Rochester Institute of Technology

RIT Scholar Works

Theses

2006

TANDI: Threat Assessment of Network Data and Information

Jared D. Holsopple

Follow this and additional works at: <https://scholarworks.rit.edu/theses>

Recommended Citation

Holsopple, Jared D., "TANDI: Threat Assessment of Network Data and Information" (2006). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.

TANDI: Threat Assessment of Network Data and Information

by

Jared D. Holsopple

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering

Supervised by

Assistant Professor, Computer Engineering Dr. Shanchieh Yang
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
May 2006

Approved By:

Shanchieh Yang

Dr. Shanchieh Yang
Assistant Professor, Computer Engineering
Primary Adviser

Moises Sudit

Dr. Moises Sudit
Associate Adjunct Professor, Industrial Engineering

Greg Semeraro

Dr. Greg Semeraro
Assistant Professor, Computer Engineering

Thesis/Dissertation Author Permission Statement

Title of thesis or dissertation: TANDI: Threat Assessment of
Network Data + Information

Name of author: Jared Holsopple
Degree: MS
Program: Computer Engineering
College: College of Engineering

I understand that I must submit a print copy of my thesis or dissertation to the RIT Archives, per current RIT guidelines for the completion of my degree. I hereby grant to the Rochester Institute of Technology and its agents the non-exclusive license to archive and make accessible my thesis or dissertation in whole or in part in all forms of media in perpetuity. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Print Reproduction Permission Granted:

I, Jared Holsopple, hereby **grant permission** to the Rochester Institute of Technology to reproduce my print thesis or dissertation in whole or in part. Any reproduction will not be for commercial use or profit

Signature of Author: Jared Holsopple Date: _____

Print Reproduction Permission Denied:

I, _____, hereby **deny permission** to the RIT Library of the Rochester Institute of Technology to reproduce my print thesis or dissertation in whole or in part.

Signature of Author: _____ Date: _____

Inclusion in the RIT Digital Media Library Electronic Thesis & Dissertation (ETD) Archive

I, Jared Holsopple, additionally grant to the Rochester Institute of Technology Digital Media Library (RIT DML) the non-exclusive license to archive and provide electronic access to my thesis or dissertation in whole or in part in all forms of media in perpetuity.

I understand that my work, in addition to its bibliographic record and abstract, will be available to the world-wide community of scholars and researchers through the RIT DML. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation. I am aware that the Rochester Institute of Technology does not require registration of copyright for ETDs.

I hereby certify that, if appropriate, I have obtained and attached written permission statements from the owners of each third party copyrighted matter to be included in my thesis or dissertation. I certify that the version I submitted is the same as that approved by my committee.

Signature of Author: Jared Holsopple Date: _____

Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title: TANDI: Threat Assessment of Network Data and Information

I, Jared D. Holsopple, hereby grant permission to the Wallace Memorial Library reproduce my thesis in whole or part.

Jared Holsopple
Jared D. Holsopple

Date _____

Dedication

This is dedicated to my parents.

Acknowledgments

I would like to first thank Dr. Yang for being a very helpful advisor who has provided valuable input towards the completion of this thesis. I would also like to thank Dr. Sudit who has provided me with an extensive background in data fusion. I am also very grateful for Dr. Semeraro's suggestions.

I would like to thank John Salerno, Doug Boulware, and George Tadda at AFRL for motivating the work and providing valuable input towards the design of TANDI, Adam Stotz and Rich Giomundo at the University of Buffalo for offering their expert knowledge on cyber attacks and INFERD, and Jason Kistner and Michael Kuhl for sharing their experience on developing a simulator for cyber attacks.

Most of this work was sponsored by the Air Force Research Labs in Rome, NY

Abstract

Current practice for combating cyber attacks typically use Intrusion Detection Sensors (IDSs) to passively detect and block multi-stage attacks. This work leverages Level-2 fusion that correlates IDS alerts belonging to the same attacker, and proposes a threat assessment algorithm to predict potential future attacker actions. The algorithm, TANDI, reduces the problem complexity by separating the models of the attacker's capability and opportunity, and fuse the two to determine the attacker's intent. Unlike traditional Bayesian-based approaches, which require assigning a large number of edge probabilities, the proposed Level-3 fusion procedure uses only 4 parameters. TANDI has been implemented and tested with randomly created attack sequences. The results demonstrate that TANDI predicts future attack actions accurately as long as the attack is not part of a coordinated attack and contains no insider threats. In the presence of abnormal attack events, TANDI will alarm the network analyst for further analysis. The attempt to evaluate a threat assessment algorithm via simulation is the first in the literature, and shall open up a new avenue in the area of high level fusion.

Contents

Dedication	iii
Acknowledgments	iv
Abstract	v
Glossary	xi
1 Introduction	1
1.1 Cyber Attacks: The Execution and Detection	3
1.2 Threat Assessment as a Data Fusion Problem	7
1.3 Statistical Methods for Threat Assessment	9
1.3.1 Bayesian Networks	9
1.3.2 Hidden Markov Models	12
1.3.3 Viability of Statistical or Probabilistic Techniques in the Cyber Do- main	13
1.4 Threat Assessment Algorithm Design Considerations	14
1.5 Vulnerability vs. Threat Assessment	15
1.6 Summary of Contributions	16
1.6.1 TANDI: Threat Assessment of Network Data and Information . . .	16
1.6.2 Threat Assessment Framework Simulation and Analysis	17
2 TANDI: Threat Assessment of Network Data and Information	18
2.1 Preliminary Design of a Threat Assessment Framework	18
2.2 Definitions	20
2.3 The Framework	21
2.3.1 Inputs	22
2.3.2 Logical Topology	23
2.3.3 Potential Attack Sequence	24
2.3.4 Information Graph	26

2.3.5	Situation Assessment Stage	28
2.3.6	Threat Assessment Stage	30
2.4	Generalization of TANDI	33
2.5	Basic Implementation Details	33
2.5.1	System Organization	35
3	Simulation and Results	38
3.1	Simulated Network Topologies	38
3.1.1	Small Network Topology	38
3.1.2	Large Network Topology	39
3.2	Attack Simulator	41
3.3	Statistics for Evaluating TANDI	42
3.3.1	Assumptions/Definitions	44
3.3.2	Metric Definitions	44
3.3.3	Average Normalized Compromising Score	44
3.3.4	Percent Threatened	46
3.4	Results and Discussion	48
3.4.1	Example Attack and Threat Scores	48
3.4.2	The Sensitivity of Weights	48
3.4.3	The Effect of Weights	52
3.4.4	Abnormalities/Insider Threats	57
3.4.5	Difference in Topology Results	59
3.4.6	Coordinated Attacks	59
4	Conclusion and Future Work	62
4.1	Conclusion	62
4.2	Future Work	63
4.2.1	More Simulation Data and More Diverse Networks	63
4.2.2	Integration with Existing Cyber Attack Detector	64
4.2.3	Inclusion of Sensor Location	64
4.2.4	Speed Optimization	64
4.2.5	Dynamic PASs and Logical Topology	65
4.2.6	New Combination Rules	65
4.2.7	Expansion of Metrics	66
4.2.8	Detection of Coordinated Attacks	66

Bibliography 67

List of Figures

1.1	Flow diagram	2
1.2	Cisco MARS Screenshot	6
1.3	Example Attack Graph of Four Computers as used in [18]	11
1.4	Connection between HMM and Graph Matching in [1]	13
2.1	Example Logical Topology	24
2.2	Potential Attack Sequence Used by TANDI	25
2.3	Information Graph Template defined for each computer or group of computers.	27
2.4	An example showing the attack sequence, the information graph, the logical topology, and how they are interconnected.	28
2.5	Incorporating Network User Accounts and Shared Files into the Automatic Generation of the Information Graph	29
2.6	Situation Assessment Feature Tree	30
2.7	Threat Assessment Feature Tree	31
2.8	Screenshot of RunCyberTA in the Middle of a Cyber Attack. A larger radius indicates a higher threat.	34
3.1	The network topology used for simulations. Blocks with a dotted background are external computers while all others are internal. The corresponding IP address(es) and services running are indicated.	39
3.2	A Larger Computer Network to Simulate	40
3.3	Example threat scores during the progression of a cyber attack with the weights of $\lambda_{A^*} = 0.2$, $\lambda_{A(A^*)} = 0.05$, $\lambda_{M^*} = 0.3$, $\lambda_{M(M^*)} = 0.1$. Both the actual and normalized threat scores are shown for each entity. Nodes compromised are indicated with an 'X'.	48
3.4	Weight Combinations	50
3.5	Average Normalized Compromising Score Sensitivity (Topology 2)	50
3.6	Average Percent False Positives Sensitivity $\beta = 1$ (Topology 2)	51
3.7	Average Percent False Negatives Sensitivity $\beta = 0.6$ (Topology 2)	51

3.8	Average Normalized Compromising Score (Set 0)	52
3.9	Average Percentile Rank (Set 0)	53
3.10	Percent False Positives (Set 0) $\beta = 1$	53
3.11	Percent False Negatives (Set 0) $\beta = 1$	53
3.12	Percent False Negatives (Set 0) $\beta = 0.6$	54
3.13	Average Percent Assessee Reduction (Set 0) $\beta = 1$	54
3.14	Percent Abnormal over Sets 1-3	57
3.15	Average of the normalized compromising scores for the set of normal attacks (set 0), a set of attacks with abnormalities (set 1), and a set of insider threats (set 2).	57
3.16	Average Percentile over Sets 1-3	58
3.17	Percent False Positives over Sets 1-3	58
3.18	Percent False Negatives over Sets 1-3 $\beta = 1$	59
3.19	Average Percent Threatened on Small Network (Set 0)	60
3.20	Average Normalized Compromising Score of a Two Zombie Coordinated Attack	60
3.21	Average Normalized Compromising Score of a Four Zombie Coordinated Attack	61

Glossary

A

alert An IDS alert or system log message., p. 1.

attack A sequence of IDS alerts or system log messages that have been grouped together to represent a single cyber attack., p. 20.

attributes Fields contained within an Event, such as Signature, Target IP, or Source IP., p. 20.

C

compromised entity An entity that has been overtaken or stolen by the hacker., p. 20.

E

entity A piece of information or data that can be compromised by the hacker. These can include: existence of computers, user accounts and critical data., p. 20.

event Any IDS alert or system log message that represents something that occurred on the network., p. 20.

I

impact assessment The act of determining the consequences of a situation., p. 8.

Information Graph A directed graph that represents logical inferencing such that each node represents an entity on a network., p. 21.

S

Sequence Graph A directed graph that represents logical inferencing such that each node represents attributes of an event., p. 21.

situation The current state of an environment., p. 8.

T

threat a situation that could lead to unauthorized access, impairment of network operations, or a loss of data integrity., p. 21.

threat assessment The act of projecting a current situation into the future to determine what could happen., p. 8.

Chapter 1

Introduction

This work develops a new threat assessment framework for cyber attacks that have been detected on a computer network. This framework is known as TANDI: Threat Assessment of Network Data and Information. In addition to developing a threat assessment framework, it also presents a framework for evaluating threat assessment methods.

Computer networks use intrusion detection sensors (IDS's) to monitor network traffic for suspicious activities. If any such activity is detected, an IDS generates an appropriate alert. These alerts vary by the IDS used, but typically include reconnaissance actions (such as “ping” or a port scan) and intrusion attempts (such as logon failures or service exploits). Since cyber attacks typically occur in multiple stages over multiple machines [14], an ideal computer security tool should be able to correlate multiple alerts generated by the same cyber attack and assess the threat associated with each attack. However, the popular tools in use today are unable to correlate IDS alerts to detect multi-stage multi-machine attacks and cannot therefore assess the threat of such an attack. These tools typically assess the threat based on the alerts generated for one machine or subnet. This method, however, is not comprehensive enough, because the alerts generated for one machine may not all be correlated to the same attack or may just be normal network traffic. Also, since most cyber attacks occur over more than one machine or subnet, the significance of an attack cannot be determined without correlating alerts generated for different computers. For large networks, this is a severe limitation. These networks are usually segmented by a firewall. Servers outside

of the firewall are vulnerable to direct attack from an Internet computer, whereas computers internal to the firewall are not. Therefore, an external hacker must first attack a server outside the firewall. If this server is compromised, the hacker could use it as a stepping stone to compromise other computers on the network. Since the hacker may be unaware of the location of critical information, multiple computers will likely be attacked before the hacker actually compromises the desired information. Research and development efforts are currently in place to correlate alerts due to multi-stage multi-machine cyber attacks; however, these tools lack *threat assessment*. In other words, they are able to detect and distinguish incoming cyber attacks, but are unable to rank them based on the threat posed to the network or network entities. One such tool that lacks threat assessment but is able to detect incoming cyber attacks is INFERD, which has been developed and currently in revision [28]. Much of the existing work on INFERD was leveraged for the development of TANDI.

Figure 1.1: Flow diagram

This work develops a novel threat assessment framework for multi-stage multi-machine cyber attacks. It is developed with the assumption that a lower-level alert correlator has already detected one or more cyber attacks. This framework was then simulated in Java using semi-randomly generated data that would be consistent with the output of an alert correlator such as INFERD [28]. Figure 1.1 illustrates how the threat assessment algorithm is

built as an additional processing unit of a cyber attack detection system. The threat assessment algorithm takes the correlated alerts, basic computer and network configurations, and topology as inputs. It outputs numerical values, known as the *threat score* representing the threat levels of different entities that have not yet been compromised by incoming cyber attacks. An *entity* in this application is something that is present logically or physically in the network, such as a user account, existence of computers and data.

The developed framework was simulated using Java. Since there is limited cyber attack data available to the public [30], attack data was generated semi-randomly that was consistent with actual cyber attacks that would occur on the simulated networks. The framework was simulated over multiple networks varying in size and connectivity.

The second contribution is a set of general threat assessment metrics that can be used to analyze different threat assessment frameworks against each other using the same data. These metrics, such as *average normalized compromising score*, *percent false positives*, and *percent false negatives*, provide the analyst with values to determine the accuracy of the past threat assessment. These metrics can be used to modify parameters or models in the threat assessment framework. Other metrics were developed that provide the analyst a means in which to rank attacks against each other.

1.1 Cyber Attacks: The Execution and Detection

Before developing a cyber attack threat assessment framework, one must first have a basic understanding of how a hacker can attack a computer network. In addition, one must also understand the process by which a network analyst typically goes through to detect cyber attacks.

A good example of a cyber attack on a Windows network can be found in [14]. The example, albeit simple, provides a good illustration of how a typical cyber attack may transpire. Note that the term “typical” is used loosely here as there are potentially an infinite number of ways a computer can be compromised. In this example, a hacker first discovers

the IP addresses of a fake organization's computers that are external to the firewall. The hacker is also able to scan the open ports of each computer as well as query these computers for the versions of software currently running. This procedure, known as network footprinting, allows the hacker to plan his first attack on the network and is usually executed by scripting. Based on information obtained through footprinting, the hacker decides to perform SQL injection on the web server to gain access to a book ordering page. SQL injection is an exploit performed on servers that do not validate usernames before querying a database for login information. In this case, the hacker passed "foo' OR 1=1;--" as the username, which, when executed as part of an SQL query, will return all results thus allowing the user to be authenticated by the SQL server. The hacker is then able to gain root privileges on the SQL server by uploading a script via TFTP (which is installed on all Windows machines by default) to the SQL server that grants the hacker administrative privileges. Now the hacker can download a user list with encrypted passwords. The hacker then runs an offline script that decrypts the passwords. The hacker then tries to login with the usernames and passwords on the data center computer until one works, which then allows him to penetrate the firewall.

While the attack presented above is now unlikely since most networks are not vulnerable to the SQL injection query anymore [14], it does provide a basic understanding of how a hacker needs to attack and compromise a network, which involves: reconnaissance, intrusion, privilege escalation, and data transfer. Note that, some hackers blindly execute service exploits before ever discovering which hosts actually exist. This is typically accomplished by a script that sends the malicious traffic to a predefined set of IP addresses and TCP ports. Also, some exploits, such as the RPC DCOM attack [20], immediately grant system level privileges on the victim computer, so privilege escalation is accomplished at the same time as intrusion. Since new exploits are being discovered and patched everyday, cyber attacks are constantly changing and evolving.

In combating cyber attacks, network administrators or security analysts (who will be referred to as network analysts) must constantly monitor the network for any abnormalities

in network traffic and performance to detect and avert incoming attacks. Network analysts typically place IDS sensors such as Snort [27], Dragon [8], and DaiWatch [3] in the network to analyze and report anomalous traffic. In addition to the IDSs, system logs on important machines are also usually configured to report any logon failures or suspicious activity. These IDS alerts and log messages are typically organized in a spreadsheet or database format that must be manually analyzed by the network analyst. Tools such as Cisco System's netForensics [12] aggregate these alerts into a spreadsheet format for the network analyst. Based on the reported alerts, netForensics classifies alerts on a scale of 1 to 5, with 5 being the most severe. It calculates the threat level on machines being attacked based on the number and severity level of alerts using the following equation:

$$Threat\ score = \sum_{n=1}^5 (Alert\ count_n) * (2^{n-1} - 1) \quad (1.1)$$

where n is the severity of the alert. This calculation is also calculated for machines that are performing the attack. The threat score defined by Cisco is basically a weight count of alerts detected for each machine.

The risk factor is then calculated for each of the organizations computers using the following equation:

$$Risk\ Score = Threat\ Score * Popularity * System\ Value * Exposure \quad (1.2)$$

where Popularity, System Value, and Exposure are assigned by the network analyst with the following definitions:

- **Popularity** indicates how often a machine might be attacked.
- **System Value** indicates the importance of the assets associated with the computer.
- **Exposure** is based on the OS and number of network services running on the machine. This value is indicative of the likelihood that the machine could be attacked. For example, a server with 10 services running should have a higher exposure value than a server with 5 services running since there are more potential vulnerabilities in the server running 10 services.

The Cisco threat assessment technique is very limited. This technique provides an interface for the network analyst to define manually. It only correlates the alerts on a by machine basis. Since cyber attacks usually progress over multiple machines, a network analyst must manually correlate alerts occurring on different computers to get an indication of what attacks a hacker has executed on the network. It can easily be seen that a multi-computer cyber attack detection scheme can drastically improve the efficiency of analyzing potentially malicious traffic.

Cisco Systems has also recently released a security program called Cisco Security Monitoring, Analysis, and Response System, or Cisco MARS for short [29]. While Cisco MARS does illustrate an ‘Attack Path’, the product literature suggests that this attack path is merely the path of a single malicious network packet. Cisco MARS provides the analyst with a spreadsheet-based output of the alerts for malicious traffic (shown in Figure 1.2) as well as a plethora of graphs for trend analysis. However, like netForensics, Cisco MARS does not seem to have the ability to correlate alerts across multiple machines to form a complete attack from a single source.

ID	Event Type	Source IP/Port	Destination IP/Port	Protocol	Time	Zone	Reporting Device	Graph	False Positive	Mitigate
10315 1390	Sudden increase of traffic to a port [3]	0.0.0.0 [3]	0.0.0.0 [3]	445 [3]	Mar 3, 2004 4:05:03 AM EDT	NUT	demios [3]	[3]	Tune	Mitigate
	AAA authorization denied due to no prior authentication [3]		Total: 25							
	AAA authorization denied due to no prior authentication [3]	130.120 [3]	Total: 3							
	AAA authorization denied due to no prior authentication [3]	131.142 [3]	Total: 2							
10544 1390	AAA authorization denied due to no prior authentication [3]	15.136.95 [3]	4049 [3]	445 [3]	May 3, 2004 5:40:05 AM EDT	NUT	cerberus2 [3]	[3]	Tune	Mitigate
	AAA authorization denied due to no prior authentication [3]	15.136.104 [3]	Total: 3							

Figure 1.2: Cisco MARS Screenshot

In spite of their short comings, Cisco netForensics and MARS seem to be the only commercially available tools that attempt to assess the threat on a computer network. It is highly likely that some companies have developed their own software to assess the threat on their network. However, if a commercial tool was developed, companies could save the time developing an in-house tool by just purchasing and configuring the commercial tool.

Despite the drawbacks of Cisco netForensics and MARS, the amount of data could still be manageable for a small network. However, such a system operating on an enterprise network could lead to information overload for the network analyst. Recent research based on data fusion, such as INFERD [28], has focused on automatically correlating alerts into separate independent attacks. INFERD creates dynamic objects known as “attack tracks” each of which represent a single attack. An attack consists of alerts generated for multiple computers providing the analyst with multi-machine cyber attacks. However, INFERD (and other such systems) lack a tool to predict what could possibly happen next. This prediction would allow the analyst to quickly assess which attacks are able to next compromise critical data or information on the network.

1.2 Threat Assessment as a Data Fusion Problem

A multisensor network is a network set up with many different types of sensors to monitor the environment or detect anomalies. The raw sensed data often has no meaningful interpretation. However, the data can be combined to provide a better view of the situation or the status of the environment. The process of combining the data into meaningful information is known as data fusion.

Data fusion is formally defined by the Joint Director’s Laboratory (JDL) [32] as

A multilevel, multifaceted process dealing with the automatic detection, association, correlation, estimation, and combination of data and information from single and multiple sources.

JDL defined 5 levels of processing in a data fusion system as summarized below [9].

Level 0 processing involves how the sensors obtain readings and also refines the signal obtained by the physical sensor. This is typically a signal processing problem.

Level 1 processing is known as object refinement. This level correlates the sensed data to identify and classify an entity. For example, a multisensor network may be set up to

detect enemy missiles. Level 1 processing combines the data from the sensors to determine if the detected object is an enemy missile.

Level 2 processing is known as situation refinement. It can also be called situation assessment. This level further refines the level 1 information by trying to determine attributes associated with each detected entity. Using the missile detection example, level 2 fusion might use temporal and spatial correlations to determine the direction and the speed of the missile.

Level 3 processing is known as threat refinement. It can also be called impact assessment or threat assessment. While most literature use impact assessment and threat assessment interchangeably, this work distinguishes between the two. Threat assessment is the process of projecting a current situation into the future. Impact assessment is the process of determining the potential consequences of a particular situation. This work focuses on threat assessment to aid in impact assessment. If level 2 fusion determined that a missile was moving towards an ally camp that was ten miles away, level 3 fusion would identify the missile as being a high threat to the ally camp.

Level 4 processing is known as process refinement. This attempts to reposition sensors or adjust sensor capabilities to gain a better view of the situation. It also attempts to automatically react to the current situation [9]. While this level was originally included in the data fusion model, recent works have suggested that this level should not be included because it is decision and reaction but not data fusion [19].

Many computer networks are set up with intrusion detection sensors (IDSs) that monitor network traffic and generate alerts when suspicious behavior occurs on a computer network. For example, an IDS would generate an alert for an ICMP message from an external IP address since that could be a signal that a hacker is trying to determine what machines on the network are available (Levels 0 and 1). These alerts can be correlated based on the source and destination IP addresses (along with other parameters) to group alerts as part of a specific attack (Level 2). The set of correlated alerts can then be analyzed to assess the threat of the attack and predict what could happen (Level 3). So it can be seen that

cyber attack detection can be a data fusion system [4]. One such tool that is still under refinement is INFERD which uses data fusion as the basis for the detection of cyber attacks [28]. INFERD currently only supports level 1 and level 2 fusion.

Little, et al. suggest that a threat assessment framework should be able to encompass capability, opportunity, and intent [17]. The capability of the attacker is determined on the education of the attacker in the domain and the tools he is able to use to execute the attack. Opportunity is determined by the current situation dictates what the attacker is able to do. Intent is simply the reason or motivation for the attack.

1.3 Statistical Methods for Threat Assessment

The authors of [2] suggest the use of Bayesian Networks (BNs) and Hidden Markov Models (HMMs) for threat assessment. This section describes how BNs and HMMs could be used to implement a threat assessment algorithm.

1.3.1 Bayesian Networks

A Bayesian network is a directed acyclic graph in which the nodes represent random variables and the edges signify conditional dependencies between the pairs of nodes. A set of posterior probabilities, $\mathbf{P}$, for each node is also needed. Formally, let $\mathbf{X} = X_1, \dots, X_n$ denote a set of random variables that correspond to one node in the Bayesian graph. Then $\mathbf{P} = \{P_i\}$ where $P_i = P(X_i | \mathbf{P}_{\mathbf{a}_i})$, $\mathbf{P}_{\mathbf{a}_i}$ is the set of predecessors for node X_i [13, 18].

If each node represents an event that could occur and each edge is assigned a transition probability, P_e , the above definition for P_i should be changed to $P_i = P(X_i | \mathbf{P}_e^*)$, where $\mathbf{P}_e^*$ is the set of incoming edge probabilities from only the predecessor nodes that have already been asserted. A node is considered to be asserted when the event it represents occurs. In other words, if a transition probability is assigned as a weight to each edge in the graph, then P_i is the joint probability of the transition probabilities from predecessor events that have already occurred.

In terms of threat assessment, Bayesian networks can be used to predict upcoming events. If one defines the nodes such that each node represents an event and the edges such that the source event can directly precede the destination event, $P_i = P(X_i|\mathbf{P}_e^*)$ indicates the probability that X_i occurred based on past events given the probabilities that the predecessors of X_i have occurred. If one assumes that event X_i can be detected when it occurs, this probability corresponds to the likelihood that X_i will occur.

Bayesian networks can be applied to cyber attacks by defining a course of action in which the hacker can penetrate the network. This course of action could be defined by the types of attacks the hacker could perform or the information the hacker has compromised during the attack. Phillips and Swiler suggest that this course of action can be generated based on the topology of the network, the services and configurations running on each machine, user groups, attacker profile and other network characteristics [23]. Each edge is assigned a probability that represents the probability of success. These edge probabilities can be used to compute the most likely path of the attacker. This can be determined using a shortest path algorithm. For this to work, the negative logarithm ($-\log$) of each probability can be used as an edge weight so that the summation of the edge weights is equivalent to the product of the corresponding probabilities. For very large networks, this attack graph can become very large and unwieldy. So the authors suggest ideas to increase the scalability – such as building blocks used to automatically create portions of the course of action.

Liu and Man [18] also develop a Bayesian vulnerability assessment technique. They use an attack graph similar to that of Phillips and Swiler, however, only root privileges, user privileges, and basic topology are used to construct the graph. An example attack graph Liu and Man use is shown in Figure 1.3. It illustrates that a hacker must first obtain user or root privileges to computer ip0 before gaining privileges to computer ip1. The rest of the attack graph is similarly structured. One limitation of this implementation is that the graph must be acyclic. So, for example, if it is possible for a hacker to gain root access to ip3, then compromise ip2, a new attack graph must be created. Since most networks have a very dense connectivity this would yield a large number of attack graphs that need to be

Figure 1.3: Example Attack Graph of Four Computers as used in [18]

analyzed.

Liu and Man assume that only privileges are needed to assess network vulnerability. While the intent of their paper was not threat assessment, this idea can be extended to threat assessment. Since different privileges allow access to different files, some privileges (such as network administrator) are more important to the integrity of the network. Different privileges typically allow different levels of access to the information and data on the network. If a hacker has gained low level privileges, such as a guest account, the critical data has not yet been compromised. So while illegal guest access to the computer should be removed as soon as possible, the situation is not in a high threat situation. The threat assessment algorithm proposed in this work distinguishes between the different privileges.

1.3.2 Hidden Markov Models

A Hidden Markov Model (HMM) is a finite set of states. Each of these states is associated with a probability distribution. State transitions are defined by a set of probabilities called transition probabilities. An observation is generated according to the associated probability distribution in a state. A state is considered “hidden” because only its observation is visible to the outside, not the state itself.

Three assumptions are made in HMM theory. First, the next state is dependent only on the current state. Mathematically, this is shown in (1.3) where a_{ij} is the transition probability from state i to state j , P denotes probability and q_t is the state at time t . This is different from a BN in that a current state in a BN may be dependent on more than one of the previous states. Therefore, an HMM is a special case of a BN.

$$a_{ij} = P\{q_{t+1} = j | q_t = i\} \quad (1.3)$$

The second assumption is that the transition probabilities do not change with time. The final assumption is that the current observation is statistically independent of the previous observations [24, 33].

While there is currently no literature supporting the use of HMM’s for cyber attack threat assessment, HMM’s have been used in other domains for threat assessment. The authors of [1, 25, 2] use HMM’s to detect, track, and predict terrorist activities. In their models, a specific sequence of events corresponds to each state. Given the actual detected sequence of events, they perform a graph matching algorithm that identifies the most likely current state of the HMM. Figure 1.4 illustrates the connection between the HMM and graph matching. Based on this current state, the model can predict future activity via the transition probabilities.

Figure 1.4: Connection between HMM and Graph Matching in [1]

1.3.3 Viability of Statistical or Probabilistic Techniques in the Cyber Domain

While the above Bayesian-based approaches are comprehensive ideas, it may not be realistic for large and complex enterprise networks if no aggregation of nodes is used. Even if the course of action can be generated for such a network, assigning the probabilities is not a trivial task. The probabilities could be assigned in three ways.

First, one or more subject matter experts (SMEs) could assign the probabilities manually [2]. However, with a large number of nodes, this could be a very time consuming and potentially inaccurate and high-maintenance task. Even if the SMEs are able to assign probabilities, there is a high likelihood that the probabilities will have a wide variance, thus making them unreliable.

The second way to assign the probabilities is by training based on historical data sets [2, 33, 7]. Unfortunately, the non-stationary nature of cyber attacks, i.e., old attacks will soon be obsolete and new attacks are being invented every day, presents a fundamental flaw

to this approach. The historical data set may never be representative for the future courses of action. However, a more limiting factor to training the probabilities is the lack of data itself [30].

The third approach is simply a combination of the first two. These observations suggest that, while the Bayesian network, at first glance, seems to be a good technique to assess threats in the cyber domain, the assignment of probabilities makes it not a viable choice.

1.4 Threat Assessment Algorithm Design Considerations

While much research has been conducted in terms of detecting cyber attacks [33, 28], little research has involved assessing the impact of a cyber attack. According to [30], the reason is because of the lack of cyber security data. Many companies will not release cyber security data publicly because of liability, loss of reputation, and competition issues. This leads to a great challenge for advancing impact assessment of cyber attacks, because any techniques that are developed cannot be compared against each other using a common set of data benchmarks. It is also currently unclear how to assess the accuracy or precision of a threat assessment algorithm. This potentially limits the credibility of a proposed impact or threat assessment algorithm for cyber attacks. This is one of the major roadblocks in the development of cyber security systems.

Salerno, et al, introduces techniques to evaluate situational awareness of data fusion systems [26]. This paper focuses on level 2 and lower data fusion but does not directly address level 3.

Kang and Mayfield propose a set of questions in [15] that should be answered in the analysis of a threat assessment algorithm for cyber attacks. Some of the important questions are:

- Can it predict attack timeline, direction, and the potential impact of the attack?
- Can it differentiate between the main attack and decoys?

- How precise and accurate is the algorithm?
- Does it generate easy to understand reports?

Despite this set of proposed questions, there are currently no common performance metrics for a threat assessment algorithm. So in addition to developing the algorithm, this work will also propose performance metrics used to evaluate such an algorithm.

1.5 Vulnerability vs. Threat Assessment

As previously mentioned, threat assessment projects an attack into the future to determine potential consequences of an attack. This work will try to determine what the hacker is attempting to do and identify the consequences of the attack. Vulnerability assessment identifies vulnerabilities *before* a system is implemented or attacked to determine how secure the system is. These vulnerability assessment techniques, however, can be used during system operation to project what could happen in the future. So while threat assessment and vulnerability assessment are different, ideas and techniques originating from vulnerability assessment can be used for threat assessment.

Impact and threat assessment are both considered to be part of level 3 of the JDL data fusion model [9]. One interpretation to distinguish impact assessment from threat assessment can be as follows. Impact assessment is the process of determining the consequence of an attack. Threat assessment models should take three variables into account – capability, opportunity, and intent [17] to predict the future actions of an attacker. Each of these variables can be used in conjunction with each other to determine what the attacker is able to do, what the attack can do, and what the attacker plans to do.

The authors of [23] set forth a definition such that an ideal network vulnerability analysis should entail the following:

Ideally, a network-vulnerability risk-analysis system should be able to model the dynamic aspects of the network (e.g. virtual topology changing), multiple

levels of attacker ability, multiple simultaneous events or multiple attacks, user access controls, and time-dependent, ordered sequences of attacks.

This definition implies what a cyber attack impact and threat assessment algorithm should include.

Changwen and You [6] use a decision making matrix to determine the threat of enemy vehicles and missiles in a military application. Their technique incorporates multiple attributes such as altitude, velocity, and attack angle. Fuzzy membership functions are defined to calculate the overall threat. These membership functions must be manually created, and may not be suitable in the cyber domain where unknown attacks are possible and being invented every day.

Vidalis and Jones [31] use vulnerability trees to model attacks where the root of the tree is the goal of the attack and the child nodes define a course of action. The possible exploitation of each vulnerability is modeled by the educational complexity of the attacker, which measures how advanced the attacker must be to exploit the vulnerability. This procedure may not be feasible for systems that have a large number of goals. A different feature tree must be developed for each goal, so the generation of these feature trees for a large number of goals could be potentially tedious and error-prone.

1.6 Summary of Contributions

1.6.1 TANDI: Threat Assessment of Network Data and Information

The main contribution of this work is the development of a novel cyber attack threat assessment framework named Threat Assessment of Network Data and Information (TANDI). TANDI is actually a combination of level 2 (situation assessment) and level 3 (threat assessment) data fusion. Besides being the first of its kind, the novelty of TANDI lies in the separation of how (Potential Attack Sequence), where (Logical Topology), and what (Information Graph) the hacker can attack. These three directed graphs can be developed

independent of each other and are fused together to determine critical network entities that have been compromised, which is situation assessment. Based on this result, TANDI will then try to predict the next critical entity that will be compromised by combining a fixed number of weights assigned to nodes in each of the three graphs. These weights are assigned based on what has already happened, and what is likely to happen next. The evaluation of a feature tree underlying each node in the information graph yields a threat score, which corresponds to the threat to the entity being compromised next. This prediction is referred to as threat assessment. While only one feature tree structure was used in this work, the structure could easily be changed to provide a different (and potentially more accurate) evaluation of the weights. The separation of how, where, and what and the feature tree make TANDI a flexible and scalable framework. Results also indicate that TANDI can be potentially used to detect coordinated attacks.

1.6.2 Threat Assessment Framework Simulation and Analysis

The second contribution involves the evaluation of the performance of a threat assessment framework. While there exist threat assessment frameworks in domains other than computer networks, there is no common way to evaluate frameworks against each other. This work provides several common metrics, such as *average normalized compromising score*, *percent false positives*, and *percent false negatives* that can be used to compare frameworks over the same set of data. Other metrics, such as *percent threatened*, can also be used to evaluate the overall threat of an incoming attack. The use of these metrics is illustrated by evaluating TANDI over different network topologies and cyber attacks with synthetic cyber attack data.

Chapter 2

TANDI: Threat Assessment of Network Data and Information

This chapter discusses the framework and implementation of Threat Assessment of Network Data and Information (TANDI). Section 2.1 discusses an analysis of the supporting work to identify key ideas that TANDI should implement. The basic framework is then illustrated in Section 2.3 and followed by a discussion of the generalization and implementation of TANDI.

2.1 Preliminary Design of a Threat Assessment Framework

To assess the threat of a cyber attack, one must first define what a threat is. This work will define *threat* in a cyber attack context as a situation that could lead to unauthorized access, impairment of network operations, or a loss of data integrity. An entity is considered to be *compromised* if it has been overtaken by the attacker.

To determine the threat on a network, one can consider the six basic questions that can be used to describe a situation: How, Where, When, What, Why, and Who. Among them, the IDS alerts provide information to indicate (or at least imply) the *how*: the methods the attackers used to penetrate the network, the *where*: the machines or subnets compromised

(or attempted to be compromised) by the attackers, and the *when*: the time the attacks took place. The *what* is defined, in this work, as the network data and information, such as the existence of a machine, the root privilege, or Oracle database, the attackers may target on in each stage of the attack. The determination of the *why* and the *who* may require forensic analysis by experts and is out of the scope of this work. Note that the *when* can be indicative to the attacker’s behavior and used to project the time of the next attack. However, there is no consensus yet among subject matter experts (SMEs) on how to use such information.

Since Bayesian networks (BNs) require the assignment of a large number of probabilities, the use of a BN is not viable for a cyber attack threat assessment. The assignment of probabilities can be avoided by the use of *logical inferencing*. Logical inferencing is simply “if A then possibly B”. For example, if it is found that a hacker has compromised a user account, then accessing the critical files associated with that user is considered to be a logical next step in the attack. However, there are many possible ways a hacker can take to compromise the same entity, therefore it is desirable if many attacks with the same intent are aggregated together. This is a valid assumption, because, for the sake of threat assessment, what could happen next is more important than the attack itself.

Network-specific knowledge is needed for accurate threat assessment. Computer networks vary widely in size and purpose, so are therefore vulnerable to some different exploits. This knowledge allows a threat assessment framework to filter out any false positives. For example, consider two networks, each of which contains a server running Windows Server 2003, where only one of them is patched against the RPC DCOM attack. The RPC DCOM attack exploits a buffer overflow in the Remote Desktop service that allows a hacker to obtain a system-level command prompt remotely [20]. If an IDS detects the RPC DCOM attack, the server on the patched network should not be compromised, since it is not vulnerable to the attack and the alert is a false positive. However, the unpatched server should be considered compromised. Now that the unpatched server is compromised, that server could then be used to compromise the patched server. Accurate threat assessment also requires knowledge of network topology. If a compromised server has no access to

another server on the network, the other server should not be threatened. This information can be automatically obtained by a network vulnerability scanner such as NESSUS [21].

In summary, this work has identified that a threat assessment framework for a cyber attack should include:

1. A definition of threat.
2. A fusion of Who, What, Where, When, Why, and How.
3. A non-Bayesian technique to determine a quantitative value for the threat.
4. Network-specific knowledge that includes (but not necessarily limited to): network connectivity, operating systems, and running services.

2.2 Definitions

To ensure clarity, the following terms are defined in alphabetical order.

- *alert* - an IDS alert or system log message.
- *attack* - a sequence of IDS alerts or system log messages that have been grouped together to represent a single cyber attack.
- *attributes* - Fields contained within an Event, such as Signature, Target IP, or Source IP.
- *compromised entity* - an entity that has been overtaken or stolen by the hacker.
- *entity* - a piece of information or data that can be compromised by the hacker. These can include: existence of computers, user accounts and critical data.
- *event* - Any IDS alert or system log message that represents something that occurred on the network.

- *Information Graph* - A directed graph that represents logical inferencing such that each node represents an entity on a network. An edge $A \rightarrow B$ implies that if A is compromised, then B can be compromised next.
- *Sequence Graph* - A directed graph that represents logical inferencing such that each node represents attributes of an event. An edge $A \rightarrow B$ implies that if A occurs, then B is able to occur
- *threat* - a situation that could lead to unauthorized access, impairment of network operations, or a loss of data integrity.

2.3 The Framework

TANDI is a novel threat assessment framework for cyber attacks that assigns numerical threat scores (also referred to as threat values) to critical network entities based on events that have occurred. This is done on a per-attack basis. The critical entities comprise the *Information Graph*, which represents a logical sequence of how the entities can be compromised. TANDI also uses two additional *Sequence Graphs* called the *Potential Attack Sequence (PAS)* and the *Logical Topology*. The PAS is a graph that describes a logical ordering of alerts that would occur in a cyber attack. This is developed separately of the logical topology, which describes the order in which computers must be compromised assuming an attack that originates outside of the network.

While TANDI is referred to as a threat assessment framework, it also incorporates level 2 fusion into its framework. This can be seen by the two processing stages. The first stage is *Situation Assessment*, which corresponds to level 2 of the JDL Data Fusion model [32]. This determines which entities in the network have already been compromised by the attack. This processing stage can be thought of as a transformation of the detected attack from the sequence of suspicious events on the network as detected by the alert correlator to specific entities that have been compromised.

The second stage is the threat assessment stage. In this stage, the PAS and the Logical Topology are processed along with the Information Graph to determine what entities are the next likely target of the attack. TANDI then calculates a threat score of each of these entities based on four weights.¹

Of the six basic questions presented in Section 2.1, three are directly addressed - how (PAS), where (Logical Topology), and what (Information Graph). *Why* and *who* are not directly addressed by this work. *Why* is simply the intent of the hacker, while *who* is the actual hacker. These will likely require a behavioral analysis out of the scope of this work. Therefore, the hacker will always be considered to be an advanced hacker with a malicious intent. TANDI incorporates *when* by factoring in the sequence of the alerts occurring within an attack, but ignores the exact time elapsed between the alerts since there is not a general consensus on how this should be interpreted.

Unlike other threat assessment or vulnerability frameworks, the PAS, Logical Topology, and Information Graph are allowed to be cyclic. This is a distinct advantage over the frameworks since TANDI is able to assess a different ordering of alerts in an attack using the same graphs, whereas the frameworks requiring a cyclic graphs must use different graphs for different ordering of alerts.

2.3.1 Inputs

TANDI requires the following inputs to construct a network-specific threat assessment model prior to running threat assessment in real-time:

1. A Logical Topology
2. A Potential Attack Sequence
3. An Information Graph

¹This work only uses one calculation which relies on four weights. This framework is general enough to support other (potentially more accurate) calculations with more or fewer parameters.

4. The Operating Systems and services running on each computer

Discussions on how to obtain these inputs will be illustrated in the next few sections.

At run time, TANDI takes the currently detected attacks as the inputs. Each attack consists of alerts that indicate the hacker's current attempts (successful and unsuccessful) to gain unauthorized access to the network and are considered to be independent of other hackers, meaning that each attack is processed separately. There are currently no commercial programs that are able to correlate alerts at run time into specific attacks. This work, thus, leverages work from the research community and mimics outputs from INFERD [28] as the inputs to TANDI.

2.3.2 Logical Topology

The logical topology incorporates the basic question of *where* into TANDI. It is a sequence graph that defines the order in which computers can be compromised on the network. The logical topology is used to predict a next possible target on the network. TANDI assumes that cyber attacks originate from computers in the Internet (i.e., outside of the network), so only computers exposed to the Internet can be compromised directly from the Internet, such as Web and VPN servers. These will be described as *external*. Computers inside a firewall are safe from direct compromise from the Internet and will be called *internal*. Many larger networks are segmented into smaller networks that have their own firewalls, so there can be multiple levels of internal computers.

Nodes on the logical topology can represent one or more computers. All computers aggregated to the same node should be running the same processes, meaning that they are all vulnerable to the same attacks. These aggregate nodes can be thought of as a subnet of workstations.

Figure 2.1 shows a small logical topology of a network with a web server, a FTP server and two subnets behind a firewall, each with two computers. Notice how the two subnets are not connected to Internet computers since they are not exposed to the Internet.

Figure 2.1: Example Logical Topology

The edges are defined by the routing rules and access-lists in the network. For example, the lack of connectivity in the logical topology between the two subnets indicates that there is a routing rule preventing traffic flow between them. Therefore, the logical topology can be generated by referencing the current configurations on the routers and switches in the network.

One current limitation of the logical topology is that it must be static. It is likely that a hacker may actually compromise a router and reconfigure the access lists and routing tables, which would alter the logical topology. Nonetheless, this requirement of a static logical topology was used to decrease complexity. So TANDI is unable to properly react to changes in the logical topology. This limitation will be the subject of future research.

2.3.3 Potential Attack Sequence

The potential attack sequence (PAS) incorporates the basic question of *how* into TANDI. It represents a logical order of how alert signatures will occur in a cyber attack and may be developed independent of the logical topology. The PAS is used to predict the next likely methods of attack by the hacker. One or more alert signatures is represented by each node in a PAS. The PAS is expected to encompass the thousands of alert signatures from IDS

and system logs. Therefore, the PAS is the most complex and error prone input to TANDI.

Figure 2.2: Potential Attack Sequence Used by TANDI

The developers of INFERD [28] had already implemented a PAS, which they call a *guidance template*, of equivalent functionality. A slightly modified version of this PAS was used for the implementation of TANDI. In the developed PAS, the alerts were classified into different categories pertaining to the type of attack. The main categories are:

1. *Reconnaissance* - alerts that indicate information gathering. These include such alerts as ICMP Pings, TCP SYN Scans, TCP Port Scans, etc.
2. *Intrusion Root* - alerts that indicate a system compromise where the hacker has obtained root or system-level privileges.
3. *Intrusion User* - alerts that indicate a system compromise where the hacker has only obtained user-level access.
4. *Intrusion Other* - alerts that indicate a system compromise where the privileges obtained by the hacker are not known.
5. *Privilege Escalation* - alerts that indicate that a user's privileges have been escalated to root or system level.
6. *Goal* - alerts that indicate that a hacker has likely accomplished a goal such as destroying or pilfering data.

The original INFERD guidance template was very dense, so the number of nodes were reduced by aggregating some nodes together to form the PAS shown in Figure 2.2, which is the same PAS that was used to implement TANDI. Each node in Figure 2.2 implicitly represents all alerts corresponding to its respective category. Note that this PAS can also easily be extended to multiple levels of internal computers. Since the PAS is used for prediction, it is necessary to filter any alerts that cannot logically occur. For example, if no external computers are running an SSH service, no alerts corresponding to SSH should be included in the external portion of the PAS. Therefore, the services of both internal and external machines must be known before creating the guidance template. Vulnerability scanners such as Nessus [21] are able to accurately identify running services automatically.

2.3.4 Information Graph

The information graph represents the relationships between network entities such as privileges, databases, proprietary files, etc. By analyzing the structure of modern machines and the typical attack sequences, an inherent three level hierarchy of network entities was observed: an attacker needs to (1) know the existence of the machine or subnet, (2) obtain appropriate access privilege, and (3) access the target information or files. Based on this observation and inspiration from the work by Phillips and Swiler [23] and that by Liu and Man [18], a template representing an information subgraph for a typical machine is developed and shown in Figure 2.3. This template is cloned with changes for each of the nodes in the logical topology graph. Together, the clones form the entire information graph.

Each of the entities in each clone is associated with the corresponding machine or subnet, as well as the attacks that can compromise the entity. Note that the edges in Figure 2.4 represent the associativity of the cloned information nodes to the nodes in the logical topology and those in the attack sequence graph. During the process of the cloning, the IDS alerts belonging to the associated attacks but cannot occur for the services running on the corresponding machines will be removed. This is done to increase the accuracy of the threat and impact assessment.

Figure 2.3: Information Graph Template defined for each computer or group of computers.

Note the “Incoming Computer” node and the “Outgoing Computer” node in the template. These two nodes are used to build the connection between network entities associated with different machines. For any directed edge $A \rightarrow B$ in the logical topology, the information nodes representing the existence of and the privileges at machine A will be connected to those of machine B (also with directed edges). These connections complete the automatic generation of the information graph.

Extending to Network Accounts and Shared Data

The automatic generation of the information discussed above only factors in user accounts and files of specific computers. While this may suffice for some networks, most networks these days contain shared files or databases that are remotely accessible from other computers on the network. These files can only be accessed by certain user accounts. It may be the case, though, that these network user accounts are only available on a certain set of computers.

Figure 2.5 incorporates network accounts and shared files to the automatic generation of the Information Graph. The network administrator is responsible for identifying the critical files or databases. Access to such files usually are organized by user groups. Therefore, aggregate nodes of user groups represent all of the user accounts contained within the user group. The user groups that are able to access these critical files are then linked to this data.

Figure 2.4: An example showing the attack sequence, the information graph, the logical topology, and how they are interconnected.

The user accounts are then linked to the cloned information graph of the computer that the user can log into. This example shows two critical files shared between three different user groups.

2.3.5 Situation Assessment Stage

Each node in the logical topology and the attack sequence graphs are to be evaluated in real time as cyber attacks occur. The *asserted* nodes reflect that the corresponding machines have been compromised or attack methods have been used. Consider this real-time situation assessment that provides indication of the attacker's capability (the types of attack methods he knows), and that of the attacker's opportunity (the machines or subnets he or she has compromised). By fusing the two, one may predict the intent of the attacker's next target, i.e., the nodes with high threat scores in the information graph. Following this intuition, TANDI performs situation and threat assessments on a per attack basis; that is, IDS alerts belonging to different attacks will be evaluated separately. Grouping alerts to different attacks is the task of an underlying alert-correlation engine.

Figure 2.5: Incorporating Network User Accounts and Shared Files into the Automatic Generation of the Information Graph

The fusion of the logical topology and the attack sequence information for predicting the threatened network entity may be represented with the use of the undirected edges connecting the nodes in the three model graphs, as shown in Figure 2.4. A network entity is connected to a machine or a subnet if the machine or the subnet contains or can access the entity. Similarly, a network entity is connected to an attack node if such type of attack can compromise the entity. The nodes in the logical topology and attack sequence can also be aggregate nodes that represent a group of computers (e.g., a subnet) or a set of alerts. These connections dictate how TANDI performs situation and threat assessments. For situation assessment, the information nodes that are associated with at least one of the asserted attack nodes and at least one of the asserted topology nodes are considered asserted or, equivalently, compromised. For example, considering Figure 2.4, if an IDS alert aggregated by the “Intrusion” node occurs on the “Web Server”, TANDI will determine that the “Web Server Privileges” is compromised by the attack because of the undirected edges between the models.

Mathematically, let A_i^* denote the i th of G alerts and M_i^* be the target(s) of A_i^* . Let A^* represent the set of alert signatures that have occurred in the current attack. M^* represents the set of target computers that have been attacked and I^* represents the entities that have been compromised. That is,

$$A^* = \bigcup_{i=1}^G A_i^* \quad (2.1)$$

$$M^* = \bigcup_{i=1}^G M_i^* \quad (2.2)$$

Let $A(X)$, $M(X)$, $I(X)$ denote the set of successor nodes to the set of nodes, X , from the guidance template, logical topology, and information graph, respectively. The set of compromised entities, I^* , can be represented by:

$$I^* = \bigcup_{i=1}^G \left(I(A_i^*) \cap I(M_i^*) \right) \quad (2.3)$$

Figure 2.6: Situation Assessment Feature Tree

Equation (2.3) is equivalent to using the feature tree in Figure 2.6 for each entity to determine if it is compromised. Note that this feature tree encompasses both (2.3) and the undirected edges between the sequence graphs and the information graph in Figure 2.4. It is possible that the undirected edges between the sequence graphs and information graph could be generated automatically. IDS websites, such as www.snort.org, provide a database of the alert name, the exploit or attack it represents, the consequences of a successful attack and the affected operating systems and services. A program could be developed to extract information from this database and automatically generate the undirected edges.

2.3.6 Threat Assessment Stage

Recall that the proposed threat assessment algorithm examines the successor nodes of those that have been asserted due to previous actions of a given attack. Denote the set of information nodes that have been compromised no later than the j th event occurring in attack A as

$I^*(e_j^A)$, and their successor nodes as $I(I^*(e_j^A))$. For completeness, $I^* = I^*(e_k^A)$ when e_k^A is the final event in attack A . Note that, in this definition, the set $I(I^*(e_j^A))$ will exclude those that are also in the set $I^*(e_j^A)$. Immediately after the j th event of attack A , a *threat score*, $0 \leq t_i(e_j^A) < 1$ will be determined for every node in the set $I(I^*(e_j^A))$. All nodes in $I^*(e_j^A)$ are assigned a threat score of one, indicating that they have been compromised. Nodes that are not in either of these two sets (i.e., nodes that are two or more hops from all nodes in $I^*(e_j^A)$) are assigned a threat score of zero. Note that, by restricting assessing threat scores for the successors of already compromised nodes, one can detect an abnormality, such as insider threats and coordinated attack.

Figure 2.7: Threat Assessment Feature Tree

To determine the threat scores for the nodes in $I(I^*(e_j^A))$ upon the occurrence of the j th event of attack A , a feature tree is evaluated for each of these nodes. Figure 2.7 shows the feature tree used for the “Web Privilege” node drawn in Figure 2.4. Note that the alerts shown in this feature tree are those associated with “Privilege Escalation” and “Intrusion” alerts that can happen on a web server. The structure of this feature tree determines the fusion rule, and is followed for all the nodes in the information graph. The feature trees for different information node differ in the alerts and the machines that are connected as leaf

nodes.

In the current implementation, a ‘Sum-of-SumSets’ operation is used for the *fusion rule*, which is the calculation used to determine the threat score. The Sum-of-SumSet operation uses only four pre-determined weights, which is a substantial reduction from the probability inferencing approach where one needs to determine the weights for all edges. The four weights: λ_{A^*} , λ_{M^*} , $\lambda_{A(A^*)}$, and $\lambda_{M(M^*)}$ correspond to the partial threat scores due to the already asserted attack nodes (A^*), the already asserted machines (M^*), the successor nodes of A^* , ($A(A^*)$), and the successors of M^* , ($M(M^*)$), respectively. The Sum-of-SumSet operation basically sums up the weights of the leaf nodes, i.e., the alerts and the machines, but only adds once for each type of weight. In other words, if two machines leaf nodes are both asserted, only one λ_{M^*} will be added to the overall threat score for the corresponding information node. This operation ensures that the threat score is less than one for all nodes that could be compromised next as long as $\lambda_{A^*} + \lambda_{M^*} + \lambda_{A(A^*)} + \lambda_{M(M^*)} < 1$.

A drawback of this approach is that the threat score does not distinguish network entities that are associated with more asserted attack nodes or machines from those associated with less. This will be addressed in future work regarding a better, if not optional, fusion rule. Intuition has that $\lambda_{A^*} > \lambda_{A(A^*)}$ and $\lambda_{M^*} > \lambda_{M(M^*)}$ should define the relationships between the weights. This intuition stems from the assumptions that a computer that has been attacked is likely to be attacked again and that an attack method that has been used is likely to be executed again. This intuition is tested in Section 3.4.3.

The prediction accuracy of TANDI relies on the correctness of the models and templates developed by SMEs, which is expected as any threat assessment algorithm may claim. As will be discussed in the next section, the current version of TANDI will provide indication for any abnormality due to possible modeling flaws.

2.4 Generalization of TANDI

While TANDI was developed for cyber security, its framework could easily be generalized for potential use in other domains. The information graph will represent entities within the domain that can be compromised by an attacker. Multiple sequence graphs can be defined to capture who, where, when, why, and how, or potential combinations thereof. The introduction of new sequence graphs will only introduce a constant number of weights per sequence graph added.

2.5 Basic Implementation Details

TANDI was implemented using Java 1.5. It was developed strictly for simulation, so the logical topologies and services were generated manually and are not integrated with any automatic network scanners. The graph classes and visualization were implemented using the Prefuse API alpha 04.01.2005 [10]. Other visual elements were developed using the Java Swing package.

TANDI was built to be highly configurable for specific networks. The logical topology, PAS, information graph, alert signatures, underlying feature trees, and attacks are all provided and managed through XML files.

Two programs were created using the TANDI core. *RunCyberTA* allows the user to visually step through the attack to see how the threat scores change as the attack progresses. A screenshot from this program is shown in Figure 2.8. This simulates how a network administrator would view an attack as it progresses in real time. It also outputs various metrics that will be discussed in the next chapter. *RunCyberSim* is a command-line simulation package that allows multiple attacks to be simulated over multiple topologies. It outputs a comma-delimited file of the results of the simulations for analysis in a spreadsheet or database program.

The source code and documentation of TANDI and associated programs can be found on the thesis CD.

Figure 2.8: Screenshot of RunCyberTA in the Middle of a Cyber Attack. A larger radius indicates a higher threat.

2.5.1 System Organization

To aid in the organization of the source code, several Java packages were developed for TANDI. This section discusses the packages developed and some of the features provided by each. The javadoc provided on the thesis CD can be referenced for more technical documentation regarding the actual method calls and class specifications.

ta

The ta package provides the core functionality of TANDI, including the RunCyberTA and RunCyberSim main classes. This package provides the data structures for the graphs and incoming attacks. The ThreatAssessor class is the processing class for TANDI and is used to create an instance of TANDI. The PAS and Logical Topology are referred to as *sequence graphs*. ThreatAssessor was defined such that the number of sequence graphs is arbitrary. This allows the user to define more or less than the two sequence graphs used in TANDI. As discussed in 2.4 these sequence graphs may capture *why* or *who*.

AssertableLogicalTreeNodeOrganizer is used to manage the possible alerts or computer nodes that could be asserted by incoming attacks. Due to the definition of a tree node in Prefuse, a tree node can only have one parent. Therefore, if a node is defined in more than one tree (such as the Information Graph and PAS), it must be cloned. AssertableLogicalTreeNodeOrganizer manages these clones so that all clones are updated properly if another changes.

This package also contains the classes used to generate the ThreatAssessor objects containing the PAS and logical topologies used in simulation. This code generates the ThreatAssessor object and writes the Information Graph template, Logical Topology, and PAS to XML files. Finally, it also includes the Attack Generator used to generate alerts for simulations.

ta.generator

The ta.generator package provides the automatic generation of an information graph from a template information graph and the logical topology. It currently does not support network resources as discussed in Section 2.3.4.

It also provides functionality to filter any alerts from the PAS that could not logically occur based on the services running in the different segments on either side of a firewall.

ta.io

The ta.io packages provides XML parsers to read in XML files to create a ThreatAssessor object and incoming attacks. It also includes an XML writer that allows a ThreatAssessor object to be written to XML format. There is a subpackage ta.io.gt that is used to parse an INFERD Guidance Template and translate into the format used for TANDI.

ta.gui

The ta.gui package provides the graphical user interface for RunCyberTA. Java swing was used along with the graphical objects provided by Prefuse for graph visualization.

ta.logicalltree

The ta.logicalltree package provides extensions to Prefuse for the data structures needed for the situation assessment and threat assessment feature trees. An AssertableLogicalTreeNode is a leaf node that can be asserted by incoming alerts. A LogicalTreeNode implements the aggregation operators such as *and*, *or*, *sum*, and *sum set* used in TANDI's feature trees. Since the situation and threat assessment feature trees differ only in the operations, the LogicalTreeNode class allows for multiple modes of operation, so the situation assessment and threat assessment feature trees are combined into only one feature tree. Situation and threat assessment functionality can be switched by simply changing the mode of evaluation of the tree. This feature was included to reduce the memory required for the feature trees.

ta.stats

This class provides statistics for the evaluation of TANDI. Each developed statistic implements the `TAAttackStatistic` interface and analyzes the collection of threat scores TANDI calculated for each entity. The statistics provided by TANDI are the metrics discussed in the next chapter.

Chapter 3

Simulation and Results

To validate and examine the performance of TANDI, it was simulated over two networks. Due to liability, loss of reputation, and competition issues, cyber attack data is not currently available in the public domain [30]. Simulations were therefore limited to attacks generated via a simple attack generator to mimic the outputs of a cyber attack detector. Multiple metrics were developed to analyze the performance of TANDI. A small network was first simulated. Then, a relatively large network was simulated to demonstrate the scalability of TANDI.

3.1 Simulated Network Topologies

3.1.1 Small Network Topology

TANDI was first simulated using three variants of the network topology shown in Figure 3.1. This is a fictitious network representing a small to medium business that will have different subnets. The three variants of this topology represent cases due to different routing rules plausible for the network. This allows the examination of the cases of external servers and internal subnets being able (or not being able) to communicate with each other. The following three variants of the topology were set up and used for simulation:

- **Network 1** assumes that all four subnets are completely segmented from each other. The only connection the four subnets have is that they all have one server connecting

to the Internet.

- **Network 2** is the same as the first topology, except that all four external servers are fully connected to each other. This setup is similar to network 1, other than external computers should be more highly threatened as alerts are seen on the external computers.
- **Network 3** is the same as the second topology, except that all internal computers in all subnets are fully connected.

Figure 3.1: The network topology used for simulations. Blocks with a dotted background are external computers while all others are internal. The corresponding IP address(es) and services running are indicated.

This was the main network for simulations. Remember that the edges are defined based on routing rules and access-lists. So the only difference between the three topologies is in the configurations of routers and switches. Each topology was simulated over different weight assignments for normal attacks, attacks with mis-detections and insider threats.

3.1.2 Large Network Topology

To illustrate the scalability of TANDI, it was simulated over a larger network. Like the smaller network, this is also a fictitious network that was meant to simulate the network of a medium to large business. It consisted of multiple subnets and many redundant servers to improve the number of users that can access the business's information. This network

will be referred to as “Network 4”. This network was simulated over different weight assignments for normal attacks.

Figure 3.2: A Larger Computer Network to Simulate

The logical topology for the large network is shown in Figure 3.2. Notice that the logical topology complexity does not vary significantly from the smaller topology (Figure 3.1). In fact, the large topology actually has *fewer* nodes in the logical topology. This is because the increased number of servers in a large typical large network form server farms, consisting of many identically configured servers. The server farms are used to allow concurrent access to data stored on the servers by multiple users. Since the servers have the same services running, they are all vulnerable to the same attacks. They can therefore be grouped into one single node in the logical topology. This network’s “Server Farm” node consists of 10 file servers. Similar to the idea of server farms, the network size increase is due to similarly configured workstations, and not necessarily a significantly larger number of subnets. There are three main subnets to this network: the Operations, Marketing, and Accounting and are their own subnet. Each subnet consists of 100 workstations and has its own server to support the needs of each group individually. In this network, vulnerability updates are handled every evening, so each workstation will have the same patches – so the workstations can be grouped into one node. It should also be noted that the accounting server is only accessible by the accounting workstations and some other servers. The reason that

the Accounting Server is not accessible from, say, the marketing workstations, is because access lists in the routers prevent this possibility of this traffic flow. So this illustrates that the edges in the Logical Topology are largely dictated by routing rules within the network itself.

3.2 Attack Simulator

A simple cyber attack simulator was developed to generate the simulation data. These generated attacks mimic the inputs that would be provided by a lower-level cyber attack detector or IDS alert correlator. Each attack is a set of IDS alerts and system log messages that represent an actual cyber attack. These generated attacks represented compromises of a single machine, multi-computer attacks that posed a large threat to the network, insider threats, and mis-correlations by the detector.

The actual generation of the attacks involved the following steps. An entity was selected to be compromised that would be realistic for the type of attack generated. An IDS alert or system log message that corresponded to the compromise of that entity was randomly chosen from a pool of probable alerts. This process is then repeated for each step in the attack. Careful consideration was taken to the order of these alerts, so they represented a realistic output from a cyber attack detector.

The small network was simulated over five different sets of attacks. It should be noted that the low number of cyber attacks do not make the results statistically significant, but do allow for the identification of basic trends that would be worthwhile to analyze over a larger set of attacks. Due to the limited availability of cyber attack data, generation of a statistically significant number of attacks was outside of the scope of this work.

The five sets used in simulating the small network are summarized as follows:

- *Set 0 (15 Attacks)* - A normal set of attacks corresponding to cyber attacks originating from the Internet. Each of these attacks first compromises at least one external computer and may penetrate into one or more internal computers. Each attack had

between 6 and 21 events.

- *Set 1 (10 Attacks)* - A set of attacks that contain abnormalities that correspond to false positives from an IDS and mis-correlations by a cyber attack detector. They could also represent errors in the information graph, logical topology or PAS. Each attack had between 4 and 12 events.
- *Set 2 (5 Attacks)* - A set of insider attacks. These attacks originate internally, so external computers are not compromised. These attacks are a more extreme version of Set 1. Each attack had between 3 and 6 events.
- *Set 3 (2 Attacks)* - One coordinated attack that was fragmented into two separate attacks. These two attacks were then combined to test the affect of combining potential coordinated attacks.
- *Set 4 (4 Attacks)* - One coordinated attack that was fragmented into four separate attacks.

The large network was only simulated over 10 normal attacks. Since the network is different, the actual attacks differ from those in “Set 0”, but the type of attacks are the same.

3.3 Statistics for Evaluating TANDI

This work defines a set of specific metrics that are used to analyze a threat assessment framework, such as TANDI. These metrics indicate the performance of the framework over a set of simulation data. Therefore, these metrics allow the comparisons between two or more threat assessment frameworks. Previously, there were not common metrics to analyze threat assessment frameworks, so there was no standard way to show that a framework is “better” than another framework. Since threat assessment involves the prediction of an attacker’s next action, and data is usually limited, it is very difficult to truly assess its accuracy.

Even though the focus of this thesis is cyber attacks, these metrics are defined independent of domain. It should be noted, however, that in some domains (including our current knowledge of the cyber domain), not all metrics defined here may make sense or even be applicable.

There is not one specific metric that can be used to encompass overall performance. Each metric provides its own analysis of the framework and incoming attacks. Different situations may deem one or more of these metrics to become unreliable, so different metrics are provided that attempt to cover the other's downfalls.

These metrics will then be collected over different sets of data for attacks on the two different networks.

The following lists six metrics that were developed to provide an indication of the performance of a threat assessment algorithm.

1. Average Normalized Compromising Score
2. Average Percentile of Compromising Score
3. Top Threat Score Prediction Percent
4. Percent False Positives
5. Percent False Negatives
6. Percent Assessee Reduction

The following two metrics are presented for each individual attack for assessment by the analyst.

1. Percent Threatened
2. Percent Abnormal

3.3.1 Assumptions/Definitions

The framework assesses the threat of specific entities within a domain and assigns a **threat score**, $0 \leq t_i(e_j^A) \leq \max_possible_threat_score$ where $0 < \max_possible_threat_score \leq \infty$, to the i th entity upon the occurrence of the j th event of an attack A , where the threat scores satisfy the following adjectives, which qualitatively describe the threat level of an entity:

- **Compromised:** $t_i(e_j^A) = \max_possible_threat_score$
- **Threatened:** $0 < t_i(e_j^A) < \max_possible_threat_score$
- **Unthreatened:** $t_i(e_j^A) = 0$

The $\max_possible_threat_score$ is simply the maximum threat score a framework can assign to an entity, which corresponds to the value indicating that an entity is *compromised*. If $\max_possible_threat_score$ is finite it is recommended that the threat scores be normalized.

The above constraints should be general enough to encompass a large range of threat assessment frameworks. Note that it is *not* required that any sequence of entities be defined (like the Information Graph in TANDI), nor does the method of assigning the threat score matter.

3.3.2 Metric Definitions

3.3.3 Average Normalized Compromising Score

An ideal threat assessment algorithm with perfect models should generate threat scores that accurately depict the sequence of attack events. In other words, the compromised entity should have the highest threat score with respect to the other threatened entities one step before it is compromised. Based on this intuition, a *normalized compromising score*, $\hat{t}_i^*(A)$, is defined as the normalized threat score for entity i one event prior to it being compromised

by attack A . That is,

$$t_i^*(A) = \left\{ \frac{t_i(e_j^A)}{\max_{k \in C_j^A} t_k(e_j^A)} \mid t_i(e_{j+1}^A) = 1, t_i(e_j^A) < 1 \right\}, \quad (3.1)$$

where C_j^A indicates the set of entities that have not been compromised prior to and including event j of attack A . The average of the normalized threat scores of entities compromised by an attack A is the **average normalized compromising score**, $\bar{t}^*(A)$. For a “perfect” assessment, $\bar{t}^*(A)$ would be 1, indicating that each compromised entity has the highest threat score just before it is compromised. Averaging $\bar{t}^*(A)$ for different attacks, denoted as $\bar{t}^*$, could then be indicative to the accuracy of a threat assessment algorithm. Note that, however, this metric can be misleading if many entities share the same highest threat score. For example, a worthless algorithm that assigns the same threat score to all network entities in every step will have $\bar{t}^* = 1$.

Average Percentile Rank

The *average percentile rank* is the average percentile rank of all entities one event prior to being compromised. Note that this includes both threatened and unthreatened entities in the percentile calculation. The idea here is to evaluate whether the compromised entities indeed have a high threat score as compared to all uncompromised entities.

Top Threat Score Prediction Frequency

Top Threat Score Prediction Percent indicates the percentage of compromised nodes such that its average normalized compromising score was one, e.g., $\hat{t}_i^*(A) = \alpha, 0 < \alpha \leq 1$. This indicates how often the next compromised entity was accurately predicted. The parameter α is a threshold that indicates the minimum normalized threat score that is high enough to be an accurate prediction. Like average normalized compromising score, this metric can become unreliable when many there are many entities with normalized threat scores higher than α .

Percent False Positives and Percent False Negatives

Let β be a minimum threshold indicating a threat score high enough for an analyst to warrant looking at. For an attack, A , that has K events, a false positive shall occur when $\max_{k \in K} t_i(e_A^k) \geq \beta$ and $\hat{t}_i^*(A)$ is undefined, meaning that the entity was never compromised. In other words, a false positive occurs when an entity has a high threat score and is never actually compromised by the attack.

A false negative shall occur when $\exists \hat{t}_i^*(A) < \beta$. In other words, a false negative occurs when an entity with a low threat score that the analyst likely would not have been immediately drawn to becomes compromised.

Percent false negatives and percent false positives can then be formally defined as follows. Let $N_A(X)$ be the number of entities in attack A that satisfy the condition X . Let C_A represent the set of compromised entities due to attack A . Percent false negatives can then be defined in (3.2) and percent false positives can then be defined in (3.3).

$$\text{Percent_False_Negatives} = \frac{N_A(\hat{t}_i^*(A) < \beta)}{|C_A|} \quad (3.2)$$

$$\text{Percent_False_Positives} = \frac{N_A((\max_{k \in K} t_i(e_A^k) \geq \beta) \text{ and } (i \notin C_A))}{|C_A|} \quad (3.3)$$

Percent Assessee Reduction

Percent assessee reduction is a measure of the percentage of nodes whose threat scores are below a threshold, β . It captures the percentage of nodes that the analyst should not have to look at since they have a lower threat.

3.3.4 Percent Threatened

Percent threatened indicates the percentage of non-compromised entities that have a threat score greater than or equal to a threshold, β , due to a specific attack. This is *not* a threat assessment accuracy metric, but rather a metric that would be presented to the analyst. With

respect the overall threat of an attack, a high percent threatened metric indicates that the corresponding attacker has the opportunity to compromise many entities, and the analyst may want to focus on treating that attack. With respect to framework performance, a high percent threatened metric may also indicate that the current prediction is too conservative, meaning that the framework is predicting too many entities to have a high probability of being attacked next.

Percent Abnormal

An abnormal entity, or abnormality, is an entity whose compromising score is zero, i.e., $\hat{t}_i^*(A) = 0$. *Percent abnormal* is the percentage of abnormalities over the set of compromised entities. Like *percent threatened*, this is *not* a threat assessment accuracy metric, but rather a metric that would be presented to the analyst.

In regards to TANDI and cyber attacks, an abnormality will occur in the any of following situations:

- Sensor readings: the abnormality could be due to a false positive or undetected event.
- Event correlator: the correlator responsible for level 2 fusion could have falsely correlated (or uncorrelated) events to an attack. A mis-correlation at this level could be due to a stealthy coordinated attack by multiple attackers.
- Threat assessment model: incomplete or inaccurate models used by the threat assessment algorithm.
- Insider threat: Any threat assessment algorithm is susceptible to not detecting insider threats if assumes that attacks will originate from outside of the organizational network.

Therefore, percent abnormal can be used to identify potential insider threats or flaws in the framework.

3.4 Results and Discussion

3.4.1 Example Attack and Threat Scores

Figure 3.3 illustrates an example simulated attack from Set 0 and the evaluated threat scores at each stage of the attack for a subset of network entities in the information graph. Network 1 of the small network is used for this example, and the weights used for threat assessment are given in the caption of the figure. The bold number for each entity indicates the threat score one step before the entity was compromised. Note that there was no abnormality in this attack, and TANDI identified the next potential target with the highest threat score, except for ‘SSH Server 35.2-Exist’ and ‘SSH Server 35.2-User’ in Step 2. Note that the average normalized compromising score is 0.74. This means that the average entity had a normalized threat score of 0.74 prior to being compromised. If this value were much lower, this likely indicates that there are deficiencies in the Logical Topology, PAS, Information Graph, or weight assignment.

Sequence Number	0		1		2		3		4	
Attack	Initial		ICMP (Ping)		imap exploit partial body overflow attempt		SSH Password Accepted		DIRECTORY TRAVERSAL	
Computer			12.30.35.1		12.30.35.1		12.30.35.2		12.30.35.2	
	Act	Norm	Act	Norm	Act	Norm	Act	Norm	Act	Norm
SSH Server 35.1-Exist	0.10	1.00	X	X						
SSH Server 35.1-Root	0.30	0.00	0.30	1.00	X	X				
SSH Server 35.1-UserFiles	0.30	0.00	0.00	0.00	0.35	1.00	0.35	1.00	0.35	1.00
SSH Server 35.2-Exist	0.00	0.00	0.10	0.33	0.10	0.29	X	X		
SSH Server 35.2-User	0.00	0.00	0.10	0.33	0.15	0.43	X	X		
SSH Server 35.2-UserFiles	0.00	0.00	0.00	0.00	0.00	0.00	0.36	1.00	X	X

Average Normalized Compromised Score: 0.74

Figure 3.3: Example threat scores during the progression of a cyber attack with the weights of $\lambda_{A^*} = 0.2$, $\lambda_{A(A^*)} = 0.05$, $\lambda_{M^*} = 0.3$, $\lambda_{M(M^*)} = 0.1$. Both the actual and normalized threat scores are shown for each entity. Nodes compromised are indicated with an ‘X’.

3.4.2 The Sensitivity of Weights

TANDI requires four parameters. To determine the values to assign to these parameters, a sensitivity analysis of the weights was first performed. Since the normalized values of the

threat scores are used in the metrics, the initial hypothesis was that only the relative values of the weight parameters significantly affect the results. The absolute values of the weight parameters *will* affect the results, but not significantly enough to warrant these values to affect the performance of TANDI.

To test this hypothesis, all four topologies were simulated using their respective Set 0 data, which contained normal attacks. Each attack was simulated over all weight combinations possible by varying each parameter by 0.1. These weight combinations were then separated into two sets. Set A contained all weight combinations such that $W(A^*) > W(A(A^*)) > 0$ and $W(M^*) > W(M(M^*)) > 0$, while set B contained all weight combinations such that $W(A(A^*)) > W(A^*) > 0$ and $W(M(M^*)) > W(M^*) > 0$. The weight combinations for each set is shown in Figure 3.4.

Topologies 1-3 did not exhibit much difference from each other. Topology 4 did yield different values, but the trends exhibited by the sensitivity analysis were the same as Topologies 1-3. Therefore, Topology 2 was arbitrarily chosen for this analysis. Figures 3.5-3.7 show the sensitivity graphs for both sets over topology 2 for average normalized compromising score, average percent false positives, and average percent false negatives. The X-axis of each of these graphs contain all weight combinations relevant for each of set A and set B. As the weight configurations progress across the X-axis, the weights for the PAS increase, while the weights for the Logical Topology decrease. Due to space limitation for the graph, the actual weight configurations for each point are not shown. Figure 3.5 shows small changes in the average normalized compromising score within each set. However, set B was consistently higher. Possible reasons for this result are discussed in the next section. This illustrates the initial conjecture regarding relative weight assignment being more important. Figure 3.6 shows virtually no change in the false positives across the different weight assignments. Figure 3.7 shows more interesting results. Both sets exhibit a general upward trend of the false negatives as the PAS weights increase and the Logical Topology weights decrease. This shows that there are generally a higher percentage of false negatives when the PAS is weighted higher than the Logical Topology.

Set A				Set B			
$W(A^*)$	$W(A(A^*))$	$W(M^*)$	$W(M(M^*))$	$W(A^*)$	$W(A(A^*))$	$W(M^*)$	$W(M(M^*))$
0.2	0.1	0.2	0.1	0.1	0.2	0.1	0.2
0.2	0.1	0.3	0.1	0.1	0.2	0.1	0.3
0.2	0.1	0.3	0.2	0.1	0.2	0.1	0.4
0.2	0.1	0.4	0.1	0.1	0.2	0.1	0.5
0.2	0.1	0.4	0.2	0.1	0.2	0.1	0.6
0.2	0.1	0.4	0.3	0.1	0.2	0.2	0.3
0.2	0.1	0.5	0.1	0.1	0.2	0.2	0.4
0.2	0.1	0.5	0.2	0.1	0.2	0.2	0.5
0.2	0.1	0.6	0.1	0.1	0.2	0.3	0.4
0.3	0.1	0.2	0.1	0.1	0.3	0.1	0.2
0.3	0.1	0.3	0.1	0.1	0.3	0.1	0.3
0.3	0.1	0.3	0.2	0.1	0.3	0.1	0.4
0.3	0.1	0.4	0.1	0.1	0.3	0.1	0.5
0.3	0.1	0.4	0.2	0.1	0.3	0.2	0.3
0.3	0.1	0.5	0.1	0.1	0.3	0.2	0.4
0.3	0.2	0.2	0.1	0.1	0.4	0.1	0.2
0.3	0.2	0.3	0.1	0.1	0.4	0.1	0.3
0.3	0.2	0.3	0.2	0.1	0.4	0.1	0.4
0.3	0.2	0.4	0.1	0.1	0.4	0.2	0.3
0.4	0.1	0.2	0.1	0.1	0.5	0.1	0.2
0.4	0.1	0.3	0.1	0.1	0.5	0.1	0.3
0.4	0.1	0.3	0.2	0.1	0.6	0.1	0.2
0.4	0.1	0.4	0.1	0.2	0.3	0.1	0.2
0.4	0.2	0.2	0.1	0.2	0.3	0.1	0.3
0.4	0.2	0.3	0.1	0.2	0.3	0.1	0.4
0.4	0.3	0.2	0.1	0.2	0.3	0.2	0.3
0.5	0.1	0.2	0.1	0.2	0.4	0.1	0.2
0.5	0.1	0.3	0.1	0.2	0.4	0.1	0.3
0.5	0.2	0.2	0.1	0.2	0.5	0.1	0.2
0.6	0.1	0.2	0.1	0.3	0.4	0.1	0.2

Figure 3.4: Weight Combinations

Figure 3.5: Average Normalized Compromising Score Sensitivity (Topology 2)

Figure 3.6: Average Percent False Positives Sensitivity $\beta = 1$ (Topology 2)

Figure 3.7: Average Percent False Negatives Sensitivity $\beta = 0.6$ (Topology 2)

3.4.3 The Effect of Weights

Now that the sensitivity of the weights was analyzed, the effect of varying the relative values of the weights was analyzed. Since the threat scores are normalized in analysis, the absolute values of the weights are not as important as the relative values of the weights. This was illustrated in the previous section. Therefore, the magnitude of the weights used in this analysis were arbitrary,

Both the large and small networks were simulated using their respective Set 0 data that contained normal attacks originating from the Internet and contained no abnormalities. Using the weight assignment intuition from Section 2.3.6, the following two hypotheses were tested:

- **Hypothesis 1:** An attack method that has already attempted is more likely to occur again than a new attack. Therefore, the weights should be set up such that $\lambda_{A^*} > \lambda_{A(A^*)}$.
- **Hypothesis 2:** A computer that has been attacked is more likely to be attacked again than a different computer. Therefore, the weights should be set up such that $\lambda_{M^*} > \lambda_{M(M^*)}$.

Figure 3.8: Average Normalized Compromising Score (Set 0)

Figures 3.8 - 3.13 illustrates the performance of the given metrics for the four simulated topologies. The average percentile rank did not exhibit any noticeable trends in Figure 3.9.

Figure 3.9: Average Percentile Rank (Set 0)

Figure 3.10: Percent False Positives (Set 0) $\beta = 1$

Figure 3.11: Percent False Negatives (Set 0) $\beta = 1$

Figure 3.12: Percent False Negatives (Set 0) $\beta = 0.6$

Figure 3.13: Average Percent Assessee Reduction (Set 0) $\beta = 1$

The far right weight assignment (where successor nodes are given a larger weight) had the highest average normalized compromising score across all topologies in Figure 3.8. The average normalized compromising across all topologies and sets score ranged from 0.69 to 0.89.

However, the far right weight assignment did yield the largest percentage of false positives but the fewest percentage of false negatives (Figures 3.10 and 3.11). For this analysis, a very strict definition of false negatives was used, meaning that any entity compromised with a normalized threat score of less than 1 were considered to be false negatives (i.e., $\beta = 1$). Since the average normalized compromising threat score ranged between 0.69 and 0.89, it indicates that the average entity was compromised with a score less than one, which explains the high number of false negatives. When this threshold was relaxed to 0.6 the percentage of false negatives improved as shown in Figure 3.12. The far right weight assignment did also yield a significantly larger number of false positives. This is likely due to the large number of alerts in the PAS and number of computers in the Logical Topology. This means that there are a significantly larger number of adjacent alerts than asserted alerts and adjacent computers than attacked computers. Therefore, a larger number of entities were highly threatened, but the same number of entities were compromised.

In threat prediction, the idea of false positives and negatives seem to be valid evaluation metrics, since they indicate the entities that were not assessed properly. However, these metrics may not be very applicable to the cyber domain. When an external hacker attacks a computer network, he often is unaware of where the important information is. Therefore, the hacker may take some unexpected steps in the execution of the attack. This would increase the number of false negatives. Likewise, the hacker may be targeting only one piece of critical information, so once the hacker compromises that information, nothing else would be compromised. This would increase the percentage of false positives. A forensic analysis of the attack would allow the evaluation of whether false positives or false negatives are actually applicable. Since TANDI is the first cyber attack threat assessment framework, it is unclear what is considered to be an acceptable or unacceptable percentage

of false positives and negatives. As the field of cyber attack threat assessment matures and more frameworks are developed, a consensus could eventually be reached on this issue.

Figure 3.13 shows that the average percent assessee reduction was also noticeably lower for the far right weight assignment, however, this is likely due to the high number of false positives. These results suggest that there could be a potential tradeoff between false positives and average normalized compromising score. However, there are not enough attacks to make these results statistically significant.

The hypotheses seem to be refuted based on these results: higher $\lambda_{A(A^*)}$ and $\lambda_{M(M^*)}$ actually exhibit better performance. An in-depth analysis of the attacks provides interesting insights towards network dependent threat assessment. Most of the attacks in this test set focus on subnets 34 and 37, which contain servers with different services running. Since different services are running on different computers, a previously executed exploit could not be used on the successor machine. Therefore, a high value of λ_{A^*} does not help to predict the next attack event. Similarly, because the test network has each computer run a single service, the number of successive attack events for the same computer is not as many as originally expected. If, however, the threat assessment is performed on the scale of subnets, i.e., network IDSs are used instead of host-based IDSs, one should expect a relatively long sequence of attack events appearing on the same subnet - the original hypothesis 2 should still hold. This analysis leads to the following two revised hypotheses:

- **Revised Hypothesis 1:** In the case where networks contain many similar computers, a weight assignment of $\lambda_{A^*} > \lambda_{A(A^*)}$ should outperform $\lambda_{A^*} < \lambda_{A(A^*)}$. In the case where networks have a wide array of services running across different computers, the opposite weight assignment should yield a better threat prediction.
- **Revised Hypothesis 2:** In the case where mainly network-based IDSs are used to detect attacks, $\lambda_{M^*} > \lambda_{M(M^*)}$ should outperform $\lambda_{M^*} < \lambda_{M(M^*)}$. In the case where host-based IDSs and extensive system logs are used to detect attacks, the opposite weight assignment should yield a better threat prediction.

3.4.4 Abnormalities/Insider Threats

Two more sets of data were generated to test how well TANDI handles abnormalities. These two attack sets were compared against the baseline Set 0 with no abnormality - the one used for the analysis in Section 3.4.3. Similar to those in Set 0, attacks from Set 1 always started by an Internet computer attacking an external machine, but a network entity that does not belongs to $I(I^*(e_j^A))$ may be compromised next in the middle of the attack. Set 2 takes one step further and contains attacks that start at an internal machine, representing insider threats. The weights used here are $\{\lambda_{A^*}, \lambda_{A(A^*)}, \lambda_{M^*}, \lambda_{M(M^*)}\} = \{0.3, 0.1, 0.4, 0.15\}$.

Figure 3.14: Percent Abnormal over Sets 1-3

Figure 3.15: Average of the normalized compromising scores for the set of normal attacks (set 0), a set of attacks with abnormalities (set 1), and a set of insider threats (set 2).

Figure 3.16: Average Percentile over Sets 1-3

Figure 3.17: Percent False Positives over Sets 1-3

Figures 3.14 - 3.18 show the metrics relevant to the analysis of these attacks. It can easily be seen in Figure 3.14 that set 0 contains no abnormalities, while the percent abnormalities increased for sets 1 and 2. Again, average percentile yielded no interesting results in Figure 3.16, therefore, this may indicate that average percentile may not be a worthwhile metric to use in analysis. While the false positives were very low in Figure 3.17, the false negatives were very high across all cases in 3.18. This is likely due to the increased number of abnormalities where unthreatened entities were being compromised without any previous indication of threat. The average normalized compromising score shows that as the abnormalities increase, this metric drastically decreases, thus showing that TANDI performed poorly in the occurrence of abnormalities. These results were expected because

Figure 3.18: Percent False Negatives over Sets 1-3 $\beta = 1$

TANDI assumes that attacks will follow the possible courses of action defined by the logical topology. The abnormalities happen when the hackers take unexpected actions or if there are IDS failures. The current implementation of TANDI will alarm the network analyst for the abnormalities, whenever a network entity is recorded a normalized compromising score of 0. This indicator will allow the network analyst to examine the potential source of insider threats, to identify coordinated attacks, or to revise the logical topology model and the IDS setup.

3.4.5 Difference in Topology Results

Interestingly, the graphs presented in the previous two sections do not show any noticeable trend between the three topologies of the small network. This could be a statistical anomaly, but this suggested that the actual topology of the network is not significant in the accuracy analysis. However, Figure 3.19 shows that topology 3 showed a significantly larger number of threatened nodes. A larger set of attacks will need to be generated to determine the actual effect of network connectivity on the prediction ability of TANDI.

3.4.6 Coordinated Attacks

A single cyber attack can often originate from multiple sources on the Internet. This can be the product of multiple hackers coordinated with each other, or several *zombie clients* being

Figure 3.19: Average Percent Threatened on Small Network (Set 0)

controlled by one hacker. A zombie client is simply a computer that has been compromised by a hacker that is used to execute another attack. This disguises the single attack from being detected.

Such coordinated attacks will often be fragmented over multiple detected attacks. These fragments, when evaluated by TANDI, would yield abnormalities, thus giving a poor detection performance for that attack. However, if the fragmented attacks were combined into a single attack, some or all of the abnormalities would not exist, thus increasing the prediction performance.

Figure 3.20: Average Normalized Compromising Score of a Two Zombie Coordinated Attack

Two such coordinated attacks were simulated in TANDI. Figure 3.20 illustrates the average normalized compromising score of three attacks. In this example, the hacker used

two zombie machines to execute the attack. The first zombie client was used to footprint the network and transfer the password files after the external computers were actually compromised by the second zombie client. Attacks 1 and 2 represent one coordinated attack that was detected as two separate attacks. Attack 3 is the combination of attacks 1 and 2. Note how the average percentile rank increased when the fragmented attacks were combined. This is because there were no abnormalities in the combined attack.

Figure 3.21: Average Normalized Compromising Score of a Four Zombie Coordinated Attack

Figure 3.21 shows another coordinated attack that was fragmented over four different attacks. Attack 4 is the combination of all of these fragmented attacks. Notice how this combination does not improve the average normalized compromising score of *all* of the attacks. However, it is significantly higher than the password transfer (PW Transfer) attack. The first two attacks are simply footprinting of external computers, which is an expected beginning to an attack. Hence, this was why attacks 0 and 1 had a high percentile rank. Attack 2 was a compromise of an external computer with no associated reconnaissance since it was performed by other zombie clients. Attack 3 is the compromise of an internal computer, which is why the attack detection was so poor. The combination of the four attacks eliminated the abnormalities and was able to better represent the overall coordinated attack.

Future work will be devoted to a more in-depth analysis on how attacks can be combined to try to detect coordinated attacks in real-time.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

The main contribution of this work was the development of a novel cyber attack threat assessment framework named Threat Assessment of Network Data and Information (TANDI). TANDI is actually a combination of level 2 (situation assessment) and level 3 (threat assessment) data fusion. Besides being the first of its kind, the novelty of TANDI lies in the separation of how (Potential Attack Sequence), where (Logical Topology), and what (Information Graph) the hacker can attack. These three directed graphs can be developed independent of each other and are fused together to determine critical network entities that have been compromised, which is situation assessment. Based on this result, TANDI will then try to predict the next critical entity that will be compromised by combining weights assigned to nodes in each of the three graphs. These weights represent what has already happened, and what is likely to happen next. They are assigned by the analyst based on the criteria determined in the simulation discussion. The evaluation of a feature tree underlying each node in the information graph yields a threat score, which corresponds to the likelihood that the entity represented in the information graph will be next compromised. This prediction is the threat assessment. While only one feature tree structure was used in this work, the structure could easily be changed to provide a different (and potentially more accurate) evaluation of the weights. The separation of how, where, and what and the feature tree make TANDI a flexible and scalable framework. Results also indicate that TANDI can

be potentially used to detected coordinated attacks.

The second contribution involves the evaluation of the performance of a threat assessment framework. While there exist threat assessment frameworks in domains other than computer networks, there is no common way to evaluate frameworks against each other. This work provided several common metrics, such as *average compromised percentile rank*, that can be used to compare frameworks over the same set of data. Other metrics, such as *percent threatened*, can also be used to evaluate the overall threat of an incoming attack. The use of the metrics was illustrated by evaluating TANDI over different network topologies and cyber attacks with synthetic cyber attack data.

4.2 Future Work

In this work, a novel framework for assessing the threat of incoming cyber attacks was developed. Since this is the first framework for such an application, many extensions to this work are worth pursuing. This section will discuss some of the possible extensions, the underlying motivation for the extensions, and a preliminary analysis of how they may be implemented.

4.2.1 More Simulation Data and More Diverse Networks

One limitation of the simulation results were that they were simulated over artificially created cyber data on fictional networks. Simulation of TANDI over more diverse, realistic networks with a larger set of cyber attack data would help to validate the results presented in this work. A more extensive simulation framework would allow for a larger number of attacks to be generated, which can help to validate some of the findings presented here.

To generate true data, a small network could be created and exposed to the Internet. Attack data could be captured and input to an alert correlator which then relays the detected attacks to TANDI. One issue with this technique may be that the network contains no important information for a hacker, so the hacker may decide to not bother compromising

the other computers, which leads to incomplete attack data.

4.2.2 Integration with Existing Cyber Attack Detector

The framework of TANDI was developed in Java, but was only programmed for simulation. More research needs to be conducted on how to integrate TANDI with existing vulnerability scanners to improve the automation of logical topology and information graph creation. TANDI then needs to be integrated with an existing cyber attack detection system, such as INFERD [28], to test its performance with actual attacks detected by such a system.

4.2.3 Inclusion of Sensor Location

The placement of IDS's can also be used to analyze the true threat of an incoming attack. For example, supposed that a network contains two network IDS's. One IDS is located external to the firewall, and the other is located behind the firewall. If an alert is reported by the external IDS with a destination of an internal computer, but not reported by the internal IDS, it is highly likely that the firewall filtered out the potentially malicious traffic, thus no compromise could have taken place. Since TANDI currently ignores the location of the sensors, the inclusion of sensor location could help to improve the accuracy of TANDI by reducing the number of false positives.

4.2.4 Speed Optimization

TANDI's code is not optimized, so there are likely different optimizations to the code that could increase the speed of TANDI. Since TANDI was simulated over only a small set of data containing only a small subset of system log messages and IDS alerts, it is possible that a non-optimized implementation of TANDI may perform too slowly to be effective when executed over a real computer network. This may prove to be necessary when TANDI is integrated with an existing cyber attack detector.

4.2.5 Dynamic PASs and Logical Topology

One assumption that TANDI makes is that the PAS and the logical topology are statically defined. However, this is a very crude assumption and may not be realistic for most networks. Since cyber attacks are constantly changing, IDSs must generate new alerts to follow suit. Also, new servers or services could be deployed on the network, thus changing the alerts that can be generated on the network. Therefore, the PAS should be able to dynamically add new attacks as they are created.

The logical topology is primarily dictated by routing and firewall rules. It is likely that a hacker may alter these rules, thus changing the logical topology. Also, the topology may naturally change as certain links may go up or down. Therefore, TANDI should be extended to allow the logical topology to change as the network changes.

It should be noted that these dynamic changes should not affect the way that TANDI calculates its threat scores and aggregate metrics. However, the dynamic nature of the guidance template and logical topology will require research on *how* to dynamically change these graphs to accurately reflect the changes.

4.2.6 New Combination Rules

TANDI currently only uses a crude summation of weights to calculate threat scores. While this has proven effective for the simulations, there may be a more accurate combination rule of the weights. The feature tree, however, does provide an easy integration of new combination rules as they are created. TANDI also currently only looks at the next steps. It may be more beneficial to predict more than one step in advance.

As discussed in Section 2.3.6, a drawback of TANDI is that it does not take into account the number of times an attack occurs on a single entity. This may or may not truly affect the threat on that entity. Further research on this topic will indicate if there is a need to factor in this multiplicity of attacks and, if so, how it will be implemented.

4.2.7 Expansion of Metrics

This work only focused on the six metrics for performance analysis. One of these metrics, *Average Compromising Score Percentile*, did not provide any useful analysis over the simulated data. There are likely to be other aggregate metrics that may give a better indication of the overall threat of the attack or even the accuracy of the current model. As suggested by [26], these metrics should also be modified to include purity in addition to accuracy.

4.2.8 Detection of Coordinated Attacks

As shown in the simulations, a combination of coordinated attacks can potentially improve the overall quality of threat assessment over the set of independent attacks. An algorithm to defragment these attacks into one coordinated attack would prove to be invaluable.

Bibliography

- [1] Jeffrey Allanach, Haiying Tu, Satnam Singh, Peter Willett, and Krishna Pattipati. Detecting, tracking and counteracting terrorist networks via hidden markov models. In *Proceedings of IEEE Aerospace Conference*, pages 3246–3257, March 2004.
- [2] Jeffrey Allanach, Haiying Tu, Satnam Singh, Peter Willett, and Krishna Pattipati. Modeling threats. *IEEE Potentials*, 23(3):18–21, 2004.
- [3] Orincon Information Assurance. Daiwatch home page. <http://www.daiwatch.com/>.
- [4] Tim Bass. Intrusion detection systems and multisensor data fusion. *Communications of the ACM*, 43(4), April 2000.
- [5] Cedric Bennett and Richard Jacik. The zen of risk assessment. *Educause Quarterly*, (2), 2005.
- [6] Qu Changwen and He You. A method of threat assessment using multiple attribute decision making. In *Proceedings of 6th International Conference on Signal Processing*, volume 2, pages 1091–1095, August 2002.
- [7] Oliver Dain and Robert K. Cunningham. Fusing a heterogeneous alert stream into scenarios. In *Proceedings of ACM Workshop on Data Mining and Security*, December 2001.
- [8] Enterasys. Enterasys intrusion defense. <http://www.enterasys.com/products/ids/>.
- [9] David L. Hall and James Llinas. An introduction to multisensor data fusion. In *Proceedings of the IEEE*, volume 85, pages 6–23, January 1997.
- [10] Jeffrey Heer. The prefuse visualization toolkit. <http://prefuse.sourceforge.net/>.

- [11] Michael L. Hinman. Some computational approaches for situation assessment and impact assessment. In *Proceedings of IEEE Information Fusion Conference*, volume 1, pages 687–693, March 2002.
- [12] Cisco Systems Inc. *netForensics: Report Guide*. April 2003. http://www.cisco.com/application/pdf/en/us/guest/products/ps5209/c1626/ccmigration_09186a008019d567.pdf.
- [13] Finn Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.
- [14] Jesper Johansson and Steve Riley. *Protect Your Windows Network: From Perimeter to Data*. The Addison-Wesley Microsoft Technology Series. Addison Wesley Professional, 1st edition, May 2005.
- [15] Myong H. Kang and Terry Mayfield. A cyber-event correlation framework and metrics. In *Proceedings of SPIE*, volume 5107, pages 72–82, 2003.
- [16] Don Koks and Subhash Challa. An introduction to bayesian and dempster-schafer data fusion. Technical Report DSTO-TR-1436, Defence Science and Technology Organisation, Department of Defense, Australian Government, August 2003.
- [17] E. Little, G. Rogova, and A.C. Bourry-Brisset. Theoretical foundations of threat ontology for data fusion applications. Technical Report TR-2005 -269, DRDC-Valcartier, November 2005.
- [18] Yu Liu and Hong Man. Network vulnerability assessment using bayesian networks. In *SPIE Proceedings of Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security*, volume 5812, pages 61–71, March 2005.
- [19] J. Llinas, C. Bowman, G. Rogova, A. Steinberg, E. Waltz, and F. White. Revisions and extensions to the jdl data fusion model II. In *Proceedings of The 7th International Conference on Information Fusion*, pages 1218–1230, June 2004.
- [20] Microsoft. Security bulletin ms03-026. <http://www.microsoft.com/technet/security/bulletin/MS03-026.msp>.
- [21] Nessus. Nessus vulnerability scanner. <http://www.nessus.org>.
- [22] Steven Noel, Eric Robertson, and Sushil Jajodia. Correlating intrusion events and building attack scenarios through attack graph distances. In *Proceedings of ACSAC*, December 2004.

- [23] Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *Proceedings of the 1998 workshop on New security paradigms*, pages 71–79, New York, NY, USA, 1998. ACM Press.
- [24] L.R. Rabiner and B.H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, January 1986.
- [25] Kenneth N. Ross and Ronald D. Chaney. Hidden markov models for threat prediction. In *Proceedings of SPIE*, volume 4051, pages 300–311, 2000.
- [26] John J. Salerno, Erik P. Blasch, Michael Hinman, and Douglas M. Boulware. Evaluating algorithmic techniques in supporting situation awareness. In *Proceedings of SPIE*, volume 5813, pages 96–104, 2005.
- [27] Sourcefire. Snort: an open source network intrusion prevention and detection system. <http://www.snort.org>.
- [28] Moises Sudit, Adam Stotz, and Michael Holender. Situational awareness of a coordinated cyber attack. In *Proceedings of SPIE*, pages 114–129, March 2005.
- [29] Cisco Systems. Cisco security monitoring, analysis and response system. <http://www.cisco.com/en/US/products/ps6241/index.html>.
- [30] Carol Taylor, Axel Krings, and Jim Alves-Foss. Risk analysis and probabilistic survivability assessment (RAPSA): An assessment approach for power substation hardening. In *Proceedings of ACM Workshop on Scientific Aspects of Cyber Terrorism*, Washington, D.C., November 2002.
- [31] S. Vidalis and A. Jones. Using vulnerability trees for decision making in threat assessment. Technical Report CS-03-2, University of Glamorgan, School of Computing, June 2003.
- [32] F. E. White. A model for data fusion. In *Proceedings of 1st National Symposium on Sensor Fusion*, 1988.
- [33] Nong Ye, Yebin Zhang, and Connie M. Borror. Robustness of the markov-chain model for cyber-attack detection. *IEEE Transactions on Reliability*, 53(1):116–123, March 2004.