

California State University, San Bernardino

CSUSB ScholarWorks

Theses Digitization Project

John M. Pfau Library

2004

Mathematical security models for multi-agent distributed systems

Chunyan Ma

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>



Part of the [Information Security Commons](#)

Recommended Citation

Ma, Chunyan, "Mathematical security models for multi-agent distributed systems" (2004). *Theses Digitization Project*. 2568.

<https://scholarworks.lib.csusb.edu/etd-project/2568>

This Thesis is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

MATHEMATICAL SECURITY MODELS FOR MULTI-AGENT
DISTRIBUTED SYSTEMS

A Thesis
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

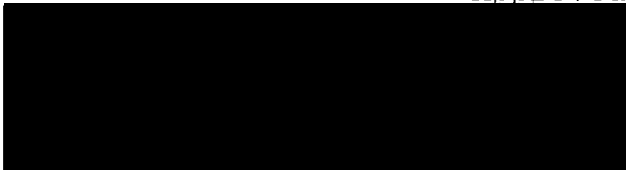
by
Chunyan Ma
December 2004

MATHEMATICAL SECURITY MODELS FOR MULTI-AGENT
DISTRIBUTED SYSTEMS

A Thesis
Presented to the
Faculty of
California State University,
San Bernardino


by
Chunyan Ma
December 2004

Approved by:



Dr. Arturo Concepcion,
Chair, Computer Science

15 Nov 2004
Date



Dr. Owen Murphy



Dr. Kay Lemoudeh

© 2004 Chunyan Ma

ABSTRACT

This thesis presents the developed taxonomy of the security threats in the agent-based distributed systems. Based on this taxonomy, a set of theories is developed to facilitate analyzing the security threats of the mobile-agent systems. We propose the idea of using the developed security risk graph to model the system's vulnerabilities. In a security risk graph, an agent or a host is a vertex and their security relationships form each edge. By using the security risk graph, we set up two mathematical models to quantitatively evaluate how secure a mobile-agent distributed system is.

In the probabilistic model, we calculate the Mean Time To Failure as the characteristic measure of the security between any two vertices. The Mean Time To Failure represents the approximate time used by the attacker to break into the target.

In the second mathematical model developed in this research, we calculate the transition time on the shortest path between any two vertices to evaluate the approximate time for the attacker to reach the target. The diameter of the whole system is used to represent the security measure of the entire system. In this way, we can compare the security level of different systems.

ACKNOWLEDGMENTS

As I approach the completion of this thesis, I cannot help looking back and express my gratitude to those who have helped me to make all these possible.

Many deepest thanks go to my thesis advisor, Dr. Arturo Concepcion, for his always support and guidance. Thank you for believing in and helping me especially in those times I struggled to continue. Your encouraging words, valuable advice and unending ideas have inspired me finding solutions to what seems an impossible work.

I would also like to thank my thesis committee members, Dr. Owen Murphy and Dr. Kay Zemoudeh. Thank you for your continuous support and valuable advice.

I would like to thank Dr. Tonglai Yu. Thank you for your sincere help and support.

Thank you for the warm-hearted assistance and support the office staffs of the Department of Computer Science have offered to me.

The award from the Instructionally Related Programs Research and Travel Fund and the Associated Students Incorporated Research and Travel Fund are gratefully acknowledged. The support of the National Science Foundation under award 9810708 is gratefully acknowledged.

A special thank goes to my family. To my dear parents, who have been the sources of my inspiration, insistance and strength.

Finally, special thanks give to all of my dear friends for your continuous inspiration, encouragement, help and support.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGMENTS	iv
LIST OF TABLES	viii
LIST OF FIGURES	ix
CHAPTER ONE: BACKGROUND	
1.1 Introduction	1
1.2 Related Works	4
1.3 Purpose of the Thesis	8
1.4 Context of the Problem	9
1.5 Significance of the Thesis	10
1.6 Organization of the Thesis	12
CHAPTER TWO: OVERVIEW AND THE TAXONOMY OF THE SECURITY THREATS IN THE AGENT-BASED SYSTEMS	
2.1 Introduction	14
2.2 Overview of the Security Threats in the Agent-based Distributed Systems	15
2.3 A Taxonomy of the Security Breaches in the Mobile-agent Systems	22
CHAPTER THREE: A PROBABILISTIC MODEL FOR AGENT- BASED DISTRIBUTED SYSTEM SECURITY THREAT EVALUATION	
3.1 Introduction	26
3.2 Introduction of the Security Risk Graph	27
3.3 Security Risks Analysis	30
3.4 Mathematical Quantification for Security Assessment	38

3.5 Illustrative Example	45
CHAPTER FOUR: DESIGN OF A MATHEMATICAL MODEL FOR SECURITY MEASURING OF AGENT-BASED DISTRIBUTED SYSTEMS	
4.1 Introduction	53
4.2 An Advanced Security Model for Security Evaluation	54
4.3 Quantification Algorithm for Security Measurement	56
CHAPTER FIVE: CONCLUSIONS AND FUTURE DIRECTIONS	
5.1 Conclusions	85
5.2 Future Directions	89
REFERENCES	90

LIST OF TABLES

Table 1. Summary of Related Works	7
Table 2. List of the Security Risks for the Agent-to-host Scenarios	17
Table 3. List of the Security Risks for the Host-to-agent Scenarios.....	19
Table 4. List of the Security Threats for the Agent-to-agent Scenarios	20
Table 5. List of the Security Threats for the Agent-to-network Scenarios	21
Table 6. Transition Time, its Corresponding Time in Weeks, Transition Rate and Graph Representation.....	46
Table 7. Part of the MTF Results, (in Number of Weeks) Calculated by Using the Proposed Method	52

LIST OF FIGURES

Figure 1. Taxonomy of the Security Threats for the Agent-based Systems.....24

Figure 2. An Example of a Security Risk Graph with Edges Labeled by Security Threats.....30

Figure 3. Security Risk Graph Example for Agent-based System.....32

Figure 4. Security Risk Graph Analyzed the Security Risks of the Host h35

Figure 5. Security Risk Graph for Agents on the Platform37

Figure 6. Security Risk Graph for B_2 as the Starting Point and B_1 the Target.....40

Figure 7. Security Risk Graph for A as the Intruder and B as the Target41

Figure 8. Security Risk Graph for Edge from One Vertex Goes Back to its Parent Case43

Figure 9. Security Risk Graph for Edge from One Vertex Goes Back to its Ancestor Case44

Figure 10. Security Risk Graph Example with Weight Demonstrated in Different Line Type48

Figure 11. Markov Graph for B_2 as the Attacker and A_2 as the Target49

Figure 12. Simplified Markov Graph for Figure 11 by Using Theorem 2 and Theorem 350

Figure 13. Simplified Markov Graph for Figure 12 Labeled by Transition Times58

Figure 14. Graph Showing Shortest Paths for B_2 as the Initial Vertex.....62

Figure 15. Graph Showing Shortest Paths for A_1 as the Initial Vertex.....66

Figure 16. Graph Showing Shortest Paths for A_2 as the Initial Vertex.....69

Figure 17. Graph Showing Shortest Paths for B_1 as the Initial Vertex.....73

Figure 18. Graph Showing Shortest Paths for B_3 as the Initial Vertex.....76

Figure 19. Graph Showing Shortest Paths for D_1 as the Initial Vertex.....80

Figure 20. Graph Showing Shortest Paths for $host_h$ as the Initial Vertex.....83

CHAPTER ONE

BACKGROUND

1.1 Introduction

Over the years computer systems have successfully evolved from centralized monolithic computing devices supporting static applications, into client-server environments that allow complex forms of distributed computing due to the advances in communication technology and the occurrence of more powerful workstations. A new phase of evolution is now under way in which complete mobility of cooperating applications among supporting platforms can form a large scale, loosely coupled distributed system. The mobile software agent is a new paradigm for structuring this new phase. A mobile agent [15] is a program that represents a user in a computer network, and is capable of migrating autonomously from node to node, to perform some computation on behalf of the user. Its tasks are determined by the agent application, and can range from online shopping to real-time device control to distributed scientific computing. Applications can inject mobile agents into a network, allowing them to roam the network either on a predetermined path, or one that the agents themselves determine based on dynamically

gathered information. Having accomplished their goals, the agents may return to their "home site" in order to report their results to the user.

Dispatching the application to other computers in the network can reduce the network communication overhead, provide access to more resources and introduce concurrency. Despite its many practical benefits, mobile agent technology results in significant new security threats from malicious agents and hosts. A malicious agent can steal or corrupt information on its host and in other agents as well. It is even more difficult to prevent a host from stealing the information from an agent, changing its states or even killing it. Therefore, solving the security problems of multi-agent distributed systems is crucial for fully utilizing its advantages. Many efforts [5, 7, 9, 10, 11, 12, 14, 15, 16, 18, 19, 22, 23, 24, 25, 26] have been made in this area. However, as of this writing there is not much work done for a quantitative evaluation on how secure an agent-based distributed system is. Most mobile agent projects provide one or more cryptographic methods to protect the agents or the hosts directly. Are these methods effective? How effective are they? And how safe one system's deployment of a set of security methods compared with another one, which has a

different set of security infrastructures? These questions cannot be easily answered without a quantitative security measurement of a system. Citations [1], [2], [6], [13], [17] include the few attempts to quantitatively measure the security of the distributed system. Among them, Chan and Lyu [2] is the only paper that proposed a mathematical model to calculate the probability of security breaching in mobile agent scenarios. In this thesis, we develop the taxonomy of the security threats in mobile agent-based distributed system. Based on the analysis of the different security threat situations that occurred in a mobile agent system, we derive probabilistic and further a mathematical security model for quantitatively evaluating the security of an agent-based distributed system.

In this research, we use security "risk" and "threat" alternatively. The security "risk" is the product of the level of threat with the level of vulnerability. It establishes the likelihood of a successful attack. And the security "threat" is a potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. So we use "threat" when we mention the potential violation of security and use "risk" when we want to express the measurement of the security.

1.2 Related Works

Presently the security schemes of the most mobile-agent systems address only the problems of protecting a machine from malicious agent and agents from each other. A growing number of mobile-agent projects are experimenting with techniques for protecting the underlying network from malicious agents and protecting agents from machines [10, 20, 24, 25]. While most efforts in security fields are devoted to cryptographic or isolation methods in protecting the systems, there is an urgent need for a quantitative assessment tool to evaluate how effective these methods are. There has been research which concentrates on quantitatively evaluating the security level of a system [1, 2, 6, 13, 17].

Brocklehurst, S. and Olovsson, T. [1] is the first one who proposed the idea to evaluate how secure a system is quantitatively. Their goal is to develop a security measure of a system which can quantitatively represent the ability of the system to resist attacks. It relates the system security with reliability. The concepts of system reliability are compared with and analogous to those in security, such as, system failure vs. system breach and the mean time to next failure vs. mean effort to next breach, etc. By analogy with the reliability function, in

the stochastic process of security breach, the distribution function of the effort, e , required for next breach is

$$F(e) = 1 - P(E > e),$$

where E is the mean effort to failure,

$P(E > e)$ is the probability of the mean effort to failure greater than the effort required for the next breach.

Continuing the work in Brocklehurst, S. and Olovsson, T. [1], Jonsson, E. and Olovsson, T. [13] conducted an empirical intrusion experiment to demonstrate the typical attacker's behavior. They divided the attacking process into three phases: the learning phase, the standard attack phase and the innovative attack phase. The probability of successful attacks during the learning and innovative phases are small. But it is considerably higher in the standard attack phase. And the collected data shows the times between breaches during this phase are exponentially distributed. This implied that traditional methods for reliability modeling could be applicable to the security evaluation.

Based on the theories developed in Brocklehurst, S. and Olovsson, T. [1] and Jonsson, E. and Olovsson, T. [13], Dacier, M. et. al. [6] proposed using privilege

graphs to model the system's vulnerabilities and calculating the characteristic measures of the distributed system security. The most important measure of the system security is Mean Time To security Failure, which is calculated by:

$$MTTF_k = T_k + \sum P_{kl} * MTTF_{kl}; P_{kl} = \lambda_{kl} * T_k,$$

where k is the initial states,

P_{kl} is the conditional probability transition from state k to state l,

T_k is the mean sojourn time in state k,

λ_{kl} is the transition rate from state k to state l.

Using the ideas developed in [6], Ortalo, R. et. al. [17] conducted an experiment for security evaluation. They modeled a large real system as a privilege graph exhibiting the security vulnerabilities. A set of tools has also been utilized to calculate the Mean Time to Failure. They found that the corresponding measure provides useful feedback to the security administrators. However, the security measure can not be always computed due to the complexity of the algorithm.

As summarized in Table 1, nearly all the works done in this area are for the traditional distributed systems. Chan and Lyu [2] is the only one who tried to model the security of mobile agent system probabilistically. It

proposed the idea about coefficient of malice k_i of each host and coefficient of vulnerability of an agent v and used them to calculate the probability of breaches on agent when it travels around on each host as

$$P(\text{breach at host } i) = 1 - \exp(-\lambda_i t_i),$$

where $-\lambda_i = vk_i$,

t_i is the amount of time during which the agent stays on host i .

The agent security E is the probability of no breach at all hosts in its itinerary,

$$E = e^{-\sum \lambda_i t_i}.$$

Table 1. Summary of Related Works

Security model for traditional distributed system	Security model for mobile agent-based distributed system
Dacier et. al. in [6]	Chan and Lyu in [2]
Jonsson and Olovsson in [13]	

1.3 Purpose of the Thesis

With the explosive development of networking and powerful workstations, the agent-based distributed system technology is becoming more and more promising. There is a great advantage if we can collect the computer's idle processing resources by sending out agents also brings about very serious security problems to us. How can we distinguish an agent from a virus? How much privilege should we grant to the agents running on our computers? What if the computer destroys the agents running on it? These are only a few questions among the scores of difficulties come with fascinating mobile agent technology. Thus it would be ideal if we can find a way to quantitatively evaluate how safe an agent is or a host computer is. As discussed in the previous section, we can see that most of attempts to quantitatively evaluate a system are for the traditional distributed systems. While [2] is the only effort to evaluate the security for the agent-based systems, it does not consider the breaches caused by a malicious agent on the host, on other agents and the underlying network. Also due to the complexity of the agent-based system, the question of how to get the coefficient of malice and vulnerability in [2] seems to be vague and impractical. The purpose of the thesis was to

develop a more practical security measure for the mobile agent-based distributed system. This measure can quantitatively represent the ability of the agent-based system to resist security attacks.

1.4 Context of the Problem

From the empirical data collected in [13], we know that most security attacks occurred during standard attack phase. So we are concentrated on the security breach behaviors in this phase.

The second model developed in this thesis is mainly designed for the system administrators who can use this model to evaluate the system's vulnerabilities. We assume the system administrators know the topology of the whole system. Thus they can use this model to monitor the security of the systems more efficiently. If this model is used for simulating the behavior of the real attackers, we also assume that the attackers have some ways to be familiar with the distribution and connections of the whole system before they start to launch any attack.

Although our work comes along with theoretical proof, this thesis just proposes the mathematical evaluation models. Future work needs to be done to test its feasibility on real mobile agent-based distributed systems.

1.5 Significance of the Thesis

The agent-based systems have become more and more attractive because this technology can provide us with more flexibilities and abilities to solve wider range of problems, some of which are not even solvable by using the classical methods. For example, in the sea-of-data applications, a huge amount of information is distributed among different locations. The information that is needed by other programs can never leave these locations. Due to the reasons, such as the ratio of the volume of information and the available bandwidth, the way in which the information is stored or the legal issues (like medical images in hospitals), the program cannot fetch the information back and process it. When we are in this kind of scenarios, agent technology seems to be the only solution. The price we need to pay for this advanced technology includes new sets of problems concerning the security. The security problem becomes even harder in the mobile-agent systems, where agents are not static and can migrate from one environment to another to continue the execution instead.

Currently, the majority of the research efforts have been emphasized the development of safer architectures or cryptographic applications to tackle the security issues

that arise in the mobile-agent systems. But at the same time we also desperately need a method to evaluate the efficiency of those security architectures or systems of cryptographic applications. This method should be able to provide quantitative measures indicating the security level of each cryptographic application deployed in mobile-agent systems. With this method one can tell a system of cryptographic application works better than another by comparing these measures. The aim of this research is to develop a security model to measure the security threat of a given agent-based distributed computing system and each element in the system.

This research also provides a taxonomy of the security risks in mobile-agent distributed systems. This taxonomy not only helps to give a clear view of the security threats in an agent-based system, but also serves as the basis of the development of our probabilistic model of agent-based system's security measurement.

The most significant contribution of this thesis is that it proposes a novel concept of security risk graph and uses the security risk graph to model the security threats in the mobile-agent distributed system.

At the time of this writing, there is only one paper about security evaluation of the agent-based distributed

systems published by Chan and Lyu [2]. As discussed in the related works section, due to the complicated nature of the mobile-agent distributed systems, their model seems to be limited in scope. There is also lack of rigorous proofs which support the claims in the paper. To overcome all these shortcomings, we provide a set of theories along with proofs to the validity of the model developed in this paper. Therefore, this thesis is the first attempt to present a complete theoretical security measurement model for the mobile agent-based distributed systems. Using this model, we can get a quantitative measure to indicate how secure an agent or a host is. Also as the first attempt in this field of study, we can evaluate how secure the whole system is.

1.6 Organization of the Thesis

The thesis is divided into five chapters. Chapter One provides an introduction to the context of the problem, related works, purpose of the thesis, and significance of the thesis. Chapter Two consists of an overview and the taxonomy of the security threats of the agent-based distributed systems. Chapter Three presents the probabilistic model for agent-based system security evaluation. Chapter Four provides an advanced mathematical

model for the security measurement of the mobile-agent systems developed in this thesis. Chapter Five gives the conclusion and future directions of the thesis. Finally, the references for the thesis are presented.

CHAPTER TWO

OVERVIEW AND THE TAXONOMY OF THE SECURITY

THREATS IN THE AGENT-BASED SYSTEMS

2.1 Introduction

There are many advantages in using mobile-agent systems. By using agent technology, we can move the code to the remote data to avoid the difficulty of moving the data, especially when the data volume is very large. We can also send out the agent to access the remote resources without keeping the network connection alive all the time. This is very useful when we have weak or expensive network links. Task parallelization and allowing the scalability of processing the information are two other benefits of using agent technology. These merits of agent technology are very important in the areas of telecommunication and massive information processing. However, more security threats emerge along with this new technology. Some of them are inherent in agent's nature and many are due to its mobility. Classic concept of security to support reliable system protection is based on the traditional taxonomy of the security threats. It is difficult to study the protection issues in the new mobile-agent scenarios. So in this chapter, we are going to focus on the security

issues in the mobile-agents systems and develop a security threats taxonomy on which we base our security model in this new paradigm.

2.2 Overview of the Security Threats in the Agent-based Distributed Systems

From the point of view of consequence of the security breaches, the traditional taxonomy of security threats identifies three main categories [11, 21] as: confidentiality, integrity and availability.

- Confidentiality is violated when unauthorized principals can learn protected information. It includes 3 subcategories of threats, they are: anonymity, traceability and traffic analysis. Common confidentiality breaches are eavesdropping, password guessing, masquerading or no password protection, etc.

- Integrity is infringed when unauthorized principals modify information. The frequent integrity threats are having improper write access, such as the ability to intercept or alter the information, or interference with the communication, etc.

- Availability is breached when the system is prevented from performing its intended function. Some common availability security risks are taking all the CPU

cycles to denial of service or jamming the communication channel, etc.

From the point of view of relationships between the actors in the mobile-agent systems, there are four main categories identified [15]: threats occurred when an agent attacks an agent platform; when an agent attacks against other agents; when an agent platform attacks an agent, and when an agent attacks the underlying network.

- The agent-to-host category includes the set of threats in which agents exploit the security weakness of an agent platform to launch attacks against their hosts. A malicious agent may steal or destroy the information on the host. It can also masquerade as another agent to the platform. Because the incoming agent has access to the CPU cycles and file systems of the host, it can install a virus or launch denial of service attacks to the host. In Table 2, there is a list of possible types of attacks we identified for the agent-to-host scenarios.

Table 2. List of the Security Risks for the Agent-to-host Scenarios

Security threat type	Threat's behavior
Masquerade	An unauthorized agent claims the identity of another agent
Resource exhaustion	An agent can consume an excessive amount of the platform's computing resources
Intercept/alter	An unauthorized agent obtains or change the data or code of the host
Eavesdropping	An unauthorized agent can monitor the communication of the host and obtain the confidential information
Repudiation	An agent participated in some transaction with the host and later denies that transaction took place

- The host-to-agent category represents the threats where platforms maliciously attack the agents running on them. Since the host controls every step in the execution of the agent, it can easily eavesdrop on the agent's communication with other agent or hosts, filch information, modify the code or state, masquerade as another platform or deny the services to the agent. As

shown in Table 3, we have a list of possible types of attacks we identified when a platform launches attacks to the agent(s).

- The agent-to-agent category represents the set of threats in which the agents exploit security weakness to launch attacks to other agents. An agent participating in a transaction or communication may repudiate the result by claiming the transaction or communication never happened. An agent may also masquerade as another agent to gain some agent's trust. Or an agent may interfere with other agents by eavesdropping the conversation or falsifying another agent's data or code. An agent can launch denial of service attacks by repeatedly sending messages to another agent too. In Table 4, we find a list of possible types of attacks when an agent launches attacks to other agents.

Table 3. List of the Security Risks for the Host-to-agent Scenarios

Security threat type	Threat's behavior
Masquerade	A platform pretends to be another platform to an agent
Resource exhaustion	A host deliberately consumes a resource of an agent so heavily that the service to other users is disrupted
Intercept/alter	A host can change, delete, or substitute data, code or in particular, the itinerary of the agent(s) running on it
Eavesdropping	A host can monitor the communication of the agent and obtain the confidential information
Repudiation	A platform participated in some transaction with the agent and later denies that transaction took place
Killing an agent	A malicious platform destroys the running agent on it
False system calls return values	A host returns the wrong values for the system calls initiated by the agents running on it
Replay	A copy of previously sent agent is retransmitted for malicious purpose

Table 4. List of the Security Threats for the Agent-to-agent Scenarios

Security threat type	Threat's behavior
Masquerade	An unauthorized agent claims the identity of another agent
Resource exhaustion	An agent deliberately consumes a resource of other agents so heavily that the service to other users is disrupted
Intercept/alter	An agent can change, delete, or substitute data, code of the other agents
Eavesdropping	An agent can monitor the communication between the other agents and obtain the confidential information
Repudiation	An agent participated in some transaction with the other agents and later denies that transaction took place
Replay	A copy of previously sent agent is retransmitted for malicious purpose
Killing an agent	An unauthorized agent has the possibility to end the life of other agents

- The agent-to-network class represents the set of threats in which the malicious agents get control of the underlying network and attack the normal communication of the network. An agent may consume excessive resources in the network by roaming through the network forever or creating endless children to exhaust the network resources. As shown in Table 5, we have a list of possible types of attacks identified when an agent launches attacks to the underlying network.

Table 5. List of the Security Threats for the Agent-to-network Scenarios

Security threat type	Threat's behavior
Masquerade	An unauthorized agent claims the identity of another agent
Resource exhaustion	An agent deliberately consumes a resource of the underlying network so heavily that the service to other users is disrupted
Intercept/alter	An agent can change, delete, or substitute data, code transmitted through the underlying network
Eavesdropping	An agent can monitor the communication traversed through the network and obtain the confidential information

2.3 A Taxonomy of the Security Breaches in the Mobile-agent Systems

As we can see from Tables 2, 3 and 4, for the mobile-agent systems, we have a new type of security threat-repudiation, i.e., when one party to a communication exchange or transaction later denies that the transaction or exchange took place. Since the repudiation relates with the trust and credit history, we name a category of security threats to which it belongs as creditability. Therefore, now we have four categories of security threats in the agent-based systems from the point of view of the consequence of the security breaches. They are: confidentiality, integrity, availability and creditability. Here we define the creditability as following:

- Creditability is violated when principals deny having performed a particular action. The most common breaches of creditability is repudiation and using some programs owned by others, etc.

The basic elements in a mobile-agent distributed system are the agents and the hosts. The hosts along with the underlying networks are the basic environment for the agents to execute. A host consists of a directory service, an agent manager and a message transport service.

Agents are software units executing on the hosts on behalf of their owners. Agents can be mobile or static, depending on the need of the agent's tasks.

Based on this structural characteristic of the agent-based systems, we can partition the security threats into two broader categories. As shown in Figure 1, security threats can be towards the host or towards the agent. They both have two subcategories as shown:

- the security breaches towards the hosts:
 - the malicious agent against agent platform
 - against the underlying network
- the security breaches towards the agents:
 - the malicious host against agent
 - agent against other agents

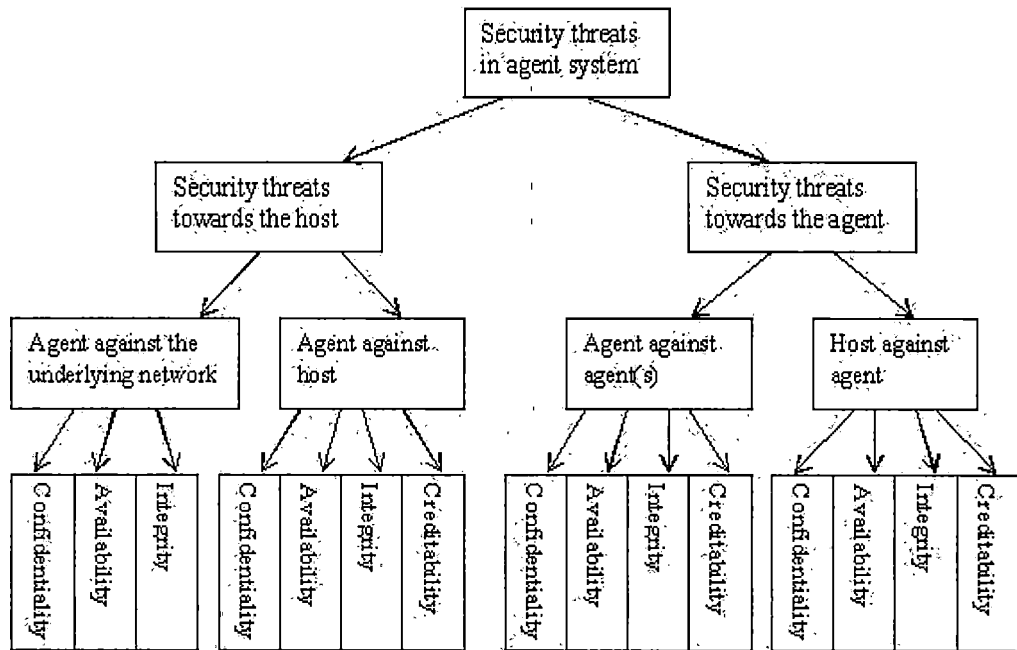


Figure 1. Taxonomy of the Security Threats for the Agent-based Systems

For each subcategories of the security threats, it also violates one of the four security requirements from the point of view of the consequence of the security breaches except for the agent-against-network category. As shown in Figure 1, the agent-against-network category will not violate the creditability security threat. If an agent has used the network to transfer information through the network connections, it has no possibility to deny that it has used the service.

From the taxonomy in Figure 1, we can develop the security model by differentiating the security risks of a host from those of an agent.

CHAPTER THREE

A PROBABILISTIC MODEL FOR AGENT-BASED DISTRIBUTED SYSTEM SECURITY THREAT EVALUATION

3.1 Introduction

Based on the taxonomy of security threats in the mobile-agent distributed systems, we design a probabilistic model for evaluating the security threats of the agents and hosts in the agent-based distributed systems.

Even though Chan and Lyu [2] have attempted to evaluate the security for the agent-based systems, their consideration does not include the security threats of agent-against-host, agent-against-agent and agent-against-network scenarios. Also the concepts of the coefficient of malice and vulnerability in Chan and Lyu [2] are not well defined in how to obtain the coefficient of malice and vulnerability is not clear. In this chapter, we develop a probabilistic security model in which all of the 4 security categories identified in the mobile agent system are taken into consideration. We develop the security risk graph and use it to set up a probabilistic model to evaluate the security of the mobile-agent systems quantitatively.

3.2 Introduction of the Security Risk Graph

The basic idea is to use a graph to describe the security threats that exist in an agent-based system. Graphs are used because they are well defined algorithmically and mathematically. The notations used in graphs are well known and easily adapted to the model we developed. We start by giving list of fundamental definitions. A security risk graph consists of a set of vertices and edges.

Definition 1. A vertex of a security risk graph is an agent or a host in the agent-based distributed system.

Definition 2. An edge in a security risk graph is an arc from vertex X to vertex Y , represented as (X, Y) .

An edge starts from vertex X and ends at vertex Y in the security risk graph means that a method exists for X to launch attack to Y .

Definition 3. A security threat of type r exists from vertex X to vertex Y means that there exists a method for vertex X to perform a type r attack to vertex Y .

For each type of security threat, there is an average access time to associate with it. We call it the transition time of a specific type of security threat.

Definition 4. Transition time is the time needed for a specific type of security threat r to succeed from one vertex to the next vertex.

Definition 5. Transition rate is the success rate of the corresponding attacks, defined as the inverse of the transition time t of the corresponding attacks.

Here the transition time or the transition rates from one vertex to another are the parameters we need to know before we can do our probabilistic evaluation. They can be obtained from the statistical estimation of agent's and host's profiles based on the analysis of the agents' and hosts' behavior and of their interaction with each other. By observing the system, we can get the transition time (which is the inverse of transition rate) indicate how hard for one node to perform one particular attack to another node and assign that value to the same kind of attack identified from the system we want to analyze.

Definition 6. The weight of each edge is the transition time of each edge.

Definition 7. A directed path in a security risk graph is a sequence of vertices along with the edges in between of them such that, for any adjacent pair of edges e_i and e_j , the ending vertex of e_i is the starting vertex of e_j . In this thesis we call a directed path as path.

We can now define the security risk graph, which we can use to model the security threats in an agent-based distributed system.

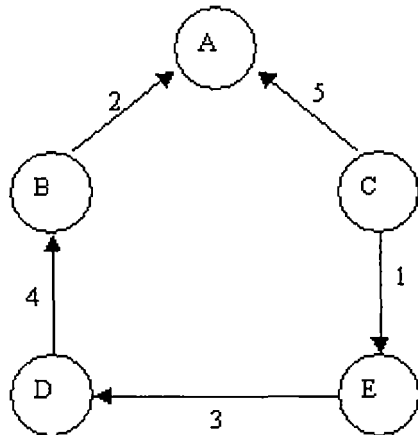
Definition 8. A security risk graph of an agent-based distributed system is a directed and weighted graph $G(V, E, W)$ where V is a set of vertices, W is the set of weights and E is a set of edges between the vertices $E = \{w(u, v) \mid u, v \in V, w \in W, u \neq v\}$.

Figure 2 shows an example of a security risk graph. A, B, C, D and E are the vertices and the edges are labeled by the security threat types. Note that only when a path between an attacker and a target exists, is there a possibility that a security breach can occur. For instance, as illustrated in Figure 2, B cannot gain any access to E. So B cannot be a potential attacker to E.

By formalizing this intuitive idea, we can get the following lemma.

Lemma 1. A security threat exists from one vertex X to another vertex Y whenever there is a path that starts from X and ends at Y .

Proof: Let us consider contrapositive statement of this Lemma. That is, "If there is no path starts from vertex X and ends at Y , there exists no security threat from vertex X to Y ". Since we know this reciprocal statement holds



- 1) X can read files from Y (intercept);
- 2) X can guess Y's password;
- 3) X can get hold of Y's CPU cycle;
- 4) Y has no password;
- 5) Y uses a program owned by X.

Figure 2. An Example of a Security Risk Graph with Edges Labeled by Security Threats

and the proof follows, we deduce that Lemma 1 is true. \square

3.3 Security Risks Analysis

Based on the taxonomy of the agent-based system vulnerabilities, we can deal with the security situation for a host and an agent separately.

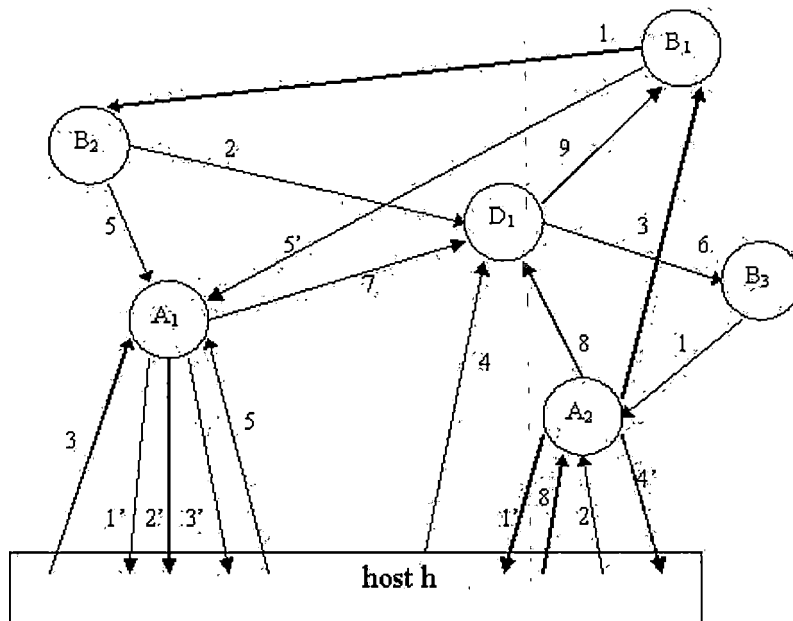
Combine the security risks related with agent against host and agent against the underlying network, we can analyze the security risks a host needs to consider. As shown in the example in Figure 3, the security risks labeled in the numerical values and a prime are the security breaches to the host. Thus when we analyze the

security risks of a host, we can discard all the edges from the host to agents and some of the edges between agents that do not count for attacking the host and the underlying network.

We find that the masquerade is a tricky type of security threat. Since both the agent and the host can masquerade as another one in the same kind. When vertex X masquerades as another vertex Y to a third vertex Z , is it an attack to Y , or Z or both? An example can be seen in Figure 3. In this example, agent $B1$ can masquerade as agent $A1$ to the host. So that this attack is toward agent $A1$ only, the host only or both is a question. To solve this problem, we need to provide more notations and definitions.

Definition 9. We call vertex X equals vertex Y if vertex X 's behavior looks the same as vertex Y 's behavior, denoted as $X = Y$.

Definition 10. Masquerade is the act of imitating the behavior of vertex X to vertex Y under false pretense, denoted as $X(Y)$.



- 1) X can guess Y's password;
- 2) X can eavesdrop Y's communication with others;
- 3) X has write access to Y (alteration);
- 4) X can masquerade as another platform to Y;
- 5) Y uses a program owned by X;
- 6) X can repudiate the result from Y;
- 7) X can copy and replay Y's information;
- 8) Y has no password;
- 9) X can deny the service to Y;
- 1') X can read files from Y;
- 2') X can write files to Y;
- 3') X can get hold of Y's CPU cycle;
- 4') X can get hold of network resources;
- 5') X can masquerade as another agent Y to the platform.

Figure 3. Security Risk Graph Example for Agent-based System

From Definition 9 and Definition 10, we know when vertex X masquerades as Y , its behavior looks the same as Y 's behavior, that is $X(Y) = Y$.

Definition 11. The behavior of B as seen by C is denoted as $B \gamma C$.

Unlike other kind of security threats, masquerade is a very special type of security threat because it has the following characteristics.

Masquerade Transition Law:

If Entity A can masquerade as Entity B to Entity C , then that is an attack from Entity A to Entity C .

Proof: To C , Entity B is as itself, so $B \gamma C = B$.

When under masquerading, Entity A acts as B , so $A \gamma C = A(B) \gamma C = B \gamma C$,

because $A(B) = B$.

So we have $B \gamma C = A \gamma C = B$.

Because A behaves itself to Entity C under the name of B , to obtain the privileges of C , so that is an attack from A to C . \square

Saying that masquerade is special is because other types of security threats do not necessarily have this feature. For example, If vertex A can intercept the information of vertex C , vertex B can also intercept the

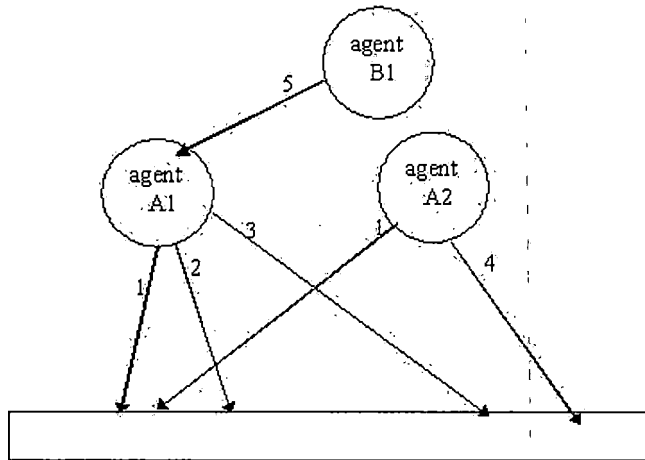
information of C, then A is not necessarily definitely able to intercept the information of C. Based on this Masquerade Transition Law, we can get and prove the following lemma.

Lemma 2. If a vertex A can masquerade as another vertex B to the third vertex C, then this is a security risk from A to B, also a security risk from A to C.

Proof: First to prove this is a security risk from A to B is trivial. Because A masquerade as B, whatever A does has affected B's reputation. So it is an indirect security risk from A to B.

Also by using the masquerade transition law, we know it is a security risk from A to C. \square

Take Figure 3 as an example, agent B1 can masquerade as agent A1 to the host. So that is a security risk to agent A1 and to the host as well. By using Lemma 2, we can leave agent B1 in the graph for the analysis of the security risks for the host. Because we discover an agent could masquerade as another agent to the platform. We regard it belongs to the security risks a host needs to face, also as a security attack form between agents. Then we can isolate the security risks the host will face, see Figure 4.



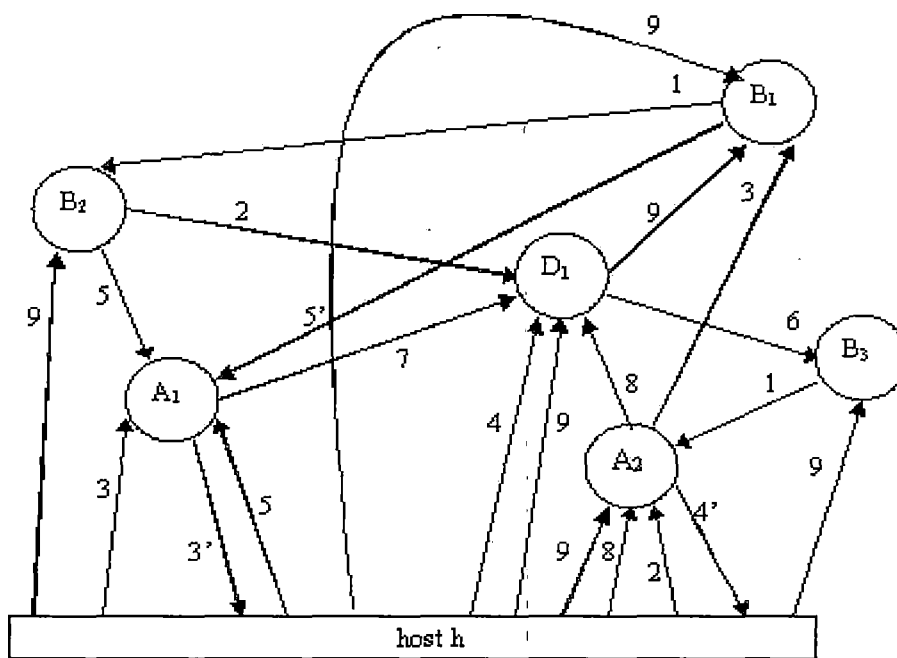
- 1) X can read files from Y (intercept);
- 2) X can write files to Y (alteration);
- 3) X can get hold of Y's CPU cycle;
- 4) X can get hold of network resources;
- 5) X can masquerade as another agent Y to the platform.

Figure 4. Security Risk Graph Analyzed the Security Risks of the Host h

By far, we have used an example to illustrate how to analyze the security risks a host will face in the system. Our basic idea is to eliminate all the unnecessary edges and vertices to make all the connections to this host stand out. To summarize, we can just consider the connections to the host and the communications between agents that can cause any breach to the host for analysis of the security risks of the host.

But for analyzing the security of the agents, using Figure 3 as an example, we cannot just discard all the

possibilities in Figure 4. Because one agent can take all host h 's CPU cycles, so as to deny the service to other agents. For instance, as shown in Figure 5, A_1 can get hold of host's CPU cycle, thus launch denial of service attack to every agent running on host h . Note that in Figure 3, we don't have the edges of type 9 from host h to each agent. By the process of analyzing the security "Towards the agent" scenarios, we found that we should add those edges. In fact, we can generalize this observation into the following theorem for analyzing the security risk graph of the agent-based system.



- 1) X can guess Y's password;
- 2) X can eavesdrop Y's communication with others;
- 3) X has write access to Y(alteration);
- 4) X can masquerade as another platform to Y;
- 5) Y uses a program owned by X;
- 6) X can repudiate the result from Y;
- 7) X can copy and replay Y's information;
- 8) Y has no password;
- 9) X can deny the service to Y;
- 3') X can get hold of Y's CPU cycle;
- 4') X can get hold of network resources;
- 5') X can masquerade as another agent Y to the platform.

Figure 5. Security Risk Graph for Agents on the Platform

Theorem 1. If an agent A on host i can take all of host i 's CPU cycles, it can in turn launch denial of service attack to all of the agents running on this host except for A itself. If more than two agents on the same host i can take all of i 's CPU cycles, the first one which launches the attack can succeed.

Proof: Since all of the agents running on a host need to utilize CPU cycles for its performance, if the CPU is hold completely by one agent, then the other agents cannot function correctly. If the first agent can get hold of all CPU cycles successfully, all the other agents running on host i can not even function correctly. They would have no chance to succeed in taking hold of all CPU cycles any more. \square

3.4 Mathematical Quantification for Security Assessment

After we have developed the security risk graph of a system, the Markov model is chosen to quantify the security risks of a multi-agent distributed system. Among various probabilistic measures derived from the Markov model, we use the MTTF (Mean Time To Failure) value which we define as follows.

Definition 12. Mean Time To Failure is the mean transition time for a potential attacker to reach the specified target, denoted as MTTF.

The MTTF is obtained by summing all the mean transition times in the edges leading to the target vertex that each edge is weighted by the transition rate of each attack. The mean time in vertex j , denoted as T_j , is given by the inverse of the sum of vertex j 's output transition rates:

$$T_j = 1 / \sum_{l \in \text{out}(j)} \lambda_{jl},$$

where λ_{jl} is the transition rate from vertex j to vertex l , and $\text{out}(j)$ is the set of all vertices to which j is connected by an edge where j is the starting point.

The MTTF_k is the mean time to failure when vertex k is the initial vertex and P_{kl} is the conditional transition probability from vertex k to vertex l . They are defined as follows:

$$\text{MTTF}_k = T_k + \sum P_{kl} * \text{MTTF}_{kl};$$

$$P_{kl} = \lambda_{kl} * T_k.$$

Though we have set up the way to quantify the security using the Markov model, before we can really start the calculation, we have some more things to do. Due to the fact that the agent-based distributed system is

quite different from the traditional distributed system as we have several agents running on a single host or a host can launch several different kinds of attacks to one particular agent, we face a problem of how to analyze and calculate the MTTF for this kind of situation. For example, as shown in Figure 6, when agent B2 is the attacker and agent B1 is the target, we have an edge loop back to its ancestor, like $host_h A_1$. How could we calculate the MTTF for this scenario? In Figure 4 and Figure 5, agent A1 can perform 3 different attacks to host h (in Figure 4) and host h can launch four different attacks to agent A2 (in Figure 5). Which one we should choose for calculating T_j ? In responding to these problems, we have developed the following theorems to handle them.

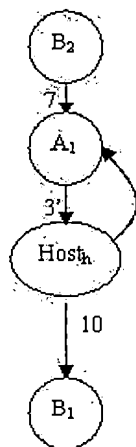


Figure 6. Security Risk Graph for B₂ as the Starting Point and B₁ the Target

Theorem 2. If there are several edges with the same direction from one vertex to the next in the security risk graph and suppose the intruders do not know the whole topology of the system, when calculating MTTF regarding to these two vertices, choosing the shortest edge with the smallest transition time will not affect the MTTF calculation.

Proof: Without losing generality, as seen in the security risk graph (Figure 7), suppose A is an intruder and B as a target. There are several edges from A to B, AB_1, AB_2, \dots, AB_n . Each edge has a transition time t_1, \dots, t_n associated with it. Suppose $t_i = \min\{ t_1, \dots, t_n \}$. Based on the assumption in Dacier, M. et. al. [6], the intruders do not know the whole topology of the security risk graph. They only know the attacks that can be directly applied in a single step. So A has the options t_1, t_2, \dots, t_n to attack B. From the empirical results obtained from Jonsson, E. and Olovsson, T. [12], the intruder A would always try to

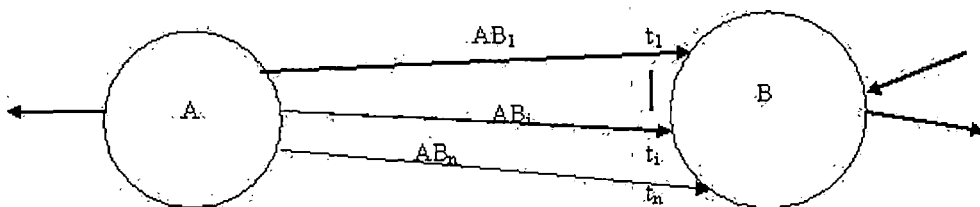


Figure 7. Security Risk Graph for A as the Intruder and B as the Target

perform the attack takes the least time which is t_i . \square

Theorem 3. If there is an edge from one vertex that goes back to its ancestor, then this edge would not be counted in calculating MTTF.

Proof: From the attacker's point of view, he would choose a route which takes the least time. Reflected in the privilege graph, the attacker's goal is to choose the branch that takes the least time.

Case I. Edge from one vertex goes back to its parent.

Without losing generality, suppose there is one branch in the security risk graph that has an edge from one vertex B_1 goes back to its parent A_i , as circled part in Figure 8. Then the time taken by each of all the other branches is just the sum of all the edges in that branch. From the security risk graph, we can see that if the time taken from R go all the way down to T is t (t is chosen by selecting the smallest value among different routes). But if we loop back at B_1 ,

for route $A_i - B_1$, total time = $t + t_{B_1i} + t_{B_1i}$, where $t_{B_1i} + t_{B_1i} > 0$.

So total time $> t$.

for route $A_i - B_j - B_1$, total time = $t + t_{B_1j} + t_{B_1j} + t_{B_1i}$, where $t_{B_1j} + t_{B_1j} + t_{B_1i} > 0$.

So total time $> t$.

That means whatever the routes in between vertex A_i and B_1 , total time $> t + t_{B1i} > t$. So when calculating METF, we discard the edges that loop back to the parent.

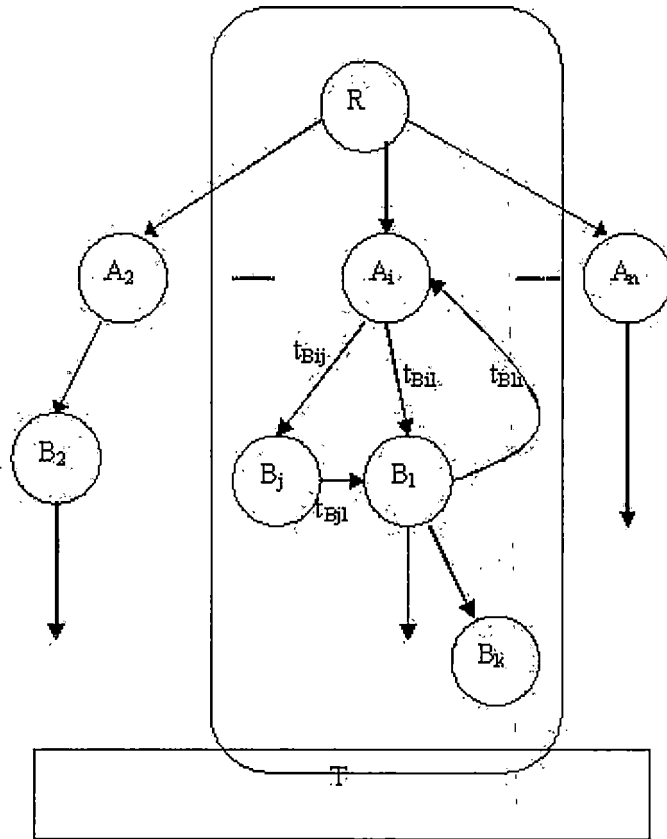


Figure 8. Security Risk Graph for Edge from One Vertex Goes Back to its Parent Case

Case II. Edge from one node goes back to its ancestor. Similar to case I, the time taken by the branch edge from one vertex n_i goes back to its ancestor is greater than

the time taken from R go all the way through A_i , n_i down to T, as in Figure 9. If t is the time taken to loop back from n_j to A_i , $t > 0$:

So when calculating MTTF, we discard the edges that loop back to the ancestor of this vertex n_i . \square

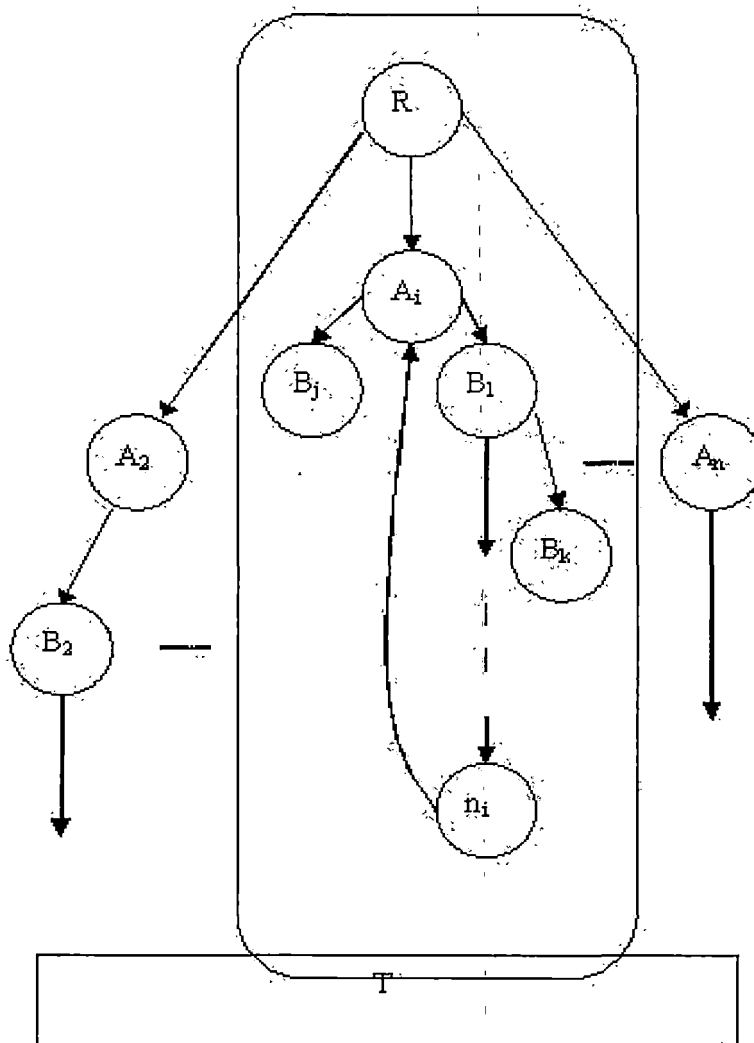
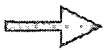




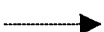


Figure 9. Security Risk Graph for Edge from One Vertex Goes Back to its Ancestor Case

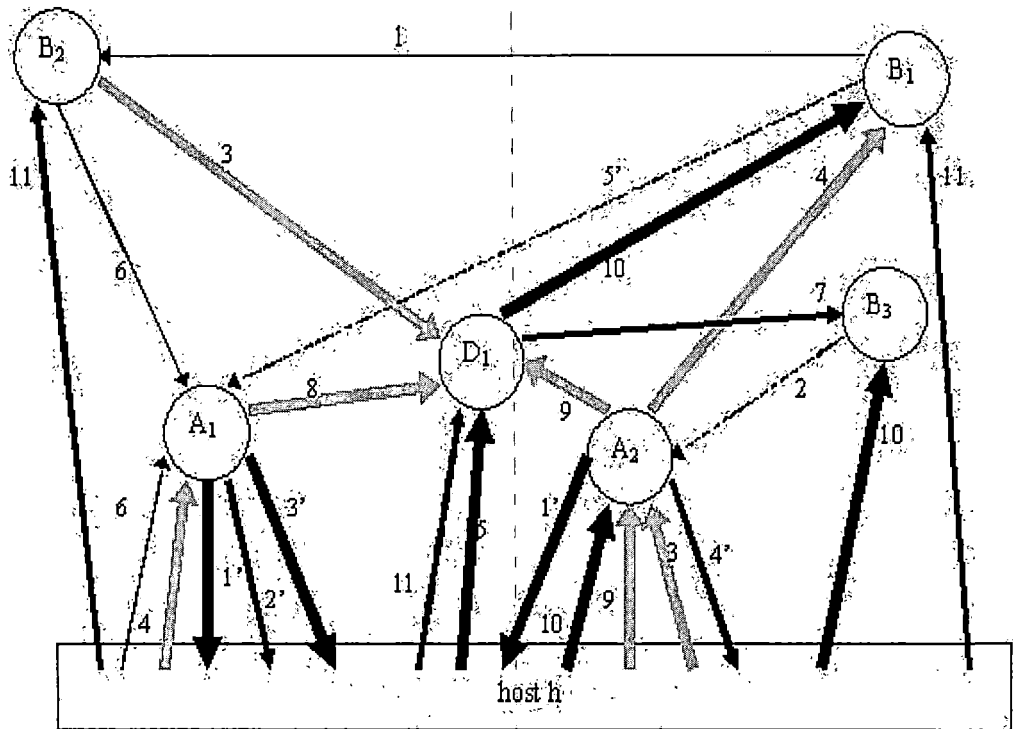
3.5 Illustrative Example

Now let us use an example to illustrate how this approach works. We use the example in Figure 3. Figure 10 shows that the edges are assigned different thicknesses to represent their weight and also to characterize the difficulty of the breaches: the thicker the edge, the easier the breach. For the convenience of calculation, we use one week as the unit of attack times. So every different transition time can be digitally quantified, as weeks, for instance, one day is approximately 0.2 week. In this way, the transition rate for each attack is 1 divided by the corresponding transition time. To represent very easy attacks (quasi-instantaneous transition firing), the transition rate is assigned a high value as 5000. Table 6 lists the transition time, its corresponding time in weeks, transition rate and the graph representation in the security risk graph for the identified transition time of the attacks.

Table 6. Transition Time, its Corresponding Time in Weeks, Transition Rate and Graph Representation

Transition time	Transition Time in weeks	Transition rate	Line type in the security risk graph
Quasi-instantaneous	0.0002	5000	
one hour	0.02	50	
one day	0.2	5	
one week	1	1	
one month	5	0.2	
one year	50	0.02	

To illustrate how to calculate the MTTF, we take B_2 as the attacker, A_2 as the target from Figure 10 and generate the Markov graph as in Figure 11.



- 1) X can guess Y's password in one week (one week →);
- 2) X can guess Y's password in one month (one month →);
- 3) X can eavesdrop Y's communication with others (quasi-instantaneous →);
- 4) X has write access to Y (alteration) (quasi-instantaneous);
- 5) X can masquerade as another platform to Y (one hour →);
- 6) Y uses a program owned by X once in a year (one year →);
- 7) X can repudiate the result from Y in one day (one day →);
- 8) X can copy and replay Y's information (quasi-instantaneous);

- 9) Y has no password (quasi-instantaneous);
- 10) X can deny the service to Y in one hour;
- 11) X can deny the service to Y in one day;
- 1') X can read files from Y in one hour;
- 2') X can write files to Y in one day;
- 3') X can get hold of Y's CPU cycle in one hour;
- 4') X can get hold of network resources in one day;
- 5') X can masquerade as another agent Y to the platform in one month.

Figure 10. Security Risk Graph Example with Weight Demonstrated in Different Line Type

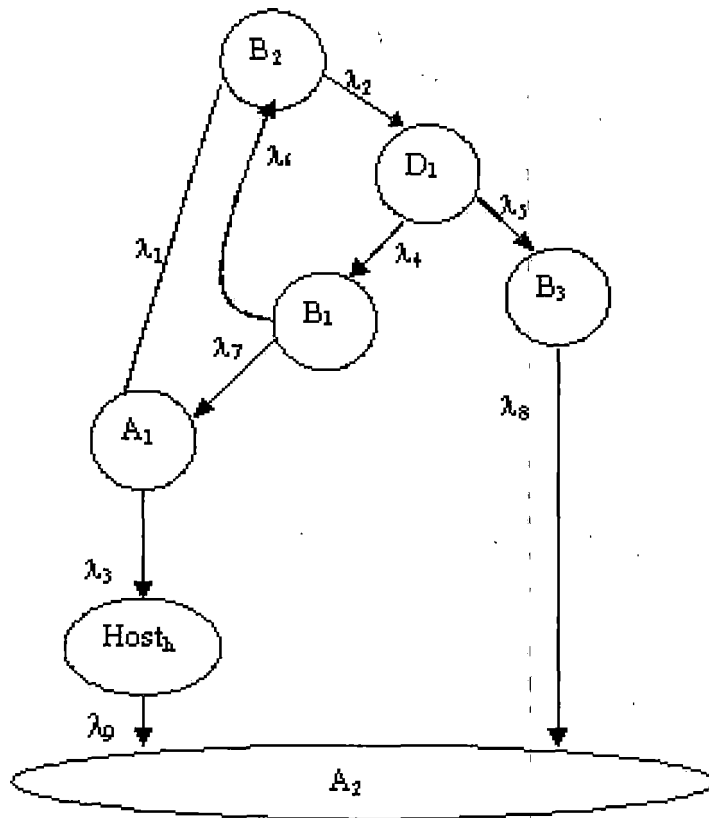


Figure 11. Markov Graph for B_2 as the Attacker and A_2 as the Target

By using Theorem 3, we can eliminate the edges B_1B_2 , to get Figure 12. Also for transition rate of edge A_1Host_h we can choose the one that gives the smallest transition time based on Theorem 2.

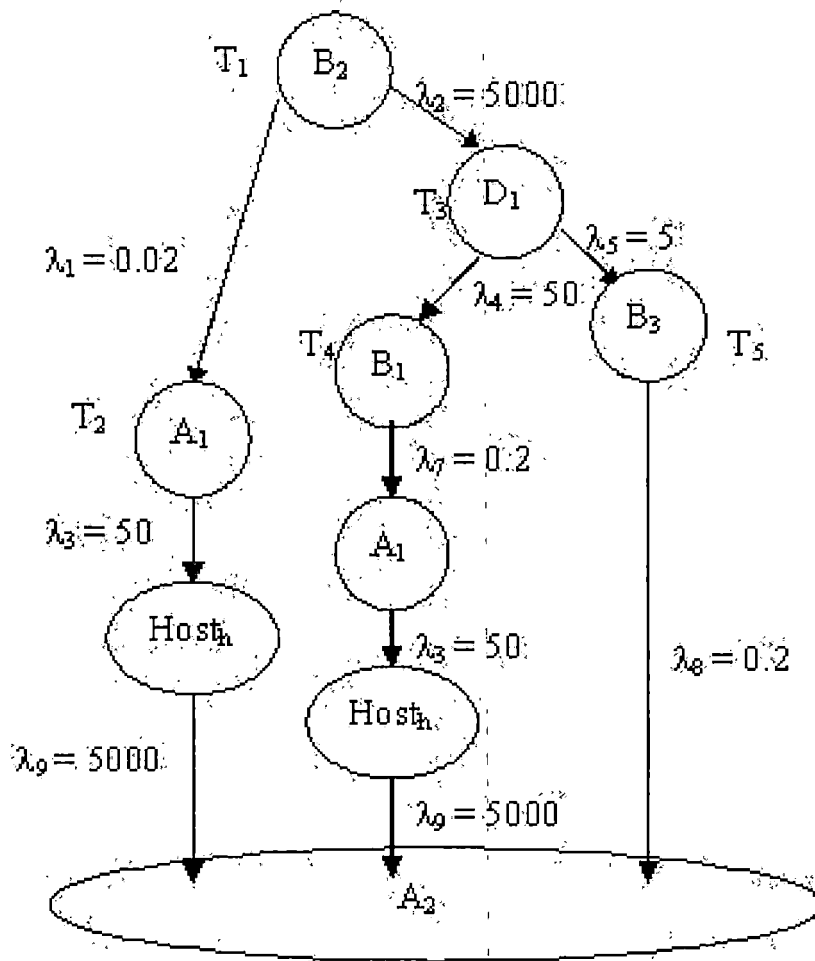


Figure 12. Simplified Markov Graph for Figure 11 by Using Theorem 2 and Theorem 3

Then we can calculate MTTF as following:

$$MTTF = T_1 + P_{12}MTTF_2 + P_{13}MTTF_3$$

$$T_1 = 1 / (\lambda_1 + \lambda_2) = 1 / (0.02 + 5000) = 0.00019999$$

$$P_{12} = \lambda_1 * T_1 = 0.02 * 0.00019999 = 0.000004$$

$$P_{13} = \lambda_2 * T_1 = 5000 * 0.00019999 = 0.999995$$

$$\text{MTTF}_2 = T_2 = 1/\lambda_3 + 1/\lambda_9 = 1/50 + 1/50000 = 0.02 + 0.0002 = 0.0202$$

$$\text{MTTF}_3 = T_3 + P_{32}\text{MTTF}_4 + P_{33}\text{MTTF}_5$$

$$T_3 = 1/(\lambda_4 + \lambda_5) = 1/(50+5) = 0.01818182$$

$$P_{32} = \lambda_4 * T_3 = 50 * 0.01818182 = 0.909091$$

$$P_{33} = \lambda_5 * T_3 = 5 * 0.01818182 = 0.0909091$$

$$\text{MTTF}_4 = 1/\lambda_7 + 1/\lambda_3 + 1/\lambda_9 = 1/0.2 + 1/50 + 1/50000 = 5 + 0.02 + 0.0002 = 5.0202$$

$$\text{MTTF}_5 = 1/\lambda_8 = 1/0.2 = 5$$

$$\text{MTTF}_3 = 0.01818182 + 0.909091 * 5.0202 + 0.0909091 * 5 = 5.036535958$$

So $\text{MTTF} = 0.00019999 + 0.000004 * 0.0202 + 0.999995 * 5.036535958 = 5.03671$, which means the average time for B_2 to attack A_2 is about 5.03671 weeks.

Part of the result is shown in Table 7. In this table, we selected 3 vertices as target and calculated the MTTF other attackers need to spend to reach them. The MTTF is represented in time duration as number of weeks. For example, take agent B_2 as the attacker and agent B_3 as the target, B_2 need 0.2002 weeks to succeed one attack to B_3 .

Table 7. Part of the MTTF Results (in Number of Weeks)
 Calculated by Using the Proposed Method

Attacker \ Target	Host h	A ₂	B ₃
B ₂	--	5.0367	0.2002
A ₁	0.02	5.148	0.1984
Host h	--	0.0002	0.02
B ₁	5.02	5.1745	0.8916

CHAPTER FOUR

DESIGN OF A MATHEMATICAL MODEL FOR SECURITY MEASURING OF AGENT-BASED DISTRIBUTED SYSTEMS

4.1 Introduction

In this research, we found that the MTTF calculation is complicated even only with basic security threats taken into consideration. If we want to evaluate a large network with numerous machines running, the calculation could be enormous. Even for the traditional distributed systems, Ortalo, R. et. al. [17] has proved that the calculation of MTTF can not be computed sometimes due to the complications by performing experiments. In order to overcome this shortcoming, we used the shortest path algorithm to reduce the amount of computation. Also by using the shortest path, we have the method to find the diameter of the security risk graph so that we can evaluate the security of the whole system. The shortest path was discussed in the study of security risks in the traditional distributed network in Dacier, M. et. al. in [6]. They claim that the shortest path can only be the major contribution to the MTTF calculation. Since an attacker does not know the whole topology of the network, they believe that the attacker could not always take the

shortest path. They showed that the mean time to reach the target using MTTF calculation is always smaller than the value calculated using the shortest path. While the MTTF calculation can only estimate the mean time to failure between any two vertices, it would be more usable if we can compare the security between two systems besides just comparing between any two vertices. Especially for the system administrators, after they perform some security upgrade, they may want to compare the upgraded system security with the original one to see how effective the new methods are. In this Chapter, we are going to present an advanced mathematical model developed in the thesis by using the shortest path.

4.2 An Advanced Security Model for Security Evaluation

As we presented in Chapter 3, we have modeled the system's security risks using the security risk graph. Based on the analysis of the security threat types, we developed Lemma 2 and Theorem 1 to identify some special kinds of attacks and add all the necessary edges. Then the developed Theorem 2 and Theorem 3 are used to simplify the generated security risk graph so that we can calculate the MTTF using Markov model. However, the MTTF calculation is too complicated and time-consuming and sometimes not even

computable by the ordinary computers [17]. Also our goal is to find a way to evaluate the entire system's security risk as opposed to finding the security risk between any two vertices. First we need to know what a shortest path is in a security risk graph.

Definition 13. Let P be a path containing vertices v_1, v_2, \dots, v_n , and $w(v_i, v_j)$ be the weight on the edge connecting v_i to v_j , then the length of path P is defined as

$$|P| = \sum_{i=1}^{n-1} w(v_i, v_{i+1}).$$

Definition 14. A shortest path from vertex u to vertex v is defined as any path with weight $\delta(u, v) = \min\{w(P) \mid P(u \sim v)\}$, where $P(u \sim v)$ is the set of paths from vertex u to vertex v , and $w(P)$ is the set of weights of each path in $P(u \sim v)$.

Despite the argument in Dacier, M. et. al. in [6] that the shortest path cannot be used for calculating the MTTF. We assume that if there exists one or more paths between two vertices in a security risk graph, the attack time taken from the starting point to the target can be represented by the sum of the transition times on the shortest path. Because the time on the shortest path

describes the least time the attacker will need to use to break into the target. If the attacker does not know the topology of the whole system, the time needed to break into the target will be definitely more than the time calculated from the shortest path or at best is equal to the time from the shortest path. Also due to the difference between the agent-based distributed systems and the traditional distributed systems, the attackers may have some means to know the shortest path from the starting point to the target or even the topology of the whole system by probing the vulnerabilities first. In either case, the shortest path is a suitable measure for the system administrators to evaluate the system's security level.

4.3 Quantification Algorithm for Security Measurement

In this section we present an algorithm to find the security measure based on the Dijkstra's algorithm and use a simple example to show how this algorithm works.

Security risks estimation algorithm:

Input: Weighted graph G , source, destination (G is the simplified graph using Theorems 2 and 3)

Output: Transition time from source to destination

Temp: temporary tree structure to hold the nodes and edges as we go through graph G

1. add source, Transition time (source) = 0 to Temp

2. while (destination \notin Temp)

find edge (u, v), where:

- a. $u \in$ Temp;

- b. $v \notin$ Temp;

- c. minimize the transition time over all (u, v)

satisfies a and b.

The resulted transition time = transition time(u) + w(u, v), where w(u, v) is the weight of (u, v).

Actually, Dijkstra algorithm can find the shortest path to every vertex to which the source vertex has a connection besides the target vertex. It has the same time complexity as the one needed for just finding the shortest path to the target.

Take Figure 12 as an example, this time we want to use the transition time instead of the transition rate and generate the Figure 13.

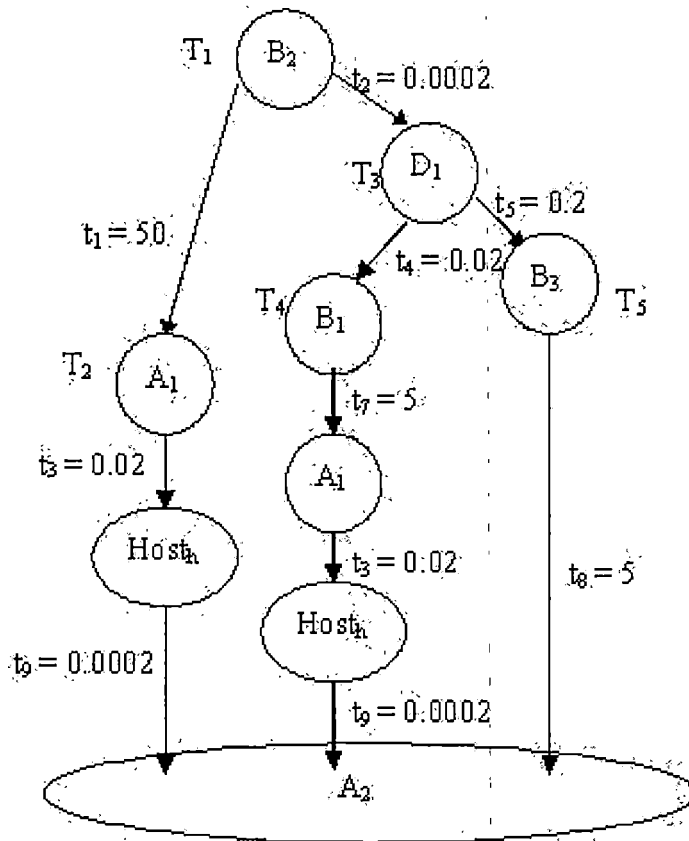


Figure 13. Simplified Markov Graph for Figure 12 Labeled by Transition Times

Following the above algorithm, the example in Figure 13 works as below:

1. Take the source vertex B_2 and put it in Temp.
Temp = $\{B_2\}$
2. Since B_2 connects to A_1 and D_1 , we mark A_1 and D_1 as candidates.
3. Compare $|B_2A_1| = 50$ and $|B_2D_1| = 0.0002$. Because $|B_2D_1|$ is smaller, we take D_1 into Temp. Now

Temp = {B₂, D₁} and we also get the shortest path between B₂ and D₁ is B₂D₁ with | B₂D₁ | = 0.0002.

4. Since D₁ is in Temp, now we need to consider the vertices connected to D₁: B₁ and B₃. After we mark them, our candidates are A₁, B₁ and B₃.

5. Compare:

$$| B_2D_1B_3 | = 0.0002 + 0.2 = 0.2002$$

$$| B_2D_1B_1 | = 0.0002 + 0.02 = 0.0202$$

$$| B_2A_1 | = 50$$

Because | B₂D₁B₁ | is smaller, we take B₁ into Temp.

Now Temp = {B₂, D₁, B₁} and the shortest path between B₂ and B₁ is B₂D₁B₁ with | B₂D₁B₁ | = 0.0202.

6. Now that B₁ is in Temp, we need to consider the vertices connected to B₁: A₁. After we mark it, our candidates are A₁, and B₃.

7. Compare:

$$| B_2D_1B_1A_1 | = 0.0002 + 0.02 + 5 = 5.0202$$

$$| B_2D_1B_3 | = 0.0002 + 0.2 = 0.2002$$

$$| B_2A_1 | = 50$$

Because | B₂D₁B₃ | is smaller, we take B₃ into Temp.

Now Temp = {B₂, D₁, B₁, B₃} and the shortest path between B₂ and B₃ is B₂D₁B₃ with | B₂D₁B₃ | = 0.2002.

8. Now that B_3 is in Temp, we need to consider the vertices connected to B_3 : A_2 . After we mark it, our candidates are A_2 , and A_1 .

9. Compare:

$$\begin{aligned} | B_2D_1B_1A_1 | &= 0.0002 + 0.02 + 5 \\ &= 5.0202 \end{aligned}$$

$$| B_2D_1B_3A_2 | = 0.0002 + 0.2 + 5 = 5.2002$$

$$| B_2A_1 | = 50$$

Because $| B_2D_1B_1A_1 |$ is smaller, we take A_3 into Temp. Now Temp = $\{B_2, D_1, B_1, B_3, A_1\}$ and the shortest path between B_2 and A_1 is $B_2D_1B_1A_1$ with $| B_2D_1B_1A_1 | = 5.0202$.

10. Since A_1 is in Temp, now we need to consider the vertices connected to A_1 : $host_h$. After we mark it, our candidates are A_2 , and $host_h$.

11. Compare:

$$| B_2D_1B_3A_2 | = 0.0002 + 0.2 + 5 = 5.2002$$

$$| B_2A_1host_h | = 50 + 0.02 = 50.02$$

$$\begin{aligned} | B_2D_1B_1A_1host_h | &= 0.0002 + 0.02 + 5 + 0.02 \\ &= 5.0402 \end{aligned}$$

Because $| B_2D_1B_1A_1host_h |$ is smaller, we take $host_h$ into Temp. Now Temp = $\{B_2, D_1, B_1, B_3, A_1, host_h\}$ and the shortest path between B_2 and $host_h$ is $B_2D_1B_1A_1host_h$ with

$$| B_2D_1B_1A_1host_h | = 5.0402.$$

12. Since $host_h$ is in Temp, now we need to consider the vertices connected to $host_h$: A_2 . After we mark it, our candidates are A_2 .

13. Compare:

$$| B_2D_1B_3A_2 | = 0.0002 + 0.2 + 5 = 5.2002$$

$$| B_2A_1host_hA_2 | = 50 + 0.02 + 0.0002 = 50.0202$$

$$\begin{aligned} | B_2D_1B_1A_1host_hA_2 | &= 0.0002 + 0.02 + 5 + 0.02 \\ &\quad + 0.0002 \\ &= 5.0404 \end{aligned}$$

Because $| B_2D_1B_1A_1host_hA_2 |$ is smaller, we take A_2 into Temp. Now $Temp = \{B_2, D_1, B_1, B_3, A_1, host_h, A_2\}$ and the shortest path between B_2 and A_2 is

$B_2D_1B_1A_1host_hA_2$ with

$$| B_2D_1B_1A_1host_hA_2 | = 5.0404.$$

By using the above algorithm in all these steps, we have found all the shortest paths starting from agent B_2 , and ending to every other vertex. We illustrate those shortest paths in Figure 14.

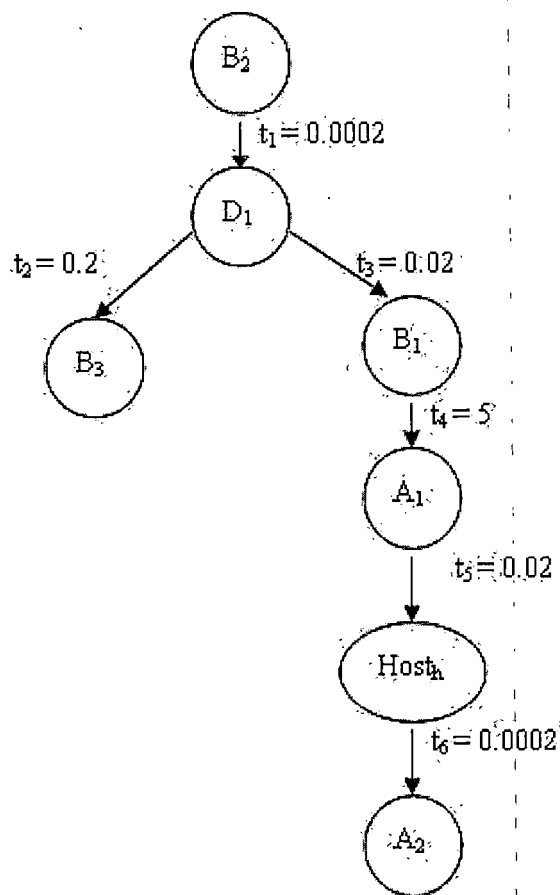


Figure 14. Graph Showing Shortest Paths for B_2 as the Initial Vertex

Next let us take A_1 as the initial vertex and try to find all the shortest paths starting from A_1 .

1. Take the source vertex A_1 and put it in Temp.
Temp = $\{A_1\}$
2. Since A_1 connects to D_1 and $host_h$, we mark D_1 and $host_h$ as candidates.

3. Compare $| A_1D_1 | = 0.0002$ and $| A_1host_h | = 0.02$.
Because $| A_1D_1 |$ is smaller, we take D_1 into Temp.

Now

Temp = $\{A_1, D_1\}$ and we also get the shortest path between A_1 and D_1 is A_1D_1 with $| A_1D_1 | = 0.0002$.

4. Since D_1 is in Temp, now we need to consider the vertices connected to D_1 : B_1 and B_3 . After we mark them, our candidates are $host_h$, B_1 and B_3 .

5. Compare:

$$| A_1D_1B_3 | = 0.0002 + 0.2 = 0.2002$$

$$| A_1D_1B_1 | = 0.0002 + 0.02 = 0.0202$$

$$| A_1host_h | = 0.02$$

Because $| A_1host_h |$ is smaller, we take $host_h$ into Temp. Now Temp = $\{A_1, D_1, host_h\}$ and the shortest path between A_1 and $host_h$ is A_1host_h with

$$| A_1host_h | = 0.02.$$

6. Now that $host_h$ is in Temp, we need to consider the vertices connected to $host_h$: A_2 , B_2 , B_1 and B_3 .
After we mark them, our candidates are A_2 , B_2 , B_1 and B_3 .

7. Compare:

$$| A_1D_1B_1 | = 0.0002 + 0.02 = 0.0202$$

$$| A_1D_1B_3 | = 0.0002 + 0.2 = 0.2002$$

$$| A_1host_hB_2 | = 0.02 + 0.2 = 0.22$$

$$| A_1 \text{host}_h B_3 | = 0.02 + 0.02 = 0.04$$

Because $| A_1 D_1 B_1 |$ is smaller, we take B_1 into Temp.

Now Temp = $\{A_1, D_1, \text{host}_h, B_1\}$ and the shortest path between A_1 and B_1 is $A_1 D_1 B_1$ with $| A_1 D_1 B_1 | = 0.0202$.

8. Now that B_1 is in Temp, we need to consider the vertices connected to B_1 : B_2 . After we mark it, our candidates are A_2 , B_2 , and B_3 .

9. Compare:

$$| A_1 D_1 B_3 | = 0.0002 + 0.2 = 0.2002$$

$$| A_1 \text{host}_h B_3 | = 0.02 + 0.02 = 0.04$$

$$| A_1 \text{host}_h A_2 | = 0.02 + 0.0002 = 0.0202$$

Because $| A_1 \text{host}_h A_2 |$ is smaller, we take A_2 into Temp. Now Temp = $\{A_1, D_1, \text{host}_h, B_1, A_2\}$ and the shortest path between A_1 and A_2 is $A_1 \text{host}_h A_2$ with $| A_1 \text{host}_h A_2 | = 0.0202$.

10. Since A_2 is in Temp, now we need to consider the vertices connected to A_2 : B_1 and host_h . Since we have found the shortest path for both of them, we do not mark them. Our candidates are B_2 and B_3 .

11. Compare:

$$| A_1 D_1 B_3 | = 0.0002 + 0.2 = 0.2002$$

$$| A_1 \text{host}_h B_2 | = 0.02 + 0.2 = 0.22$$

$$| A_1 \text{host}_h B_3 | = 0.02 + 0.02 = 0.04$$

$$| A_1D_1B_1B_2 | = 0.0002 + 0.02 + 1 = 1.0202$$

Because $| A_1\text{host}_hB_3 |$ is smaller, we take B_3 into Temp. Now $\text{Temp} = \{A_1, D_1, \text{host}_h, B_1, A_2, B_3\}$ and the shortest path between A_1 and B_3 is $A_1\text{host}_hB_3$ with $| A_1\text{host}_hB_3 | = 0.04$.

12. Since B_3 is in Temp, now we need to consider the vertices connected to B_3 : A_2 . Since we have found the shortest path for it, we do not mark it. Our candidates are B_2 .

13. Compare:

$$| A_1\text{host}_hB_2 | = 0.02 + 0.2 = 0.22$$

$$| A_1D_1B_1B_2 | = 0.0002 + 0.02 + 1 = 1.0202$$

Because $| A_1\text{host}_hB_2 |$ is smaller, we take B_2 into Temp. Now $\text{Temp} = \{A_1, D_1, \text{host}_h, B_1, A_2, B_3, B_2\}$ and the shortest path between A_1 and B_2 is $A_1\text{host}_hB_2$ with $| A_1\text{host}_hB_2 | = 0.22$.

From the above steps, we have found all the shortest paths starting from agent A_1 , and ending to every other vertex. We illustrate those shortest paths in Figure 15.

Let us take A_2 as our next initial vertex and try to find all the shortest paths starting from A_2 .

1. Take the source vertex A_2 and put it in Temp.

Temp = {A₂}

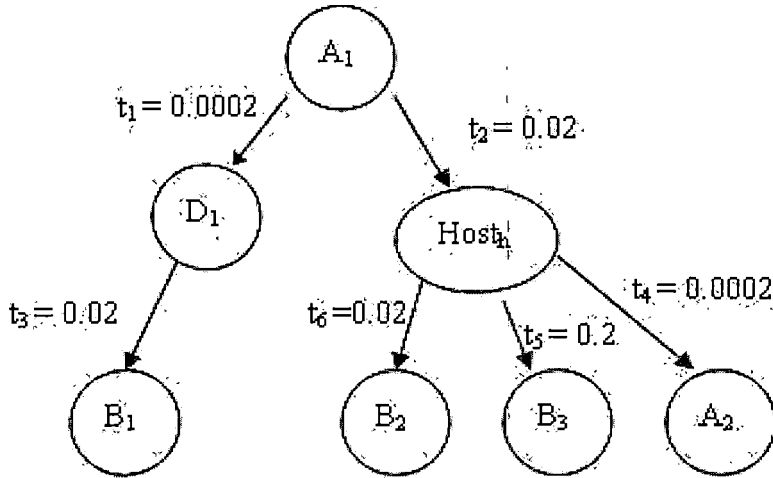


Figure 15. Graph Showing Shortest Paths for A₁ as the Initial Vertex

2. Since A₂ connects to D₁, B₁ and host_h, we mark D₁, B₁ and host_h as candidates.
3. Compare

$$| A_2 D_1 | = 0.0002$$

$$| A_2 \text{host}_h | = 0.02$$

$$| A_2 B_1 | = 0.0002$$

Because $| A_2 D_1 | = | A_2 B_1 |$ are the smallest, we take D₁ and B₁ into Temp. Now Temp = {A₂, D₁, B₁} and we also get the shortest path between A₂ and D₁ is A₂D₁ with $| A_2 D_1 | = 0.0002$ and between A₂ and B₁ is A₁B₁ with $| A_2 B_1 | = 0.0002$.

4. Since D_1 and B_1 are in Temp, now we need to consider the vertices connected to D_1 and B_1 : B_2 , B_3 , A_1 , B_1 and $host_h$. After we mark some of them, our candidates are $host_h$, A_1 , B_2 and B_3 .

5. Compare:

$$| A_2 host_h | = 0.02$$

$$| A_2 D_1 B_3 | = 0.0002 + 0.2 = 0.2002$$

$$| A_2 B_1 B_2 | = 0.0002 + 1 = 1.0002$$

$$| A_2 B_1 A_1 | = 0.0002 + 5 = 5.0002$$

Because $| A_2 host_h |$ is smaller, we take $host_h$ into Temp. Now Temp = $\{A_2, D_1, B_1, host_h\}$ and the shortest path between A_2 and $host_h$ is $A_2 host_h$ with $| A_2 host_h | = 0.02$.

6. Now that $host_h$ is in Temp, we need to consider the vertices connected to $host_h$: A_1 , D_1 , B_2 , B_1 and B_3 . After we mark some of them, our candidates are A_1 , B_2 and B_3 .

7. Compare:

$$| A_2 D_1 B_3 | = 0.0002 + 0.2 = 0.2002$$

$$| A_2 host_h B_3 | = 0.02 + 0.02 = 0.04$$

$$| A_2 B_1 B_2 | = 0.0002 + 1 = 1.0002$$

$$| A_2 host_h B_2 | = 0.02 + 5 = 5.02$$

$$| A_2 B_1 A_1 | = 0.0002 + 5 = 5.0002$$

$$| A_2 host_h A_1 | = 0.02 + 0.0002 = 0.0202$$

Because $| A_2 \text{host}_h A_1 |$ is smaller, we take A_1 into Temp. Now $\text{Temp} = \{A_2, D_1, B_1, \text{host}_h, A_1\}$ and the shortest path between A_2 and A_1 is $A_2 \text{host}_h A_1$ with $| A_2 \text{host}_h A_1 | = 0.0202$.

8. Now that A_1 is in Temp, we need to consider the vertices connected to A_1 : D_1 and host_h . Since we have them in the Temp already, we do not mark them. Our candidates are B_2 and B_3 .

9. Compare:

$$| A_2 D_1 B_3 | = 0.0002 + 0.2 = 0.2002$$

$$| A_2 \text{host}_h B_3 | = 0.02 + 0.02 = 0.04$$

$$| A_2 \text{host}_h D_1 B_3 | = 0.02 + 0.02 + 0.2 = 0.24$$

$$| A_2 B_1 B_2 | = 0.0002 + 1 = 1.0002$$

$$| A_2 \text{host}_h B_2 | = 0.02 + 5 = 5.02$$

Because $| A_2 \text{host}_h B_3 |$ is smaller, we take B_3 into Temp. Now $\text{Temp} = \{A_2, D_1, B_1, \text{host}_h, A_1, B_3\}$ and the shortest path between A_2 and B_3 is $A_2 \text{host}_h B_3$ with $| A_2 \text{host}_h B_3 | = 0.04$.

10. Now that B_3 is in Temp, we need to consider the vertices connected to B_3 : A_2 . As we do not need to mark it, our candidates are B_2 .

11. Compare:

$$| A_2 B_1 B_2 | = 0.0002 + 1 = 1.0002$$

$$| A_2 \text{host}_h B_2 | = 0.02 + 5 = 5.02$$

Because $|A_2B_1B_2|$ is smaller, we take B_2 into Temp.

Now Temp = $\{A_2, D_1, B_1, \text{host}_h, A_1, B_3, B_2\}$ and the

shortest path between A_2 and B_2 is $A_2B_1B_2$ with

$$|A_2B_1B_2| = 1.0002.$$

Thus we have found all the shortest paths starting from agent A_2 , and ending to every other vertex. We illustrate those shortest paths in Figure 16.

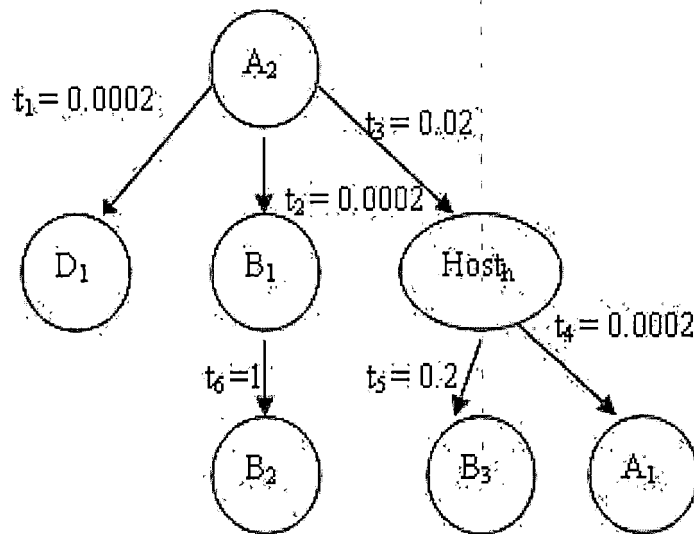


Figure 16. Graph Showing Shortest Paths for A_2 as the Initial Vertex

Our next target is to take B_1 as the initial vertex and try to find all the shortest paths starting from B_1 .

1. Take the source vertex B_1 and put it in Temp.

$$\text{Temp} = \{B_1\}$$

2. Since B_1 connects to A_1 and B_2 , we mark A_1 and B_2 as candidates.

3. Compare

$$| B_1A_1 | = 5$$

$$| B_1B_2 | = 1$$

Because $| B_1B_2 | = 1$ is smaller, we take B_2 into Temp. Now $\text{Temp} = \{B_1, B_2\}$ and we also get the shortest path between B_1 and B_2 is B_1B_2 with

$$| B_1B_2 | = 1.$$

4. Since B_2 is in Temp, now we need to consider the vertices connected to B_2 : A_1 and D_1 . After we mark some of them, our candidates are A_1 and D_1 .

5. Compare:

$$| B_1A_1 | = 5$$

$$| B_1B_2A_1 | = 1 + 50 = 51$$

$$| B_1B_2D_1 | = 0.0002 + 1 = 1.0002$$

$$| B_1A_1D_1 | = 0.0002 + 5 = 5.0002$$

Because $| B_1B_2D_1 |$ is smaller, we take D_1 into Temp.

Now $\text{Temp} = \{B_1, D_1, B_2\}$ and the shortest path between B_1 and D_1 is $B_1B_2D_1$ with $| B_1B_2D_1 | = 1.0002$.

6. Now that D_1 is in Temp, we need to consider the vertices connected to D_1 : B_3 . After we mark it, our candidates are A_1 and B_3 .

7. Compare:

$$| B_1A_1 | = 5$$

$$| B_1B_2A_1 | = 1 + 50 = 51$$

$$| B_1B_2D_1B_3 | = 1 + 0.0002 + 0.2 = 1.2002$$

$$| B_1A_1D_1B_3 | = 0.0002 + 5 + 0.2 = 5.2002$$

Because $| B_1B_2D_1B_3 |$ is smaller, we take B_3 into Temp. Now Temp = $\{B_1, D_1, B_2, B_3\}$ and the shortest path between B_1 and B_3 is $B_1B_2D_1B_3$ with

$$| B_1B_2D_1B_3 | = 1.2002.$$

8. Now that B_3 is in Temp, we need to consider the vertices connected to B_3 : A_2 . After we mark it, our candidates are A_1 and A_2 .

9. Compare:

$$| B_1A_1 | = 5$$

$$| B_1B_2A_1 | = 1 + 50 = 51$$

$$| B_1B_2D_1B_3A_2 | = 1 + 0.0002 + 0.2 + 5 = 6.2002$$

Because $| B_1A_1 |$ is smaller, we take A_1 into Temp.

Now Temp = $\{B_1, D_1, B_2, B_3, A_1\}$ and the shortest path between B_1 and A_1 is B_1A_1 with

$$| B_1A_1 | = 5.$$

10. Now that A_1 is in Temp, we need to consider the vertices connected to A_1 : $host_h$ and D_1 . Since D_1 has already been in the Temp, so we only we mark $host_h$, our candidates are $host_h$ and A_2 .

11. Compare:

$$| B_1A_1host_h | = 5 + 0.02 = 5.02$$

$$| B_1B_2A_1host_h | = 1 + 50 + 0.02 = 51.02$$

$$| B_1B_2D_1B_3A_2 | = 1 + 0.0002 + 0.2 + 5 = 6.2002$$

Because $| B_1A_1host_h |$ is smaller, we take $host_h$ into Temp. Now $Temp = \{B_1, D_1, B_2, B_3, A_1, host_h\}$ and the shortest path between B_1 and $host_h$ is $B_1A_1host_h$ with $| B_1A_1host_h | = 5.02$.

12. Now that $host_h$ is in Temp, we need to consider the vertices connected to $host_h$: B_2, B_3, A_2 and D_1 .

Since B_2, B_3 and D_1 has already in the Temp, so our candidate is only A_2 .

13. Compare:

$$| B_1A_1host_hA_2 | = 5 + 0.02 + 0.02 = 5.04$$

$$| B_1B_2D_1B_3A_2 | = 1 + 0.0002 + 0.2 + 5 = 6.2002$$

Because $| B_1A_1host_hA_2 |$ is smaller, we take A_2 into Temp. Now $Temp = \{B_1, D_1, B_2, B_3, A_1, host_h, A_2\}$ and the shortest path between B_1 and A_2 is $B_1A_1host_hA_2$ with $| B_1A_1host_hA_2 | = 5.04$.

By now we have found all the shortest paths starting from agent B_1 , and ending to every other vertex. We illustrate those shortest paths in Figure 16.

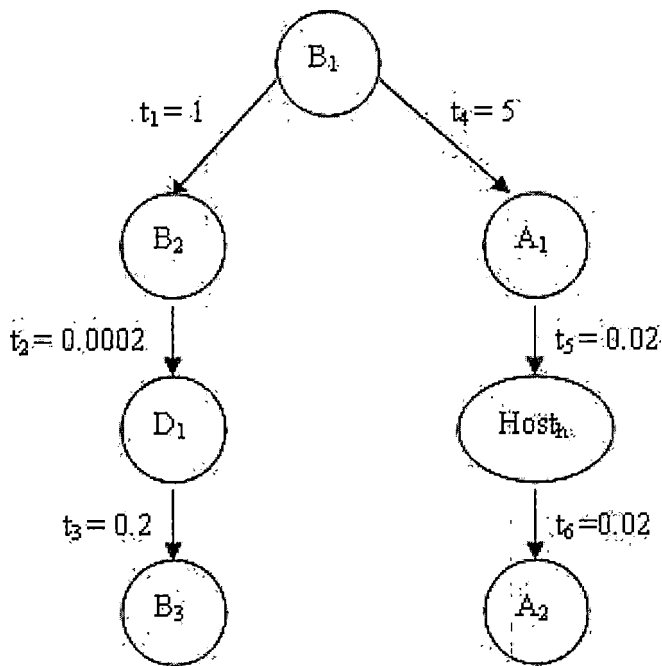


Figure 17. Graph Showing Shortest Paths for B_1 as the Initial Vertex

Next, let us analyze the situations when B_3 as the initial vertex and try to find all the shortest paths starting from B_3 .

1. Take the source vertex B_3 and put it in Temp.
Temp = $\{B_3\}$
2. Since B_3 connects to A_2 , we mark A_2 as candidate.
3. Since $|B_3A_2| = 5$ and we have nothing to compare with it, we take A_2 into Temp. Now Temp = $\{B_3, A_2\}$ and we also get the shortest path between B_3 and A_2 is B_3A_2 with $|B_3B_2| = 5$.

4. Since A_2 is in Temp, now we need to consider the vertices connected to A_2 : D_1 , B_1 and $host_h$. After we mark them, our candidates are D_1 , $host_h$ and B_1 .

5. Compare:

$$| B_3A_2B_1 | = 5 + 0.0002 = 5.0002$$

$$| B_3A_2D_1 | = 0.0002 + 5 = 5.0002$$

$$| B_3A_2host_h | = 0.2 + 5 = 5.2$$

Because $| B_3A_2D_1 | = | B_3A_2B_1 |$ are smaller, we take D_1 and B_1 into Temp. Now Temp = $\{B_3, A_2, D_1, B_1\}$.

The shortest path between B_3 and D_1 is $B_3A_2D_1$ with $| B_3A_2D_1 | = 5.0002$ and between B_3 and B_1 is $B_3A_2B_1$ with $| B_3A_2B_1 | = 5.0002$.

6. Now that D_1 and B_1 are in Temp, we need to consider the vertices connected to them: B_2 , B_1 and A_1 . After we mark some of them, our candidates are A_1 , B_2 and $host_h$.

7. Compare:

$$| B_3A_2host_h | = 0.2 + 5 = 5.2$$

$$| B_3A_2B_1B_2 | = 1 + 5 + 0.0002 = 6.0002$$

$$| B_3A_2B_1A_1 | = 0.0002 + 5 + 5 = 10.0002$$

Because $| B_3A_2host_h |$ is smaller, we take $host_h$ into Temp. Now Temp = $\{B_3, A_2, D_1, B_1, host_h\}$ and the shortest path between B_3 and $host_h$ is $B_3A_2host_h$ with $| B_3A_2host_h | = 5.2$.

8. Now that $host_h$ is in Temp, we need to consider the vertices connected to $host_h$: A_1 , B_2 and D_1 . After we mark some of them, our candidates are A_1 and B_2 .

9. Compare:

$$| B_3A_2B_1B_2 | = 1 + 5 + 0.0002 = 6.0002$$

$$| B_3A_2B_1A_1 | = 0.0002 + 5 + 5 = 10.0002$$

$$| B_3A_2host_hA_1 | = 5 + 0.2 + 0.0002 = 5.2002$$

$$| B_3A_2host_hB_2 | = 0.2 + 5 + 0.2 = 5.4$$

Because $| B_3A_2host_hA_1 |$ is smaller, we take A_1 into Temp. Now $Temp = \{B_3, A_2, D_1, B_1, host_h, A_1\}$ and the shortest path between B_3 and A_1 is $B_3A_2host_hA_1$ with $| B_3A_2host_hA_1 | = 5.2002$.

10. Now that A_1 is in Temp, we need to consider the vertices connected to A_1 : $host_h$ and D_1 . Since D_1 and $host_h$ have already been in the Temp, so our candidate is only B_2 .

11. Compare:

$$| B_3A_2B_1B_2 | = 1 + 5 + 0.0002 = 6.0002$$

$$| B_3A_2D_1B_1B_2 | = 5 + 0.0002 + 0.02 + 1 = 6.0202$$

$$| B_3A_2host_hB_2 | = 0.2 + 5 + 0.2 = 5.4$$

Because $| B_3A_2host_hB_2 |$ is smaller, we take B_2 into Temp. Now $Temp = \{B_3, A_2, D_1, B_1, host_h, A_1, B_2\}$ and

the shortest path between B_3 and B_2 is $B_3A_2\text{host}_hB_2$
with $| B_3A_2\text{host}_hB_2 | = 5.4$.

Therefore we have found all the shortest paths
starting from agent B_3 , and ending to every other vertex.
We illustrate those shortest paths in Figure 18.

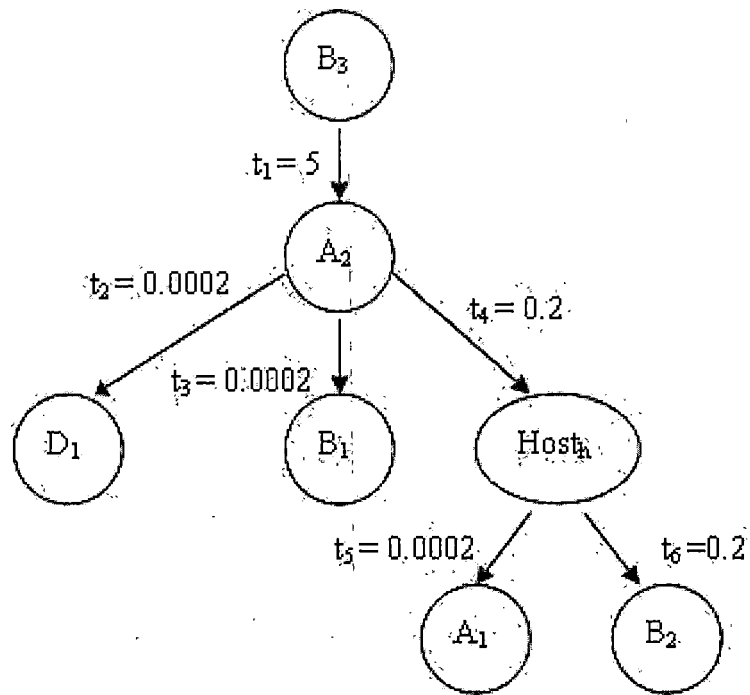


Figure 18. Graph Showing Shortest Paths for B_3 as the
Initial Vertex

Next, let us take D_1 as the initial vertex and try to
find all the shortest paths starting from D_1 .

1. Take the source vertex D_1 and put it in Temp.

$$\text{Temp} = \{D_1\}$$

2. Since D_1 connects to B_1 and B_3 , we mark B_1 and B_3 as candidates.

3. Compare:

$$| D_1B_1 | = 0.02$$

$$| D_1B_3 | = 0.2$$

Because $| D_1B_1 |$ is smaller, we take B_1 into Temp.

Now $\text{Temp} = \{D_1, B_1\}$ and we also get the shortest path between D_1 and B_1 is D_1B_1 with $| D_1B_1 | = 0.02$.

4. Since B_1 is in Temp, now we need to consider the vertices connected to B_1 : A_1 and B_2 . After we mark them, our candidates are A_1 , B_2 and B_3 .

5. Compare:

$$| D_1B_1A_1 | = 5 + 0.02 = 5.02$$

$$| D_1B_1B_2 | = 0.02 + 1 = 1.02$$

$$| D_1B_3 | = 0.2$$

Because $| D_1B_3 |$ is smaller, we take B_3 into Temp.

Now $\text{Temp} = \{D_1, B_1, B_3\}$. The shortest path between D_1 and B_3 is D_1B_3 with $| D_1B_3 | = 0.2$.

6. Now that B_3 is in Temp, we need to consider the vertices connected to it: A_2 . After we mark it, our candidates are A_1 , B_2 and A_2 .

7. Compare:

$$| D_1B_1A_1 | = 0.02 + 5 = 5.02$$

$$| D_1B_1B_2 | = 1 + 0.02 = 1.02$$

$$| D_1B_3A_2 | = 0.2 + 5 = 5.2$$

Because $| D_1B_1B_2 |$ is smaller, we take B_2 into Temp.

Now Temp = $\{D_1, B_1, B_3, B_2\}$ and the shortest path between D_1 and B_2 is $D_1B_1B_2$ with $| D_1B_1B_2 | = 1.02$.

8. Now that B_2 is in Temp, we need to consider the vertices connected to B_2 : A_1 and D_1 . After we mark A_1 , our candidates are A_1 and A_2 .

9. Compare:

$$| D_1B_1A_1 | = 0.02 + 5 = 5.02$$

$$| D_1B_3A_2 | = 0.2 + 5 = 5.2$$

$$| D_1B_1B_2A_1 | = 1 + 50 + 0.02 = 51.02$$

Because $| D_1B_1A_1 |$ is smaller, we take A_1 into Temp.

Now Temp = $\{D_1, B_1, B_3, B_2, A_1\}$ and the shortest path between D_1 and A_1 is $D_1B_1A_1$ with

$$| D_1B_1A_1 | = 5.02.$$

10. Now that A_1 is in Temp, we need to consider the vertices connected to A_1 : $host_h$ and D_1 . Since D_1 has already been in the Temp, so our candidates are A_2 and $host_h$.

11. Compare:

$$| D_1B_3A_2 | = 5 + 0.2 = 5.2$$

$$\begin{aligned} | D_1B_1A_1host_hA_2 | &= 5 + 0.02 + 0.02 + 0.0002 \\ &= 5.0406 \end{aligned}$$

$$| D_1B_3A_2host_h | = 0.2 + 5 + 0.02 = 5.22$$

$$| D_1B_1A_1\text{host}_h | = 0.02 + 5 + 0.02 = 5.04.$$

Because $| D_1B_1A_1\text{host}_h |$ is smaller, we take host_h into Temp. Now $\text{Temp} = \{D_1, B_1, B_3, B_2, A_1, \text{host}_h\}$ and the shortest path between D_1 and host_h is $D_1B_1A_1\text{host}_h$ with $| D_1B_1A_1\text{host}_h | = 5.04$.

12. Now that host_h is in Temp, we need to consider the vertices connected to host_h : A_2, B_3, B_2, A_1 and B_1 . Since most of them have already been in the Temp, so our candidate is only A_2 .

13. Compare:

$$| D_1B_3A_2 | = 5 + 0.2 = 5.2$$

$$\begin{aligned} | D_1B_1A_1\text{host}_hA_2 | &= 5 + 0.02 + 0.02 + 0.0002 \\ &= 5.0406 \end{aligned}$$

Because $| D_1B_1A_1\text{host}_hA_2 |$ is smaller, we take A_2 into Temp. Now $\text{Temp} = \{D_1, B_1, B_3, B_2, A_1, \text{host}_h, A_2\}$ and the shortest path between D_1 and A_2 is $D_1B_1A_1\text{host}_hA_2$ with $| D_1B_1A_1\text{host}_hA_2 | = 5.0406$.

In this way we have found all the shortest paths starting from agent D_1 , and ending to every other vertex. We illustrate those shortest paths in Figure 19.

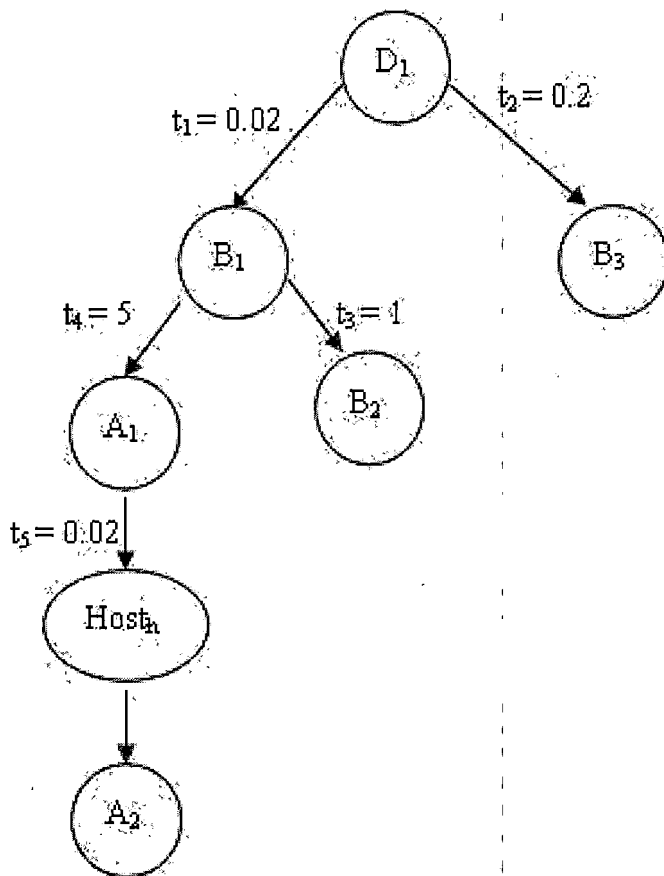


Figure 19. Graph Showing Shortest Paths for D_1 as the Initial Vertex

Finally, we are going to take $host_h$ as the initial vertex and try to find all the shortest paths starting from $host_h$.

1. Take the source vertex $host_h$ and put it in Temp.
Temp = { $host_h$ }
2. Since $host_h$ connects to every agent running on it in this example, we mark all of them as candidates.
3. Compare:

$$| \text{host}_h B_1 | = 0.2$$

$$| \text{host}_h B_2 | = 0.2$$

$$| \text{host}_h B_3 | = 0.02$$

$$| \text{host}_h A_1 | = 0.0002$$

$$| \text{host}_h A_2 | = 0.0002$$

$$| \text{host}_h D_1 | = 0.02$$

Because $| \text{host}_h A_1 | = | \text{host}_h A_2 |$ are the smallest, we take A_1 and A_2 into Temp. Now $\text{Temp} = \{\text{host}_h, A_1, A_2\}$. We also get the shortest path between host_h and A_1 is $\text{host}_h A_1$ with $| \text{host}_h A_1 | = 0.0002$ and the shortest path between host_h and A_2 is $\text{host}_h A_2$ with $| \text{host}_h A_2 | = 0.0002$.

4. Since A_1 and A_2 are in Temp, now we need to consider the vertices connected to A_1 and A_2 : D_1 and B_1 . After we mark them, our candidates are D_1 , B_1 , B_2 and B_3 .

5. Compare:

$$| \text{host}_h B_1 | = 0.2$$

$$| \text{host}_h B_2 | = 0.2$$

$$| \text{host}_h B_3 | = 0.02$$

$$| \text{host}_h A_2 B_1 | = 0.0002 + 0.0002 = 0.0004$$

$$| \text{host}_h A_2 D_1 | = 0.0002 + 0.0002 = 0.0004$$

$$| \text{host}_h A_1 D_1 | = 0.0002 + 0.0002 = 0.0004$$

Because $| \text{host}_h A_2 B_1 | = | \text{host}_h A_2 D_1 | = | \text{host}_h A_1 D_1 |$ are the smallest, we take B_1 and D_1 into Temp. Now $\text{Temp} = \{\text{host}_h, A_1, A_2, D_1, B_1\}$. We also get the shortest path between host_h and B_1 is $\text{host}_h A_2 B_1$ with $| \text{host}_h A_2 B_1 | = 0.0004$ and the shortest path between host_h and D_1 is $\text{host}_h A_2 D_1$ with $| \text{host}_h A_2 D_1 | = 0.0004$ or $\text{host}_h A_1 D_1$ with $| \text{host}_h A_1 D_1 | = 0.0004$.

6. Now that B_1 and D_1 are in Temp, we need to consider the vertices connected to them: A_1, B_2, B_1, B_3 . After we mark some of them, our candidates are B_2 and B_3 .

7. Compare:

$$| \text{host}_h B_2 | = 0.2$$

$$| \text{host}_h B_3 | = 0.02$$

$$| \text{host}_h D_1 B_3 | = 0.02 + 0.2 = 0.22$$

$$| \text{host}_h A_2 B_1 B_2 | = 0.0002 + 0.0002 + 1 = 1.0004$$

$$| \text{host}_h A_1 D_1 B_3 | = 0.0002 + 0.2 = 0.2002$$

Because $| \text{host}_h B_3 |$ is smaller, we take B_3 into Temp. Now $\text{Temp} = \{\text{host}_h, A_1, A_2, D_1, B_1, B_3\}$ and the shortest path between host_h and B_3 is $\text{host}_h B_3$ with $| \text{host}_h B_3 | = 0.02$.

8. Now that B_3 is in Temp, we need to consider the vertices connected to B_3 : A_2 . Since A_2 has already

been in Temp, we do not mark it. So our candidate B_2 .

9. Compare:

$$| \text{host}_h B_2 | = 0.2$$

$$| \text{host}_h A_2 B_1 B_2 | = 0.0002 + 0.0002 + 1 = 1.0004$$

Because $| \text{host}_h B_2 |$ is smaller, we take B_2 into Temp. Now $\text{Temp} = \{ \text{host}_h, A_1, A_2, D_1, B_1, B_3, B_2 \}$ and the shortest path between host_h and B_2 is $\text{host}_h B_2$ with $| \text{host}_h B_2 | = 0.2$.

In this way we have found all the shortest paths starting from host host_h , and ending to every other vertex. We illustrate those shortest paths in Figure 20.

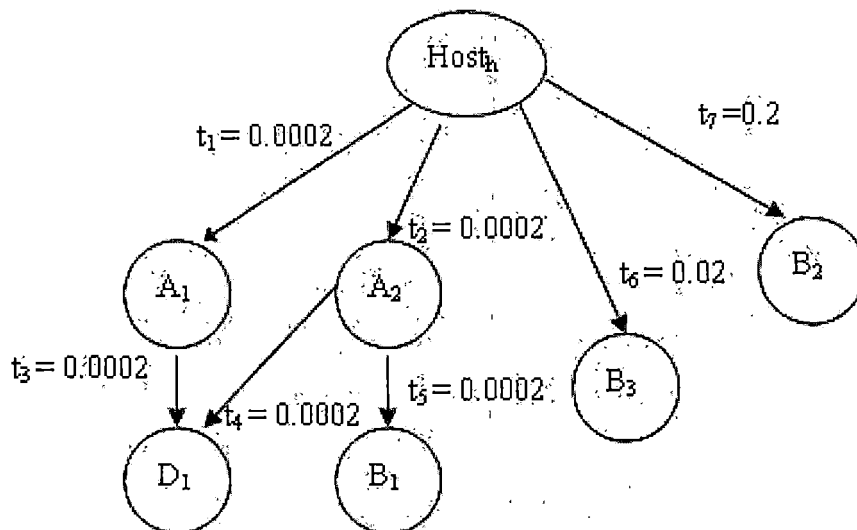


Figure 20. Graph Showing Shortest Paths for host_h as the Initial Vertex

After all these steps, we have used this algorithm to find the shortest paths between every pair of vertices of the entire system.

The shortest minimum length path between any two vertices represents the weakest security point and the longest shortest path describes the ultimate time the attacker needs to break the whole system at most. Here we are more interested in the latter one and we have the following definition.

Definition 14. The diameter of a security risk graph is the length of the longest shortest path between any two vertices.

In the above example, since $| B_3A_2host_nB_2 | = 5.4$ is the longest shortest path for all the vertices. The diameter for this example is 5.4 weeks. The diameter can be used to represent the security level of the whole system because it is the least time the attacker needs to break into the whole system because it is the least time needed for the toughest point in the whole system. Thus we can use the diameters to compare the security of different system. If after reconfiguration, the diameter of the whole system increases, we can say that the whole system's security increases because the time needed to break into the hardest point of the system increases.

CHAPTER FIVE

CONCLUSIONS AND FUTURE DIRECTIONS

5.1 Conclusions

In this research, we have developed security models to evaluate the security levels of the agent-based distributed systems by giving a mathematical measure to tell how secure a system is. First, we presented the overview of the agent-based distributed systems, security evaluation of the distributed systems and the summary of the related works. Then, by identifying the security threats in the mobile-agent distributed systems, we developed a taxonomy of the security threats in the mobile-agent distributed systems. There are four types of security threats identified in the mobile-agent distributed systems from the point of view of the consequence of the security breaches. They are: confidentiality, integrity, availability and creditability. And they are falling into four categories of security threats from the point of view of the relationships between the actors in the agent-based systems: agent-against-host, agent-against-agent, host-against-agent and agent-against-network. To facilitate us analyzing and evaluating the security of the agent-based systems, we

combine the agent-against-host and the agent-against-network scenarios into towards-host category and the agent-against-agent and the host-against-agent scenarios into towards-agent category. We defined the security risk graph as the basis of our research. By using the security risk graph, we can put the security threat relationships between agents and hosts, agents and agents and that of hosts and hosts into a snapshot graph. After we have divided the security threats and have set up the security risk graph to reflect the system's vulnerabilities, we developed a set of theories to analyze and simplify the graph so that we can do further calculation.

By using the simplified security risk graphs, we have set up two models for the security evaluation of the agent-based systems. One is a probabilistic model by using Markov chain. Another one is a mathematical model based on the shortest path.

In the probabilistic model, we calculated the Mean Time To Failure to evaluate the approximate time needed for an intruder to reach the target. As we summarized in the related works section, currently only Chan and Lyu's model [2] deals with agent-based system. Compared with their method, the following characteristics can be observed in our model:

- It is more generalized and complete

The model developed in this paper not only can evaluate how secure an agent is, but also can evaluate the security of the host. While Chan and Lyu's method only considered about the security risk of agent.

- It is more practical and feasible

The coefficient of malice and vulnerability proposed by Chan and Lyu [2] are hard and vague to be obtained. While in our model, we first generate Markov model from the privilege graph. Then the MTTF can be calculated accurately by using the set of well-developed formulas for solving the Markov Chain problem in reliability field.

Our method can be used to dynamically monitor the security level of each host and agent in the system. Combined with the auditing log history technology, each host can decide to accept an agent or not based on the corresponding MTTF value and the credit history of this agent as well as its owner. Also, if we want to test some new technologies to improve the security, we can compare the MTTF before the experiment with that of after the test to see and analyze that if there is any enhancement.

While the Mean Time To Failure can provide a stochastic evaluation of the intruder's performance, a measurement of the whole system's security is also needed from the administrator's view of point. In the mathematical model using the shortest path, the system's administrator not only can evaluate the approximate breach success time (transition time) between any two vertices, but also can evaluate the whole system's security risk. Thus we can have a way to compare the security between different systems.

This work demonstrated the security measurement models that can be used to evaluate the security levels between any two objects in the agent-based distributed systems as well as the whole system's security level. These models can be used to monitor the security evolution of the agents and hosts running in the system dynamically. They can also help the system administrators to manage the system's security and performance. The system administrators can evaluate the effectiveness of different configurations by comparing the values obtained from these different configurations.

5.2 Future Directions

Even though we have achieved the objectives and goals that we aimed in this thesis, there are still some points needed to be addressed for future directions due to its potential practical usefulness.

By monitoring the system's risks, we can get the profile of the transition time of each type of security risks. We plan to use some probabilistic model to process the empirical data obtained from the observation.

Also, it would be desirable to apply some probabilistic method to the time value from the calculation so that it describes the security measure more accurately.

We plan to apply these models in Spider III, the multi-agent distributed system developed in CSUSB to study its feasibility.

REFERENCES

- [1] Brocklehurst, S. and Olovsson, T. , On measurement of Operational Security, IEEE, 1994, Pages 257 - 266.
- [2] Chan, A. and Lyu, M., Security Modeling and Evaluation for Mobile Code Paradigm, In, proceedings of the Asian Computing Science Conference, 1997, Pages 371 - 371.
- [3] Concepcion, A. et. al., Spider: A Multi-Agent Architecture for Internet Distributed Computing System, In proceedings of the ISCA 15th International Conference on Parallel and Distributed Computing System, September, 2002.
- [4] Concepcion, A. and Ma, C., A Probabilistic Security Model for Multi-Agent Distributed Systems, In proceedings of the 6th International Conference on Business Information Systems, June 2003.
- [5] Cramer, R. and Shoup, V., Signature schemes based on the strong RSA assumption, ACM Transactions on Information and System Security, Vol. 3, No. 3, August 2000, Pages 161 - 185.
- [6] Dacier, M. et. al., Quantitative Assessment of Operational Security: Models and Tools, LAAS Research Report, 96493, May 1996.
- [7] Fong, P., Viewer's discretion: Host security in mobile code systems, <ftp://fas.sfu.ca/pub/cs/techreports/1998>, November1998.
- [8] Gattiker, U., The Information Security Dictionary, ISBN: 1-4020-7927-3, 2004.
- [9] Gray, R. et. al., D'Agents: Security in a multiple-language, mobile-agent System, Lecture Notes in Computer Science on "Mobile Agents and Security", 1419, 1998, Pages 154 - 187.
- [10] Hohl, F., Time limited blackbox security: Protecting mobile agents from malicious hosts, National Institute of Standards and Technology, 1999.

- [11] Humphries, J. et. al., Secure mobile agent for network vulnerability scanning, Proceedings of the 2000 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, 6-7 June, 2000.
- [12] Jansen, W. and Karygiannis T., Mobile agent security, NIST Special Publication, 800-19, October 1999.
- [13] Jonsson, E. and Olovsson, T., A Quantitative Model of the Security Intrusion Process Based on Attacker Behavior, IEEE Transactions on Software Engineering, Vol. 23, No. 4, April 1997.
- [14] Kaeo, M., Designing network security, Macmillan Technical Publishing, 1999.
- [15] Karnik, N. and Tripathi A., Design issues in mobile agent programming systems, Department of Computer Science, University of Minnesota, June 1998.
- [16] Karjoth, G. et. al., A security model for aglets, IEEE Internet Computing, July-August, August 1997, Pages 68 - 77.
- [17] Ortalo, R. et. al., Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security, LAAS Research Report, 1997.
- [18] Rubin, A. and Geer D., Mobile code security, IEEE Internet, November- December, December 1998, Pages 30 - 34.
- [19] Sander, T. and Tschudin, C., Towards mobile cryptography, International Computer Science Institute, November 1997.
- [20] Sander, T. and Tschudin, C., Protecting mobile agents against malicious hosts, Lecture Notes in Computer Science on "Mobile Agents and Security", 1419, 1998, Pages 44 - 60.
- [21] SANS: The Trusted Source for Computer Security Training, Certification and Research
<http://www.sans.org/resources/glossary.php>

- [22] Schneider, F., Security in Tacoma Too, In proceedings of the 1997 DAGSTUHL Workshop on Mobile Agents, September 1997.
- [23] Stajano, F. and Anderson, R., The Resurrecting Duckling: Security Issues for Ubiquitous Computing, first Security & Privacy supplement to IEEE Computer, April 2002.
- [24] Vitek, J. and Bryce, C., Secure mobile code: The JAVASEAL experiment, University of Geneva, 1999.
- [25] Yee, B., A sanctuary for mobile agents, Proceedings of the DARPA workshop on foundations for secure mobile code, Monterey CA, USA, March 1997.
- [26] Ylitalo, J., Secure platforms for mobile agents, <http://www.hut.fi/~jylitalo/seminar99>, January 2000.