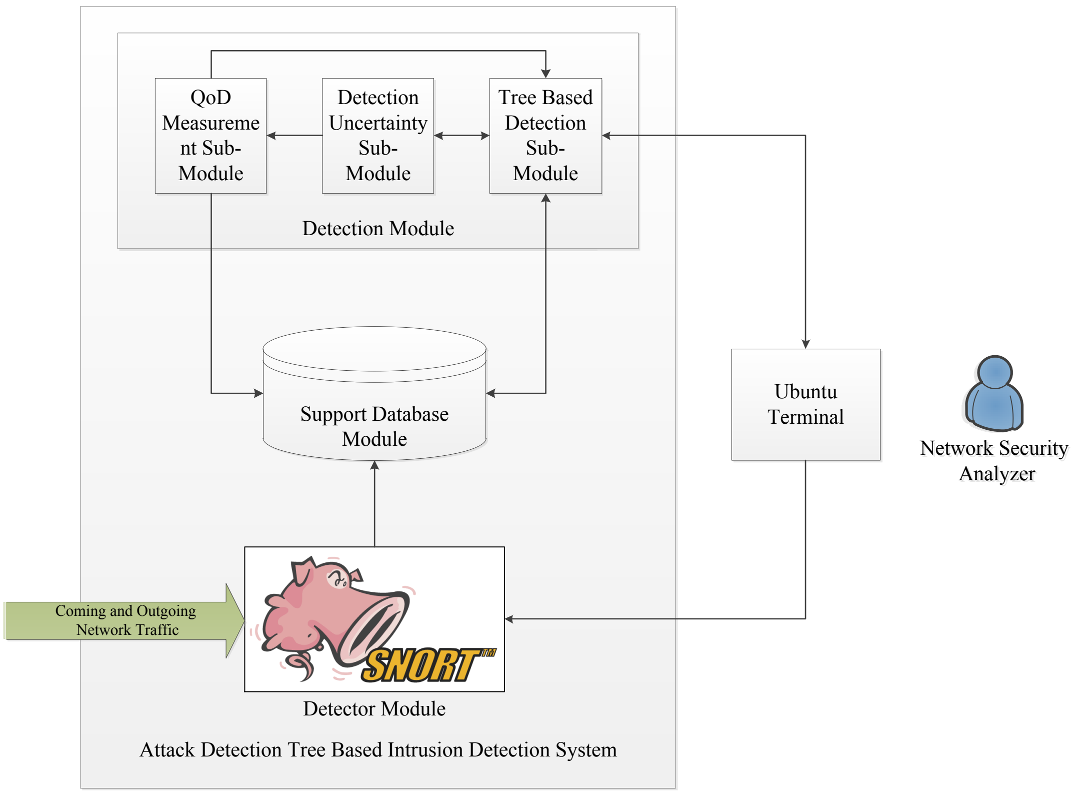

QoD Measurement Sub-Module
Detection Uncertainty Sub-Module
Tree Based Detection Sub-Module
Detection Module
Support Database Module
Ubuntu Terminal
Network Security Analyzer
Coming and Outgoing Network Traffic
SNORT
Detector Module
Attack Detection Tree Based Intrusion Detection System

Figure 7.1: Architecture of ATIDS

In addition, the *Detection Module* is subdivided into the following three subsections: (I) *Tree Based Detection Sub-Module*; (II) *Detection Uncertainty Sub-Module*; and (III) *QoD Measurement Sub-Module*. *Tree Based Detection Sub-Module* implements the fundamental detection process according to the ADtT based intrusion detection algorithm. *Detection Uncertainty Sub-Module* deals with the D-S evidence theory based detection uncertainty analysis as part of the tree based detection. Moreover, *QoD Measurement Sub-Module* measures the relevant QoD metrics values according to the achieved detection results.

Both *Detector Module* and *Detection Module* are controlled by the network security analyzer through the command terminal.

7.1.2 Support Database Schema

Figure 7.2 illustrates the visual overview of the support database module in ATIDS with the entity relations between the tables. The support database module can be classified into two main sections: *tree detection section* and *detector support section*. The left dash line box in Figure 7.2 shows the *tree detection section*, while the right dash line box in Figure 7.2 shows the *detector support section*. The tables in the *tree detection section* represent the relevant database tables relating to the modelled tree representation and the detection procedure recording. The tables in the *detector support section* are Snort provided standard tables, for example, *event*, *signature*. The corresponding Snort database schema can be found in [4].

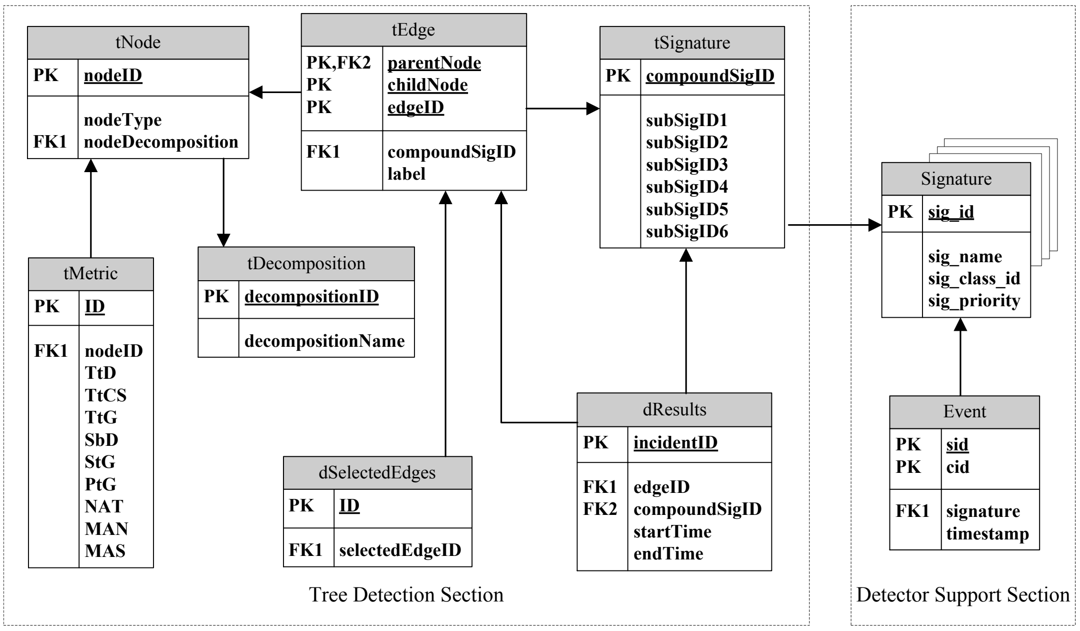


Figure 7.2: ER Diagram of Support Database

The *tree detection section* can be further divided into two parts: *tree structure part* and *detection procedure part*. The *tree structure part* tables are established based on the ADtT formalisation. We set five tables: *tNode*, *tEdge*, *tSignature*, *tDecomposition* and *tMetric* to represent ADtT's node $\mathcal{N}$, edge $\mathcal{E}$, signature $SIG_{u,v}$, node decomposition $\mathcal{D}$ and metric $\mathcal{M}$, respectively. All values from the modelled attack detection tree are stored in the relevant tables. In the *detection procedure part*, *dResults* table stores the detected high-level atomic attack information, whereas, *dSelectedEdges* table records the selected edges during detection process about the possible edges which may be checked in the next DW.

Within the range of *tree detection section*, *tNode* and *tEdge* are most fundamental modelled tree elements due to each edge having two nodes and the edge provides the framework of the tree. Hence, there is one relation between *tNode* and *tEdge*. In addition, each node entity has been attributed with the corresponding decomposition and metric entities. Thus, another two relations exist between *tNode* and *tDecomposition*, and between *tNode* and *tMetric*. For each edge entity, there is another relation between itself and the signature entity, since each edge has its own unique signature. Besides the entity relations within the *tree detection section*, there is one relation that connects from *tSignature* entity to *signature* entity in the *detector support section*. Consequently, the modelled Attack Detection Tree is virtually represented in the database.

7.2 Experimental Initialization

7.2.1 Testbed Setup

A testbed had been constructed to implement the proposed ATIDS. Figure 7.3 illustrates the network structure of the built testbed. Two laptops with installed Ubuntu OS connect to each other via a network hub. The version of applied Ubuntu on both laptops is v9.10. According to Figure 7.3, the left laptop plays as the attack traffic provision server, irrespective of whether it is a compromised bot, or the root attack source from an adversary; whereas the right laptop plays as the detection server. In the left one, a piece of network traffic replay software had been installed to replay the captured attack traffic in .PCAP format. In the right one, the developed ATIDS had been deployed to implement the high-level detection on the attack traffic.

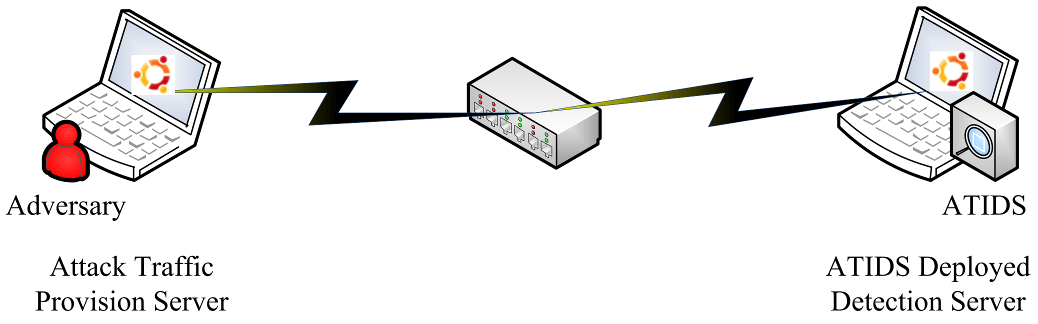

Adversary
ATIDS
Attack Traffic Provision Server
ATIDS Deployed Detection Server

Figure 7.3: Constructed Testbed

During the ATIDS development process, the development platform is also Ubuntu v9.10; the main developing language is *C* and the applied C compiler is the *GNU C Compiler* (v4.4.4.1-1ubuntu2) in Ubuntu; the applied database is *MySQL* (v5.1.37-1ubuntu5.4). In addition, the attack traffic provision server installed replay software is *tcpreplay* (v3.4.1). Furthermore, *Snort* (v2.8.4.1) is utilised as the low-level detector of ATIDS.

7.2.2 Experimental Data Set Selection

As the proof of concept, DARPA2000 data sets [55] have been applied to examine the proposed ATIDS. Compared with other well-known intrusion data sets (for example, KDD CUP 1999 [1], DARPA1998 [52], DARPA1999 [53]), the significant feature of DARPA2000 is that DARPA2000 data sets are multi-step intrusions for the unique ultimate attack goal (to compromise host by DDoS) by the single adversary or the multiple cooperating adversaries. Thus, there are causal relations between two consecutive intrusion steps. In contrast, for the other data sets, though they contain large volume of attack traffic, these attacks target on different attack goals (for example, in DARAP1999, DoS targets on the disruption of a host or network service, Remote to Local (R2L) targets on the remote and illegal access to the local machine, User to Root (U2R) targets on the local user illegally obtain the administrator privilege [64].) and may launched by different adversaries from different sources without any comprehensive causal relations. From the view of attack tree modelling, it is difficult to build an attack tree with multiple ultimate attack goals (root nodes). Therefore, we apply DARPA2000, which includes two separate data sets: LLDOS1.0 and LLDOS2.0.2.

The simulated network of DARPA2000 is divided into three segments representing the networks *inside* an Air Force base, which is the target network system of the adversary; the *outside* Internet of the Air Force base; and the De Militarized Zone (*DMZ*) that connects the *inside* and *outside*. Within each data set, there are two groups of provided traffic files in .PCAP format. One group is a set of traffic files from the "Inside" network, and another is a set of traffic files from the "DMZ" network. The abstracted network topology is shown as Figure 7.4. Both "Inside" traffic and "DMZ" traffic are captured simultaneously when the adversary conducting the multi-step attacks. Thus, both of "Inside" traffic and "DMZ" traffic have same attack process and most of the attack packets are same.

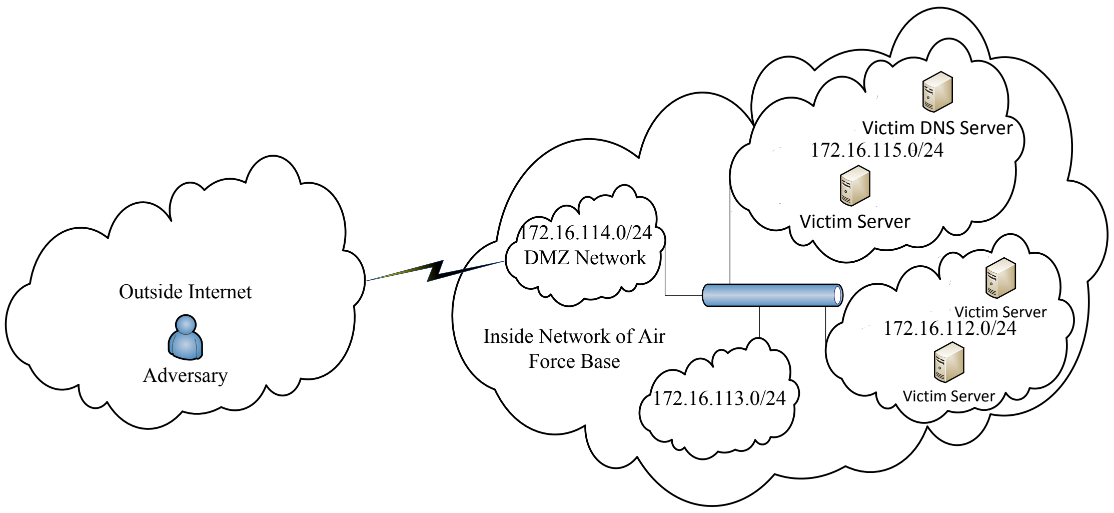


Figure 7.4: Network Topology of DARPA2000 Data Sets

In LLDOS1.0 data set, the adversary targets on four subnets within the inside network and compromises three victim servers to launch DDoS attacks. In LLDOS2.0.2 data set, the adversary targets on the DNS victim server to exploit the same victim servers to launch DDoS attacks. In addition, both LLDOS1.0 data set and LLDOS2.0.2 data set have five intrusion steps, the main process of each step is briefly described in Table 7.1. More statistics information of each phase will be presented in Section 7.2.3.

We apply the “Inside” files to establish the model of ADtT and initialize the detection performance of Snort. We then apply both the “Inside” and “DMZ” files to evaluate our proposed ATIDS.

Table 7.1: Typical Attack Phases Matching in DARPA2000 Data Sets

Data Set	Phase	Attack Description
LLDOS1.0	Phase 1	IP Sweep
	Phase 2	Probe by Sadmind Vulnerability
	Phase 3	Break into Victims
	Phase 4	Install DDoS Daemons on Victims
	Phase 5	Launch Attacks to Primary Victim
LLDOS2.0.2	Phase 1	DNS Probe
	Phase 2	Break into Victim
	Phase 3	Install DDoS Daemon on Victim
	Phase 4	Install DDoS Daemons on Victims
	Phase 5	Launch Attacks to Primary Victim

7.2.3 Prerequisite Detection Evaluation

Since Detection Uncertainty Analysis requests the calculation of total uncertainty (as Equation (5.24) in Section 5.2.3) during the attack detection tree based intrusion detection process, it is necessary to obtain the uncertainties from both the membership functions and detector’s detection performance. This subsection provides the corresponding process to evaluate the detector’s detection performance.

In order to identify Snort’s detection performance on DARPA2000 data sets, the first essential task is to investigate the ground truth of the examined traffic. Though DARPA2000 is one of the widely applied data sets for intrusion detection related research, there is limited information about the ground truth of DARPA2000 from MIT Lincoln laboratory. In addition, few relevant publications clarify the detailed ground truth of those data sets. Thus, we have conducted the ground truth analysis on DARPA2000. The detailed process is presented in Appendix B. Table 7.2 and Table 7.3 show the measured ground truth including the total number of traffic packets from each “Inside” .PCAP file, the number of identified attack packets and the number of identified normal packets in both LLDOS1.0 and LLDOS2.0.2 data sets.

After that, the next key process is to determine Snort’s detection precision. The general idea is to examine whether the alert generated packet is the attack

Table 7.2: Ground Truth of DARPA2000 LLDOS1.0 Inside Traffic

Phase	Number of Total Packets	Number of Attack Packets	Number of Normal Packets
Phase 1	40	40	0
Phase 2	158	151	7
Phase 3	225	111	114
Phase 4	520	190	330
Phase 5	73924	34059	39865

Table 7.3: Ground Truth of DARPA2000 LLDOS2.0.2 Inside Traffic

Phase	Number of Total Packets	Number of Attack Packets	Number of Normal Packets
Phase 1	4	2	2
Phase 2	6	4	2
Phase 3	72	42	30
Phase 4	203	135	68
Phase 5	954	609	345

packet or the normal packet according to the measured ground truth. The detailed process is presented in Appendix C.

Refer to the four fundamental intrusion detection metrics in Section 2.2.3, the packet which had been labelled as positive in both ground truth and detection result is determined as *TP*; the packet which had been labelled as negative in both ground truth and detection result is determined as *TN*; the packet which had been labelled as positive in ground truth and negative in detection result is determined as *FN*; the packet which had been labelled as negative in ground truth and positive in detection result is determined as *FP*. Table 7.4 shows the measured four common detection metrics (TP, FP, FN, TN) of each phase on the DARPA2000 LLDOS1.0 Inside traffic. The measured results for DARPA2000 LLDOS2.0.2 are shown in Table 7.5.

Then, it is possible to calculate the relevant detection performance for each phase by applying the detection performance examination equations as Equation (2.1), Equation (2.2), Equation (2.3) and Equation (2.4) in Section 2.2.3. Table 7.6 shows the measured detection performance in terms of *DR*, *FAR*, *Precision* and *Accuracy* in the corresponding columns. Note that "NAN" in "LLDOS2.0.2 Inside Phase 1" row and "LLDOS2.0.2 Inside Phase 3" represents *Not A Number*. The reason to "NAN" is due to there being no packet detected in both

Table 7.4: Measured Detection Metrics of DARPA2000 LLDOS1.0 Inside Traffic

Phase	Number of TP	Number of FP	Number of FN	Number of TN
Phase 1	40	0	0	0
Phase 2	151	0	0	7
Phase 3	28	7	83	107
Phase 4	7	4	183	326
Phase 5	267	311	33792	39554

Table 7.5: Measured Detection Metrics of DARPA2000 LLDOS2.0.2 Inside Traffic

Phase	Number of TP	Number of FP	Number of FN	Number of TN
Phase 1	0	0	2	2
Phase 2	4	0	0	2
Phase 3	0	0	42	30
Phase 4	4	2	131	66
Phase 5	3	2	606	343

phases, thus, the corresponding 0 *TP* and 0 *FP* cause the division 0/0.

Table 7.6: Detection Performance of Each Phase

Phase	DR	FAR	Precision	Accuracy
LLDOS1.0 Inside Phase 1	100%	0%	100%	100%
LLDOS1.0 Inside Phase 2	100%	0%	100%	100%
LLDOS1.0 Inside Phase 3	25.23%	20%	80%	60%
LLDOS1.0 Inside Phase 4	3.7%	36.36%	63.64%	64.04%
LLDOS1.0 Inside Phase 5	0.78%	53.81%	46.19%	53.87%
LLDOS2.0.2 Inside Phase 1	0%	NAN	NAN	50%
LLDOS2.0.2 Inside Phase 2	100%	0%	100%	100%
LLDOS2.0.2 Inside Phase 3	0%	NAN	NAN	41.67%
LLDOS2.0.2 Inside Phase 4	2.96%	33.33%	66.67%	34.48%
LLDOS2.0.2 Inside Phase 5	0.49%	40%	60%	36.27%

As all four metrics represent the detection performance, the following task is to determine the most suitable metric to be applied in D-S analysis. For *DR*, *FAR* and *Precision*, they determine the detection performance by only investigating the fact of generated alerts. *Accuracy* measures the detection performance by examining not only the correctly labelled attack packets, but also the correctly classified normal packets. Though the former set shows the performance about how the detector can correctly detect the attack packets, the latter metric represents the comprehensive detection performance on both attack packets detection and normal packets labelling. Therefore, *Accuracy* is a more appropriate metric to represent the detector's general performance. The values from all phases will be utilised for the following detection process.

7.2.4 Attack Tree Modelling

To build the attack detection tree model for DARPA2000 data sets, there are two essential steps: (1) *tree framework construction*, and (2) *attack signature creation*. According to ADtT formalisation in Definition 23, the infrastructure of the tree consists of node $\mathcal{N}$, edge $\mathcal{E}$ and decomposition $\mathcal{D}$, whereas $SIG_{u,v}$ represents the attack signature and can be applied in intrusion detection. Hence, the first phase determines the fundamental structure of the tree according to the data set's attack process. Meanwhile, the second phase composes the attack signatures for the corresponding tree edges.

- *Tree framework construction.* The files are the network captures within each of the DARPA2000 data set containing five attack phases. Generally, the adversary probes the target network system to determine the live hosts, then, compromises the live hosts and installs the DDoS daemons, and finally, launches the DDoS attack on the primary victim. More detailed information can be found from the DARPA2000 website of MIT Lincoln Laboratory [55].

  With the attack description information, we can construct the DARPA2000 Attack Detection Tree model based on the analysis of *the general attack step sequence* and *the causal relations* between attack phases. *The general attack step sequence* analysis is applied to construct the overall tree structure of the five phases in DARPA2000, whereas *the causal relations* analysis is applied to determine the appropriate position of each phase within the overall tree structure.

  In practical computer network attacks, irrespective of how the attacks are different from each other, there is typically a *general attack step sequence*: *reconnaissance*; *exploit vulnerable port* and *attack goal*.

  In LLDOS1.0, the main operation of Phase 1 is to identify the live machines in the target network by applying the IP sweep attack. Then, in Phase 2, the adversary explores the flaws of targeted machines based on the port information. Thus, both of phases are classified as *reconnaissance* in the general attack step sequence. Phase 3 and Phase 4 are the *exploit vulnerable* step as the adversary breaks into the secondary victims and installs the DDoS attack daemon programs. The DDoS attack had been launched targeting on the primary victim in Phase 5, which is the ultimate attack goal as the final step. Table 7.7 concludes the steps in the *general attack step sequence* attack phases and the corresponding phase and attack description for LLDOS1.0.

Table 7.7: Typical Attack Phases Matching in LLDOS1.0 Data Set

Typical Attack Phase	**Phase**	**Attack Description**
Reconnaissance	Phase 1	IP Sweep
	Phase 2	Probe by sadmind Vulnerability
Exploit Vulnerability	Phase 3	Break into Victims
	Phase 4	Install DDoS Daemons on Victims
Attack	Phase 5	Launch Attacks to Primary Victim

In DDOS2.0.2, the operation of Phase 1 is to probe the DNS server by conducting HINFO query as *reconnaissance*. Then, the adversary breaks into the secondary victims and installs the DDoS daemon in Phase 2, Phase 3 and Phase 4 as *exploit vulnerable* process. At last, the adversary launches the DDoS attacks by controlling these secondary bots to the primary victim. Table 7.8 concludes the steps in the *general attack step sequence* attack phases and the corresponding phase and attack description for LLDOS2.0.2.

Table 7.8: Typical Attack Phases Matching in LLDOS2.0.2 Data Set

Typical Attack Phase	**Phase**	**Attack Description**
Reconnaissance	Phase 1	Probe by DNS HINFO Query
	Phase 2	Break into one Victim
Exploit Vulnerability	Phase 3	Install DDoS Daemons on Victim
	Phase 4	Install DDoS Daemons on More Victims
Attack	Phase 5	Launch Attacks to Primary Victim

Therefore, we determine that all of these five phases in both LLDOS1.0 and LLDOS2.0.2 follow the generic attack sequence, and the overall structure of these phases is one phase linking to another subsequent phase. The determined edges, relevant nodes and labels are listed in Table 7.9. The syntax of a node name is represented as “NX.X”, which the prefix “N” indicates “Node”, the first “X” indicates data set and the second “X” indicates the corresponding phase index. The “R” indicates the root node.

- *Attack signature creation.* As defined in Definition 28, an atomic attack

Table 7.9: Framework of Modelled Attack Detection Tree

Data Set	Edge ID	Child	Parent	Label
LLDOS1.0	E1.1	N1.1	N1.2	IP Sweep
	E1.2	N1.2	N1.3	sadmind Probe
	E1.3	N1.3	N1.4	Breakin Victims
	E1.4	N1.4	N1.5	DDoS Daemons Installation
	E1.5	N1.5	R	DDoS Attacks
LLDOS2.0.2	E2.1	N2.1	N2.2	DNS Probe
	E2.2	N2.2	N2.3	Breakin Victim
	E2.3	N2.3	N2.4	DDoS Daemon Installation
	E2.4	N2.4	N2.5	More DDoS Daemons Installation
	E2.5	N2.5	R	DDoS Attacks

signature ($SIG_{u,v}$) of ADtT is a group of combined signatures from related incidents. As all of the incident signatures (SIG_I) appear with a particular order in each phase, it is important to construct $SIG_{u,v}$ with multi SIG_I obeying that specific sequence. In addition, because most SIG_I must appear in the signature to fully represent the attack step, we apply logical AND to connect the incident signatures. The symbol $\cup$ indicates the AND connection in the $SIG_{u,v}$.

From the view of detection on DARPA2000 data sets, the possible incidents are the network traffic packets, which are labelled as attack packets by Snort. While, the signature of each incident is the corresponding Snort signature. Thus, we may obtain SIG_Is by implementing the Snort detection on the provided sample data. Table 7.10 lists all of the 16 kinds of Snort alert names as the low-level incident types within DARPA2000. Some of the incidents appear in both LLDOS1.0 data set and LLDOS2.0.2 data set. Note that our obtained attack incident list is slightly different from other researchers' work (for example, 15 in total in [5]) even though we applied the same data files. The possible reason could be due to our applied Snort versions being different (for example, Snort 2.8.2 in [5], but ours is Snort 2.8.4.).

Table 7.11 shows the details of the incidents that make up the atomic attack signatures and corresponding edges in the attack tree.

Figure 7.5 displays the modelled attack detection tree of DARPA2000. The left branch represents the LLDOS1.0, whose initial phase is Phase 1; the right branch represents the LLDOS2.0.2, whose initial phase is Phase 2. Note that Phase 1 and Phase 3 in the right branch have been represented with the dash line. This is

Table 7.10: Summary of Low-Level Incident Types in DARPA2000

Index	**Name of Incident**
I_1	ICMP PING
I_2	ICMP Echo Reply
I_3	RPC portmap sadmind request UDP
I_4	RPC sadmind UDP Ping
I_5	ICMP Destination Unreachable Port Unreachable
I_6	RPC sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow attempt
I_7	RPC sadmind query with root credentials attempt UDP
I_8	INFO TELNET access
I_9	INFO TELNET login incorrect
I_{10}	INFO TELNET Bad Login
I_{11}	RESERVICES rsh root
I_{12}	COMMUNITY SIP TCP/IP message flooding directed to SIP proxy
I_{13}	(snort decoder) Bad Traffic Loopback IP
I_{14}	BAD-TRAFFIC loopback traffic
I_{15}	SNMP AgentX/tcp request
I_{16}	BAD-TRAFFIC tcp port 0 traffic

Table 7.11: Atomic Attack Signatures of DARPA2000

LLDOS1.0			LLDOS2.0.2		
Edge	Sig ID	Sig Value	Edge	Sig ID	Sig Value
E1.1	$SIG_{N1.2,N1.1}$	$I_1 \cup I_2$	E2.1	N/A	N/A
E1.2	$SIG_{N1.3,N1.2}$	$I_3 \cup I_4 \cup I_5$	E2.2	$SIG_{N2.4,N2.2}$	$I_3 \cup I_6 \cup I_7$
E1.3	$SIG_{N1.4,N1.3}$	$I_6 \cup I_7 \cup I_8 \cup I_9 \cup I_{10} \cup I_3$	E2.3	N/A	N/A
E1.4	$SIG_{N1.4,N1.4}$	$I_{11} \cup I_8$	E2.4	$SIG_{N2.5,N2.4}$	$I_3 \cup I_6 \cup I_7 \cup I_8$
E1.5	$SIG_{R,N1.5}$	$I_8 \cup I_{12} \cup I_{13} \cup I_{14} \cup I_{15} \cup I_{16}$	E2.5	$SIG_{R,N2.5}$	$I_8 \cup I_{12} \cup I_5$

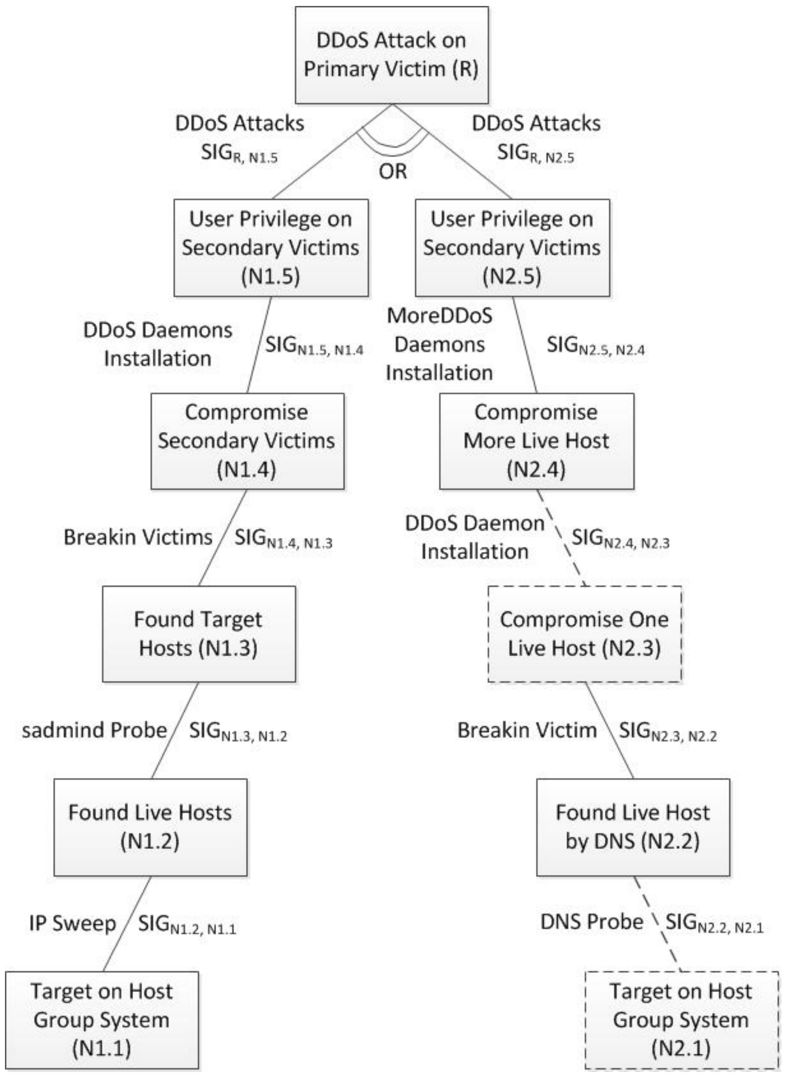

DDoS Attack on Primary Victim (R)
DDoS Attacks SIG$_{R, N1.5}$
OR
DDoS Attacks SIG$_{R, N2.5}$
User Privilege on Secondary Victims (N1.5)
User Privilege on Secondary Victims (N2.5)
DDoS Daemons Installation
SIG$_{N1.5, N1.4}$
MoreDDoS Daemons Installation
SIG$_{N2.5, N2.4}$
Compromise Secondary Victims (N1.4)
Compromise More Live Host (N2.4)
Breakin Victims
SIG$_{N1.4, N1.3}$
DDoS Daemon Installation
SIG$_{N2.4, N2.3}$
Found Target Hosts (N1.3)
Compromise One Live Host (N2.3)
sadmind Probe
SIG$_{N1.3, N1.2}$
Breakin Victim
SIG$_{N2.3, N2.2}$
Found Live Hosts (N1.2)
Found Live Host by DNS (N2.2)
IP Sweep
SIG$_{N1.2, N1.1}$
DNS Probe
SIG$_{N2.2, N2.1}$
Target on Host Group System (N1.1)
Target on Host Group System (N2.1)

Figure 7.5: Constructed Attack Detection Tree of DARPA2000

due there being no Snort alerts generated in Phase 1 and Phase 3 in LLDOS2.0.2 according to Section 7.2.3. In addition, the DDoS attack on the primary victim is the ultimate attack goal of both two datasets, thus it is the root node of the tree. Since the adversary had implemented only one attack operation to achieve each single sub-goal in the dataset, the modelled tree has only two direct branches without any more leaf nodes or additional paths, where achieving the sub-goals of any branch leads to the final goal (root node). The label and the signature information are drawn on both sides of each edge. Each edge contains the atomic attack signature ID as detailed in Table 7.11.

7.3 Experimental Results

Once we have established our testbed and generated ADtT for DARPA2000, the experiments are conducted by executing ATIDS on testbed and replaying

DARPA2000 data sets. The experimental input is the malicious network traffic from either LLDOS1.0 data set or LLDOS2.0.2 data set, while the experimental outputs include the generated high-level alert and the measured on-going intrusion detection metrics. The whole experiments last two weeks from 20^{th} October 2011 to 2^{nd} November 2011, in which we are kept ATIDS executing and replayed the attack data set into the monitored network at any time as we wish.

The general detection process of ATIDS is stated as follows. ATIDS conducts the high-level intrusion detection within each specified time period according to DW setting. The detection process follows strictly on the modelled multi-step attack signatures and further implements the detection uncertainty analysis to allow any wrong detection (Please refer to Chapter 6 for the detailed intrusion detection process). On GUI of ATIDS, it shows the time information of each DW. ATIDS additionally shows attack information, which including *Attack Phase*, *DW start time*, *DW end time* and *the name of high-level attack step*, as soon as any high-level attack step been detected. All the experimental data shown in the tables of this section are obtained from the detection process.

We define two DW settings: (1) *Non-overlapped setting* is configured with $\mathfrak{T} = now$, $\mathbb{S} = 6$ and $\Delta = 6$; (2) *Overlapped setting* is configured with $\mathfrak{T} = now$, $\mathbb{S} = 6$ and $\Delta = 5$. The former DW represents that ATIDS examines Snort generated low-level alerts every 6 seconds and the investigated time period is latest 6-second period without the overlapping between two consecutive DWs. The latter DW also represents that ATIDS examines Snort generated low-level alerts every 6 seconds, but the investigated time period is latest 5-second period plus with an additional overlapped 1-second period between the current DW and the last DW.

However, the time constraint is always exist as the research limitation. The reason is that the relevant detection process only investigates the low-level alerts within the specified time period, no matter it is overlapped or not. Any generated low-level alerts beyond the current DW cannot be examined. Thus, the hidden logic and relation may be difficult fully identified.

We conduct three groups of experiments: (1) AAT based intrusion detection; (2) ADtT based intrusion detection with sequential based assessment; and (3) ADtT based intrusion detection with combination based assessment. The measured experimental results of each group are as follows.

7.3.1 AAT Based Intrusion Detection

With the *AAT Based Intrusion Detection* approach, we have conducted three sets of experiments with two targets. The first two experiment sets target on the detection performance evaluation on “Inside” traffic with both the non-overlapped

DW setting and the overlapped DW setting, while the third experiment set targets on the detection performance evaluation on "DMZ" traffic.

Non-overlapped DW Setting

Table 7.12 and Table 7.13 show the relevant detection results of each attack phase on LLDOS1.0 "Inside" attack traffic and LLDOS2.0.2 "Inside" attack traffic.

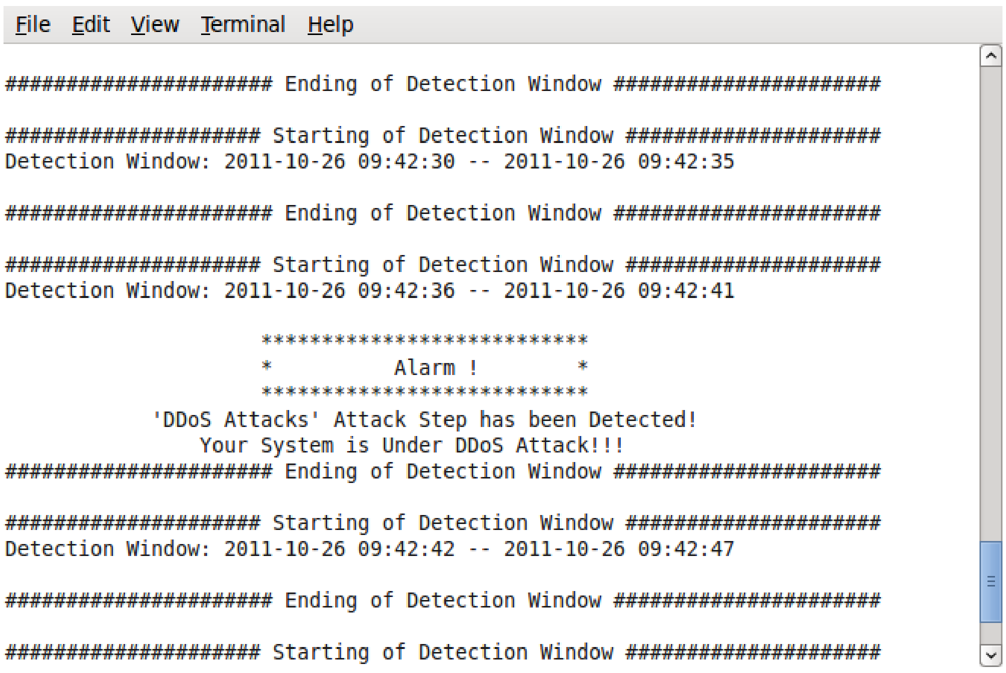
In both tables, the "Detection" column represents the detection results of that phase. The possible detection results are *True*, *False* and *N/A*. *True* indicates that the attack phase is detected through detection process. *False* indicates that the ATIDS has conducted the relevant detection on the attack phase, but the detection result is negative. *N/A* indicates the corresponding detection is unable to be implemented due to the unachieved detection of last attack phase. Then, the "DW Start" column and the "DW End" column represent the start time and the end time of the DW, which is the corresponding detection time window of the attack phase detection. The last "High-Level Alert" column represents the generated high-level alert of each phase according to the edge label from Table 7.9.

It is clear that ATIDS generates the high-level alerts for all modelled five attack phases in LLDOS1.0 Inside data set and three attack phases in LLDOS2.0.2 Inside data set. Figure 7.6 illustrates the snapshot of LLDOS1.0 Phase 5 detection and the corresponding high-level alarm generation on the Ubuntu terminal.

Table 7.12: AAT Based Detection Results of LLDOS1.0 Inside Traffic with Non-overlapped DW Setting

Attack Phase	**Detection**	**DW Start**	**DW End**	**High-Level Alert**
Phase 1	True	2011-10-26 09:41:06	2011-10-26 09:41:11	IP Sweep
Phase 2	True	2011-10-26 09:41:30	2011-10-26 09:41:35	Sadmind Probe
Phase 3	True	2011-10-26 09:41:54	2011-10-26 09:41:59	Breakin Victims
Phase 4	True	2011-10-26 09:42:18	2011-10-26 09:42:23	DDoS Daemons Installation
Phase 5	True	2011-10-26 09:42:36	2011-10-26 09:42:41	DDoS Attacks

Overlapped DW Setting



```
###################### Ending of Detection Window ######################

##################### Starting of Detection Window #####################
Detection Window: 2011-10-26 09:42:30 -- 2011-10-26 09:42:35

###################### Ending of Detection Window ######################

##################### Starting of Detection Window #####################
Detection Window: 2011-10-26 09:42:36 -- 2011-10-26 09:42:41

                     ***************************
                     *          Alarm !        *
                     ***************************
            'DDoS Attacks' Attack Step has been Detected!
                Your System is Under DDoS Attack!!!
###################### Ending of Detection Window ######################

##################### Starting of Detection Window #####################
Detection Window: 2011-10-26 09:42:42 -- 2011-10-26 09:42:47

###################### Ending of Detection Window ######################

##################### Starting of Detection Window #####################
```

Figure 7.6: Output Snapshot of AAT Based Intrusion Detection

Table 7.13: AAT Based Detection Results of LLDOS2.0.2 Inside Traffic with Non-overlapped DW Setting

Attack Phase	Detection	DW Start	DW End	High-Level Alert
Phase 2	True	2011-10-26 09:46:07	2011-10-26 09:46:12	Breakin Victim
Phase 4	True	2011-10-26 09:46:31	2011-10-26 09:41:36	More DDoS Daemon Installation
Phase 5	True	2011-10-26 09:46:55	2011-10-26 09:47:00	DDoS Attacks

This experiment set investigates the detection performance with the overlapped DW scenario. The measured detection results of LLDOS1.0 data set and LLDOS2.0.2 data set are shown in Table 7.14 and Table 7.15.

Same as the detection results in the Non-overlapped DW Setting, all of the modelled atomic attacks have been detected successfully. However, note that the detection of LLDOS1.0 Phase 1 and LLDOS2.0.2 Phase 2 are repeated once after the first detected DW. This demonstrates that ATIDS has the capability to detect the atomic attack with the historic information. In addition, any historic information is helpful to identify any hidden relation between the past and the on-going intrusion.

DMZ Traffic

Table 7.14: AAT Based Detection Results of LLDOS1.0 Inside Traffic with Overlapped DW Setting

Attack Phase	**Detection**	**DW Start**	**DW End**	**High-Level Alert**
Phase 1	True	2011-10-31 08:36:51	2011-10-31 08:36:56	IP Sweep
Phase 1	True	2011-10-31 08:36:56	2011-10-31 08:37:01	IP Sweep
Phase 2	True	2011-10-31 08:37:11	2011-10-31 08:37:16	Sadmind Probe
Phase 3	True	2011-10-31 08:37:31	2011-10-31 08:37:36	Breakin Victims
Phase 4	True	2011-10-31 08:37:56	2011-10-31 08:38:01	DDoS Daemons Installation
Phase 5	True	2011-10-31 08:39:01	2011-10-31 08:39:06	DDoS Attacks

Table 7.15: AAT Based Detection Results of LLDOS2.0.2 Inside Traffic with Overlapped DW Setting

Attack Phase	**Detection**	**DW Start**	**DW End**	**High-Level Alert**
Phase 2	True	2011-10-31 08:39:36	2011-10-31 08:39:41	Breakin Victim
Phase 2	True	2011-10-31 08:39:41	2011-10-31 08:39:46	Breakin Victim
Phase 4	True	2011-10-31 08:40:21	2011-10-31 08:40:26	More DDoS Daemon Installation
Phase 5	True	2011-10-31 08:40:36	2011-10-31 08:40:41	DDoS Attacks

The experiment set with DMZ traffic applies non-overlapped DW setting. Though the applied "DMZ" traffic files and the applied "Inside" traffic files are captured simultaneously on the same network traffic, there are some differences between them, such as the total number of captured packets in each phase, the number of incident types. Take Phase 5 of LLDOS1.0 for example, the modelled attack signature has six modelled incident types in the "Inside" file, but, there are only three incident types in the "DMZ" file.

The detection results of DMZ traffic are shown in Table 7.16 and Table 7.17 to represent LLDOS1.0 data set and LLDOS2.0.2 data set. In Table 7.16, the first four attack phases have been detected with "True" results, while the last attack phase had been evaded. In Table 7.17, the detection results of LLDOS2.0.2 DMZ traffic show that only the first modelled attack phase had been detected, the second modelled attack phase had been evaded due to the unmatched signature, while the last attack phase cannot be achieved since the Phase 4 is unachieved yet.

Table 7.16: AAT Based Detection Results of LLDOS1.0 DMZ Traffic

Attack Phase	**Detection**	**DW Start**	**DW End**	**High-Level Alert**
Phase 1	True	2011-10-26 09:48:30	2011-10-26 09:48:35	IP Sweep
Phase 2	True	2011-10-26 09:48:36	2011-10-26 09:48:41	Sadmind Probe
Phase 3	True	2011-10-26 09:48:54	2011-10-26 09:48:59	Breakin Victims
Phase 4	True	2011-10-26 09:49:12	2011-10-26 09:49:17	DDoS Daemons Installation
Phase 5	False	N/A	N/A	N/A

Table 7.17: AAT Based Detection Results of LLDOS2.0.2 DMZ Traffic

Attack Phase	**Detection**	**DW Start**	**DW End**	**Alert**
Phase 2	True	2011-10-26 09:52:48	2011-10-26 09:52:53	Breakin Victim
Phase 4	False	N/A	N/A	N/A
Phase 5	False	N/A	N/A	N/A

Summary

According to the above generated alerts information, we summarise that the proposed *AAT based intrusion detection* approach can detect the atomic attacks which satisfy the modelled atomic attack signatures from the attack traffic, irrespective of whether the applied DW has been set with either the non-overlapped or overlapped DW setting.

With the non-overlapped DW setting, ATIDS detects the atomic attacks by investigating only the Snort generated alerts within that time range without binding any historic information. If DW contains all of the relevant incidents, ATIDS generates the high-level alerts as TP. Otherwise, it generates FN.

With the overlapped DW setting, ATIDS is able to detect any atomic attacks which are low-level incidents that happened in the two consecutive DWs. Thus, the reason why the Phase 1 high-level alert in LLDOS1.0 and the Phase 2 high-level alert in LLDOS2.0.2 repeat is because the low-level incidents are generated in the exactly overlapped part.

However, ATIDS lacks detection capability to identify any signature non-fully matched atomic attacks as demonstrated by the experiment set on DMZ traffic files. These attacks can be regarded as the uncertainties in the detection process. Without access to the traffic ground truth, the possible reasons of the undetected incidents may be due to the FNs generated by Snort or the inexistence of incidents from traffic. In practice, it is possible for a sophisticated adversary to achieve any atomic attacks by using only a subset of the incidents modelled in the atomic attack plus other unmodelled incidents. Therefore, the next detection approaches will conduct the high-level atomic attack detection and additionally try to address the detection uncertainty as the problem of incidents not fully modelled within atomic attacks (that is, no relevant incidents within attack traffic, low-level intrusion detector unable to detect incidents).

7.3.2 ADtT Based Intrusion Detection with Sequential Based Detection Uncertainty Assessment

In this category, the attack data sets are detected by applying the *ADtT based intrusion detection* with *sequential based assessment* approach as described in Section 5.2.4. Generally, the main idea of this approach is to determine the achievement of every single modelled incident within the atomic attack through the *sequential detection uncertainty assessment* to cause the achievement of an attack tree edge.

The main process of this approach is as follows. ATIDS firstly detects the achievement of modelled atomic attack signatures in each DW as the *AAT based approach* does. If any of the atomic attacks have been matched, ATIDS generates a "caution" instead of a high-level alarm to indicate the signature matching. Next, the *sequential based approach* implements the detection uncertainty assessment on each modelled incident within the atomic attack based on the obtained real-time assessment factors. The assessment mainly measures the identified incidents with evidence to trigger the alert. If the measured evidence supports the identified incident, then, ATIDS labels that incident to be achieved. Otherwise, ATIDS labels it as unachieved. Once all of the modelled incidents within an atomic attack have been achieved, ATIDS generates a high-level alert to confirm the detection on the corresponding edge. If not, ATIDS continually examines the currently unachieved incident without high-level alert generation.

Dissimilar to the previous tables in the last approach, the tables in this experiment set record more detailed information. The "Detection" column shows the results of the basis signature matching detection. The "Assessment" column shows the results of the *sequential based assessment*. Two "True" values from both "Detection" and "Assessment" columns indicate the achievement of the attack phase. Otherwise, the attack phase cannot be achieved by ATIDS, and the following attack phase cannot be examined. In addition, the "Achieved Incidents" column and the "Unachieved Incidents" column list the relevant index of achieved and unachieved modelled incidents during the assessment. Moreover, the "DW Start" column and the "DW End" column represent the corresponding DW start and end time.

Non-overlapped DW Setting

Table 7.18 and Table 7.19 show the detection results on LLDOS1.0 Inside attack traffic and LLDOS2.0.2 Inside attack traffic by applying *ADtT based intrusion detection* with *sequential based assessment* and the non-overlapped DW setting.

Table 7.18: ADtT Sequential Based Detection Results of LLDOS1.0 Inside Traffic with Non-overlapped DW Setting

Attack Phase	**Dete- ction**	**Asse- ssment**	**Achieved Incidents**	**Unachieved Incidents**	**DW Start**	**DW End**
Phase 1	True	True	I_1, I_2	NULL	2011-10-31 16:50:25	2011-10-31 16:50:30
Phase 2	True	True	I_3, I_4, I_5	NULL	2011-10-31 16:50:37	2011-10-31 16:50:42
Phase 3	True	False	I_6, I_7, I_8, I_9	I_{10}	2011-10-31 16:50:55	2011-10-31 16:51:00
Phase 4	N/A	N/A	N/A	N/A	N/A	N/A
Phase 5	N/A	N/A	N/A	N/A	N/A	N/A

According to Table 7.18, Phase 1 and Phase 2 within LLDOS1.0 Inside attack traffic have been successfully detected since the atomic attack signatures have been matched and all of the modelled incidents (that is, I_1 and I_2 in Phase 1, I_3, I_4 and I_5 in Phase 2) have been determined. However, in Phase 3, I_{10} can not be achieved through the computation of the detection uncertainty assessment. Therefore, I_{10} has been recorded in the "Unachieved Incidents" column. Additionally, "False" value has been generated for the whole edge assessment due to not all of the incidents achieved in current DW. The unachieved Phase 3 causes the detection process to continually check the current phase. Any further detection on Phase 4

Table 7.19: ADtT Sequential Based Detection Results of LLDOS2.0.2 Inside Traffic with Non-overlapped DW Setting

Attack Phase	**Dete-ction**	**Asse-ssment**	**Achieved Incidents**	**Unachieved Incidents**	**DW Start**	**DW End**
Phase 2	True	False	NULL	I_3	2011-10-26 16:53:26	2011-10-26 16:53:31
Phase 4	N/A	N/A	N/A	N/A	N/A	N/A
Phase 5	N/A	N/A	N/A	N/A	N/A	N/A

and Phase 5 have been evaded.

Table 7.19 shows the detection results on LLDOS2.0.2 Inside attack traffic. The assessment of first attack phase has failed because the first modelled incident I_3 cannot be achieved based on the measured assessment factors, even though the signatures are matched.

Overlapped DW Setting

With the utilisation of the overlapped DW setting, Table 7.20 and Table 7.21 display the obtained results of this approach. Same as the last experiment with the non-overlapped DW setting, Phase 3 in LLDOS1.0 and Phase 2 in LLDOS2.0.2 have been achieved with failure due to the same reason: I_{10} in Phase 3 and I_3 are not determined according to the detection uncertainty assessment.

As one of the possible characteristics of the overlapped scheme, Phase 1 and Phase 3 in LLDOS1.0 repeat in the two consecutive DWs since the atomic attacks are conducted in the overlapped part.

DMZ Traffic

The detection results of LLDOS1.0 DMZ traffic and LLDOS2.0.2 DMZ traffic are shown in Table 7.22 and Table 7.23. Only the first attack phase in LLDOS1.0 has been detected and achieved by ATIDS's process.

Summary

We conclude that the proposed *ADtT based intrusion detection* with *sequential based assessment* approach generates poor results. Since this approach requires that all of the modelled incidents within the atomic attack must be achieved

Table 7.20: ADtT Sequential Based Detection Results of LLDOS1.0 Inside Traffic with Overlapped DW Setting

Attack Phase	**Dete-ction**	**Asse-ssment**	**Achieved Incidents**	**Unachieved Incidents**	**DW Start**	**DW End**
Phase 1	True	True	I_1, I_2	NULL	2011-10-31 16:54:38	2011-10-31 16:54:43
Phase 1	True	True	I_1, I_2	NULL	2011-10-31 16:54:43	2011-10-31 16:54:48
Phase 2	True	True	I_3, I_4, I_5	NULL	2011-10-31 16:54:58	2011-10-31 16:55:03
Phase 3	True	False	I_6, I_7, I_8, I_9	I_{10}	2011-10-31 16:55:13	2011-10-31 16:55:18
Phase 3	True	False	I_6, I_7, I_8, I_9	I_{10}	2011-10-31 16:55:18	2011-10-31 16:55:23
Phase 4	N/A	N/A	N/A	N/A	N/A	N/A
Phase 5	N/A	N/A	N/A	N/A	N/A	N/A

Table 7.21: ADtT Sequential Based Detection Results of LLDOS2.0.2 Inside Traffic with Overlapped DW Setting

Attack Phase	**Dete-ction**	**Asse-ssment**	**Achieved Incidents**	**Unachieved Incidents**	**DW Start**	**DW End**
Phase 2	True	False	NULL	I_3	2011-10-26 16:57:40	2011-10-26 16:57:45
Phase 4	N/A	N/A	N/A	N/A	N/A	N/A
Phase 5	N/A	N/A	N/A	N/A	N/A	N/A

through both the signature matching and detection uncertainty assessment, any of the unachieved incidents handicap the successful detection of the atomic attack. Therefore, the limitation of this approach is that once any of the modelled incidents cannot been achieved, the detection of that whole attack phase will fail.

According to the above generated alerts information, the main reason of the unachieved incident is due to the weakness of the measured evidence. However, it is difficult to conclude that the obtained weak evidence cannot support the achievement of the incident, or any missed evidence can support the unachievement of the incident. Therefore, we will solve this problem in the next experiment set by applying the *combination based assessment* approach.

Table 7.22: ADtT Sequential Based Detection Results of LLDOS1.0 DMZ Traffic

Attack Phase	**Dete-ction**	**Asse-ssment**	**Achieved Incidents**	**Unachieved Incidents**	**DW Start**	**DW End**
Phase 1	True	True	I_1, I_2	NULL	2011-10-31 16:59:16	2011-10-31 16:59:21
Phase 2	True	False	I_3, I_4	I_5	2011-10-31 16:59:40	2011-10-31 16:59:45
Phase 3	N/A	N/A	N/A	N/A	N/A	N/A
Phase 4	N/A	N/A	N/A	N/A	N/A	N/A
Phase 5	N/A	N/A	N/A	N/A	N/A	N/A

Table 7.23: ADtT Sequential Based Detection Results of LLDOS2.0.2 DMZ Traffic

Attack Phase	**Dete-ction**	**Asse-ssment**	**Achieved Incidents**	**Unachieved Incidents**	**DW Start**	**DW End**
Phase 2	True	False	NULL	I_3	2011-10-26 17:01:00	2011-10-26 17:01:05
Phase 4	N/A	N/A	N/A	N/A	N/A	N/A
Phase 5	N/A	N/A	N/A	N/A	N/A	N/A

7.3.3 ADtT Based Intrusion Detection with Combination Based Detection Uncertainty Assessment

In this experiment set, the attack data sets are detected by applying the *ADtT based intrusion detection* with *combination based assessment* approach as described in Section 5.2.5. The main background idea of this approach is to determine the detection of an atomic attack with general detection uncertainty analysis. Specifically, the atomic attack detection is examined by fusing the multiple sets of measured evidence which are provided from each single incident.

For each atomic attack detection, ATIDS firstly detects the achievement of modelled atomic attack signatures in each DW as *AAT based approach* does. Next, ATIDS obtains the assessment factors as the evidence to measure the certainty of every modelled incident as the *ADtT sequential based approach* does. Then, instead of the determination on each incident, ATIDS *combination based approach* applies the measured results as another evidence set about uncertainty to determine the detection of the atomic attack. Through this way, it is helpful to determine the achievement of atomic attack if any of the incidents are missed or have been detected falsely by Snort. Once the atomic attack has been achieved through both signature matching and detection uncertainty analysis, ATIDS generates the high-level alert to confirm the detection on the corresponding edge. If not, ATIDS continually examines the currently unachieved incident without high-level alert generation. Furthermore, ATIDS implements the QoD mechanism to monitor and record the on-going intrusion detector's detection performance and the on-going intrusion progress.

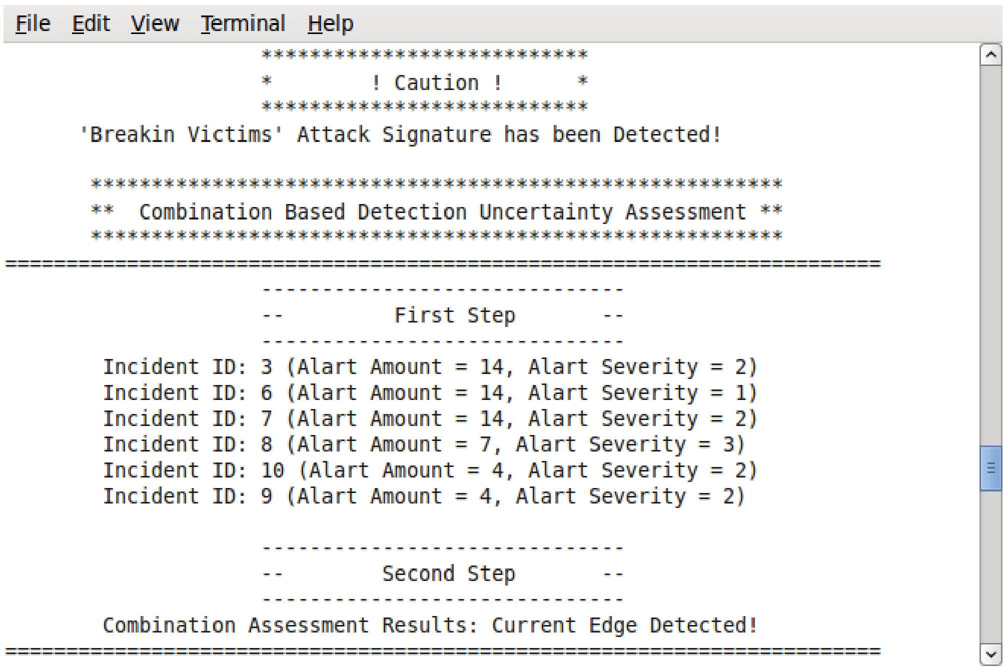
In this experiment set, we conduct the experiments by applying the non-overlapped DW setting in ATIDS to detect the Inside traffic files, and additionally detect the DMZ traffic files.

Non-overlapped DW Setting

Non-overlapped DW setting is applied first to evaluate the detection of *ADtT Based Intrusion Detection* with *Combination Based Detection Uncertainty Assessment*. The detection results on both LLDOS1.0 data set and LLDOS2.0.2 data set are as follows.

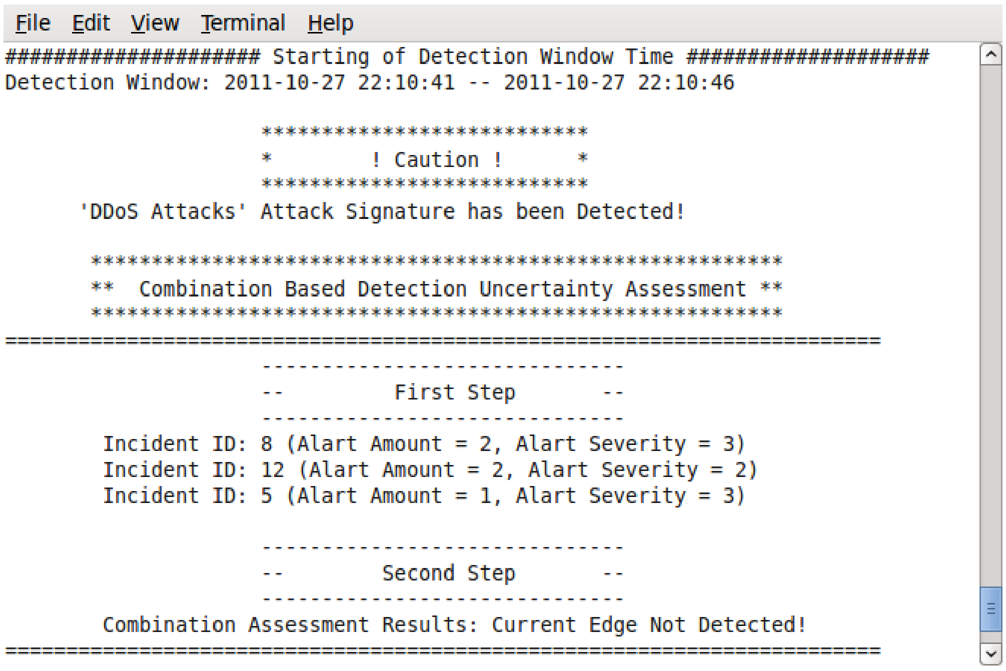
In contrast to detection results of ADtT sequential based detection approach, the *combination based approach* has better detection performance as the first four attack phases have been identified. In the last phase, though the atomic attack signature has been matched, the detection uncertainty assessment determines that the result is weak to generate the final alarm. Specifically, there are three modelled incidents I_8, I_{12} and I_{16}, each of which has only one generated low-level alert. Thus, the proposed D-S evidence based combination analysis cannot support the atomic attack achievement due to three sets of weak evidence from these modelled incidents. Table 7.24 displays the detection results on LLDOS1.0 Inside traffic. Table 7.25 shows the detection results on LLDOS2.0.2 Inside traffic. Although the last attack phase is not achieved, the first two atomic attacks have been detected and achieved. Figure 7.7 illustrates the snapshot on the relevant detection uncertainty assessment results on LLDOS1.0 Phase 3. Figure 7.8 illustrates the output snapshot on LLDOS2.0.2 Phase 5 detection.

The measured QoD metrics of each attack phase in LLDOS1.0 are shown in Table 7.26. In addition, the measured on-going QoD metrics of LLDOS2.0.2 are displayed in Table 7.27. The measurement of QoD metrics are based on the proposed computation methods in Section 5.1. The logical steps based metrics (that is, StD, StG and PtG) are computed according to Equation (5.1), Equation (5.2) and Equation (5.3). The time based metrics (that is, TtD, TtCS and TtG) are computed according to Equation (5.6), Equation (5.7) and Equation (5.8). While, the alert based metrics (that is, NAT, MAN and MAS) are computed according to



```
                     ***************************
                     *        ! Caution !      *
                     ***************************
      'Breakin Victims' Attack Signature has been Detected!

       *********************************************************
       **  Combination Based Detection Uncertainty Assessment **
       *********************************************************
========================================================================
                     -----------------------------
                     --         First Step       --
                     -----------------------------
        Incident ID: 3 (Alart Amount = 14, Alart Severity = 2)
        Incident ID: 6 (Alart Amount = 14, Alart Severity = 1)
        Incident ID: 7 (Alart Amount = 14, Alart Severity = 2)
        Incident ID: 8 (Alart Amount = 7, Alart Severity = 3)
        Incident ID: 10 (Alart Amount = 4, Alart Severity = 2)
        Incident ID: 9 (Alart Amount = 4, Alart Severity = 2)

                     -----------------------------
                     --        Second Step       --
                     -----------------------------
        Combination Assessment Results: Current Edge Detected!
========================================================================
```

Figure 7.7: Output Snapshot of ADtT Combination Based Intrusion Detection on Phase 3



```
#################### Starting of Detection Window Time ####################
Detection Window: 2011-10-27 22:10:41 -- 2011-10-27 22:10:46

                     ***************************
                     *        ! Caution !      *
                     ***************************
      'DDoS Attacks' Attack Signature has been Detected!

       *********************************************************
       **  Combination Based Detection Uncertainty Assessment **
       *********************************************************
========================================================================
                     -----------------------------
                     --         First Step       --
                     -----------------------------
        Incident ID: 8 (Alart Amount = 2, Alart Severity = 3)
        Incident ID: 12 (Alart Amount = 2, Alart Severity = 2)
        Incident ID: 5 (Alart Amount = 1, Alart Severity = 3)

                     -----------------------------
                     --        Second Step       --
                     -----------------------------
        Combination Assessment Results: Current Edge Not Detected!
========================================================================
```

Figure 7.8: Output Snapshot of ADtT Based Intrusion Detection on LLDOS2.0.2 Phase 5

Table 7.24: ADtT Combination Based Detection Results of LLDOS1.0 Traffic with Non-overlapped DW Setting

Attack Phase	**Detection**	**Assessment**	**DW Start**	**DW End**	**High-Level Alert**
Phase 1	True	True	2011-10-27 22:02:11	2011-10-27 22:02:16	IP Sweep
Phase 2	True	True	2011-10-27 22:02:23	2011-10-27 22:02:28	Sadmind Probe
Phase 3	True	True	2011-10-27 22:02:35	2011-10-27 22:02:40	Breakin Victims
Phase 4	True	True	2011-10-27 22:02:53	2011-10-27 22:02:58	DDoS Daemons Installation
Phase 5	True	False	2011-10-27 22:03:23	2011-10-27 22:03:28	N/A

Table 7.25: ADtT Combination Based Detection Results of LLDOS2.0.2 Inside Traffic with Non-overlapped DW Setting

Attack Phase	**Detection**	**Assessment**	**DW Start**	**DW End**	**High-Level Alert**
Phase 2	True	True	2011-10-27 22:10:11	2011-10-27 22:10:16	Breakin Victim
Phase 4	True	True	2011-10-27 22:10:29	2011-10-27 22:10:34	More DDoS Daemon Installation
Phase 5	True	False	2011-10-27 22:10:41	2011-10-27 22:10:46	N/A

Equation (5.11), Equation (5.12) and Equation (5.13).

In this experiment, we assign $T_{min} = 10$ sec to avoid unnecessary computation complexity and unreasonable attack time consumption. The "Edge Metrics" columns display the measured metrics about the alert-based information from the "Detector Module". The "Node Metrics" columns display the measured logic and time metrics according to the modelled tree information with ADtT. The "Phase" column in table denotes the corresponding atomic attack on edge and the "NodeID" column denotes the corresponding node of each edge ("Phase"). Since the achievement of parent node indicates the compromise of the (sub)goal, we assign the node metrics of parent node on the same row with the "Phase" (edge). Note that we initialise the leaf nodes of the bottom edges as the abstracted starting

Table 7.26: Measured QoD Metrics of LLDOS1.0 Inside Traffic Detection

Phase	Edge Metrics			NodeID	Node Metrics					
	NAT	MAN	MAS		TtD	TtCS	TtG	StD	StG	PtG
					(Unit: Second)			(Unit: Step)		
P1.1	2	20.00	3.00	N1.1	0	0	50	0	5	0.00
				N1.2	10	10	40	1	4	1.00
P1.2	3	50.33	2.46	N1.3	22	12	38	2	3	1.00
P1.3	6	9.50	1.88	N1.4	34	12	26	3	2	1.00
P1.4	3	8.00	1.54	N1.5	52	18	8	4	1	1.00
P1.5	6	289.50	2.67	R	82	30	0	5	0	1.00

Table 7.27: Measured QoD Metrics of LLDOS2.0.2 Inside Traffic Detection

Phase	Edge Metrics			NodeID	Node Metrics					
	NAT	MAN	MAS		TtD	TtCS	TtG	StD	StG	PtG
					(Unit: Second)			(Unit: Step)		
P2.2	3	2.00	1.67	L2.2	0	0	30	0	3	0.00
				L2.4	10	10	20	1	2	1.00
P2.4	4	2.00	2.00	L2.5	28	18	12	2	1	1.00
P2.5	3	1.67	2.60	R	40	12	0	3	0	1.00

point of attack and store the leaf node metrics on the row of the bottom edges. ATIDS generates a separated alert with a set of corresponding QoD metrics in each DW.

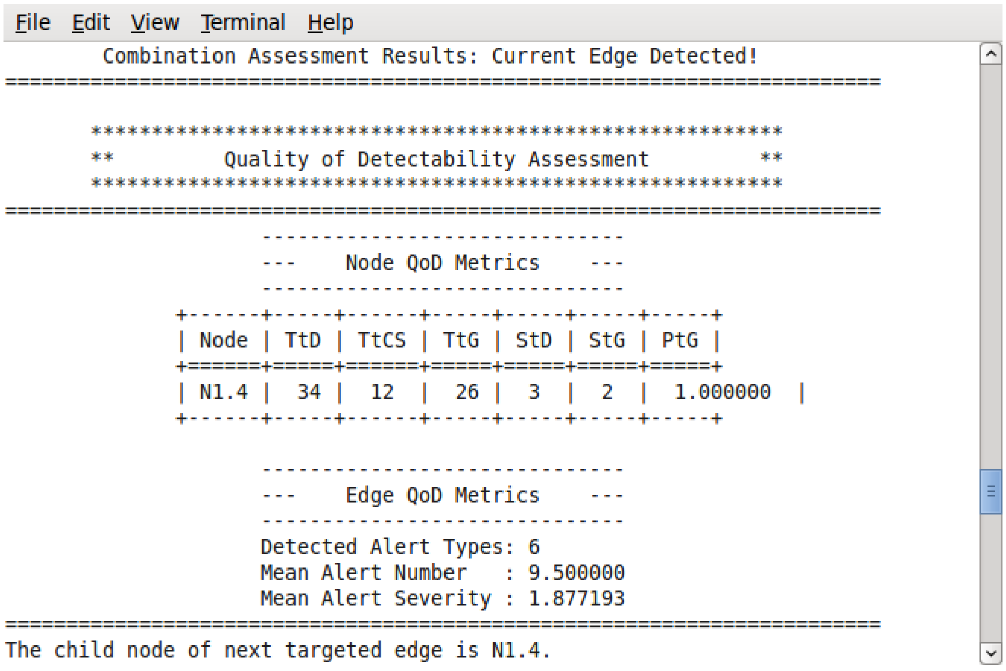
According to the measured edge metrics from the *Edge Metrics* columns, the system administrator can quickly have a general idea about the attack traffic. As the measured NAT is exactly same as the modelled incident type number on each edge, the system administrator can know that the adversary follows the exactly modelled attack process to compromise the system without any other attack attempts. MAN and MAS metrics can assist the system administrator to understand the average attack incident number and the seriousness of the detected attack incidents. Take Phase 5 from Table 7.26 for example, the measured MAN = 289.50 and MAS = 2.67. The system administrator knows that there are lots of malicious traffic with low risk level, which may be DoS/DDoS attack. While the generated high-level alert may confirm his understanding of the attack process.

From the *Node Metrics* columns, it is clear that there is one incremental factor in StD metric column and one decremental factor in StG metric column from the start phase to the end phase in both datasets. With the progress of the attack, the system administrator can identify that the attack is getting close to the ultimate goal of the simulated attack. Take Phase 4 for example, the measured StD = 4 steps and StG = 1 step. Thus, the system administrator can determine that the adversary had already achieved the 4^{th} attack step and with only one step to compromise the whole system in attack scenario LLDOS1.0.

In columns of M_T, the value of TtD follows the increasing tendency and the value of TtG follows the decreasing tendency. Once ATIDS detects the achievement of the first attack phase, the corresponding metrics can be measured on the parent node *N1.2*. Since the values of TtD and TtG have been obtained with the assistance of T_{min}, ATIDS simply assumes that the initial values of TtD, TtCS and TtG are 0, 0 and 50, respectively. As the attack progress, the values of TtD, TtCS and TtG are dynamically updated till the achievement of the root node. We can also identify the hidden relation between these three metrics. The current value of TtD equals to the sum of the last measured TtD value with current TtCS value (for example, TtD = 34 sec in *Phase 3*, TtCS = 18 sec in *Phase 4*, so, TtD = 34 + 18 = 52 sec in *Phase 4*). This represents that the current detection time since the beginning is based on the detection time cost of the last phase and the current detection time consumption. In addition, the current value of TtG equals to the difference between the previously measured TtG value in last phase with current TtCS value (for example, TtG = 34 sec in *Phase 3*, TtCS = 8 sec in *Phase 4*, so, TtG = 34 − 18 = 16 sec in *Phase 4*). This represents that the achievement of the current phase is closer to the ultimate than the last phase. Though TtCS and T_{min} are similar since both of them represent the time on each single edge, we found that there is no constraint between the actually measured TtCS and the theoretically assumed T_{min}. In order to better assign a value on T_{min}, we suppose that there should be one justifiable mechanism on T_{min} as the threshold according to the feedback from the detection result and the measured metrics value. Figure 7.9 illustrates the snapshot on the relevant QoD measured results on the LLDOS1.0 Phase 3.

DMZ Traffic

Since some attack phases in DMZ data set (for example, Phase 5 in LLDOS1.0) have incomplete incidents compared with the modelled incidents in the built Attack Detection Tree, thus, the assistance of DMZ traffic can help us to identify the detection capability of the *combination based detection* with the missing alerts as uncertainty. It is possible for us to regard the missed incidents as the unable to be detected incidents since Snort cannot generate alerts fort them. In that case, the



```
        Combination Assessment Results: Current Edge Detected!
========================================================================

       ********************************************************
       **         Quality of Detectability Assessment        **
       ********************************************************
========================================================================
                     -----------------------------
                     ---    Node QoD Metrics    ---
                     -----------------------------
              +------+-----+------+-----+-----+-----+-----+
              | Node | TtD | TtCS | TtG | StD | StG | PtG |
              +======+=====+======+=====+=====+=====+=====+
              | N1.4 |  34 |  12  |  26 |  3  |  2  |  1.000000  |
              +------+-----+------+-----+-----+-----+-----+

                     -----------------------------
                     ---    Edge QoD Metrics    ---
                     -----------------------------
                     Detected Alert Types: 6
                     Mean Alert Number   : 9.500000
                     Mean Alert Severity : 1.877193
========================================================================
The child node of next targeted edge is N1.4.
```

Figure 7.9: QoD Output Snapshot of ADtT Combination Based Intrusion Detection on LLDOS1.0 Phase 3

assessment factors can be assigned with 0 for the alert amount, while maintaining the value of the alert severity. The results are stated as follows.

Table 7.28 and Table 7.29 show the measured detection results and QoD results on LLDOS1.0 DMZ files. Figure 7.10 illustrates the output snapshot on the DMZ Phase 5 detection. As Snort only generates the alerts for I_8 and I_{16} in that phase, the numbers of the undetected incidents are assigned with 0. Through D-S evidence assessment, Phase 5 has not been achieved according to Snort generated evidence. Table 7.30 and Table 7.31 show the measured detection results and QoD results on LLDOS2.0.2 DMZ files. Figure 7.11 illustrates the output snapshot on the DMZ Phase 4 detection.

Summary

According to the obtained detection results, we can summarise that the proposed *ADtT based intrusion detection* with the *combination based assessment* approach can detect most of the modelled atomic attacks by handling the uncertainty issues during the detection process. Since this approach fuses all of measured evidence from each single modelled incidents within the atomic attack, the D-S evidence based assessment determines the detection of the atomic attack by fusing all of the evidence into the ultimate evidence to support the atomic attack detection. Therefore, the *combination based assessment* approach is more flexible and more general to determine the detection achievement compared with the

Table 7.28: ADtT Combination Based Detection Results of LLDOS1.0 DMZ Traffic

Attack Phase	**Detection**	**Assessment**	**DW Start**	**DW End**	**High-Level Alert**
Phase 1	True	True	2011-11-01 11:18:18	2011-11-01 11:18:23	IP Sweep
Phase 2	True	True	2011-11-01 11:18:30	2011-11-01 11:18:35	Sadmind Probe
Phase 3	True	True	2011-11-01 11:18:42	2011-11-01 11:18:47	Breakin Victims
Phase 4	True	True	2011-11-01 11:18:55	2011-11-01 11:19:00	DDoS Daemons Installation
Phase 5	True	False	2011-11-01 11:19:07	2011-11-01 11:19:12	N/A

sequential based assessment.

However, we notice that some of the modelled incidents within the typical undetected atomic attack (that is, in LLDOS1.0 Phase 5 Inside traffic, the generated low-level alert number of I_8, I_{12} and I_{16} are 1, 1, 1, respectively) only have a few number of the generated low-level alerts. In addition, according to Table 7.6, we notice that the applied values of P_{IDS} in the assessment of the typical undetected atomic attacks (for example, $P_{IDS} = 53.87\%$ in LLDOS1.0 Phase 5, $P_{IDS} = 34.48\%$ in LLDOS2.0.2 Phase 4) are lower than the values of detected atomic attacks (for example, $P_{IDS} = 100\%$ in LLDOS1.0 Phase 1, $P_{IDS} = 100\%$ in LLDOS2.0.2 Phase 2). As these applied values are required for the D-S evidence based detection uncertainty assessment, we will discuss whether these different values may affect the assessment results in the next section.

7.4 Discussion

We discuss two points in this section based on two settings within the *detection uncertainty assessment*. The first discussion targets on how the different detector precision P_{IDS} value within the *basic probability assignment* process may affect the assessment result. The second discussion targets on how the different threshold γ_2 value within in the *alert amount membership function* may affect the assessment result.

Our discussions are conducted based on the *ADtT combination based intrusion detection approach* on LLDOS1.0 Inside traffic. Every discussion is classified into

Table 7.29: Measured QoD Metrics of LLDOS1.0 DMZ Traffic Detection

Phase	Edge Metrics			NodeID	Node Metrics					
	NAT	MAN	MAS		TtD	TtCS	TtG	StD	StG	PtG
					(Unit: Second)			(Unit: Step)		
P1.1	2	392.50	3.00	N1.1	0	0	50	0	5	0.00
				N1.2	10	10	40	1	4	1.00
P1.2	3	45.00	1.96	N1.3	22	12	38	2	3	1.00
P1.3	6	23.00	1.99	N1.4	35	13	25	3	2	1.00
P1.4	2	5.50	1.55	N1.5	47	12	13	4	1	1.00
P1.5	N/A	N/A	N/A	R	N/A	N/A	N/A	N/A	N/A	N/A

Table 7.30: ADtT Combination Based Detection Results of LLDOS2.0.2 DMZ Traffic

Attack Phase	**Detection**	**Assessment**	**DW Start**	**DW End**	**High-Level Alert**
Phase 2	True	True	2011-11-01 11:11:28	2011-11-01 11:11:33	Breakin Victim
Phase 4	True	False	2011-11-01 11:11:40	2011-11-01 11:11:45	N/A
Phase 5	N/A	N/A	N/A	N/A	N/A

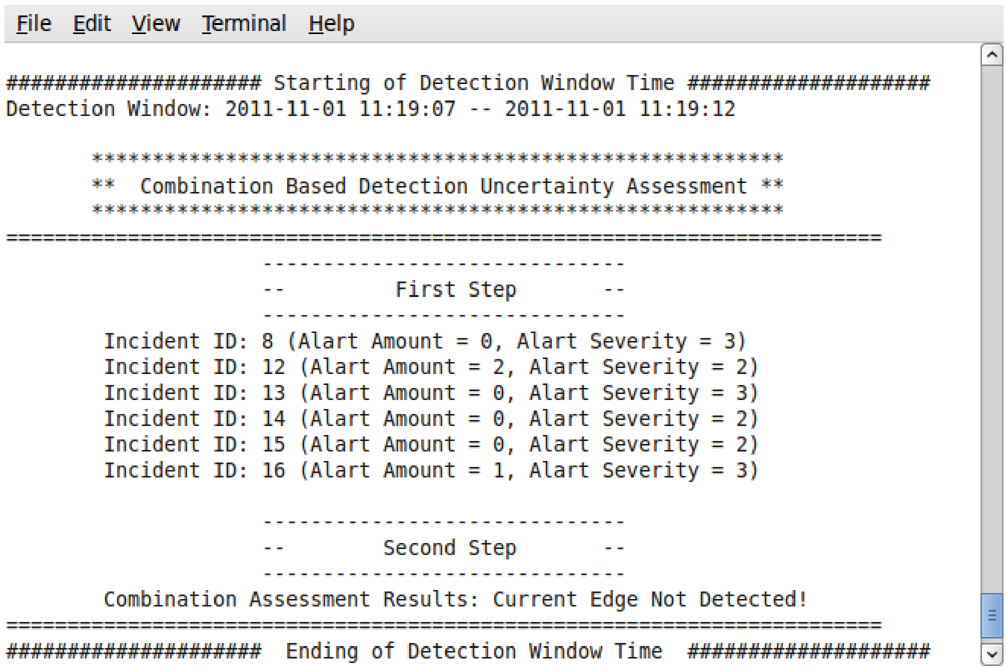
two steps. The first step discusses the affect on the measured results from the *basic probability assignment* process, while the second step discusses the affect on the measured results from the *fusion* process.

7.4.1 Detector Precision Value Setting for Basic Probability Assignment

The *basic probability assignment* with the detector precision P_{IDS} value has been described in Section 5.2.3. P_{IDS} represents the detection precision of the low-level detector, the value of P_{IDS} determines the certainty on the correctly classified low-level incidents and low-level normal actions. While the value of $1 - P_{IDS}$ represents the uncertainty on the incorrectly classification. Since the applied P_{IDS} value may be assigned with different values within ADtT based intrusion detection approaches, this subsection discusses how the different P_{IDS} values may affect the

Table 7.31: Measured QoD Metrics of LLDOS2.0.2 DMZ Detection

Phase	Edge Metrics			NodeID	Node Metrics					
	NAT	MAN	MAS		TtD	TtCS	TtG	StD	StG	PtG
					(Unit: Second)			(Unit: Step)		
P2.2	3	2.00	1.67	N2.2	0	0	30	0	3	0.00
				N2.4	10	10	20	1	2	1.00
P2.4	N/A	N/A	N/A	N2.5	N/A	N/A	N/A	N/A	N/A	N/A
P2.5	N/A	N/A	N/A	R	N/A	N/A	N/A	N/A	N/A	N/A



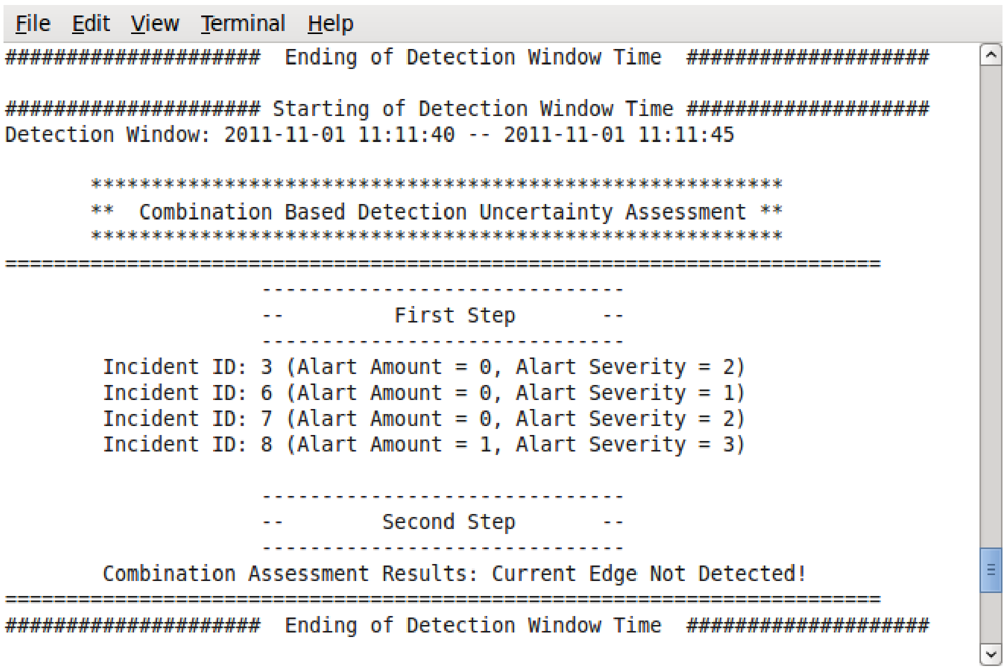
```
#################### Starting of Detection Window Time ####################
Detection Window: 2011-11-01 11:19:07 -- 2011-11-01 11:19:12

       ********************************************************
       **  Combination Based Detection Uncertainty Assessment **
       ********************************************************
========================================================================
                     -----------------------------
                     --         First Step      --
                     -----------------------------
        Incident ID: 8 (Alart Amount = 0, Alart Severity = 3)
        Incident ID: 12 (Alart Amount = 2, Alart Severity = 2)
        Incident ID: 13 (Alart Amount = 0, Alart Severity = 3)
        Incident ID: 14 (Alart Amount = 0, Alart Severity = 2)
        Incident ID: 15 (Alart Amount = 0, Alart Severity = 2)
        Incident ID: 16 (Alart Amount = 1, Alart Severity = 3)

                     -----------------------------
                     --        Second Step      --
                     -----------------------------
        Combination Assessment Results: Current Edge Not Detected!
========================================================================
####################  Ending of Detection Window Time  ####################
```

Figure 7.10: Output Snapshot of ADtT Combination Based Intrusion Detection on LLDOS1.0 DMZ Phase 5

atomic attack detection results.

We set P_{IDS} with three typical probabilities: the *maximal*, the *medium* and the *minimal*. Note that all these values should be *measurable* and *reasonable*. For the former, the probabilities should be obtained directly according to any relevant analysis or computation (that is, Snort detection performance evaluation as described in Appendix C). For the latter, the applied values should be *meaningful* (for example it should not be "NAN" in Table 7.6) and *possible* (for example 0 is the minimal probability to represent that an IDS misses all of the intrusions at all, but it is impractical because the normal IDS should generate some TP and TN).

According to the "Accuracy" values from Table 7.6, which records the meas-



```
####################  Ending of Detection Window Time  ###################

#################### Starting of Detection Window Time ###################
Detection Window: 2011-11-01 11:11:40 -- 2011-11-01 11:11:45

       ********************************************************
       **  Combination Based Detection Uncertainty Assessment **
       ********************************************************
========================================================================
                    -----------------------------
                    --        First Step       --
                    -----------------------------
        Incident ID: 3 (Alart Amount = 0, Alart Severity = 2)
        Incident ID: 6 (Alart Amount = 0, Alart Severity = 1)
        Incident ID: 7 (Alart Amount = 0, Alart Severity = 2)
        Incident ID: 8 (Alart Amount = 1, Alart Severity = 3)

                    -----------------------------
                    --        Second Step      --
                    -----------------------------
        Combination Assessment Results: Current Edge Not Detected!
========================================================================
####################  Ending of Detection Window Time  ###################
```

Figure 7.11: Output Snapshot of ADtT Combination Based Intrusion Detection on LLDOS2.0.2 DMZ Phase 4

ured detection performance through the ground truth analysis and detection evaluation, the possible probabilities are as follows: 100% as the *maximal*, 75% as the *medium* and 53.87% as the *minimal*. The reason to set the first probability with 100% is because 100% is the largest one from the stored data, while it is also the maximal probability boundary. Likewise, we select the least one 53.87% as the minimal probability. The reason to assign the second probability with 75% is because 75% is the average of five accuracy values in the LLDOS1.0 inside traffic. As the known least value is 53.87% from the "Accuracy" column of Table 7.6, any values which are less than 53.87% cannot be considered for this discussion.

In our previous experiments in Section 7.3, P_{IDS} has been assigned with different probabilities according to Table 7.6 for each particular atomic attack. We set this group of fused evidence as "*Group 1*". Besides, we set a fixed P_{IDS} in all attack phases in *Group 2*, *Group 3* and *Group 4* with $P_{IDS} = 100\%$, $P_{IDS} = 75\%$ and $P_{IDS} = 53.87\%$, respectively, to make the comparison against the previous experiment and also between each other.

Effect on the Basic Probability Assignment

According to Equation (5.24) in the *Basic Probability Assignment* process in Section 5.2.3, the proposed divisor takes the uncertainty $1 - P_{IDS}$ into account by adding the measured evidence from the membership functions together with the uncertainty on the misclassification. Hence, the lower value of P_{IDS} causes the higher value of the divisor, and additionally causes the lower value of the quotient. While the higher value of P_{IDS} causes the lower value of the divisor, and additionally causes the higher value of the quotient.

Table 7.32 takes Phase 2 of LLDOS1.0 for example and shows the measured detection evidence of I_3, I_4 and I_5, which are obtained from the *basic probability assignment* process.

Table 7.32: Belief Values on Every Single Incident in LLDOS1.0 Phase 2 with Different P_{IDS}

Attack Incident	**Group 1** P_{IDS}	**Group 2** $P_{IDS} = 100\%$	**Group 3** $P_{IDS} = 75\%$	**Group 4** $P_{IDS} = 53.87\%$
I_3	$m(V_1) = 0.08$	$m(V_1) = 0.00$	$m(V_1) = 0.07$	$m(V_1) = 0.09$
	$m(V_1) = 0.82$	$m(V_2) = 1.00$	$m(V_2) = 0.88$	$m(V_2) = 0.80$
	$m(V_1) = 0.10$	$m(V_3) = 0.00$	$m(V_3) = 0.05$	$m(V_3) = 0.11$
I_4	$m(V_1) = 0.00$	$m(V_1) = 0.00$	$m(V_1) = 0.00$	$m(V_1) = 0.00$
	$m(V_1) = 0.78$	$m(V_2) = 1.00$	$m(V_2) = 0.84$	$m(V_2) = 0.73$
	$m(V_1) = 0.22$	$m(V_3) = 0.00$	$m(V_3) = 0.16$	$m(V_3) = 0.27$
I_5	$m(V_1) = 0.21$	$m(V_1) = 0.00$	$m(V_1) = 0.19$	$m(V_1) = 0.21$
	$m(V_1) = 0.65$	$m(V_2) = 1.00$	$m(V_2) = 0.74$	$m(V_2) = 0.65$
	$m(V_1) = 0.14$	$m(V_3) = 0.00$	$m(V_3) = 0.07$	$m(V_3) = 0.14$

Therefore, we may conclude that, generally, the higher P_{IDS} value generates strong evidence, for example, $m(V_2)$ is higher where V_2 corresponds to belief in there being an attack, in contrast with the lower P_{IDS} value with the same assessment factors. The strong evidence may be helpful to determine the achievement of the atomic attack.

Effect on the Final Fusion Process

This part discusses how P_{IDS} values may affect the final fusion process. The measurement of the atomic attack detection combines the multiple evidences from each modelled incident as Equation (5.19) in Section 5.2.1, thus, the evidence values of incidents from basic probability assignment determine the final fused results. According to the measured results from Table 7.33, it is clear that the first four phases have been achieved since the obtained belief value of the second focal element exceeds the summation of the first belief value and the third belief value (that is, $m(V_2) > m(V_1) + m(V_3)$). However, the last attack phase is unachieved as the obtained belief value of the second focal element less than the summation of the first belief value and the third belief value (that is, $m(V_2) < m(V_1) + m(V_3)$).

Table 7.33: Belief Values on LLDOS1.0 Inside Traffic with Different P_{IDS}

Attack Phase	**Group 1** Origin P_{IDS}	**Group 2** $P_{IDS} = 100\%$	**Group 3** $P_{IDS} = 75\%$	**Group 4** $P_{IDS} = 53.87\%$
Phase 1	$m(V_1) = 0.00$ $m(V_2) = 1.00$ $m(V_3) = 0.00$	$m(V_1) = 0.00$ $m(V_2) = 1.00$ $m(V_3) = 0.00$	$m(V_1) = 0.15$ $m(V_2) = 0.84$ $m(V_3) = 0.01$	$m(V_1) = 0.22$ $m(V_2) = 0.73$ $m(V_3) = 0.04$
Phase 2	$m(V_1) = 0.00$ $m(V_2) = 1.00$ $m(V_3) = 0.00$	$m(V_1) = 0.00$ $m(V_2) = 1.00$ $m(V_3) = 0.00$	$m(V_1) = 0.01$ $m(V_2) = 0.99$ $m(V_3) = 0.03$	$m(V_1) = 0.02$ $m(V_2) = 0.97$ $m(V_3) = 0.01$
Phase 3	$m(V_1) = 0.00$ $m(V_2) = 0.99$ $m(V_3) = 0.00$	$m(V_1) = 0.00$ $m(V_2) = 1.00$ $m(V_3) = 0.00$	$m(V_1) = 0.01$ $m(V_2) = 0.99$ $m(V_3) = 0.00$	$m(V_1) = 0.01$ $m(V_2) = 0.99$ $m(V_3) = 0.00$
Phase 4	$m(V_1) = 0.11$ $m(V_2) = 0.83$ $m(V_3) = 0.06$	$m(V_1) = 0.00$ $m(V_2) = 1.00$ $m(V_3) = 0.00$	$m(V_1) = 0.08$ $m(V_2) = 0.89$ $m(V_3) = 0.03$	$m(V_1) = 0.12$ $m(V_2) = 0.79$ $m(V_3) = 0.09$
Phase 5	$m(V_1) = 0.56$ $m(V_2) = 0.44$ $m(V_3) = 0.00$	$m(V_1) = \text{NAN}$ $m(V_2) = \text{NAN}$ $m(V_3) = \text{NAN}$	$m(V_1) = 0.60$ $m(V_2) = 0.39$ $m(V_3) = 0.00$	$m(V_1) = 0.56$ $m(V_2) = 0.44$ $m(V_3) = 0.00$

Therefore, we may conclude that different value of P_{IDS} could not affect the final results on any atomic attack detection, though the measured evidences in each group are slightly different.

7.4.2 Threshold Setting for Alert Amount Membership Function

In the applied *ADtT combination based intrusion detection approach*, three possible thresholds γ_1, γ_2 and γ_3 of the Alert Amount membership functions (that is, Equation (5.20) and Equation (5.21) in Section 5.2.2) are defined with 1, 3 and 30, respectively. The value assignment on γ_1 and γ_3 are direct. $\gamma_1 = 1$ is because that 1 represents the existence of the incident as the minimal. $\gamma_3 = 30$ is because that 30 is the measured average incident number in LLDOS1.0 Inside data sets. However, the assignment of γ_2 is indirect. Thus, in this subsection, we discuss how different settings on γ_2 may affect the evidence measurement in the *basic probability assignment* process and the detection results of the atomic attack detection.

We define four threshold setting groups: *Set 1*, *Set 2*, *Set 3* and *Set 4*. γ_2 in these four groups are 3, 5, 10, 15, respectively. While $\gamma_1 = 1$ and $\gamma_3 = 30$ are fixed in each group. The applied P_{IDS} value in each attack phase sets as the corresponding "Accuracy" value from Table 7.6.

Effect on the Basic Probability Assignment

Table 7.34 shows the measured evidence on I_8 and I_{10} of Phase 4 by the basic proability assignment process. According to Equation (5.20) in Section 5.2.2, γ_2 is only relating to the evidence measurement on the first focal element V_1, which represent *no risk* as the non-achievement. It is clear that all measured values of $m(V_1)$ in both I_8 and I_{10} increase from *Set 1* to *Set 4*. The reason to this trend is because the higher value of γ_2, the more possibility the measured evidence to represent the non-achievement of the modelled incident.

Table 7.34: Belief Values on Every Single Incident in LLDOS1.0 Phase 4 with Different P_{IDS}

Attack Incident	**Set 1** $\gamma_2 = 3$	**Set 2** $\gamma_2 = 5$	**Set 3** $\gamma_2 = 10$	**Set 4** $\gamma_2 = 15$
I_8	$m(V_1) = 0.45$	$m(V_1) = 0.69$	$m(V_1) = 0.76$	$m(V_1) = 0.78$
	$m(V_2) = 0.31$	$m(V_2) = 0.18$	$m(V_2) = 0.13$	$m(V_2) = 0.12$
	$m(V_3) = 0.24$	$m(V_3) = 0.13$	$m(V_3) = 0.10$	$m(V_3) = 0.10$
I_{10}	$m(V_1) = 0.00$	$m(V_1) = 0.00$	$m(V_1) = 0.04$	$m(V_1) = 0.14$
	$m(V_2) = 0.83$	$m(V_2) = 0.84$	$m(V_2) = 0.82$	$m(V_2) = 0.73$
	$m(V_3) = 0.06$	$m(V_3) = 0.16$	$m(V_3) = 0.14$	$m(V_3) = 0.13$

Figure 7.12 illustrates an abstracted relation between γ_1, γ_2 and γ_3. The left part of γ_2 represents the area of "No Risk" as the non-achievement, whereas the right part of γ_2 represents the area of "Risk" as the achievement. As γ_2 increasing from 3 to 15, the "no risk" area becomes wider. In that case, there is higher possibility about the non-achievement of the incident.

Therefore, we may conclude that the higher value of γ_2 increase the certainty of $m(V_1)$ as the non-achievement. Next, we will identify how γ_2 may affect the final fusion on the atomic attack detection.

Effect on the Final Fusion Process

Table 7.35 shows the measured belief values on the detection of each atomic attack with different thresholds. According to the provided data from Table 7.35, the phases from Phase 3 to Phase 5 share the same tendency: with the value of γ_2 increases, the value of $m(V_1)$ also increases, but the value of $m(V_2)$ decreases. The reason of the values unchanged in Phase 1 and Phase 2 is because the applied $P_{IDS} = 100\%$ without any detection uncertainty.

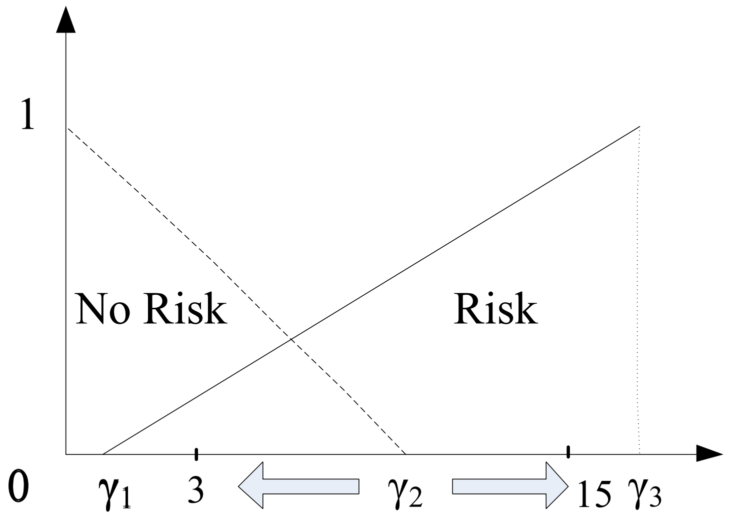


Figure 7.12: Abstracted Relation Between Three Thresholds

As the evidence combination approach considers all the measured evidence (that is, $m(V_1)$, $m(V_2)$ and $m(V_3)$), the higher $m(V_1)$ value of the incident leads the higher $m(V_1)$ value of the atomic attack after the fusion. Therefore, we can summarise that the higher γ_2 value cause the generation of higher $m(V_1)$ evidence value. Note that, the generated detection results are different in Phase 4 of *Set 4*. Because Phase 4 in *Set 1*, *Set 2* and *Set 3* are achieved, but the highest γ_2 value in *Set 4* causes that attack phase unachieved.

From the view of intrusion detection, the security administrator aware of a security issue once an alert had been generated by IDS. By setting the lower value of γ_2, it is possible to make the "No Risk" field smaller and the "Risk" field bigger. Compared with other three groups, *Set 1* is the optimum group because it can generate least "No Risk" field and most "Risk" field. Therefore, it is easier to generate evidence to support the achievement of the detection uncertainty assessment.

7.4.3 Discussion Summary

Through the discussions on the different values of P_{IDS} and γ_2 about the potential effect on the *basic probability assignment* and the *final fusion process*, we can make the following conclusions: (1) the higher value of P_{IDS} generates the higher value of $m(V_2)$ on single incident detection; (2) the value of P_{IDS} could not affect the final results on any atomic attack detection; (3) the higher value of γ_2 leads the higher value of $m(V_1)$ on single incident detection; (4) the value of γ_2 may affect the final results on any atomic attack detection.

Therefore, several ideas are made to suggest the corresponding value settings.

Table 7.35: Belief Values on LLDOS1.0 Inside Traffic with Different Thresholds

Attack Phase	**Set 1** $\gamma_2 = 3$	**Set 2** $\gamma_2 = 5$	**Set 3** $\gamma_2 = 10$	**Set 4** $\gamma_2 = 15$
Phase 1	$m(V_1) = 0.00$ $m(V_2) = 0.10$ $m(V_3) = 0.00$	$m(V_1) = 0.00$ $m(V_2) = 0.10$ $m(V_3) = 0.00$	$m(V_1) = 0.00$ $m(V_2) = 0.10$ $m(V_3) = 0.00$	$m(V_1) = 0.00$ $m(V_2) = 0.10$ $m(V_3) = 0.00$
Phase 2	$m(V_1) = 0.00$ $m(V_2) = 0.10$ $m(V_3) = 0.00$	$m(V_1) = 0.00$ $m(V_2) = 0.10$ $m(V_3) = 0.00$	$m(V_1) = 0.00$ $m(V_2) = 0.10$ $m(V_3) = 0.00$	$m(V_1) = 0.00$ $m(V_2) = 0.10$ $m(V_3) = 0.00$
Phase 3	$m(V_1) = 0.00$ $m(V_2) = 0.99$ $m(V_3) = 0.00$	$m(V_1) = 0.01$ $m(V_2) = 0.99$ $m(V_3) = 0.00$	$m(V_1) = 0.04$ $m(V_2) = 0.96$ $m(V_3) = 0.00$	$m(V_1) = 0.11$ $m(V_2) = 0.89$ $m(V_3) = 0.00$
Phase 4	$m(V_1) = 0.11$ $m(V_2) = 0.83$ $m(V_3) = 0.06$	$m(V_1) = 0.25$ $m(V_2) = 0.70$ $m(V_3) = 0.05$	$m(V_1) = 0.39$ $m(V_2) = 0.57$ $m(V_3) = 0.04$	$m(V_1) = 0.55$ $m(V_2) = 0.42$ $m(V_3) = 0.03$
Phase 5	$m(V_1) = 0.55$ $m(V_2) = 0.44$ $m(V_3) = 0.00$	$m(V_1) = 0.77$ $m(V_2) = 0.23$ $m(V_3) = 0.00$	$m(V_1) = 0.84$ $m(V_2) = 0.15$ $m(V_3) = 0.00$	N/A

(1) No particular requirement on the P_{IDS} value assignment. (2) The γ_2 value should be set closer to the threshold γ_1.

7.5 Summary

This chapter has presented our conducted experiments and relevant analysis. Firstly, the design of ATIDS and the design of corresponding database have been described. Second, the relevant experimental initialization process has presented including how we setup the testbed, how we select the experimental data set, how we analyse the ground truth of data set and how we generate the attack detection tree against the data set. Then, we presented three set of experimental results by applying three attack tree based intrusion detection approaches. Finally, we discussed how the different value settings may affect the detection uncertainty analysis during the intrusion detection process.

According to the obtained experimental results, *AAT based intrusion detection approach* conducts the detection process as a typical SID. It can detect intrusions, which are fully modelled as the signatures. As a common SID limitation, it is unable to generate intrusion alarm if the identified low-level alerts not matching the modelled how-level signature. *ADtT sequential approach* conducts the detection process by considering the detection uncertainty analysis. However, since it has been proposed with the strict signature matching process (that is, every modelled low-level incident must be achieved by both low-level detection and detection uncertainty assessment), any generated weak evidence from the detection uncertainty assessment will cause the whole step detection fail. Finally, *ADtT combination approach* deals with the detection uncertainty assessment to accept any low-level incident detection missing or ignorance.

By comparing the obtained detection results, though *AAT based intrusion detection approach* and *ADtT combination approach* have better detection performance, it is difficult for us to determine which approach is the best one. The reason is because each approach has its own feature and mechanism. For the *AAT based approach*, it achieves the detection with the full signature matching without any detection uncertainty assessment. Hence, any missed low-level detection makes the high-level attack step detection fail. For the *ADtT sequential approach*, it achieves the detection with the not only the full signature matching, but also the evidence assessment achieving on each low-level incident. For the ADtT combination approach, it achieves the detection if the evidence assessment been achieved with either full or partial signature matching. Even if some of the low-level intrusions are missed by the detector, it is still possible to generate high-level alert if the generated evidence supports the detection uncertainty assessment.

In the next chapter, we will make the conclusion of the whole thesis. Additionally, we will summarise the limitations and future work of this research.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This thesis has presented our research on the advanced attack tree based intrusion detection to detect high-level atomic attack of the multi-step attack. In this thesis, Chapter 2 had reviewed the relevant background on intrusion detection and couple of related work on detection uncertainty analysis. Chapter 3 had reviewed the background on attack graph and attack tree modelling techniques, which have particularly investigated how attack graph modelling technique assists intrusion detection research and how attack tree modelling technique and components may be extended for intrusion detection. Chapter 4 had presented our proposed theoretical contributions on the Unified Parametrisable Attack Tree and attack resistance attribution within a general attack tree. Then, targeted on the intrusion detection, Chapter 5 had described our proposed Quality of Detectability metrics and Detection Uncertainty Analysis mechanisms, and additionally given the formalisation of Attack Detection Tree. Our proposed intrusion detection algorithms based on advanced attack trees are presented in Chapter 6. Next, the design of our intrusion detection system, the experimental initialisation, the measured experimental results and further discussions have been described in Chapter 7.

Generally, our research can be split into two parts: the research on the *advanced attack tree* and the research on the *intrusion detection*. Table 8.1 summarises the main research contributions and research weaknesses. Two main contributions are an advanced attack tree modelling technique for intrusion detection (that is, ADtT) and an attack tree based IDS with uncertainty assessment (that is, ATIDS). ADtT is the first known attack tree modelling technique specialised for intrusion detection purpose, while ATIDS is the first known IDS based on modelled attack tree road map and provides detection uncertainty assessment. However, there are two weaknesses in our research. The first weakness is that ATIDS is unable to detect the unknown high-level attack. Since the detection of ATIDS is mainly depends on the modelled signature, any unmodelled attacks cannot be detected by ATIDS. This weakness indicates that ATIDS is a signature based IDS. The second weakness is that there is no metrics been modelled from the applied data sets, because no detailed state information been provided from the data sets. Thus, it is difficult for us to conduct our proposed aggregations within our experiments. Table 8.2 summarises the information of each main contribution, including the elementary minor contributions and the corresponding description.

Table 8.1: Summary of Research

Main Research Contributions	**Research Weaknesses**
1. ADtT (Attack tree modelling technique for intrusion detection.) 2. ATIDS (IDS based on attack tree model and additionally with uncertainty assessment.)	1. ATIDS is unable to detect the unknown high-level attack. 2. No node metrics provided from the applied data sets to verify the proposed aggregation approaches.

Table 8.2: Summary of Research Contributions

Main Contributions	**Minor Contributions**	**Description**
ADtT	QoD	Provides the **real-time** tree structure based and alarms based metrics to describe the security situation.
	DUA	Provides the **real-time** detection uncertainty assessment during intrusion detection process.
	UPAT	Provides the **theoretical** complete attack tree formalisation for any attack tree extensions.
ATIDS	Aggregations	Provides the **theoretical** metrics computation approaches.
	ADtT	Provides the intrusion detection capability for attack tree modelling.
	Algorithms	Provides the **real-time** intrusion detection approaches based on ADtT.

The research on the *advanced attack tree* is the foundation to *intrusion detection*. Our proposed *Attack Detection Tree* modelling technique targets on not only the modelling of the multi-step attack, but also the framework for intrusion detection. The Attack Detection Tree modelling technique applies the notion of Augmented Attack Tree, and furthermore provides two advanced mechanisms to facilitate the intrusion detection. These two proposed mechanisms are the *Quality of Detectability mechanism* and the *detection uncertainty assessment mechanism*.

The proposed *Quality of Detectability mechanism* measures the real-time metrics to represent the on-going intrusion detection progress according to the modelled tree structure, the detection time and the alert information. While the proposed *detection uncertainty assessment mechanism* measures the evidence from the real-time generated alerts to conduct the relevant detection uncertainty analysis, and especially investigates the situations such as any modelled low-level attack incidents are missed by Snort.

Specifically, two detection uncertainty assessment approaches have been proposed: the *sequential based approach* and the *combination based approach*. The sequential one examines each single modelled incident one-by-one within the atomic attack according to the captured evidence. The achievement of the atomic attack is indicated by the achievement of all modelled incidents. Any undetected incident within this approach is unacceptable, as it causes the unachievement of the atomic attack. The combination approach examines the achievement of the atomic attack by fusing the measured evidence from all modelled incidents. Any undetected incident within this approach is acceptable, since the achievement of the atomic attack is determined by fusing all of the captured evidence together.

According to the original *Augmented Attack Tree* and our proposed *Attack Detection Tree*, we have proposed two main intrusion detection approaches: the *AAT based intrusion detection* and the *ADtT based intrusion detection*. The AAT based approach targets on the validation of the high-level atomic attack detection. The ADtT based approach targets on not only the high-level atomic attack detection, but also the *Quality of Detectability mechanism* measurement and the *detection uncertainty assessment mechanism* analysis.

The experimental results show that both of the proposed detection approaches can detect the high-level atomic attacks from the generated low-level incidents alerts. For the *ADtT based intrusion detection* approaches, they generate different results with the *sequential based assessment approach* and the *combination based assessment approach*. With the sequential based detection approach, the experimental results show that it generates less TP. The main reason is that any unachieved modelled incidents lead to the non-achievement of the atomic attack detection. With the combination based detection approach, the experimental results show that it generates more TP. The main reason is that the atomic attack detection is by fusing all of the measured evidence from all modelled incidents, irrespective of whether that incident is missed or not. Moreover, the experimental results show the measured QoD metrics can represent the progress of on-going intrusion and intrusion detection.

Besides the aforementioned practical detection process, we have additionally proposed several theoretical mechanisms to enhance the research on the attack tree: (1) *Unified Parametrisable Attack Tree*; and (2) *attack resistance aggregation* in the attack tree. Our *Unified Parametrisable Attack Tree* gives the theoretical prototype for the attack tree extensions, including our proposed Attack Detection Tree. Since the possible extension locations are tree node, tree edge and tree connector, the relevant extensions should fall in that scope. While the *attack resistance aggregation* shows the metrics attribution approaches on the attack tree.

In general, this research explores a new attack tree examination with uncertainty and tree based intrusion detection techniques. The framework presented here is hoped to be the foundation for the security solutions of the future.

8.1.1 Limitations

There are several limitations within this research.

Limitation 1. *Detection Window constraint.*

Detection window assists the proposed detection mechanism to identify the low-level incidents and the corresponding evidence within the determined time period, and not alerts outside that window. However, the detection window constraint exists no matter how we set the window length, for example, 1 second, 1 hour, 1 day, 1 year, etc, even if we apply the overlapped setting.

Limitation 2. *Lack of capability to distinguish the generated low-level alerts are from multiple intrusions.*

In our proposed detection approach, we take all of the Snort generated alerts into the account assuming that they are all relating to the current on-going intrusion. However, it is possible that there are multiple adversaries conducting the intrusions on the same victim system simultaneously. For instance, one of the adversaries conduct the multi-step attack, but the rest of them simply launch any intrusions without any particular attack goals.

8.2 Future Work

The possible future work are as follows.

Future Work 1. *Automatic Attack Detection Tree construction.*

The main reason of this future work is to address the limitation of **Assumption 2**. In this piece of future work, the Attack Detection Tree would be generated automatically by software, and would combine the attack information from any open attack databases (for example, Common Vulnerability Scoring System (CVSS)).

For one, the software based generation is more efficient and effective compared with the manual work. For another, the software based process may generate the more comprehensive attack tree by analysing and applying different attack databases, that provided various network intrusions. Therefore, the constructed attack detection tree may be more precise and can represent more sophisticated multi-step attacks.

Future Work 2. *Attack tree based intrusion detection in real network.*

The main reason of this future work is to address the problem of **Limitation 2**. This piece of future work will apply the attack detection tree mechanism within the real network. Therefore, the attack tree based intrusion detection should be capable to identify the low-level alerts which are corresponding to the on-going multi-step attack, in contrast to those which are the intrusions launched by other adversaries not mounting multi-step attacks. Therefore, the proposed attack tree based intrusion detection can be practically applied to implement the high-level intrusion detection on the modelled attack tree in any real network.

References

[1] KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[2] Snort. http://www.snort.org/.

[3] The Bro Network Security Monitor. http://bro-ids.org/.

[4] Snort database schema v1.06. http://cvs.snort.org/viewcvs.cgi/*checkout*/snort/doc/snort_schema_v106.pdf?rev=1.3, 2004.

[5] F. Alserhani, M. Akhlaq, I.U. Awan, A.J. Cullen, and P. Mirchandani. MARS: Multi-Stage Attack Recognition System. In *Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications (AINA2010)*, pages 753–759, 2010.

[6] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, Graph-Based Network Vulnerability Analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*, pages 217–224, 2002.

[7] N.B. Anuar, M. Papadaki, S. Furnell, and N. Clarke. An Investigation and Survey Of Response Options For Intrusion Response Systems (IRSs). In *Proceedings of Information Security for South Africa (ISSA)*, pages 1–8, 2010.

[8] D. Ariu. *Host and Network based Anomaly Detectors for HTTP Attacks.* PhD thesis, University of Cagliari, 2010.

[9] S. Axelsson. The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):186–205, 2000.

[10] J.S. Balasubramaniyan, J.O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni. An Architecture for Intrusion Detection using Autonomous Agents. In *Proceedings of 14th Annual Computer Security Applications Conference*, pages 13–24, 1998.

[11] M. Benini and S. Sicari. Risk Assessment in Practice: a Real Case Study. *Journal of Computer Communications*, 31(15):3691–3699, 2008.

[12] E. Biermann, E. Cloete, and L.M. Venter. A Comparison of Intrusion Detection Systems. *Computers & Security*, 20(8):676–683, 2001.

[13] S. Bistarelli, F. Fioravanti, and P. Peretti. Defense Trees For Economic Evaluation Of Security Investments. In *Proceedings of The First International Conference on Availability, Reliability and Security*, 2006.

[14] D. Bolzoni, S. Etalle, P. Hartel, and E. Zambon. POSEIDON: A 2-Tier Anomaly-Based Network Intrusion Detection System. In *Proceedings of the Fourth IEEE International Workshop on Information Assurance (IWIA'06)*, pages 144–156, 2006.

[15] D. Bolzoni, S. Etalle, and P.H. Hartel. Panacea: Automating Attack Classification for Anomaly-based Network Intrusion Detection Systems. In *Proceedings of 12th International Symposium on Recent Advances in Intrusion Detection (RAID'09), Lecture Notes in Computer Science*, pages 1–20, 2009.

[16] J. Botwicz, P. Buciak, and P. Sapiecha. Building Dependable Intrusion Prevention Systems. In *Proceedings of International Conference on Dependability of Computer Systems (DepCos-RELCOMEX'06)*, pages 135–142, 2006.

[17] P.J. Brooke and R.F. Paige. Fault Trees for Security System Design and Analysis. *Journal of Computers & Security*, 22(3):256–264, 2003.

[18] A. Buldas, P. Laud, J Priisalu, M. Saarepera, and J. Willemson. Rational Choice of Security Measures Via Multi-parameter Attack Trees. In *Proceedings of Critical Information Infrastructures Security, Lecture Notes in Computer Science*, pages 235–248, 2006.

[19] S.A. Camtepe and B. Yener. Modeling and Detection of Complex Attacks. In *Proceedings of Third International Conference on Security and Privacy in Communications Networks and the Workshops (SecureComm 2007)*, pages 234–243, 2007.

[20] A.A. Cardenas, J.S. Baras, and K. Seamon. A Framework for the Evaluation of Intrusion Detection Systems. In *Proceedings of 2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 50–61, 2001.

[21] T.M. Chen and V. Venkataramanan. Dempster-shafer Theory for Intrusion Detection in Ad Hoc Networks. *IEEE Internet Computing*, pages 35–41, 2005.

[22] F. Cheng, S. Roschke, and C. Meinel. An Integrated Network Scanning Tool for Attack Graph Construction . In *Proceedings of Advances in Grid and Pervasive Computing (GPC 2011), Lecture Notes in Computer Science*, pages 138–147, 2011.

[23] G.D. Crescenzo, A. Ghosh, and R. Talpade. Towards a Theory of Intrusion Detection. In *Proceedings of the 10th European conference on Research in Computer Security (ESORICS 2005), Lecture Notes in Computer Science*, pages 267–286. Springer, 2005.

[24] F. Cuppens and A. Miege. Alert Correlation in a Cooperative Intrusion Detection Framework. In *Proceedings of 2002 IEEE Symposium on Security and Privacy (S&P'02)*, pages 202–215, 2002.

[25] K. Daley, R. Larson, and J. Dawkins. A Structural Framework for Modeling Multi-Stage Network Attacks. In *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02)*, 2002.

[26] J. Dawkins and J. Hale. A Systematic Approach to Multi-Stage Network Attack Analysis. In *Proceedings of the Second IEEE International Information Assurance Workshop (IWIA'04)*, pages 48–56, 2004.

[27] H. Debar, D.A. Curry, and B.S. Feinstein. RFC4765: The Intrusion Detection Message Exchange Format (IDMEF). *The IETF Trust*, 2007.

[28] D.E. Denning. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, 1987.

[29] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley. Optimal Security Hardening Using Multi-Objective Optimization on Attack Tree Models of Networks. In *Proceedings of the 14th ACM Conference on Computer and communications Security*, pages 204–213, 2007.

[30] H. Du, D.F. Liu, J. Holsopple, and S.J. Yang. Toward Ensemble Characterization and Projection of Multistage Cyber Attacks. In *Proceeings of 19th International Conference on Computer Communications and Networks (ICCCN2010)*, pages 1–8, 2010.

[31] K. Edge, R. Raines, M. Grimaila, R. Baldwin, R. Bennington, and C. Reuter. The Use of Attack and Protection Trees to Analyze Security for an Online Banking System. In *Proceedings of 40th Annual Hawaii International Conference on System Sciences (HICSS2007)*, pages 144b–144b, 2007.

[32] K.S. Edge. *A Framework for Analyzing and Mitigating the Vulnerabilities of Complex Systems Via Attack and Protection Trees.* PhD thesis, Air Force Institute Of Technology, 2007.

[33] L. Fang, C. Wang, and G. Ma. A Framework for Network Security Situation Awareness Based on Knowledge Discovery. In *Proceedings of 2nd International Conference on Computer Engineering and Technology (ICCET2010)*, pages 226–231, 2010.

[34] T. Fawcett. An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27(4):861C–874, 2006.

[35] J.E. Gaffney and J.W. Ulvila. Evaluation of Intrusion Detectors: A Decision Theory Approach. In *Proceedings of 2001 IEEE Symposium on Security and Privacy (S&P'01)*, pages 50–61, 2001.

[36] G. Gu, P. Fogla, D. Dagon, W. Lee, and B. Skoric. Measuring Intrusion Detection Capability: An Information-Theoretic Approach. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications (ASIACCS '06)*, pages 90–101, 2006.

[37] G. Gu, P. Fogla, D. Dagon, W. Lee, and B. Skoric. Towards an Information-Theoretic Framework for Analyzing Intrusion Detection Systems. In *Proceedings of the 11th European conference on Research in Computer Security (ESORICS 2006), Lecture Notes in Computer Science*, pages 527–546. Springer, 2006.

[38] G. Gu, A. McCallum, and D. Towsley. Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement (IMC'05)*, 2005.

[39] J. Holsopple and S.J. Yang. FuSIA: Future Situation and Impact Awareness. In *Proceedings of 2008 11th International Conference on Information Fusion*, pages 1–8, 2008.

[40] K. Ingols, R. Lippmann, and K. Piwowarski. Practical Attack Graph Generation For Network Defense. In *Proceedings of 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 121–130, 2006.

[41] S. Jajodia, S. Noel, and B. O'Berry. Topological Analysis of Network Attack Vulnerability. *Managing Cyber Threats*, 5:247–266, 2005.

[42] F. Jemili, M. Zaghdoud, and M.B. Ahmed. A Framework for an Adaptive Intrusion Detection System using Bayesian Network. In *Proceedings of 2007 IEEE Intelligence and Security Informatics*, pages 66–70, 2007.

[43] S. Jha, O. Sheyner, and J. Wing. Two Formal Analysis Of Attack Graphs. In *Proceedings of the 15th IEEE Computer Security Foundations Workshops (CSFW'02)*, pages 49–63, 2002.

[44] A. Jürgenson and J. Willemson. Processing Multi-Parameter Attacktrees with Estimated Parameter Values. In *Proceedings of Advances in Information and Computer Security, Lecture Notes in Computer Science*, pages 308–319, 2007.

[45] A. Jürgenson and J. Willemson. Computing Exact Outcomes of Multi-Parameter Attack Trees. In *Proceedings of On the Move to Meaningful Internet Systems, Lecture Notes in Computer Science*, pages 1036–1051, 2008.

[46] A. Jürgenson and J. Willemson. Serial Model for Attack Tree Computations. In *Proceedings of Information, Security and Cryptology (ICISC 2009), Lecture Notes in Computer Science*, pages 118–128, 2010.

[47] R.A. Kemmerer and G. Vigna. Intrusion Detection: A Brief History And Overview. *Computer*, 35(4):27–30, 2002.

[48] P.A. Khand. System Level Security Modeling Using Attack Trees. In *Proceedings of 2nd International Conference on Computer, Control and Communication*, pages 1–6, 2009.

[49] B. Kordy, S. Mauw, S. Radomirovic, and P. Schweitzer. Foundations of Attack-Defense Trees. In *Proceedings of Formal Aspects of Security and Trust, Lecture Notes in Computer Science*, pages 80–95. Springer, 2011.

[50] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian Event Classification For Intrusion Detection. In *Proceedings of 19th Annual Computer Security Applications Conference (ACSAC'03)*, pages 14–23, 2003.

[51] C. Kruegel, G. Vigna, and W. Robertson. A Multi-Model Approach To The Detection Of Web-Based Attacks. *Computer Networks*, 48(5):717–738, 2005.

[52] MIT Lincoln Laboratory. 1998 DARPA Intrusion Detection Evaluation Data Set. `http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1998data.html`.

[53] MIT Lincoln Laboratory. 1999 DARPA Intrusion Detection Evaluation Data Set. http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1999data.html.

[54] MIT Lincoln Laboratory. 1999 DARPA Intrusion Detection Evaluation Plan. http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/docs/id99-eval-ll.html.

[55] MIT Lincoln Laboratory. 2000 DARPA Intrusion Detection Scenario Specific Data Sets. http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/2000data.html.

[56] D. Lathrop, J.M.D. Hill, and J.R. Surdu. Modeling Network Attacks. In *Proceedings of the 12th Conference on Behavior Representation in Modeling and Simulation*, pages 401–407, 2003.

[57] W. Lee and S.J. Stolfo. Data Mining Approaches For Intrusion Detection. In *Proceedings of SSYM'98 Proceedings of the 7th conference on USENIX Security Symposium*, 1998.

[58] W. Lee and S.J. Stolfo. A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security*, 3(4):227–261, 2001.

[59] W. Lee, S.J. Stolfo, and K.W. Mok. A Data Mining Framework For Building Intrusion Detection Models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy (S&P1999)*, pages 120–132, 1999.

[60] J. Lei and Z. Li. Using Network Attack Graph to Predict the Future Attacks. In *Proceedings of Second International Conference on Communications and Networking in China (CHINACOM'07)*, pages 403–407, 2008.

[61] J. Li, X. Ou, and R. Rajagopalan. Uncertainty and Risk Management in Cyber Situational Awareness. *Cyber Situational Awareness, Advances in Information Security*, pages 51–68, 2010.

[62] Z. Li, A. Das, and J. Zhou. Theoretical Basis For Intrusion Detection. In *Proceedings from the Sixth Annual IEEE Information Assurance Workshop (IAW'05)*, pages 184–192, 2005.

[63] Z. Li, J. Lei, L. Wang, and D. Li. A Data Mining Approach to Generating Network Attack Graph for Intrusion Detection. In *Proceedings of Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, pages 307–311, 2007.

[64] R. Lippmann, J.W. Haines, D.J. Fried, J. Korba, and K. Das. The 1999 DARPA Off-Line Intrusion Detection Evaluation. *Computer Networks*, 34(4):579–595, 2000.

[65] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, and R. Cunningham. Validating And Restoring Defense In Depth Using Attack Graphs. In *Proceedings of Military Communications Conference (MILCOM)*, pages 1–10, 2006.

[66] R.P. Lippmann and K.W. Ingols. An Annotated Review of Past Papers on Attack Graphs. Technical report, Lincoln Laboratory, Massachusetts Institute Of Technology, 2005.

[67] X. Liu and D. Xiao. Using Vulnerability Analysis to Model Attack Scenario for Collaborative Intrusion Detection. In *Proceedings of 10th International Conference on Advanced Communication Technology (ICACT 2008)*, pages 1273–1277, 2008.

[68] X. Liu, D. Xiao, T. Gu, and H. Xu. Scenario Recognition based on Collaborative Attack Modeling in Intrusion Detection. In *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS 2008)*, 2008.

[69] M.V. Mahoney and P.K. Chan. Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'02)*, pages 376–385, 2002.

[70] M.F. Marhusin, D. Cornforth, and H. Larkin. An Overview of Recent Advances in Intrusion Detection. In *Proceedings of 8th IEEE International Conference on Computer and Information Technology (CIT 2008)*, pages 432–437, 2008.

[71] S. Mauw and M. Oostdijk. Foundations of Attack Trees. In *Proceedings of Information Security and Cryptology - ICISC 2005, Lecture Notes in Computer Science*, pages 186–198. Springer, 2006.

[72] A. Morais, E. Martins, A. Cavalli, and W. Jimenez. Security Protocol Testing Using Attack Trees. In *Proceeings of 2009 International Conference on Computational Science and Engineering (CSE'09)*, pages 690–697, 2009.

[73] B. Morin and L. Mé. Intrusion Detection and Virology: an Analysis of Differences, Similarities and Complementariness. *Journal in Computer Virology*, 3(1):39–49, 2007.

[74] B. Morin, L. Mé, H. Debar, and M. Ducassé. M2D2 a formal data model for IDS alert correlation. In *Proceedings of the 5th International Conference on Recent Advances in Intrusion Detection (RAID2002), Lecture Notes in Computer Science*, pages 115–127, 2002.

[75] C.P. Mu, X.J. Li, H.K. Huang, and S.F. Tian. Online Risk Assessment of Intrusion Scenarios Using D-S Evidence Theory. In *Proceedings of the 11th European conference on Research in Computer Security (ESORICS 2006), Lecture Notes in Computer Science*, pages 35–48. Springer, 2006.

[76] B. Mukherjee, L.T. Heberlein, and K.N. Levitt. Network Intrusion Detection. *IEEE Network*, 8(3):26–41, 1994.

[77] S. Neol, M. Jacobs, P. Kalap, and S. Jajodia. Multiple Coordinated Views fro Network Attack Graphs. In *Proceedings of Workshop on Visualization for Computer Security*, pages 99–106, 2005.

[78] S. Neol and S. Jajodia. Understanding Complex Network Attack Graphs Through Clustered Adjacency Matrices. In *Proceedings of 21st Annual Computer Security Applications Conference (ACSAC2005)*, 2005.

[79] S. Neol, E. Robertson, and S. Jajodia. Correlating Intrusion Events and Building Attack Scenarios Through Attack Graph Distance. In *Proceedings of 20th Annual Computer Security Applications Conference (ACSAC'04)*, pages 350–359, 2004.

[80] S. Neol, E. Robertson, and S. Jajodia. Correlating Intrusion Events and Building Attack Scenarios Through Attack Graph Distances. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*, pages 350–359, 2004.

[81] P. Ning, Y. Cui, and D.S. Reeves. Constructing Attack Scenarios Through Correlation of Intrusion Alerts. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 245–254, 2002.

[82] S. Noel and S. Jajodia. Optimal IDS Sensor Placement and Alert Prioritization Using Attack Graphs. *Journal of Network System Management*, 16:259–275, 2008.

[83] S. Noel, S. Jajodia, L. Wang, and A. Singhal. Measuring Security Risk of Networks Using Attack Graphs. *International Journal of Next-Generation Computing*, 1(1):135–147, 2010.

[84] NSTAC. Network group intrusion detection subgroup report. `http://www.ncs.gov/nstac/reports/1997/FIDSGREP.pdf`, December 1997.

[85] X. Ou, S.R. Rajagopalan, and S. Sakthivelmurugan. An Empirical Approach to Modeling Uncertainty in Intrusion Analysis. In *Proceedings of 2009 Annual Computer Security Applications Conference (ACSAC2009)*, pages 494–503, 2009.

[86] A. Patcha and J.M. Park. An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends. *Computer Networks*, pages 51–68, 2007.

[87] W. Peng, Z. Wang, and J. Chen. Research on Attack Intention Recognition based on Graphical Model. In *Proceedings of 2009 Fifth International Conference on Information Assurance and Security*, pages 360–363, 2009.

[88] N. Pham and M. Riguidel. Security Assurance Aggregation for IT Infrastructures. In *Proceedings of Second International Conference on Systems and Networks Communications (ICSNC 2007)*, pages 1–8, 2007.

[89] C. Phillips and L.P. Swiler. A Graph-Based System For Network-Vulnerability Analysis. In *Proceedings of the 1998 Workshop on New Security Paradigms*, pages 71–79, 1998.

[90] X. Qin and W. Lee. Attack Plan Recognition and Prediction Using Causal Networks. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*, pages 370–379, 2004.

[91] I. Ray and N. Poolsapassit. Using Attack Trees to Identify Malicious Attacks from Authorized Insiders. In *Proceedings of the 10th European conference on Research in Computer Security (ESORICS 2005), Lecture Notes in Computer Science*, pages 231–246. Springer, 2005.

[92] I. Ray and N. Poolsapassit. Investigating Computer Attacks Using Attack Trees. In *Proceedings of Advances in Digital Forensics III, International Federation for Information Processing*, pages 331–343. Springer, 2007.

[93] M. Rehák, E. Staab, V. Fusenig, J. Stiborek, M. Grill, K. Bartoš, M. Pěchouček, and T. Engel. Threat-model-driven runtime adaptation and evaluation of intrusion detection system. In *Proceedings of the 6th international conference on Autonomic computing (ICAC'09)*, pages 65–66, 2009.

[94] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX conference on System administration*, pages 229–238, 1999.

[95] S. Roschke, F. Cheng, and C. Meinel. Using Vulnerability Information And Attack Graphs For Intrusion Detection . In *Proceedings of 2010 Sixth International Conference on Information Assurance and Security (IAS)*, pages 68–73, 2010.

[96] A. Roy, D.S. Kim, and K.S. Trivedi. Cyber Security Analysis Using Attack Countermeasure Trees. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, pages 1–4, 2010.

[97] B. Schneier. Attack Trees. *Dr. Dobb's Journal*, 24(12):21–29, 1999.

[98] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[99] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated Generation and Analysis of Attack Graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P'02)*, pages 273–284, 2002.

[100] O. Sheyner and J. Wing. Tools for Generating and Analyzing Attack Graphs. In *Proceedings of Formal Methods For Components and Objects (FMCO 2003), Lecture Notes In Computer Science*, pages 344–371, 2004.

[101] J. Steffan and M. Schumacher. Collaborative Attack Modeling. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 253–259, 2002.

[102] P.K. Sujatha, A. Kannan, S. Ragunath, K.S. Bargavi, and S. Githanjali. A Behavior Based Approach to Host-Level Intrusion Detection Using Self-Organizing Maps. In *Proceedings of First International Conference on Emerging Trends in Engineering and Technology (ICETET'08)*, pages 1267–1271, 2008.

[103] L.P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-Attack Graph Generation Tool. In *Proceedings of DARPA Information Survivability Conference & Exposition II (DISCEX'01)*, pages 307–321, 2001.

[104] S.J. Templeton and K. Levitt. A Requires/Provides Model For Computer Attacks. In *Proceedings of the 2000 Workshop On New Security Paradigms*, pages 31–38, 2001.

[105] C. Thomas and N. Balakrishnan. Advanced Sensor Fusion Technique for Enhanced Intrusion Detection. In *Proceedings of IEEE International Conference on Intelligence and Security Informatics (ISI 2008)*, pages 173–1798, 2008.

[106] J. Tian, T. Liu, and H. Jiao. Entropy Weight Coefficient Method for Evaluating Intrusion Detection Systems. In *Proceedings of International Symposium on Electronic Commerce and Security (ISECS2008)*, pages 592–598, 2008.

[107] T. Tidwell, R. Larson, K. Fitch, and J. Hale. Modeling Internet Attacks. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, pages 54–59, 2001.

[108] T. Tidwell, R. Larson, K. Fitch, and J. Hale. The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems. In *Proceedings of International Infrastructure Survivability Workshop (IISW'04)*, 2004.

[109] A.G. Tokhtabayev, V.A. Skormin, and A.M. Dolgikh. Expressive, Efficient and Obfuscation Resilient Behavior Based IDS. In *Proceedings of the 15th European conference on Research in Computer Security (ESORICS 2010), Lecture Notes in Computer Science*, pages 698–715. Springer, 2010.

[110] I.A. Tøndel, J. Jensen, and L. Røstad. Combining Misuse Cases with Attack Trees and Security Activity Models. In *Proceedings of 2010 International Conference on Availability, Reliability and Security (ARES2010)*, pages 438–445, 2010.

[111] R. Trost. *Practical Intrusion Analysis: Prevention and Detection for the Twenty-First Century.* Addison-Wesley Professional, 2010.

[112] A. Valdes and K. Skinner. Probabilistic Alert Correlation. In *Proceedings of the 4th International Conference on Recent Advances in Intrusion Detection (RAID2001), Lecture Notes in Computer Science*, pages 54–68, 2001.

[113] F. Valeur, G. Vigna, C. Kruegel, and R.A. Kemmerer. Comprehensive Approach to Intrusion Detection Alert Correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, 2004.

[114] H. Wang, S. Liu, and X. Zhang. A Prediction Model of Insider Threat Based on Multi-Agent. In *Proceedings of 1st International Symposium on Pervasive Computing and Applications*, pages 273–278, 2006.

[115] J. Wang, R.C.W Phan, J.N. Whitley, and D.J. Parish. Augmented Attack Tree Modeling of Distributed Denial of Services and Tree Based Attack

Detection Method. In *Proceedings of The 10th IEEE International Conference on Computer and Information Technology (CIT2010)*, pages 1009–1014, 2010.

[116] J. Wang, R.C.W Phan, J.N. Whitley, and D.J. Parish. Quality of Detectability (QoD) and QoD-Aware AAT-based Attack Detection. In *Proceedings of The 5th International Conference for Internet Technology and Secured Transactions (ICITST-2010)*, pages 152–157, 2010.

[117] J. Wang, J.N. Whitley, R.C.W. Phan, and D.J. Parish. Unified Parametrizable Attack Tree. *International Journal for Information Security Research*, 1(1):20–26, 2011.

[118] K. Wang and S.J. Stolfo. Anomalous Payload-Based Network Intrusion Detection. In *Proceedings of 7th International Symposium on Recent Advances in Intrusion Detection (RAID'04), Lecture Notes in Computer Science*, pages 203–222, 2004.

[119] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An Attack Graph-Based Probabilistic Security Metric. *Data and Applications Security XXII, Lecture Notes in Computer Science*, pages 283–296, 2008.

[120] L. Wang, A. Singhal, and S. Jajodia. Measuring The Overall Security Of Network Configurations Using Attack Graphs. In *Proceedings of the 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 98–112, 2007.

[121] L. Wang, A. Singhal, and S. Jajodia. Toward Measuring Network Security Using Attack Graphs. In *Proceedings of the 2007 ACM workshop on Quality of Protection (QoP'07)*, pages 49–54, 2007.

[122] L. Wang, A. Singhal, S. Jajodia, and S. Noel. *k*-Zero Day Safety: Measuring the Security Risk of Networks Against Unknown Attacks. In *Proceedings of the 15th European conference on Research in Computer Security (ESORICS 2010), Lecture Notes in Computer Science*, pages 573–587. Springer, 2010.

[123] Y. Wang, H. Yang, X. Wang, and R. Zhang. Distributed Intrusion Detection System Based on Data Fusion Method. In *Proceedings of Fifth World Congress on Intelligent Control and Automation (WCICA 2004)*, pages 4331–4334, 2004.

[124] J.N. Whitley, R.C.W. Phan, J. Wang, and D.J. Parish. Attribution of Attack Trees. *Computers & Electrical Engineering*, 37(4):624–628, 2011.

[125] Z. Wu, D. Xiao, H. Xu, X. Peng, and X. Zhuang. Virtual Inline: A Technique of Combining IDS and IPS Together in Response Intrusion. In *Proceedings of First International Workshop on Education Technology and Computer Science (ETCS'09)*, pages 1118–1121, 2009.

[126] G. Xiang and Y.D. Cao. Generating IDS Attack Pattern Automatically Based on Attack Tree. *Journal of Beijing Institute of Technology*, 12(2):138–142, 2003.

[127] Z. Zhang, P.H. Ho, and L. He. Measuring IDS-Estimated Attack Impacts for Rational Incident Response: A Decision Theoretic Approach. *Computers & Security*, 28(7):605–614, 2009.

[128] K. Zhao, F. Ren, Nurbol, and L. Hu. LDLB: A Light Intrusion Prevention System In Data Link Layer. In *Proceedings of 2nd International Conference on Anti-counterfeiting, Security and Identification (ASID 2008)*, pages 190–193, 2008.

[129] N. Zhu, X.Y. Chen, Y.F. Zhang, and S.Y Xin. Design and Application of Penetration Attack Tree Model Oriented to Attack Resistance Test. In *Proceeings of 2008 International Conference on Computer Science and Software Engineering*, pages 622 – 626, 2008.

[130] J. Zimmermann, L. Me, and C. Bidan. An Improved Reference Flow Confreol Model for Policy-Based Intrusion Detection. In *Proceedings of the 8th European conference on Research in Computer Security (ESORICS 2003), Lecture Notes in Computer Science*, pages 291–308. Springer, 2003.

[131] S.A. Zonouz, H. Khurana, W.H. Sanders, and T.M. Yardley. RRE: A Game-Theoretic Intrusion Response And Recovery Engine. In *Proceedings of IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 439–448, 2009.

Appendix A

Formalisation of Augmented Attack Tree

Augmented Attack Tree (AAT) [92, 91] extended the conventional attack tree by associating additional information on the tree edge. There are two different formalized AATs for two different research. In the former, AAT [91] had been proposed by augmenting the attack probability label on each edge to identify malicious attacks from authorized insiders. with each branch a sequence of malicious operations that could have been used in the attack. In the latter, AAT [92] had been proposed by augmenting the attack signature on each edge to implement attack forensic analysis. The applied AAT in this thesis is the second one. The formalisation of applied AAT is as follows.

Definition 30 ***Augmented Attack Tree.*** *An augmented attack tree is a rooted labelled tree given by* $AAT = (V, E, \varepsilon, Label, SIG_{u,v})$*, where*

- *V is the set of nodes in the tree representing the different states of partial compromise or sub-goals that an adttacker needs to move through in order to fully compromise a system. $\mathcal{V} \in V$ is a special node, distinguished from others, that forms the root of the tree. It represents the ultimate goal of the attacker, namely system compromise. The set V can be partitioned into two subsets, leaf_nodes and internal_nodes, such that*

  - ∗ $leaf\_nodes \bigcup internal\_nodes = V$,
  - ∗ $leaf\_nodes \bigcap internal\_nodes = \emptyset$, *and*
  - ∗ $\mathcal{V} \in internal\_nodes$

- *$E \subseteq V \times V$ constitutes the set of edges in the attack tree. An edge $(u,v) \in E$ defines an atomic attack and represents the state transition from a child node*

v to a parent node u, $u, v \in V$. An atomic attack is a sequence of incidents. The edge (u, v) is said to be "emergent from" v and "incident to" u.

- *ε is a set of tuples of the form $\langle v$, decomposition$\rangle$ such that*
  - $*$ *$v \in$ internal_nodes and*
  - $*$ *decomposition $\in$ [AND-decomposition, OR-decomposition]*
- *Label is the name of the exploit associated with each edge.*
- *$SIG_{u,v}$ is an attack signature which is defined as Definition 32 below.*

Definition 31 ***Incident-choice.*** *An incident-choice is a group of related incidents, the occurrence of any one of which can contribute towards the state transition in the attack tree.*

Definition 32 ***Attack Signature.*** *An attack Signature $SIG_{u,v}$ is a sequence of incident-choices (incident-choice$_1$, incident-choice$_2$,..., incident-choice$_n$) such that the sequence (incident$_{i,1}$, incident$_{j,2}$, ...,incident$_{m,n}$) constitute an atomic attack.*

Definition 33 ***AND-decomposition.*** *Given a node, v in an attack tree such that $v \in$ internal_nodes, the node is an AND-decomposition if all edges incident to the node are connected by the AND operation, or there is exactly one edge incident to the node.*

Definition 34 ***OR-decomposition.*** *Given a node v of an attack tree such that $v \in$ internal_nodes, the node is an OR-decomposition if all edges incident to the node are connected by the OR operation.*

Appendix B

Process of Data Set Ground Truth Analysis

This appendix describes the process to examine the ground truth of DARPA2000 data sets.

The data information of applied LLDOS1.0 and LLDOS2.0.2 "Inside" traffics are provided in Table B.1 and Table B.2, respectively. Note that all of the applied files are from the "inside" network category of the DARPA2000 data sets.

Table B.1: Applied Files for LLDOS1.0 Inside Traffic Ground Truth Analysis

Phase	IDMEF Alert File	TCPDUMP Traffic File	Session List File
Phase 1	mid-level-phase-1.xml	phase-1-dump.pcap	phase-1.list
Phase 2	mid-level-phase-2.xml	phase-2-dump.pcap	phase-2.list
Phase 3	mid-level-phase-3.xml	phase-3-dump.pcap	phase-3.list
Phase 4	mid-level-phase-4.xml	phase-4-dump.pcap	phase-4.list
Phase 5	mid-level-phase-5.xml	phase-5-dump.pcap	phase-5.list

Table B.2: Applied Files for LLDOS2.0.2 Inside Traffic Ground Truth Analysis

Phase	IDMEF Alert File	TCPDUMP Traffic File	Session List File
Phase 1	mid-level-phase-1.xml	phase-1-dump.pcap	phase-1.list
Phase 2	mid-level-phase-2.xml	phase-2-dump.pcap	phase-2.list
Phase 3	mid-level-phase-3.xml	phase-3-dump.pcap	phase-3.list
Phase 4	mid-level-phase-4.xml	phase-4-dump.pcap	phase-4.list
Phase 5	mid-level-phase-5.xml	phase-5-dump.pcap	phase-5.list

MIT Lincoln laboratory provides a list of IDMEF alert files in XML for each attack phase. Although each XML alert is a part of the attack, it represents the

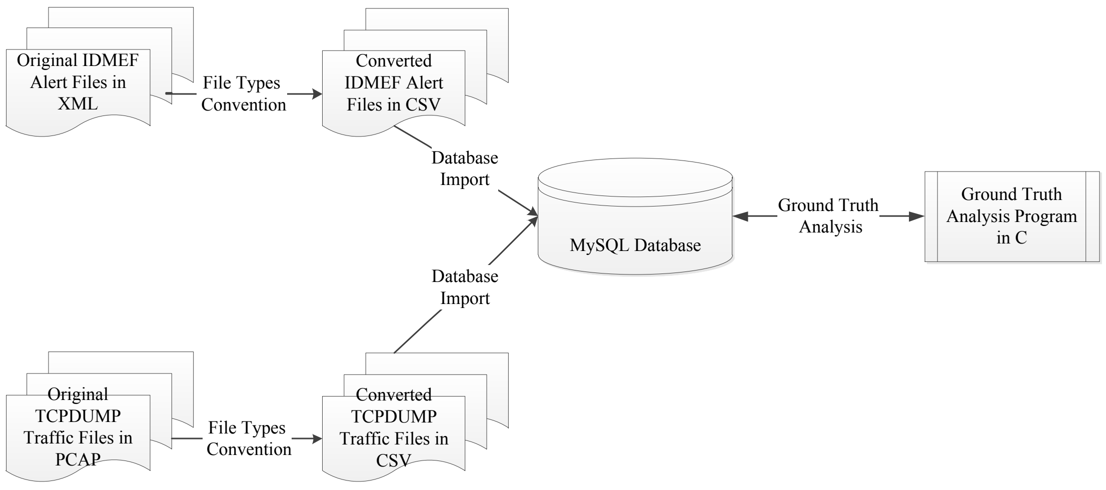

Original IDMEF Alert Files in XML
File Types Convention
Converted IDMEF Alert Files in CSV
Database Import
MySQL Database
Ground Truth Analysis
Ground Truth Analysis Program in C
Database Import
Original TCPDUMP Traffic Files in PCAP
File Types Convention
Converted TCPDUMP Traffic Files in CSV

Figure B.1: Abstracted Process of Data Set Ground Truth Analysis

alert as the network session instead of the network packet. As Snort implements the intrusion detection by examining every single network packet, it is necessary to accurately identify the malicious attack network packets from the TCPDUMP file relating to each provided attack session. Thus, the IDMEF alert file is traversed to obtain the key information (for example, attack time stamp, source IP, destination IP) of alert session record, to identify the corresponding attack packet in TCPDUMP file. Note that "*sessionduration*" field in IDMEF file represents the amount of time between the attack start and end times [54].

Figure B.1 displays the general attack packets determination process. Briefly, both sets of IDMEF alert files in XML format and the set of TCPDUMP traffic files in PCAP format are converted into files in CSV format, and then imported into MySQL database. The tables which contain IDMEF alert files are called as *IDMEF alert tables*, while, the tables which contain network traffic files are called as *network traffic tables*.

To each attack phase in DARPA2000, the developed C ground truth analysis program extracts each attack session's key information from *IDMEF alert table*. After that, the program selects the corresponding attack packet(s) with the exactly same information in *network traffic table* according to *sessionduration* value. If *sessionduration* equals to 0 second, the matched single attack packet in *network traffic table* will be labelled as attack packet; if *sessionduration* exceeds 0 second, a number of matched attack packets in *network traffic table* within session will be labelled as attack packets. The analysis program traverses the *IDMEF alert table* from the first alert record till the last one. Figure B.2 illustrates the flow chart of attack packets determination for an attack phase. The determination process repeats in all of the phases in DARPA2000. Therefore, the ground truth of data set is determined.

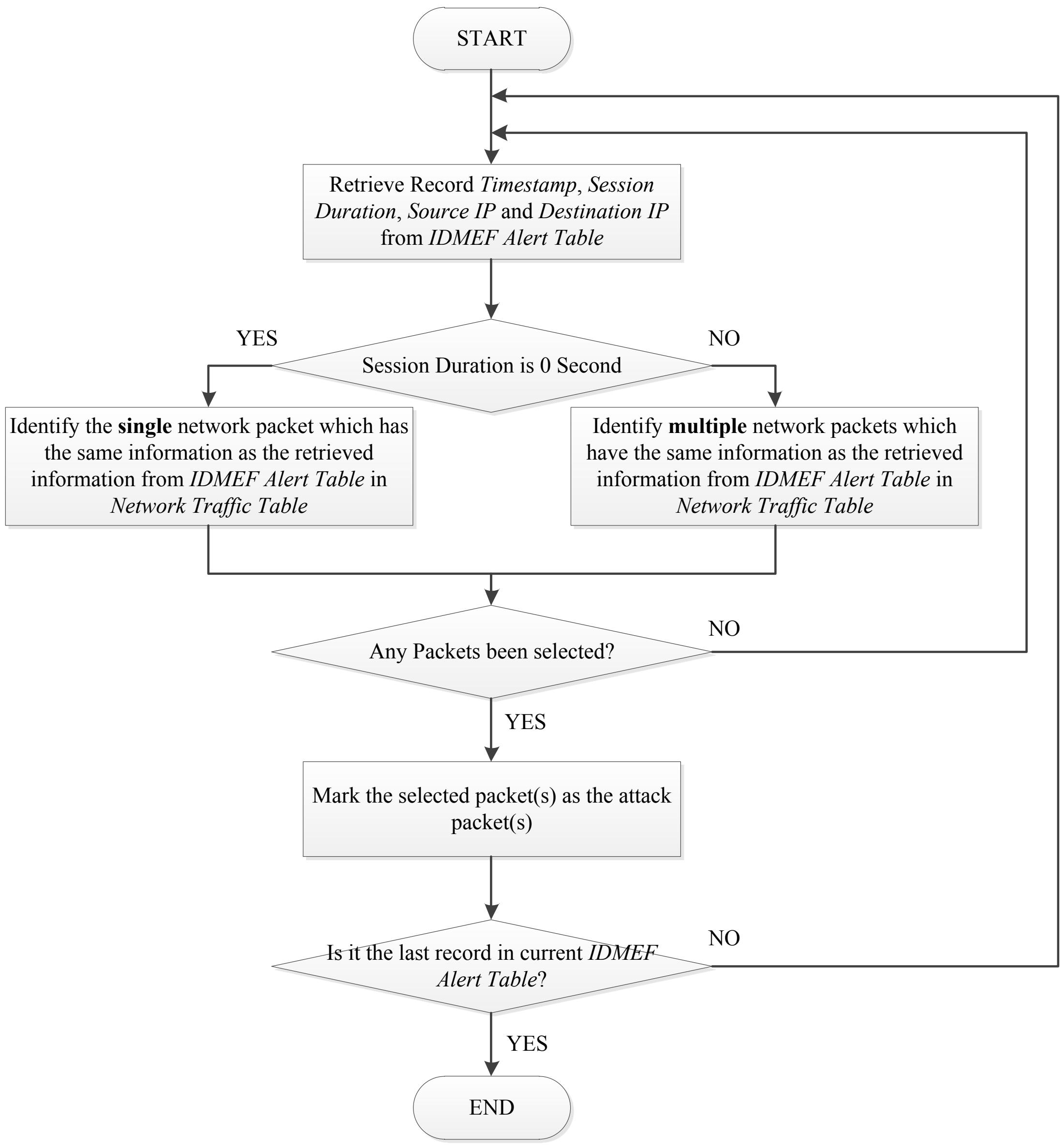

START
Retrieve Record *Timestamp*, *Session Duration*, *Source IP* and *Destination IP* from *IDMEF Alert Table*
YES
NO
Session Duration is 0 Second
Identify the **single** network packet which has the same information as the retrieved information from *IDMEF Alert Table* in *Network Traffic Table*
Identify **multiple** network packets which have the same information as the retrieved information from *IDMEF Alert Table* in *Network Traffic Table*
NO
Any Packets been selected?
YES
Mark the selected packet(s) as the attack packet(s)
NO
Is it the last record in current *IDMEF Alert Table*?
YES
END

Figure B.2: Flow Chart of Data Set Ground Truth Analysis

Appendix C

Process of Snort Detection Evaluation

This appendix describes the detailed process how we examined Snort's detection results against the obtained data sets ground truth.

As the typical attack alert from the intrusion detector contains the same characteristic values as the originally network packet has, such as source IP, destination IP, source port, destination port and protocol, the key mechanism of this process is to identify these essential alert information from each Snort alert, then, apply these information to match the corresponding attack packet from TCPDUMP traffic file.

Figure C.1 illustrates the abstracted process to examine Snort's detection performance. By replaying the exactly same TCPDUMP traffic files as *Ground Truth Determination* phase, Snort generates the relevant alerts for each phase. Once again, five alert CSV files have been generated by setting Snort alarming with CSV output and also been imported into MySQL database. Additionally, all of the converted TCPDUMP traffic files in CSV format are imported into MySQL database. The tables contain Snort alert information are termed as *Snort alert tables*, while, the tables contain raw network traffic information are termed as *TCPDUMP traffic tables*.

In each phase, the developed Snort detection examination program in C retrieves the alert information from the first alert till the last one in each *Snort alert table* from database. Firstly, the program selects a set of selected alert information of each alert. Then, the program identifies the corresponding matched attack packet in *TCPDUMP traffic table* by selecting the retrieved alert information. Once an attack packet had been matched, the examination program retrieves the next alert information and implements the matching process till all of the alerts in current *Snort alert table* been retrieved.

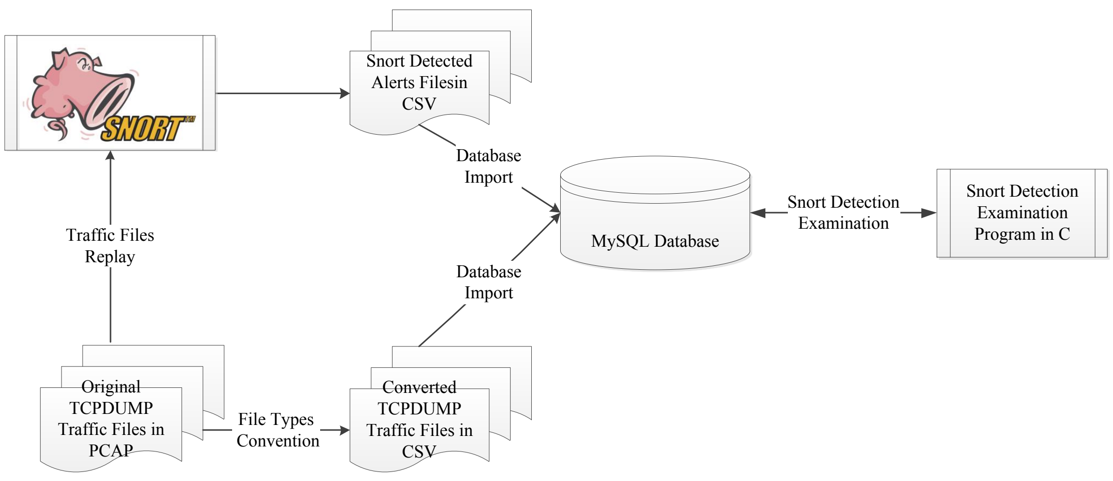

SNORT
Snort Detected Alerts Filesin CSV
Database Import
MySQL Database
Snort Detection Examination
Snort Detection Examination Program in C
Traffic Files Replay
Database Import
Original TCPDUMP Traffic Files in PCAP
File Types Convention
Converted TCPDUMP Traffic Files in CSV

Figure C.1: Abstracted Process of Snort Detection Examination

Note that there is a special scenario, where several successive alerts in *Snort alert table* have exactly same packet characteristics (for example, source IP, same destination IP, source port, destination port and protocol) but with different alert types. This phenomena is generated due to the packet information from one single packet has been matched by multiple Snort attack signatures (rules). Thus, these successive alerts only match with the relevant single attack packet.

Figure C.2 shows the process flow of Snort detection examination.

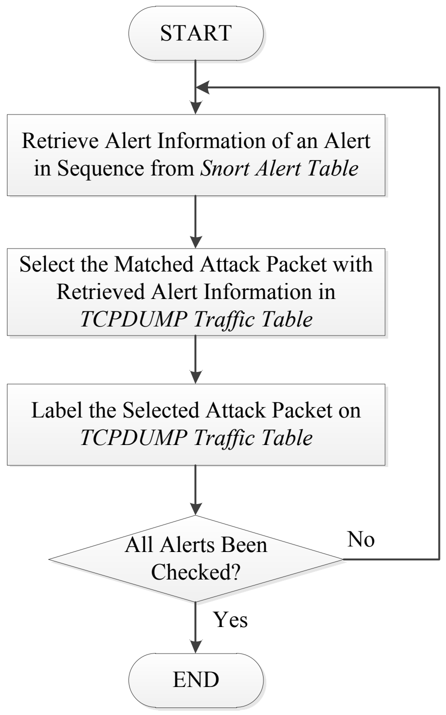

START
Retrieve Alert Information of an Alert in Sequence from Snort Alert Table
Select the Matched Attack Packet with Retrieved Alert Information in TCPDUMP Traffic Table
Label the Selected Attack Packet on TCPDUMP Traffic Table
All Alerts Been Checked?
No
Yes
END

Figure C.2: Flow Chart of Data Set Ground Truth Analysis