Summer 2006

Stochastic propagation modeling and early detection of malicious mobile code

Xin Xu
Louisiana Tech University

STOCHASTIC PROPAGATION MODELING AND EARLY DETECTION OF MALICIOUS MOBILE CODE

By

Xin Xu, M.S.

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

August 2006

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

May 17, 2006
Date

We hereby recommend that the thesis prepared under our supervision

by Xin Xu

entitled

Stochastic Propagation Modeling and Early Detection of Malicious Mobile Code

be accepted in partial fulfillment of the requirements for the Degree of

Doctor of Philosophy in Computational Analysis and Modeling

[illegible]
Supervisor of Thesis Research

Richard [illegible]
Head of Department

CAM
Department

Recommendation concurred in:

[illegible]

[illegible]

[illegible]

Advisory Committee

Approved:
[illegible]
Director of Graduate Studies

[illegible]
Dean of the College

Approved:
[illegible]
Dean of the Graduate School

GS Form 13
(5/03)

ABSTRACT

Epidemic models are commonly used to model the propagation of malicious mobile code like a computer virus or a worm. In this dissertation, we introduce stochastic techniques to describe the propagation behavior of malicious mobile code. We propose a stochastic infection-immunization (INIM) model based on the standard Susceptible-Infected-Removed (SIR) epidemic model, and we get an explicit solution of this model using probability generating function (pgf.). Our experiments simulate the propagation of malicious mobile code with immunization. The simulation results match the theoretical results of the model, which indicates that it is reliable to use INIM model to predict the propagation of malicious mobile code at the early infection stage when immunization factor is considered.

In this dissertation, we also propose a control system that could automatically detect and mitigate the propagation of malicious mobile programs at the early infection stage. The detection method is based on the observation that a worm always opens as many connections as possible in order to propagate as fast as possible. To develop the detection algorithm, we extend the traditional statistical process control technique by adding a sliding window. We do the experiment to demonstrate the training process and testing process of a control system using both real and simulation data set. The experiment results show that the control system detects the propagation of malicious mobile code with zero false negative rate and less than 6% false positive rate. Moreover, we introduce risk analysis using Sequential Probability Ratio Test (SPRT) to limit the false positive rate. Examples of risk control using SPTR are presented. Furthermore, we analyze the network behavior using the propagation models we developed to evaluate the effect of the control system in a network environment. The theoretical analysis of the model shows that the propagation of malicious program is reduced when hosts in a network applied the control system. To verify the theoretical result, we also develop the experiment to simulate the propagation process in a network. The experiment results match the mathematical results.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author [illegible]

Date 7-27-2006

GS Form 14
(5/03)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF NOTATIONS

B	Number of contactors a host has
B_1	Limited number of contactors a host could have when infection detected
B_2	Number of contactors a host could have when infection not detected
d	Detection rate of the control system
f	Updating frequency of the upper control limit
$I(t)$	Number of Infected hosts at time t
I_1	Number of infected hosts whose connection rated is limited, $I_1 = I \cdot p \cdot d$
I_2	Number of infected hosts whose connection rate is not limited, $I_2 = I \cdot (1-p) + I \cdot p \cdot (1-d) = I - I_1$
$i(t)$	Proportion of infected hosts, $i(t) = I(t)/N$
i_1	Fraction of infected hosts whose connection rate is limited, $i_1 = I_1 / N$,
i_2	Fraction of infected hosts whose connection rate is not limited, $i_2 = I_2 / N$
k	Infection rate, $k = B\beta$
k_1	Infection rate with control system, $k_1 = \beta \cdot B_1$
k_2	Infection rate without control system $k_2 = \beta \cdot B_2$
$M(t)$	Number of immunized hosts at time t
N	Size of a population; total number of hosts in a network

p	Percentage of hosts that installed the control system in a network
$R(t)$	Number of removed hosts at time t
$r(t)$	Proportion of removed hosts, $r(t)= R(t)/N$
$S(t)$	Number of susceptible hosts at time t
$s(t)$	Proportion of susceptible hosts, $s(t) = S(t)/N$
t_1	The moment that immunization method of certain malicious code is available
w	Size of the sliding window
α	Immunization rate of healthy machine
β	Pair wise infection rate
γ	Immunization rate of infected machine

ACKNOWLEDGEMENTS

I would like to thank Dr. Vir Phoha, principle advisor, for his guidance, encouragement and generous support throughout my study at Louisiana Tech University.

I would also like to thank my committee members, Dr. Raja Nassar, Dr. Weizhong Dai, and Dr. Andrea Paun for their generous help, valuable advice and guidance.

Thank you Dr. Greechie for providing me the opportunity to study in CAM program and always being so supportive.

Thank you Weihua Song for the valuable discussion about the research, and the support she gave me.

Special thanks for my parents. They have been patient, supportive and considerate for all the years. Without their love and support, I could never make what I have done so far. Also, I want to thank my cousin, Xi Xu, from the bottom of my heart. She is always so considerate and always gives as much help as she could.

CHAPTER 1

INTRODUCTION

1.1 Computer Security

The security of a computer system or a network system includes the *confidentiality*, *integrity* and *assurance* of the system. The definitions of *confidentiality*, *integrity* and *assurance* are as follows [Phoha 2002]:

Confidentiality: "The property of not being divulged to the unauthorized parties. A confidentiality service assists in the prevention of disclosure of information to unauthorized parties."

Integrity: "a condition in which data or a system itself has not been modified or corrupted without authorization."

Assurance: "ensuring the availability, integrity, authentication, confidentiality and non-repudiation of information and information systems by incorporating protection, detection and reaction capabilities to restore information systems."

An intrusion is a series of malicious activities that attempts to comprise the security of a computer or a network system [Ye 2000]. An intrusion detection system analyzes the activities performed in a computer or network to look for evidence of malicious behavior. There are two categories of intrusion detection method. One is anomaly detection, and the other is misuse detection. Intrusion detection systems using the anomaly detection method generally build profiles for normal activities, and identify system activities which vary from the established profile as intrusion attempts [Eckmann 2002]. An intrusion detection system using misuse detection technique builds a profile with signatures of known attacks and compares current activities with those signatures. An intrusion is signaled when there is a match between current activities and the profiled intrusion activities.

Both anomaly detection and misuse detection have advantages and disadvantages. Anomaly detection is able to detect unknown attacks, but there is always a trade off between false negative and false positive. False negative is defined as "events that are not flagged intrusive, although they actually are," and false positive is defined as "anomalous activities that are not intrusive but are flagged as intrusive [Denning 1990]." Anomaly detection systems are also computationally expensive because we need to keep track of, and update all system profiles. Misuse detection has a relatively low false positive rate, but it cannot detect unknown attacks.

Intrusions exploit the flaws of the system architecture, the operating system, the server system, or the other software systems. A complete secure system is not really feasible because designing and implementing a totally secure system is an extremely difficult task. Flaws in the programs and operating systems are prevalent [Miller 1995]. In practice, it is not possible to build a completely secure system.

The financial loss caused by the intrusion of malicious programs accounts for a large amount of losses caused by computer security problems [Usa 2001]. The following section gives a brief view of malicious mobile code.

1.2 Malicious Mobile Code

This section talks about malicious mobile code. We will discuss the current defense methods against malicious code and the general idea of mathematical modeling of malicious code propagation.

1.2.1 What Is Malicious Mobile Code?

A malicious mobile code is a software program intentionally designed to move from computer to computer or from network to network and modify the system without the consent of the user [Grimes 2001]. Major types of malicious mobile code include viruses, worms, Trojans, and rogue Internet content. The first malicious program is a computer virus developed by Fred Cohen [Cohen 1985, 1987] for research purpose. In the early 1980s, Cohen did extensive theoretical research as well as setting up and performing numerous practical experiments regarding viral type programs. Cohen's definition [Cohen 1994] of a computer virus is "a program that can 'infect' other programs by modifying them to include a version of itself." This definition has been generally accepted as a standard definition of a computer virus [Fites 1992] [Levin 1990].

Worms are very similar to viruses in that they are computer programs that replicate themselves and often, but not always contain some malicious functions that will disrupt the normal use of a computer system or a network system [Grimes 2001] [Denning 1990] [Levin 1990]. Unlike viruses, worms exist as separate entities; they do not attach themselves to other files or programs. Worms were first noticed as a potential computer security threat when *Christmas Tree* [Denning 1990] attacked IBM mainframes in December 1987. *Christmas Tree* is an executable file attached in an e-mail. Once executed, it displays a Christmas tree and sends a copy to everyone in the victim's address list. Someone argued that *Christmas Tree* is not a true computer worm program but just a Trojan program with replicating mechanism [Denning 1990] [NISTIR4939]. The first computer worm, *Morris*, was released on November 2, 1988 [Spafford 1989a] [Spafford 1989b]. It utilized the TCP/IP protocols, common application layer protocols, operating system bugs, and a variety of system administration flaws to propagate. *Morris* infected approximately three thousand computers during eight hours of activity [Spafford 1989a].

Malicious programs are created by exploiting the flaws of the system. Since it is impossible to design a perfect system, there is always a possibility for new malicious codes to be designed. The malicious function of these programs might be different, but they usually have similar infection strategy. For instance, one class of worm programs always try to connect to as many hosts as possible so that they can be distributed easily and quickly through the network. The increasing connectivity of network and the growing use of computers have led to more and more concerns about security problems caused by malicious mobile codes like worms. In the past few years, the fast spreading malicious mobile codes have disrupted tens of thousands of businesses and homes worldwide and caused millions of dollars in loss [Usa 2001]. Famous ones include Code Red [CERT01-19] and Nimda [CERT01-26] in 2001, SQL Slammer [CERT03-04], Blaster [CERT03-20] and Welchia [SYMANTEC03] in 2003, and Netsky [CERT04-02] and Sasser [CERT04-05] in 2004.

1.2.2 Defense against Malicious Mobile Code

Anti-virus tools are now installed on almost all computers to detect and prevent the spread of such programs. Common techniques applied by these tools are activity monitors, integrity management systems and virus scanners [Kumar 1992]. Activity monitors alert users about system activity that is commonly associated with viruses. Integrity management system warns the user of suspicious changes that have been made to files. These two methods are quite generic, and can be used to detect unknown viruses in the system. The drawback of these two methods is that they often flag or prevent legitimate activities, and hence, disrupt normal work. As a consequence, the user may ignore their warnings altogether. Virus scanning is the most commonly used method for anti-virus tools because it is the most simple, economical way for virus detection. Virus scanners search files, boot records, memory and other locations where executable codes can be stored for characteristic byte patterns that occur in one or more known viruses. The drawback is that virus scanners rely on the priori knowledge of the viral code, which means they can only detect previously known viruses, but not new viruses. Thus, the scanner has to be updated frequently [Forrest 1994] [Kumar 1992] [Wang 2000]. [Xu 2002] [Phoha 2003] and [Xu 2004] introduce a novel approach to control the spread of virus and presents a technique to make them ineffective which is a complement of current virus detection techniques. This approach models the process as a discrete event system such that supervisory control theory can be applied to control the reproduction and propagation of the malicious code. The drawback is that this approach is only effective for executable files whose execution process could be modeled as a discrete event system.

The detection of worm programs is still an open problem [White 1998], especially for unknown worms. Currently, the most general way is to embed the worm detection component in the anti-virus tools or intrusion detection systems. The methodology is similar as those used in the virus detection tools. In other words, use known signatures to catch the known worms. For unknown worms, we do not have a general solution yet.

1.2.3 Epidemic Modeling of Malicious Code

The epidemic modeling of biological viruses and their dissemination has a history of about three hundred years [Andersson 2000] [Daley 2001]. Daniel Bernoullli presented the first theoretical approach about the effects of the disease in 1760 [Daley 2001]. At that time, the smallpox was widespread in Europe and affected a large proportion of the population. In the early twentieth century, Ross and Hudson, Scoper, as well as Kermack and Mckendrick, began to provide a firm theoretical framework for the investigation of the infectious diseases [Anderson 1992]. The mathematical models they provided help to understand the mechanism by which diseases spread to predict the future spreading of the epidemic and to control the spread of the diseases.

Epidemic modeling of malicious code has become a popular research topic for computer scientists since computer worm *Morris* was released in 1988 [Spafford 1989a] [White 1998]. Propagation modeling helps us to understand the life cycle and fast propagation nature of such malicious mobile codes. It also helps us understand the impact of countermeasures [Chen 2004] [Serazzi 2003], network traffic, and network topology [Satorras 2001][Satorras 2002]. The propagation models of malicious code are extensions of the classic epidemic models [Zou 2003] [Zou 2002] [Kephart 1991] [Boguna 2002] [Staniford 2002] [Wang 2003] [Chen 2003]. This dissertation will give the general classic epidemic models and the related works of malicious code propagation modeling in Chapter 2.

1.3 The Contributions of This Dissertation

This dissertation first models the propagation of the malicious code using stochastic technique, then proposes a control system that automatically detects and mitigates the propagation of such a malicious code. The control system works as a complement to the current intrusion detection systems. The detection method of the control system belongs to the anomaly detection method. We analyze the normal connection behavior of a host and compare the current connection behavior with the normal behavior to identify the anomaly. The simulation experiment results match the theoretical results.

1.3.1 Propagation Modeling of Malicious Code

This dissertation introduces stochastic techniques to model the propagation of malicious mobile code. We build a stochastic propagation model that considers the factors of recovering and immunization. This model gives the probability that an infection will or will not happen instead of a deterministic yes-or-no answer that relies on the law of large numbers. This model also allows probabilistic analysis of the virus and propagation phenomenon. It is more precise than the deterministic method when we study the infection scale and speed inside a community or an organization with varying population size.

1.3.2 Early Detection and Propagation Mitigation of Malicious Code

This dissertation proposes a control system to detect the propagation of malicious code at the early infection stage. It also mitigates the propagation of malicious code over the network so that the overall damage to our society could be reduced. It is novel to apply the statistic process control technique to detect the malicious code. The general steps of building a control system are given and the details of each step are presented in simulation experiments. The framework of building the control system can be easily extended and applied to other scenarios.

1.3.3 Risk Control and Network Performance Analysis

This dissertation also introduced Sequential Probability Ratio Test (SPRT) to control the false positive rate of the control system. Examples of risk control using SPRT are presented using simulation data. To analyze the network effect of the control system, we give a quantitative analysis of propagation mitigation. We build propagation models to describe the propagation behavior of malicious code inside a network with and without the control system. Mathematical analysis of both models shows a significant difference in propagation speed and scale when the control system is applied on every machine in the network. To verify the theoretical results, we simulate malicious code propagation on virtual network using computer programs. All simulation results match the theoretical results from the propagation model.

1.4 The Organization of This Dissertation

The rest of the dissertation is organized as follows. Chapter 2 gives the background information and related research in both propagation modeling of malicious mobile code using epidemic models and the early detection of malicious code. Chapter 3 introduces the stochastic method to describe the propagation models and proposes the Infection-Immunization (INIM) model using the stochastic techniques. Chapter 4 gives the framework of building a control system to detect and mitigate the malicious code, and then shows the experimental details of the training process and testing process of the control system. The testing results are presented in Chapter 4. Chapter 4 also presents the effect of the control system by analyzing the network performance. Both theoretical results and simulation results are presented. Chapter 5 is the conclusion and suggestions for future work.

CHAPTER 2

BACKGROUND AND RELATED RESEARCH

This dissertation first uses stochastic techniques to model the propagation of malicious mobile code, and then proposes a control system to detect and mitigate the propagation of malicious mobile code at the early infection stage. The detection algorithm of the control system uses extended Process Control technique. This chapter provides background information of epidemic modeling, statistical Process Control, and related research in areas of malicious code propagation modeling as well as early detection of malicious code.

2.1 Epidemic Models

The mathematical modeling of diseases and their propagation has a history of about three hundred years [Daley 2001]. Epidemic modeling has three main aims [Daley 2001]. The first is to understand the mechanism by which diseases spread. The second aim is to predict the future course of the epidemic. The third aim is to understand how we may control the spread of the epidemic. A good epidemic model captures the essential features of the epidemic, makes reasonable predictions, and evaluates the effect of control method. The following two subsections give a brief review of the early mathematical models for the spread of infectious diseases. Readers who want a more detailed overview are referred to [Bailey 1975] and [Anderson 1992].

2.1.1 Deterministic Modeling

In this section, we introduce two classical deterministic epidemic models, the Susceptible Infected (SI) model and Susceptible, Infected and Removed (SIR) model. Both models assume that the population is homogeneously mixed. If we consider each individual as a vertex in a graph, from the graph theory point of view, a homogeneously mixed population means a fully connected graph.

2.1.1.1 Deterministic SI model

In this model, each host stays in one of the two states: susceptible (S) or infectious (I). SI model assumes that once a host is infected, it becomes infectious and it will never become susceptible again. The only state transition is: S → I (see Figure 2.1).

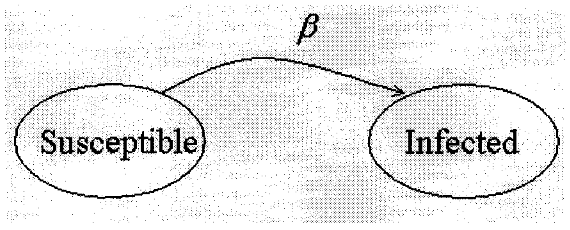


Figure 2.1: State transition of SI model.

N is the size of the population;

S(t) = *S* is the number of susceptible hosts at time t,

I(t) = *I* is the number of infected hosts at time t.

β is the pair wise infection rate.

At any time *t*, we have *S(t)* + *I(t)* = *N.*

Using Ordinary Differential Equation, we have

$$\frac{dI(t)}{dt} = \beta S(t)I(t) \qquad 2.1$$

which is the same as

$$\frac{dI(t)}{dt} = \beta(N - I(t))I(t) \qquad 2.2$$

Let $k = \beta N$, $i(t) = I(t)/N$, Equation 2.2 becomes

$$\frac{di(t)}{dt} = k(1 - i(t))i(t)$$

$$\Rightarrow \qquad i(t) = \frac{e^{kt}}{e^{kt} - c}$$

Let $I(0) = I_0$, so $i(0) = I_0/N$, we get $c = \frac{I_0 - N}{I_0}$. Plugging c into the above equation,

we get

$$i(t) = \frac{I_0 e^{kt}}{(N - I_0) + I_0 e^{kt}},$$

which is the same as

$$I(t) = \frac{I_0 N}{I_0 + (N - I_0)e^{-kt}} \qquad 2.3$$

Figure 2.2 shows the infection evolution process given N = 10000, $\beta = 1/N$ $I_0 = 1$.

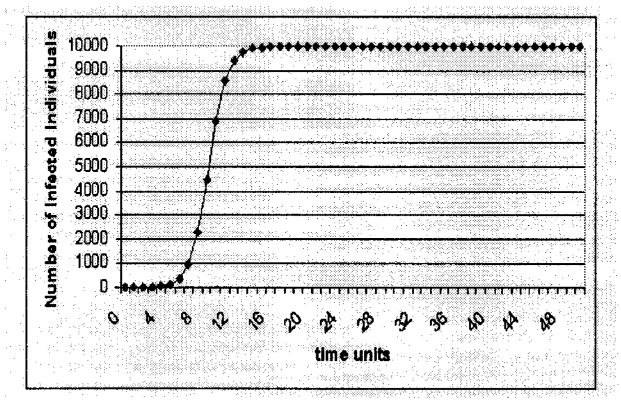


Figure 2.2: Infection evolution of SI Model.

Figure 2.2 illustrates that the infection evolution procedure can be roughly divided into three stages: the slow starting stage, the fast spreading stage, and the saturating stage. Let *i(t)* = *I(t)/N*, at the beginning, when $i(t) \to 0$, the number of infectious hosts grows almost exponentially.

From Equation 2.3, time t needed to infected I individuals of the whole population is derived as

$$t = \frac{1}{\beta N} \ln \frac{(N - I_0)I}{(N - I)I_0} \qquad 2.4$$

Table 2.1 presents the time steps needed to infect certain proportion of the population with a different number of initially infected individual I_0. The time step values in all the three tables (Table 2.1, Table 2.2, and Table 2.3) are calculated using Equation 2.4.

Table 2.1: Time steps needed to infect I individuals when I_0 varies (N = 10000, $\beta = 0.0001$)

I_0 \ I	100	1000	5000	9999
1	4.61	6.91	9.21	18.42
10	2.31	4.71	6.91	16.12
100	—	2.40	4.59	13.81

Table 2.1 shows that as the initial number of infected nodes I_0 increases, the time steps needed to infect the population decrease dramatically at the beginning. For example, the time steps needed to infect 1,000 individuals in a population of 10,000 when I_0 is 100 is almost one third of the time steps needed when I_0 is 1. But the time steps needed to infect the whole population does not change that much as we can see from

Table 2.1. Table 2.2 presents the time steps needed to infect a certain proportion of the population when population size *N* varies.

Table 2.2: Time steps needed to infect certain percentage of *N* when *N* varies
$(I_0 = 1,\ \beta = 1/N)$

N \ *I*	*N/100*	*N/10*	*N/2*	*N-1*
1000	2.31	4.71	6.91	13.81
10000	4.61	7.01	9.21	18.42
100000	6.92	9.31	11.80	23.02

Table 2.2 shows that when the population size increases, the time steps needed to infect the same proportion of the population also increase, and they increase faster at the beginning (*I* = *N/100*) than near the end (*I* = *N/2*). From Table 2.2, we know that when population N increases, the slow starting stage gets longer. Table 2.3 gives the time steps needed to infect a certain number of individuals when *N* varies.

Table 2.3: Time steps needed to infect *I* individuals when *N* varies
$(I_0 = 1,\ \beta = 1/N)$

N \ *I*	5	10	100	500	999
1000	1.61	2.31	4.71	6.91	13.81
10000	1.61	2.30	4.62	6.27	7.01
100000	1.61	2.30	4.61	6.22	6.92

From Table 2.3, we observe that the infection is slow at the very beginning. It infects less than 10 individuals during the first two time steps. Then suddenly it increases almost exponentially. During the next two time steps, more than 50 individuals are infected. The infection slows down when the infection comes to the third stage. It takes almost the same time steps to infect a certain number of individuals (see Table 2.3), but it takes more time steps to infect the whole population if the population size is larger (see the last column of Table 2.2).

2.1.1.2 Deterministic SIR model

The first complete mathematical model for the propagation of infectious diseases was a deterministic model given by Kermack and McKendrick in 1927 [Dailey 2001]. In this model, each host is in one of the three states: susceptible (S), infectious (I), or removed (R). This model assumes that once a host is infected, it will recover or die in the end. Whether dead or recovered, it will never be susceptible to the same disease; therefore, it will stay in the removed state forever. So the state transitions of the SIR model is: S→I →R (see Figure 2.3).

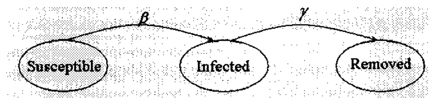


Figure 2.3: State transition of SIR model.

If we let *R(t)* represent the number of removed hosts at time t, then we have $N = S(t) + I(t) + R(t)$. Let γ represent the removal rate. In a homogeneously mixed community, we have

$$\begin{cases} \dfrac{dS(t)}{dt} = -\beta S(t)I(t) \\ \dfrac{dI(t)}{dt} = \beta S(t)I(t) - \gamma I(t) \\ \dfrac{dR(t)}{dt} = \gamma I(t) \end{cases}$$

with initial conditions $S(0) = S_0$, $I(0) = I_0$, and $R(0) = 0$.

The Kermack-McKendrick model improves the SI model by considering that the infectious hosts may recover or die after some time. But this model does not consider the immunization of the susceptible hosts while immunization has become a very popular way, not only to prevent the outbreak of the infectious diseases, but also to prevent the outbreak of the infectious malicious code.

2.1.2 Stochastic Modeling

Stochastic modeling has been used to model the growth of population, the price changing of the stock market, the queuing process, etc. This section gives a brief introduction to stochastic process and stochastic epidemic modeling of diseases.

2.1.2.1 Introduction to stochastic process

A stochastic process is a family of random variables *X(t)* describing an empirical process whose development is governed by probability laws [Chiang 1980]. The time parameter *t* could be either discrete or continuous. In diffusion processes, both *X(t)* and *t* are continuous variables, while in Markov chains, *X(t)* and *t* take discrete values. The main interest is the probability distribution $p_k(t) = \Pr\{X(t) = k\}, \quad k = 0, 1, 2, 3 \cdots$.

2.1.2.2 Stochastic epidemic modeling

A simple stochastic epidemic modeling assumes that the population consists of only susceptible individuals and infective individuals. Once a susceptible individual is infected, it becomes infective and stays at the infected state forever. Let random variable *X(t)* = *S(t)*, and *Y(t)* = *I(t)*, recall that *S(t)*+*I(t)* = *N*, so *X(t)*+*Y(t)* = *N*, then the only transition from t to $t + \Delta$ is (*S*, *I*) to ($S - I$, $I + 1$) with probability β.

We have $\Pr\{(X,Y)(t+\Delta)=(S-1,I+1)\mid(X,Y)(t)=(S,I)\}=\beta SI\Delta+o(\Delta)$,

and $\Pr\{(X,Y)(t+\Delta)=(S,I)\mid(X,Y)(t)=(S,I)\}=1-\beta SI\Delta-o(\Delta)$

where $o(\Delta)$ represents the higher order function of Δ such that $o(\Delta)/\Delta\to 0$ when $\Delta\to 0$.

Then, the forward Kolmogorov equation system for the state probability $p_k(t)=\Pr\{X(t)=S\mid X(0)=N-1\}$ is

$$\frac{dp_N}{dt}=-\beta(N-I)Ip_N(t)$$

$$\frac{dp_i}{dt}=-\beta i(N-I)p_i(t)+\beta(I+1)(N-I-1)p_{i+1}(t)$$

$$\frac{dp_0}{dt}=-\beta(N-1)Ip_1(t)$$

In matrix form, for $P(t)=(p_N(t),p_{N-1}(t),\cdots,p_0(t))^T$, we have

$$\frac{P(t)}{dt}=-\beta\begin{bmatrix}(N-I)I & 0 & 0 & \cdots & 0\\ -(N-I)I & (N-I-1)(I+1) & 0 & \cdots & 0\\ 0 & -(N-I-1)(I+1) & (N-2)(I+2) & \cdots & 0\\ \vdots & \vdots & \ddots & \ddots & \vdots\\ 0 & 0 & \cdots & -1(N-1) & 0\end{bmatrix}\begin{bmatrix}p_N(t)\\ p_{N-1}(t)\\ p_{N-2}(t)\\ \vdots\\ p_0(t)\end{bmatrix}$$

$$=-\beta AP(t)$$

We can use the matrix analysis [Dailey 2001] to solve the equations, and the solution is known to be $P(t)=e^{-\beta At}p(0)$. But the explicit result is not easy to be derived with this method. We may use Laplace transform or probability generation function or a mixture of both to obtain the explicit solution. This dissertation uses the probability generation function (pgf) to get the probability that I individuals are infected at time t. We will introduce pgf in Chapter 3 and show the details of using pgf technique to solve the stochastic propagation model of malicious mobile code.

2.2 Literature Review of Malicious Mobile Code Propagation Modeling

A reliable propagation model helps us to understand the life cycle of a self-replicating program, to predict the propagation scale and speed, and to estimate the effect of factors like network topology, network traffic, and countermeasure techniques [Chen 2004] [Serazzi 2003] [Satorras 2001].

Kepart and White [Kephart 1991] built a SIS (Susceptible-Infected-Susceptible) model to model the virus propagation, and use deterministic Ordinary Differential Equations (ODEs) to approximate the SIS model. They also present hierarchical model and spatial model in [Kephart 1991]. Later, they introduced the *Kill* signal as a countermeasure to reduce the spreading of computer virus and build a model for virus propagation with the *Kill* signals and concluded that the *Kill* signal is effective in reducing the spread of the virus [Kephart 1993]. Staniford *et al.* [Staniford 2002] constructed deterministic SI (Susceptible-Infected) model based on the empirical data from the outbreak of the Code Red worm. Serazzi and Zanero [Serazzi 2003] surveyed the existing models for virus and worm and came out with a compartment-based model that deals with the propagation inside and outside of an Autonomous System (a sub-network administered by a single authority). Zou *et al.* [Zou 2002] gave a model for Code Red Worm propagation based on the classical SIR (Susceptible-Infected-Removed) model. They introduced two factors that might affect the worm propagation; that is, the countermeasure effect and decreased infection rate because of Internet congestions caused by the worm. Ramualado Pastor-Satorras *et al.* studied the effects of network topology on epidemic models [Boguna 2002] [Satorras 2001] [Satorras 2002].

All the above models use a deterministic approach to represent the models; that is, the models are described by a system of Ordinary Differential Equations (ODEs) except [Kephart 1991]. [Kephart 1991] gave a linear birth and death process when discussing the expected lifetime of the infection.

Under the homogeneous assumption, every individual in the population is assumed to be equally likely to infect or to be infected by every other individual. This approximation works well when each individual has many randomized contacts with others. However, if the number of contacts that a typical individual has with others is fairly small and/or the pattern of contacts is more or less localized, the homogeneous approximation fails. We suspect that the majority of today's computer populations are characterized by a degree of sparsity and locality that invalidates the homogeneous mixing approximation. In this dissertation, we introduced a factor *B*, the average number of contactors an individual could have, into the epidemic models.

Although ODEs can be safely used to approximate a stochastic process when the population size is large, it is more accurate to use stochastic models when the population size varies. Moreover, the spread of infectious disease or malicious programs is actually stochastic [Zou 2003] [Daley 2001] [Andersson 2000], so it is natural to model it with the stochastic model. The stochastic model gives the probability that an event will happen instead of deterministic yes-or-no answer relying on the law of large numbers [Andersson 2000]. Actually, when the population size is big, it shows that the deterministic model is the convergence of a stochastic model. We believe both models are important to understand the propagation of a malicious mobile program like a worm or a virus.

Andersson and Britton [Andersson 2000] concluded that stochastic models are preferred when their analysis is possible; otherwise, the deterministic model should be used.

This dissertation focuses on the stochastic propagation characteristics of malicious mobile programs; thus, we use stochastic models to describe the spreading of a malicious program over the Internet. The benefits of the stochastic model are: (1) It gives the probability of whether or not an infection will happen instead of a deterministic yes-or-no answer relying on the law of large numbers; (2) It allows probabilistic analysis of the malicious code propagation phenomenon; (3) It is more precise than the deterministic method when we study the infection process inside a community or organization where the population size varies; and (4) We could further derive the waiting time for the occurrence of the *k*th infection based on the stochastic model.

2.3 Literature Review of Early Detection of Malicious Mobile Code

The propagation speed of malicious code has increased dramatically in recent years. As we pointed out before, a malicious code spreads almost exponentially at the early infectious stage when there is no counter action taken, so we need to respond automatically before it is identified. Cliff Zou [Zou 2003] proposed an early detection system by monitoring the illegitimate network traffic. A *Kalman* filter is used to detect the presence of a worm by detecting the trend. When the monitoring system encounters a surge of illegitimated network traffic, the *Kalman* filter is activated. The traffic is claimed to be caused by worm propagation when the estimated infection rate stabilized and oscillated a little bit around a constant positive value. One disadvantage of the approach is that the machine will be quarantined when illegitimated traffic is detected. This may irritate the users when false alarm happens. Users tend to disable the intrusion detection system even when the false alarm rate is not very high.

[Williamson 2002] proposed a filter algorithm based on the observation of connection behavior. Evidence from [Heberlein 1990] and [Hofmeyr 1999] showed that during virus propagation, an infected machine will connect to as many machines as possible in order to spread as fast as possible. The idea of the filter algorithm is to use a series of timeouts to restrict the rate of connections to the new hosts; any traffic that attempts to connect at a higher rate is delayed. The filtering mechanism is user transparent, which means a user cannot take active actions to remove the malicious code and fix the system flaws the malicious program exploits.

The control system we proposed here is based on the same observation as in [Williamson 2002]. We applied statistical Process Control technique to automatically detect and mitigate the propagation of the malicious code. The advantages of this approach are: (1) The detection delay is small; the propagation of a malicious code can be detected as the very first beginning; (2) An explicit message will be given when anomaly connection behavior is detected so that a user could take active counteractions to fight for the malicious program; (3) The propagation rate of the malicious code is reduced automatically so that the overall damage is reduced; and (4) The control system does not disconnect the machine from the Internet so that users will not feel annoyed caused by the false alarms. Furthermore, we could give a mathematical estimation of the propagation mitigation scale using mathematical propagation models. Besides, the hypothesis test underlying the detection algorithm gives quantitative evaluation of the false alarm rate.

2.4 Statistical Process Control

2.4.1 A Brief Review of Process Control

The use of statistical method in process control began at the Bell Lab in the 1920s [Hansen 1987]. It has been widely used in the industry to manage, monitor and control the production quality [Hansen 1987]. The basic idea of Process Control is collecting and analyzing the past data, and comparing new data with past data to identify process violations.

Figure 2.4 gives the general outline of statistical Process Control. To apply Process Control techniques, we assume that the sample data follows normal distribution. Thus, when a process is in control, we know (1) about 68% of the plotted points lie in one standard deviation of the central line and 34% at each side; (2) about 13.5% of the plotted points lie in between one and two standard deviations on both sides of the central line; and (3) about 2.5% of the plotted points lie in between two standard deviations and three standard deviations. If the quality of a product changes, the plotted points will not follow the variation patterns given above. The operator needs to investigate the possible causes and adjuss it so that the quality of production is consistent. Control charts, like $\overline{X}$ chart and R chart, are useful tools to help us visualize the quality control so that we can identify the change of quality more easily and straightforward.

The hypothesis being tested at the monitoring period is

$$H_0 : \mu_x = \hat{\mu}_0$$

against the alternative

$$H_1 : \mu_x \neq \hat{\mu}_0$$

where $\hat{\mu}_0$ comes from the base period and μ_x is the mean of the sample data we just collected. There are two types of error defined with this hypothesis test. One is called type I error (α), which is defined as the probability that we reject H_0 when H_0 is true. Type I error also means the probability when a point is beyond the control limit, and we identify it as a signal of quality change but actually it is not. Another one is called type II error (β), which is defined as the probability that we accept H_0 when H_0 is not true. Type II error is also the probability of a point is inside the control limit, and we identify the process is under control, but the process is actually out of control. The power of the hypothesis test is given by $(1-\beta)$. In industry, control limits are usually established at three standard deviations from the central line. Then, the probability that a type I error will occur is 0.26%.

```
Algorithm: Outline of Process Control
Phase 1: Base period
Step 1: Collect sample data
Step 2: Estimate the parameters
Step 3: Calculate the control limits
Step 4: Check each observation of the base data
Step 5: If it is over control limits, remove it and back to step 2
Step 6: If all observations are within the control limits
        Extend the control limits to monitoring period
Phase 2: Monitoring period
Use the control limits established during base period to test the hypothesis
that "the process is in control".
```

Figure 2.4: Outline of process control.

We can see the idea of Process Control is very similar to the idea of anomaly detection, which is profiling the normal pattern of an object and comparing the current pattern with the profile to determine the possible violation. We will apply Process Control technique to design the detection algorithm of the control system.

2.4.2 An Overview of Building the Control System Using Process Control Techniques

The control system includes a controller and a monitor. It takes three phases to build the proposed control system. Phase one is the training period during which the monitor collects normal connection behavior as the base data, then the control rule is defined for the controller using the base data. Phase two is the testing phase during which the reliability of the control system is tested using both normal and abnormal connection data. Phase three is the monitoring period during which the controller checks each and every observation the monitor collected to determine whether the current activity of the machine is legal.

This dissertation extends the traditional Process Control technique by adding a sliding window so that the base data always includes the most recent normal observations. This makes our system adaptive to the changes of process mean, and the false alarm rate is therefore reduced. The experiments demonstrate the reliability and flexibility of the control system. Furthermore, we present and compare the propagation models of malicious mobile code with and without the control system in order to evaluate the effect of the control system. The theoretical analysis shows that the detection method is effective and the propagation is reduced more when more hosts adopt the control system. Later, we simulate the propagation of malicious mobile programs in a network with a certain number of nodes. The simulation results match our theoretical analysis, which indicate a success of the propagation models and verify the effectiveness of the control system.

The idea of our approach is simple and straightforward, and so is Williamson's approach [Williamson 2002]. People did not think about it before, mainly because most people concentrate on how to protect ourselves from being infected. We install anti-virus tools, firewalls, filters, and intrusion detection systems to keep us secure. But people seldom think about minimizing the damage over the whole network if we have been infected unfortunately by a malicious program. This dissertation provides a mechanism to not only detect but also reduce the propagation of malicious codes at the early infection stage so that human beings could gain precious time to take counter actions like patching their system or upgrading their anti-virus tools to fight for the malicious mobile code.

2.5 Summary

This chapter provided the background information and related works of this dissertation. We first introduced the deterministic and stochastic models of epidemic modeling, and then we showed how epidemic modeling has been used to model the propagation of malicious code. We talked about the related work of early detection of malicious code and discussed the advantages of our detection system. After that, we introduced the process control technique and showed the framework of building the proposed control system using the process control technique.

CHAPTER 3

STOCHASTIC PROPAGATION MODELING OF MALICIOUS MOBILE CODE

This chapter uses stochastic modeling to model the propagation of malicious mobile code. Standard SI model and SIR model assume that the population is homogeneous. In fact, the number of individuals a given individual contacted in a certain period of time is limited. In this chapter, we introduce a new factor *B*, the average number of neighbors an individual has, into our model. In this dissertation, machine *g* becomes one of the neighbors of machine *h* only when machine *g* and machine *h* have direct contact with each other; for example, g sends an email to *h*, *g* downloads a file from *h*, *g* visits the website provided by *h*, etc., and vice versa.

We first present a stochastic SI model, and show how to get the explicit solution of the SI model using the pgf technique; then we present the stochastic INIM (Infection-Immunization) model which models the propagation of malicious code at the early infection stage. We get the approximate solution of INIM model using the pgf technique. We simulate the propagation evolution of a malicious mobile code using the INIM model. The theoretical results approximate the simulation results, which indicates that we can use the theoretical model to predict the propagation of malicious mobile code.

3.1 Stochastic SI Model for Malicious Code Propagation

The standard SI model divides the computer systems into two groups: one is susceptible, and the other is infected. Any system is at either susceptible state or at infected state. In the SI model, we assume that if a machine gets infected, it will never recover and become susceptible again, so the only state transition is S $\rightarrow$ I.

Let,

N be the size of the population.

$I(t) = i$: denote the number of infected machine at time t.

$p_{i,j}(0,t) = \Pr(I(t) = j \mid I(0) = i)$ $\quad i, j = 0, 1, 2, \cdots$ where $p_{i,j}(0,t)$ denotes the probability of j infections at time t, given that there are i infections at time zero.

β : The infection rate. In reality, this rate varies over time because the propagation of malicious code depends on the network bandwidth. At the beginning, a copy of the malicious code could always infect a susceptible machine successfully since the bandwidth of the network is enough to transfer all the byte stream of the malicious code, while later on, each copy of the malicious code could not reach the susceptible machine successfully because of the network traffic congestion; therefore, the infection rate decreases over time. In our model, for simplicity, we assume the infection rate to be a constant.

B : The number of machines that could be contacted by each machine during certain time unit. Generally, B would vary, but for simplicity, we assume that B is a constant. B is the same as the population size N in a homogeneous network, because any machine could contact any other machine in the population at any time.

Assume that one machine is infected at the beginning, and Δ is a small time interval in which no more than one infection could happen within it. Let $o(\Delta)$ represent any function of Δ which tends to 0 faster than Δ. Following the analysis of a simple Poisson process, we have:

(1) The probability that exactly one infection event happens in $(t, t+\Delta)$ is $\beta B(1-i/N)(i-1)\Delta + o(\Delta)$. This is so because in time interval Δ, one infected machine could contact B machine, and $B(1-i/N)$ of those are susceptible, so $\beta B(1-i/N)\Delta$ machines could be infected by one infected machine. Now, the number of infected machines is $(i-1)$, so the total number of machines that could be infected in time interval Δ is $\beta B(1-i/N)(i-1)\Delta$. Hence, $\beta B(1-i/N)(i-1)\Delta$ is the probability that exactly one infection occurs. At the early infection stage $i << N$, so the probability that one infection event occurs is approximately $\beta B(i-1)\Delta$. Since the probability that *i-1* infection happened in (0, *t*) is $p_{1,i-1}(0,t)$, the probability that i infection events occur in time interval $(0, t+\Delta)$ is $p_{1,i-1}(0,t)[\beta B(i-1)\Delta + o(\Delta)]$.

(2) The probability that more than one event, say q events, occurs during time interval Δ is $o(\Delta)$. The probability that $i-q$ infection events happen in time interval (0, *t*) is $p_{1,i-q}(0,t)$. So, the probability that i infection happen in time interval $(0, t+\Delta)$ is $p_{1,i-q}(0,t) \cdot o(\Delta)$.

(3) The probability that no event occurs in time interval Δ is $1-\beta Bi\Delta - o(\Delta)$. The probability that all infection events happen in $(0, t)$ is $p_{1,i}(0,t)$. So, the

probability that i infection events happen in time interval $(0, t+\Delta)$ is $p_{1,i}(0,t)[1-\beta B i\Delta - o(\Delta)]$.

Since only one of the above three is possible and they are mutually exclusive, we can combine all the three possibilities, so we have

$$p_{1,i}(0,t+\Delta) = p_{1,i-1}(0,t)[\beta B(i-1)\Delta + o(\Delta)] + p_{1,i}(0,t)[1-\beta B i\Delta - o(\Delta)]$$
$$+ p_{1,i-q}(0,t)\cdot o(\Delta).$$

Moving $p_{1,i}(0,t)$ from the right side to the left, and dividing both sides by Δ, since $\frac{o(\Delta)}{\Delta} \to 0$, we get

$$\frac{p_{1,i}(0,t+\Delta) - p_{1,i}(0,t)}{\Delta} = -p_{1,i}(0,t)\beta B i + p_{1,i-1}(0,t)\beta B(i-1).$$

Therefore, we have

$$\frac{dp_{1,i}(0,t)}{dt} = -p_{1,i}(0,t)\beta B i + p_{1,i-1}(0,t)\beta B(i-1) \qquad 3.1.1$$

The initial conditions are as follows. According to our assumption that only one machine is infected at time $t=0$, so $p_{1,1}(0,0)=1$, and the probability of more than one machine being infected is zero, so $p_{1,i}(0,0)=0, \quad i>1$.

We use probability generation function (pgf) to solve Equation 3.1.1. The pgf is given by

$$G_I(s,t) = \sum_{i=1}^{\infty} p_{1,i}(0,t)s^i \qquad 3.1.2$$

By differentiating Equation 3.1.2, and substituting Equation 3.1.1, we get

$$\frac{dG_I}{dt} = \sum_i \frac{dp_{1,i}(0,t)}{dt}s^i = \sum_i -B\beta i p_{1,i}(0,t)s^i + \sum_i B\beta(i-1)p_{1,i-1}(0,t)s^i$$

$$= -B\beta s \sum_{i\geq 1} i p_{1,i}(0,t) s^{i-1} + B\beta s^2 \sum_{i\geq 2} (i-1) p_{1,i-1}(0,t) s^{i-2}$$

We can ignore the values of the first term when i = 0 and of the second term when i = 1, because the probabilities are not valid.

$$= -B\beta s \frac{dG_I}{ds} + B\beta s^2 \frac{dG_I}{ds} \qquad 3.1.3$$

which can be written as

$$\frac{dG_I}{dt} = B\beta s(s-1)\frac{dG_I}{ds}$$

$$\Rightarrow \frac{dG_I}{dt} - B\beta s(s-1)\frac{dG_I}{ds} = 0 \qquad 3.1.4$$

To solve the partial differential equation of 3.1.4, we write the auxiliary equations

$$\Rightarrow \frac{ds}{s(1-s)} = B\beta dt \quad \text{and} \quad dG_I(s;t) = 0.$$

Using the following equality,

$$\int \frac{1}{ax^2+bx+c} = \frac{1}{\sqrt{b^2-4ac}} \ln \frac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}} \quad \text{if } b^2-4ac > 0$$

we get,

$$\Rightarrow \ln\frac{s}{s-1} = B\beta t + c \;\Rightarrow \frac{s}{s-1} = ce^{B\beta t} \;\Rightarrow \frac{s-1}{s}e^{B\beta t} = \text{constant}$$

From the second auxiliary equation, we get $G_I(s;t) = \text{constant}$.

Therefore, the general solution of 3.1.4 is

$$G_I(s;t) = \Phi(\frac{s-1}{s}e^{B\beta t}) \text{ where } \Phi \text{ is an arbitrary differentiable function.}$$

To obtain the particular solution, plug in the initial condition, and we get

$$G_I(s;0) = s = \Phi(\frac{s-1}{s})$$

Let $\theta = \frac{s-1}{s} \Rightarrow s = \frac{1}{1-\theta}$, we get $\Phi(\theta) = \frac{1}{1-\theta}$.

So, $G_I(s;t) = \Phi(\frac{s-1}{s}e^{B\beta t}) = \frac{1}{1-\frac{s-1}{s}e^{B\beta t}} = \frac{s}{s-(s-1)e^{B\beta t}}$

The solution of Equation 3.1.4 is

$$G_I(s;t) = \frac{s}{s-(s-1)e^{B\beta t}} \qquad 3.1.5$$

Taking the first order derivatives of $G_I(s;t)$, we get the expectation of I as

$$E[I(t)] = \frac{dG_I}{ds}\Big|_{s=1} = \frac{e^{B\beta t}}{(s-(s-1)e^{B\beta t})^2}\Big|_{s=1} = e^{B\beta t} \qquad 3.1.6$$

Taking the second order derivative, we get

$$E[I(I-1)] = \frac{d^2G_I}{ds^2} = (-2)\frac{1-e^{B\beta t}}{(s-(s-1)e^{B\beta t})^3}e^{B\beta t}\Big|_{s=1} = (-2)(1-e^{B\beta t})e^{B\beta t} \qquad 3.1.7$$

So, the variance of I is $\sigma^2 = E[I^2] - (E[I])^2 = E[I(I-1)] + E[I] - (E[I])^2$

$= (-2)(1-e^{B\beta t})e^{B\beta t} + e^{B\beta t} - (e^{B\beta t})^2 = e^{B\beta t}(e^{B\beta t}-1)$

Taking the ith order derivative, we could get

$$\frac{d^iG}{ds^i} = \frac{1}{i!}(-1)^{i-1}i!\frac{(1-e^{B\beta t})^{i-1}e^{B\beta t}}{(s-(s-1)e^{B\beta t})^{i+1}}\Big|_{s=0} = (-1)^{i-1}\frac{(1-e^{B\beta t})^{i-1}e^{B\beta t}}{(e^{B\beta t})^{i+1}}$$
$$= \frac{(e^{B\beta t}-1)^{i-1}}{e^{B\beta it}} = \left(\frac{e^{B\beta t}-1}{e^{B\beta t}}\right)^{i-1}\frac{1}{e^{B\beta t}} = \left(1-\frac{1}{e^{B\beta it}}\right)^{i-1}\frac{1}{e^{B\beta t}}$$

According to the properties of pgf, we get

$$p_{1,i}(0,t) = \left(1 - \frac{1}{e^{B\beta it}}\right)^{i-1} \frac{1}{e^{B\beta t}} \quad 3.1.8$$

3.2 Infection-Immunization (INIM) Model for Worm Propagation

In the Infection-Immunization model, the population is divided into three groups; that is, *susceptible*, *infected*, and *removed*, which is the same as the standard SIR model. The difference is that instead of recovered and therefore immunized after infected, a machine could be immunized when it is healthy, which makes the propagation model closer to reality because, for example, a computer user could immunize a machine by downloading the patch or updating the anti-virus tool when the users get the message about the malicious program. Both [Zou 2002] and [Wong 2004] present similar ideas of immunization from a healthy machine, but neither of them gives a stochastic analysis of their models.

Figure 3.1 shows the state transitions of epidemic propagation. The *removed* machines include those immunized when still healthy and those recovered after infected. A recovered machine is immunized and cannot be infected by the same worm again. If we do not specify, the immunized machine includes both cases. In Figure 3.1, α, β, and γ are the immunization rate, infection rate and recover rate, respectively.

$S(t) = s$ is the number of susceptible machines at time *t*.

$I(t) = i$ is the number of infected machines at time *t*.

$M(t) = m$ is the number of immunized machines at time *t*.

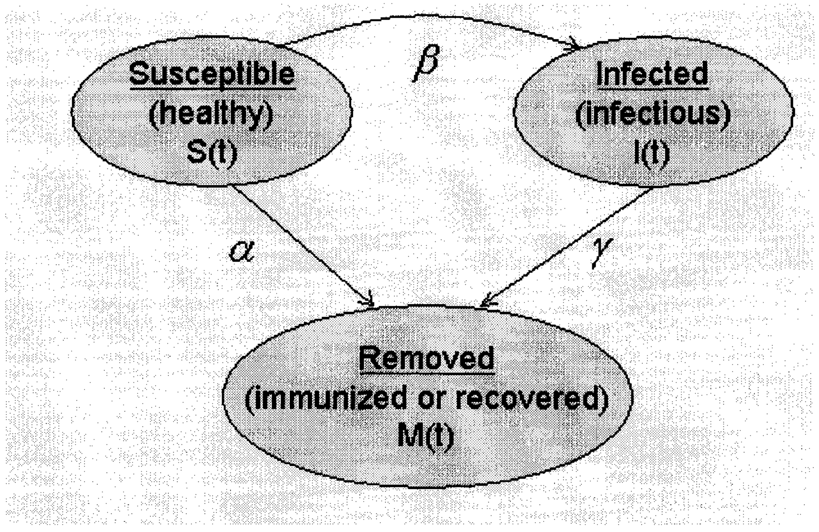


Figure 3.1: State transition diagram of INIM model.

3.2.1 Stochastic Analysis of INIM Model

Assume that only one machine is infected at time zero, and Δ is a small time interval in which no more than one infection could happen within it. To build the INIM model, following the similar analysis of SI model, we have:

(1) The probability that exactly one infection event happens in $(t,\ t+\Delta)$ is $\beta B(1-(i-1)/N-m/N)(i-1)\Delta+o(\Delta)$. This is so because in time interval Δ, one infected machine could contact B machine, and $(1-(i-1)/N-m/N)\Delta$ of those are susceptible, so $\beta B(1-(i-1)/N-m/N)\Delta$ machines could be infected by one infected machine. Now, the total number of infected machine is $(i-1)$, so the number of machines that could be infected in time interval Δ is $\beta B(1-(i-1)/N-m/N)(i-1)\Delta$. We assumed that Δ is a small time interval that no more than one event could occur within it. Hence, $\beta B(1-(i-1)/N-m/N)(i-1)\Delta$ is the probability that exactly one infection occurs. At the early stage of infection, since $i << N$ and $m << N$, the probability of one infection is approximately $B\beta(i-1)\Delta+o(\Delta)$. The probability that

$i-1$ infection events happen in $(0,t)$ is $p_{1,i-1}(0,t)$; therefore, the probability that i infection occur in time interval $(0,\ t+\Delta)$ is $p_{1,i-1}(0,t)(\beta B(i-1)\Delta+o(\Delta))$.

(2) The probability that exactly one recover event occurs in $(t,\ t+\Delta)$ is $(i+1)\gamma\Delta+o(\Delta)$. This is so because the probability that one infected machine is recovered during time interval Δ is $\gamma\Delta$. The total number of infected machine is $i+1$, so the number of machine that could be recovered during Δ is $(i+1)\gamma\Delta$. We assumed that Δ is a small time interval that no more than one event could occur within it. Hence, the probability that exactly one recover occurs is $(i+1)\gamma\Delta$. The probability that $i+1$ infection event happens in $(0,t)$ is $p_{1,i+1}(0,t)$; therefore, the probability that i infection occurs in time interval $(0,\ t+\Delta)$ is $p_{1,i+1}(0,t)((i+1)\gamma\Delta+o(\Delta))$.

(3) The probability that exactly i infection events happen in $(0,t)$ and the number of infection events does not change, i.e., no infection or recover occurs, during $(t,\ t+\Delta)$ is $(1-\beta Bi\Delta-i\gamma\Delta-o(\Delta))$. So the probability that i infection occurs in time interval $(0,\ t+\Delta)$ is $p_{1,i}(0,t)(1-\beta Bi\Delta-i\gamma\Delta-o(\Delta))$.

(4) The probability that more than one infection, say q infections, occurs during time interval Δ, but no recover occurs is $o(\Delta)$. The probability that $i-q$ infection events happen in time interval $(0,t)$ is $p_{1,i-q}(0,t)$. So, the probability that i infection happens in time interval $(0,t+\Delta)$ is $p_{1,i-q}(0,t)\cdot o(\Delta)$.

(5) The probability that more than one recover events, say q recover events, occur during time interval Δ but no infection occurs is $o(\Delta)$. The probability that $i+q$ infection

events happen in time interval (0, t) is $p_{1,i+q}(0,t)\cdot$ So, the probability that i infection occur in time interval $(0, t+\Delta)$ is $p_{1,i+q}(0,t)\cdot o(\Delta)$.

Since only one of the above six is possible and they are mutually exclusive, we can combine all three possibilities, so we have

$$\begin{aligned} p_{1,i}(0,t+\Delta) = {} & (1-\beta Bi\Delta - i\gamma\Delta - o(\Delta))p_{1,i}(0,t) \\ & + p_{1,i-1}(0,t)[\beta B(i-1)\Delta + o(\Delta)] \\ & + p_{1,i+1}(0,t)[(i+1)\gamma\Delta + o(\Delta)] \\ & + p_{1,i-q}(0,t)\cdot o(\Delta) + p_{1,i+q}(0,t)\cdot o(\Delta) \end{aligned}$$

Moving $p_{1,i}(0,t)$ from the right side to the left, and dividing both sides by Δ, since $\frac{o(\Delta)}{\Delta} \to 0$, we get

$$\begin{aligned} \frac{p_{1,i}(0,t+\Delta) - p_{1,i}(0,t)}{\Delta} = {} & (-\beta Bi - i\gamma)p_{1,i}(0,t) \\ & + p_{1,i-1}(0,t)\beta B(i-1) \\ & + p_{1,i+1}(0,t)(i+1)\gamma \end{aligned}$$

Therefore, the differential equation is

$$\begin{aligned} \frac{dp_{1,i}(0,t)}{dt} = {} & -(\beta Bi - i\gamma)p_{1,i}(0,t) \\ & + p_{1,i-1}(0,t)\beta B(i-1) \\ & + p_{1,i+1}(0,t)(i+1)\gamma \end{aligned} \qquad 3.2.1$$

The initial conditions are as follows. According to our assumption that only one machine is infected at time $t=0$, so $p_{1,1}(0,0)=1$, and the probability of more than one machine being infected is zero, so $p_{1,i}(0,0)=0, \quad i>1$.

Define pgf as

$$G_I(s,t) = \sum_{i=1}^{\infty} p_{1,i}(0;t)s^i \qquad 3.2.2$$

Differentiating Equation 3.2.2 and substituting Equation 3.2.1, we have

$$\frac{dG}{dt}=\sum_i \frac{dp_{1,i}(0,t)}{dt}=\sum_i s^i[-(\beta Bi-i\gamma)p_{1,i}(0,t)+p_{1,i-1}(0,t)\beta B(i-1)+p_{1,i+1}(0,t)(i+1)\gamma]$$

By extending the summation to each term, we get

$$=\sum_i\ [-(\beta Bi-i\gamma)p_{1,i}(0,t)]s^i$$
$$+\sum_i\ p_{1,i-1}(0,t)\beta B(i-1)s^i$$
$$+\sum_i\ p_{1,i+1}(0,t)(i+1)\gamma s^i$$

Rearranging each term, we get

$$=s\sum_i\ [-(\beta B-\gamma)ip_{1,i}(0,t)]s^{i-1}$$
$$+s^2\sum_i\ p_{1,i-1}(0,t)\beta B(i-1)s^{i-2}\ ,$$
$$+\sum_i\ p_{1,i+1}(0,t)(i+1)\gamma s^i$$

which can be written as

$$\frac{dG}{dt}=-(B\beta+\gamma)s\frac{dG}{ds}+B\beta s^2\frac{dG}{ds}+\gamma\frac{dG}{ds}.$$

Let $B\beta=\lambda$, we get

$$\frac{dG}{dt}=-(\lambda+\gamma)s\frac{dG}{ds}+\lambda s^2\frac{dG}{ds}+\gamma\frac{dG}{ds}\Rightarrow\frac{dG}{dt}-(\lambda s^2-(\lambda+\gamma)s+\gamma)\frac{dG}{ds}=0 \qquad 3.2.3$$

To solve partial differential Equation 3.2.3, we write the auxiliary equations

$$dt=\frac{ds}{-(\lambda s^2-(\lambda+\gamma)s+\gamma)}\ \text{ and } dG_I(s;t)=0$$

If $\lambda>\gamma$

From the first auxiliary equation, we get

$$\int dt=\int\frac{1}{-(\lambda s^2-(\lambda+\gamma)s+\gamma)}ds.$$

Using the following equality

$$\int \frac{1}{ax^2+bx+c} = \frac{1}{\sqrt{b^2-4ac}} \ln \frac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}} \quad \text{if } b^2-4ac>0,$$

(In this case, a=$-\lambda$, b= $(\lambda+\gamma)$, c=$-\gamma$, $\sqrt{b^2-4ac} = \sqrt{(\lambda-\gamma)^2} = \lambda-\gamma$)

w get $C+t = \frac{1}{\lambda-\gamma} \ln \frac{-2\lambda s+(\lambda+\gamma)-(\lambda-\gamma)}{-2\lambda s+(\lambda+\gamma)+(\lambda-\gamma)} = \frac{1}{\lambda-\gamma} \ln \frac{\lambda s-\gamma}{\lambda(s-1)}$

$\Rightarrow c_1 e^{(\lambda-\gamma)t} = \frac{\lambda s-\gamma}{\lambda(s-1)}$ where c_1 is the new constant contains C.

$\Rightarrow \lambda c_1 e^{(\lambda-\gamma)t} = \frac{\lambda s-\gamma}{(s-1)}$

$\Rightarrow c_2 e^{(\lambda-\gamma)t} = \frac{\lambda s-\gamma}{(s-1)}$ where c_2 is the new constant contains c_1.

$\Rightarrow e^{(\lambda-\gamma)t} \frac{(s-1)}{\lambda s-\gamma} = c_2$

From the second auxiliary equation, we get $G_I(s;t) =$ constant. Therefore, the general solution of $G_I(s;t)$ is

$$G(s;t) = \Phi\{ \frac{(s-1)}{\lambda s-\gamma} e^{(\lambda-\gamma)t} \}$$

To obtain the particular solution, plug in the initial condition, we get

$$G(s;0) = p_{1,1}(0,0)s^1 = s \text{ (see initial conditions of Equation 3.2.1)}$$

so $G(s;0) = \Phi\{ \frac{(s-1)}{\lambda s-\gamma} e^{(\lambda-\gamma)0} \} = \Phi\{ \frac{(s-1)}{\lambda s-\gamma} \} = s$.

Let $\theta = \frac{(s-1)}{\lambda s-\gamma} \Rightarrow s = \frac{\theta\gamma-1}{\theta\lambda-1}$, we get $\Phi(\theta) = .\frac{\theta\gamma-1}{\theta\lambda-1}$

Therefore, the particular solution is

$$G(s;t)=\Phi\{\frac{(s-1)}{\lambda s-\gamma}e^{(\lambda-\gamma)t}\}=\frac{\gamma\frac{s-1}{\lambda s-\gamma}e^{(\lambda-\gamma)t}-1}{\lambda\frac{s-1}{\lambda s-\gamma}e^{(\lambda-\gamma)t}-1}=\frac{\gamma(s-1)e^{(\lambda-\gamma)t}-(\lambda s-\gamma)}{\lambda(s-1)e^{(\lambda-\gamma)t}-(\lambda s-\gamma)}$$

$$=\frac{\gamma(1-s)e^{(\lambda-\gamma)t}+(\lambda s-\gamma)}{\lambda(1-s)e^{(\lambda-\gamma)t}+(\lambda s-\gamma)} \quad 3.2.4$$

If $\lambda<\gamma$, from the first auxiliary equation, we get

$$\int dt=\int\frac{1}{-(\lambda s^2-(\lambda+\gamma)s+\gamma)}ds.$$

Using the following equality,

$$\int\frac{1}{ax^2+bx+c}=\frac{1}{\sqrt{(b^2-4ac)}}\ln\frac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}} \quad \text{if } b^2-4ac>0,$$

(In this case, a= $-\lambda$, b= $(\lambda+\gamma)$, c= $-\gamma$, $\sqrt{b^2-4ac}=\sqrt{(\gamma-\lambda)^2}=\gamma-\lambda$)

we get

$$t+C=\frac{1}{\gamma-\lambda}\ln\frac{2\lambda s-(\lambda+\gamma)-(\lambda-\gamma)}{2\lambda s-(\lambda+\gamma)+(\lambda-\gamma)}=\frac{1}{\gamma-\lambda}\ln\frac{\lambda(s-1)}{\lambda s-\gamma}$$

Similar to solving the equation when $\lambda>\gamma$, we have

$$c_1e^{(\gamma-\lambda)t}=\frac{\lambda(s-1)}{\lambda s-\gamma}$$

$$\Rightarrow c_1e^{(\gamma-\lambda)t}/\lambda=\frac{(s-1)}{\lambda s-\gamma}$$

$$\Rightarrow e^{(\gamma-\lambda)t}\frac{\lambda s-\gamma}{s-1}=c_2$$

The general solution of the second auxiliary equation is $G(s;t)=\Phi\{\frac{\lambda s-\gamma}{s-1}e^{(\gamma-\lambda)t}\}$.

Using the initial condition of Equation 3.2.1, we have $G(s;0)=p_{1,1}(0,0)s^1=s$.

So, $G(s;0) = \Phi\{\frac{\lambda s - \gamma}{s-1} e^{(\gamma-\lambda)0}\} = \Phi\{\frac{\lambda s - \gamma}{s-1}\} = s$.

Let $\theta = \frac{\lambda s - \gamma}{s-1} \Rightarrow s = \frac{\gamma - \theta}{\lambda - \theta}$, we get $\Phi(\theta) = \frac{\gamma - \theta}{\lambda - \theta}$.

Therefore, the particular solution is

$$G(s;t) = \Phi\{\frac{\lambda s - \gamma}{s-1} e^{(\gamma-\lambda)t}\} = \frac{\gamma - \frac{\lambda s - \gamma}{s-1} e^{(\gamma-\lambda)t}}{\lambda - \frac{\lambda s - \gamma}{s-1} e^{(\gamma-\lambda)t}} = \frac{\gamma(s-1) - (\lambda s - \gamma)e^{(\gamma-\lambda)t}}{\lambda(s-1) - (\lambda s - \gamma)e^{(\gamma-\lambda)t}} \qquad 3.2.5$$

If $\lambda = \gamma$, we have

$$\int dt = \int \frac{1}{-(\lambda s^2 - (\lambda + \gamma)s + \gamma)} ds.$$

Using the following equality,

$$\int \frac{1}{ax^2 + bx + c} = -\frac{2}{2ax + b} \quad \text{if } b^2 - 4ac = 0,$$

(In this case, a=$-\lambda$, b= $(\lambda + \gamma)$, c=$-\gamma$, $\sqrt{b^2 - 4ac} = \sqrt{(\gamma - \lambda)^2} = 0$)

we get

$$t + C = -\frac{2}{-2\lambda s + (\lambda + \gamma)} = \frac{2}{2\gamma s - (\gamma + \gamma)} = \frac{1}{\gamma(s-1)}$$

$$\frac{1}{\gamma(s-1)} - t = C$$

The general solution of the second auxiliary equation is

$$G(s;t) = \Phi\{\frac{1}{\gamma(s-1)} - t\}$$

Using the initial condition of Equation 3.2.1, we have $G(s;0) = \Phi\{\frac{1}{\gamma(s-1)} - 0\} = s$

Let $\theta = -\dfrac{1}{\gamma(s-1)} \Rightarrow s = 1+\dfrac{1}{\theta\gamma}$,

we get $\Phi(\theta) = 1+\dfrac{1}{\theta\gamma}$.

Therefore, the particular solution is

$$G(s;t) = \Phi\{\frac{1}{\gamma(s-1)} - t\} = 1+\frac{1}{(\frac{1}{\gamma(s-1)}-t)\gamma} = 1+\frac{1}{\frac{1-t\gamma(s-1)}{\gamma(s-1)}\gamma}$$

$$= 1+\frac{s-1}{1-t\gamma(s-1)} = \frac{1-t\gamma(s-1)+(s-1)}{1-t\gamma(s-1)} = \frac{s-t\gamma(s-1)}{1-t\gamma(s-1)}.$$

Rearrange the terms, we get

$$G\left(s;t\right) = \frac{s(1-t\gamma)+t\gamma}{1+t\gamma-t\gamma s}. \qquad 3.2.6$$

We apply the properties of pgf to get the mean and variance of *I*.

If $\lambda > \gamma$, taking the first order derivative of Equation 3.2.4, we get

$$\frac{dG}{ds} = \frac{\gamma(-1)e^{(\lambda-\gamma)t}+\lambda}{\lambda(1-s)e^{(\lambda-\gamma)t}+(\lambda s-\gamma)} + [\gamma(1-s)e^{(\lambda-\gamma)t}+(\lambda s-\gamma)]\frac{-\lambda e^{(\lambda-\gamma)t}+\lambda}{(-1)[\lambda(1-s)e^{(\lambda-\gamma)t}+(\lambda s-\gamma)]^2}$$

Let $s=1$, we get

$$\frac{dG}{ds}|_{s=1} = \frac{\gamma(-1)e^{(\lambda-\gamma)t}+\lambda}{(\lambda-\gamma)} + (\lambda-\gamma)\frac{-\lambda e^{(\lambda-\gamma)t}+\lambda}{(-1)[(\lambda-\gamma)]^2}$$

$$= \frac{\gamma(-1)e^{(\lambda-\gamma)t}+\lambda-(-\lambda e^{(\lambda-\gamma)t}+\lambda)}{(\lambda-\gamma)} = \frac{(\lambda-\gamma)e^{(\lambda-\gamma)t}}{(\lambda-\gamma)} = e^{(\lambda-\gamma)t}.$$

According to the properties of pgf, we get

$$E[I] = \frac{dG}{ds}|_{s=1} = e^{(\lambda-\gamma)t}$$

Taking the second derivative of Equation 3.2.4, we get

$$\frac{d^2G}{ds^2}=\frac{(-\gamma e^{(\lambda-\gamma)t}+\lambda)(-\lambda e^{(\lambda-\gamma)t}+\lambda)}{(-1)[\lambda(1-s)e^{(\lambda-\gamma)t}+(\lambda s-\gamma)]^2}+[-\gamma e^{(\lambda-\gamma)t}+\lambda]\frac{-\lambda e^{(\lambda-\gamma)t}+\lambda}{(-1)[\lambda(1-s)e^{(\lambda-\gamma)t}+(\lambda s-\gamma)]^2}$$

$$+[\gamma(1-s)e^{(\lambda-\gamma)t}+(\lambda s-\gamma)](-2)\frac{-\lambda e^{(\lambda-\gamma)t}+\lambda}{(-1)[\lambda(1-s)e^{(\lambda-\gamma)t}+(\lambda s-\gamma)]^3}$$

$$\frac{d^2G}{ds^2}\Big|_{s=1}=\frac{(-\gamma e^{(\lambda-\gamma)t}+\lambda)(-\lambda e^{(\lambda-\gamma)t}+\lambda)}{(-1)[(\lambda-\gamma)]^2}+[-\gamma e^{(\lambda-\gamma)t}+\lambda]\frac{-\lambda e^{(\lambda-\gamma)t}+\lambda}{(-1)[(\lambda-\gamma)]^2}$$

$$+2[(\lambda-\gamma)]\frac{-\lambda e^{(\lambda-\gamma)t}+\lambda}{[(\lambda-\gamma)]^3}$$

$$=(-2)[-\gamma e^{(\lambda-\gamma)t}+\lambda]\frac{-\lambda e^{(\lambda-\gamma)t}+\lambda}{(\lambda-\gamma)^2}+2\frac{-\lambda e^{(\lambda-\gamma)t}+\lambda}{(\lambda-\gamma)^2}$$

$$=2\frac{-\lambda e^{(\lambda-\gamma)t}+\lambda}{(\lambda-\gamma)^2}(1+\gamma e^{(\lambda-\gamma)t}-\lambda]$$

According to the properties of pgf, we get

$$E[I(I-1)]=\frac{d^2G}{ds^2}\Big|_{s=1}=2\frac{\lambda(1-e^{(\lambda-\gamma)t})}{(\lambda-\gamma)^2}[1+\gamma e^{(\lambda-\gamma)t}-\lambda]$$

Since $\sigma^2=E[I^2]-(E[I])^2=E[I(I-1)]+E[I]-(E[I])^2$, we get

$$\sigma^2=2\frac{\lambda(1-e^{(\lambda-\gamma)t})}{(\lambda-\gamma)^2}[1+\gamma e^{(\lambda-\gamma)t}-\lambda]+e^{(\lambda-\gamma)t}-[e^{(\lambda-\gamma)t}]^2$$

$$=2\frac{\lambda(1-e^{(\lambda-\gamma)t})}{(\lambda-\gamma)^2}[1+\gamma e^{(\lambda-\gamma)t}-\lambda]+e^{(\lambda-\gamma)t}\frac{(1-e^{(\lambda-\gamma)t})(\lambda-\gamma)^2}{(\lambda-\gamma)^2}$$

$$=\frac{(1-e^{(\lambda-\gamma)t})}{(\lambda-\gamma)^2}[(\lambda^2+\gamma^2)e^{(\lambda-\gamma)t}+2\lambda-2\lambda^2] \qquad 3.2.7$$

If $\lambda<\gamma$, taking the first order derivative of Equation 3.2.5, we get

$$\frac{dG}{ds}=\frac{\gamma-\lambda e^{(\gamma-\lambda)t}}{\lambda(s-1)-(\lambda s-\gamma)e^{(\gamma-\lambda)t}}+\frac{\gamma(s-1)-(\lambda s-\gamma)e^{(\gamma-\lambda)t}}{(-1)[\lambda(s-1)-(\lambda s-\gamma)e^{(\gamma-\lambda)t}]^2}(\lambda-\lambda e^{(\gamma-\lambda)t})$$

Let $s=1$, we get

$$\frac{dG}{ds}\Big|_{s=1}=\frac{\gamma-\lambda e^{(\gamma-\lambda)t}}{-(\lambda-\gamma)e^{(\gamma-\lambda)t}}+\frac{(\lambda-\gamma)e^{(\gamma-\lambda)t}}{[-(\lambda-\gamma)e^{(\gamma-\lambda)t}]^2}(\lambda-\lambda e^{(\gamma-\lambda)t})$$

$$=\frac{\gamma-\lambda e^{(\gamma-\lambda)t}}{-(\lambda-\gamma)e^{(\gamma-\lambda)t}}+\frac{1}{[(\lambda-\gamma)e^{(\gamma-\lambda)t}]}(\lambda-\lambda e^{(\gamma-\lambda)t})$$

$$=\frac{\lambda e^{(\gamma-\lambda)t}-\gamma+(\lambda-\lambda e^{(\gamma-\lambda)t})}{(\lambda-\gamma)e^{(\gamma-\lambda)t}}=\frac{-\gamma+\lambda}{(\lambda-\gamma)e^{(\gamma-\lambda)t}}=e^{(\lambda-\gamma)t}$$

So, $E[I]=\frac{dG}{ds}\Big|_{s=1}=e^{(\lambda-\gamma)t}$ 3.2.8

Taking the second derivative of Equation 3.2.5 $G\ (s;t)$, we get

$$\frac{d^2G}{ds^2}=(-1)\frac{(\gamma-\lambda e^{(\gamma-\lambda)t})(\lambda-\lambda e^{(\gamma-\lambda)t})}{[\lambda(s-1)-(\lambda s-\gamma)e^{(\gamma-\lambda)t}]^2}+\frac{\gamma-\lambda e^{(\gamma-\lambda)t}}{(-1)[\lambda(s-1)-(\lambda s-\gamma)e^{(\gamma-\lambda)t}]^2}(\lambda-\lambda e^{(\gamma-\lambda)t})$$

$$+(-2)\frac{[\gamma(s-1)-(\lambda s-\gamma)e^{(\gamma-\lambda)t}][\lambda-\lambda e^{(\gamma-\lambda)t}]}{(-1)[\lambda(s-1)-(\lambda s-\gamma)e^{(\gamma-\lambda)t}]^3}(\lambda-\lambda e^{(\gamma-\lambda)t})$$

$$\frac{d^2G}{ds^2}\Big|_{s=1}=(-1)\frac{(\gamma-\lambda e^{(\gamma-\lambda)t})(\lambda-\lambda e^{(\gamma-\lambda)t})}{[-(\lambda-\gamma)e^{(\gamma-\lambda)t}]^2}+\frac{\gamma-\lambda e^{(\gamma-\lambda)t}}{(-1)[-(\lambda-\gamma)e^{(\gamma-\lambda)t}]^2}(\lambda-\lambda e^{(\gamma-\lambda)t})$$

$$+(-2)\frac{[-(\lambda-\gamma)e^{(\gamma-\lambda)t}][\lambda-\lambda e^{(\gamma-\lambda)t}]}{(-1)[-(\lambda-\gamma)e^{(\gamma-\lambda)t}]^3}(\lambda-\lambda e^{(\gamma-\lambda)t})$$

$$=(-2)\frac{(\gamma-\lambda e^{(\gamma-\lambda)t})(\lambda-\lambda e^{(\gamma-\lambda)t})}{[(\lambda-\gamma)e^{(\gamma-\lambda)t}]^2}+2\frac{(\lambda-\gamma)e^{(\gamma-\lambda)t}[\lambda-\lambda e^{(\gamma-\lambda)t}]}{[(\lambda-\gamma)e^{(\gamma-\lambda)t}]^3}(\lambda-\lambda e^{(\gamma-\lambda)t})$$

$$=(-2)\frac{(\gamma-\lambda e^{(\gamma-\lambda)t})(\lambda-\lambda e^{(\gamma-\lambda)t})}{[(\lambda-\gamma)e^{(\gamma-\lambda)t}]^2}+2\frac{[\lambda-\lambda e^{(\gamma-\lambda)t}]}{[(\lambda-\gamma)e^{(\gamma-\lambda)t}]^2}(\lambda-\lambda e^{(\gamma-\lambda)t})$$

$$=2\frac{[\lambda-\lambda e^{(\gamma-\lambda)t}]}{[(\lambda-\gamma)e^{(\gamma-\lambda)t}]^2}[(\lambda-\lambda e^{(\gamma-\lambda)t})-(\gamma-\lambda e^{(\gamma-\lambda)t})]$$

$$=2\frac{[\lambda-\lambda e^{(\gamma-\lambda)t}]}{[(\lambda-\gamma)e^{(\gamma-\lambda)t}]^2}(\lambda-\gamma)$$ 3.2.9

Therefore, the variance of $I(t)$ could be obtained as

$$\sigma^2 = E[I^2] - (E[I])^2 = E[I(I-1)] + E[I] - (E[I])^2$$

$$= 2\frac{[\lambda - \lambda e^{(\gamma-\lambda)t}]}{[(\lambda-\gamma)e^{(\gamma-\lambda)t}]^2}(\lambda-\gamma) + e^{(\lambda-\gamma)t} - (e^{(\lambda-\gamma)t})^2 \qquad 3.2.10$$

From the above derivations, we can see the solution for the expectation of I is the same when $\lambda \neq \gamma$; that is,

$$I(t) = e^{(\lambda-\gamma)t} \qquad 3.2.11$$

But the solution for the variance of I is different (See Equation 3.2.7 and 3.2.10).

When $\lambda = \gamma$, we have $G(s;t) = \frac{s(1-t\gamma)+t\gamma}{1+t\gamma-t\gamma s}$. Taking the first order derivative of Equation 3.2.6, we get

$$\frac{dG}{ds} = \frac{1-\gamma t}{1+\gamma t-\gamma ts} + [t\gamma + (1-t\gamma)s]\frac{-t\gamma}{(-1)(1+t\gamma-t\gamma s)^2}$$

According to the property of pgf, the expectation of I is

$$E[I] = \frac{dG}{ds}|_{s=1} = \frac{-\gamma t-1}{\gamma t-\gamma t-1} + [-1]\frac{-t\gamma}{(-1)[-1]^2} = \gamma t + 1 - \gamma t = 1 \qquad 3.2.12$$

Taking the second derivative of Equation 3.2.7, we get

$$\frac{d^2G}{ds^2} = \frac{(1-\gamma t)(-\gamma t)}{(-1)(1+\gamma t-\gamma ts)^2} + t\gamma[\frac{1-\gamma t}{(1+t\gamma-t\gamma s)^2} + (t\gamma + (1-\gamma t)s)\frac{(-2)(-\gamma t)}{(1+t\gamma-t\gamma s]^3}]$$

Let s = 1, and according to the properties of pgf, we get

$$E[I(I-1)] = \frac{d^2G(s;t)}{ds^2}|_{s=1} = \frac{(1-\gamma t)(-\gamma t)}{(-1)} + (t\gamma)[\frac{1-\gamma t}{1} + \frac{2\gamma t}{[1]^3}]$$
$$= \gamma t - (\gamma t)^2 + t\gamma[1+\gamma t] = 2\gamma t$$

So the variance of I could be

$$\sigma^2 = E[I^2] - (E[I])^2 = E[I(I-1)] + E[I] - (E[I])^2 = 2\gamma t + 1 - 1^2 = 2\gamma t \qquad 3.2.13$$

We can see that when $\lambda = \gamma$, the expected number of infected machines is a constant as the initial value (See Equation 3.2.12), and the variance is a linear function of t.

3.2.2 Discussion about Stochastic Model and Deterministic Model

The corresponding deterministic model of the INIM model is

$$\begin{cases} \frac{dI(t)}{dt} = B\beta I(t) - \gamma I(t) \\ \frac{dM(t)}{dt} = \alpha S(t) + \gamma I(t) \\ S(t) = N - I(t) - M(t) \end{cases} . \qquad 3.2.14$$

with initial conditions $I(0) = 1$, $S(0) = N - 1$, and $M(0) = 0$.

The first equation of 3.2.14 can be written as

$$\frac{dI(t)}{I(t)} = (B\beta - \gamma)dt .$$

Integrate both sides, we get $\int \frac{dI(t)}{I(t)} = \int (B\beta - \gamma)dt \Rightarrow \ln I(t) = (B\beta - \gamma)t + c$

$\Rightarrow I(t) = c_1 e^{(B\beta - \gamma)t}$

Use initial condition $I(0) = 1$, we get $c_1 = 1$. So, the number of infected machines at time t is

$$I(t) = e^{(B\beta - \gamma)t} \qquad 3.2.15$$

Notice that if we assume the machines are fully connected (homogenous network), B equals to *S(t)*, and the first equation in equation system 3.2.14 becomes

$$\frac{dI(t)}{dt} = \beta S(t) I(t) - \gamma I(t) \qquad 3.2.16$$

In [Wong 2004], the authors also give a deterministic model with immunization rate μ. They call the model a "delayed immunization model" because immunization starts at a time moment when a certain proportion of hosts is infected. In their model, they use $(N - I)$ to represent the number of suspected nodes $S(t)$, but it should be $(N - I - M)$ as given in the third equation of equation system 3.2.14 if we consider the immunization from the healthy machine. The authors of [Wong 2004] ignored the number of immunized node M when building the propagation model, but they did not specify this approximation. As [Wong 2004] stated, the number of immunized nodes and immunization rate is not easily observable, so we are not sure how much this approximation affects the model's accuracy. We can only ignore the number of immunized machine M and approximate the number of susceptible node $S(t)$ as $N - I$ at the early stage of infection. That is why we assume $I(t) << N$ and $M(t) << N$ and get equation system 3.2.14. This assumption makes our model more applicable at the early stage of infection which includes both the starting stage and the fast-growing stage. This is also why the infected nodes grow exponentially as shown in Equation 3.2.15 instead of logistically as given in Chapter 2.

From Equation 3.2.15, we can see the number of infected machines we get from the deterministic model is exactly the same as the expected value of the stochastic model. The solutions agree with the theory that the stochastic model converges to the deterministic model when the population size is large [Andersson 2000].

From the stochastic model, we get both the expectation and variance of those values at any time t so that we can evaluate the best and the worst infection situation instead of a single number of infected machines. This evaluation could be important under certain circumstances. For example, when the infection rate and the recover rate are the same, the deterministic model shows the number of infected machines will not change and keep the same number as the initial state. The expected value of the stochastic model agrees with this result. But from the stochastic model, we also know that the variance is increasing as time goes on, and the number of infected machines could be infinite when $t \to \infty$.

3.3 Simulation Analysis

3.3.1 Simulation Setup

The simulation program is written using C++ compiled with Microsoft Visual C++ compiler under Windows XP environment. First, we randomly generate a simulated network with a given number of nodes, say N nodes. Each node has r neighbors; r is a random number ranging from a to b, where a and b are given real numbers and $\frac{a+b}{2} = B$, so $0 < a < r < b < N$. Then for each node g, we randomly choose r nodes out of the N nodes as its neighbor. When node h is selected as the neighbor of g, we also add g as one of the neighbors of h. When we generate the neighbors for h, we will randomly choose ($r -$existing number of neighbors) as h's neighbors so that the total number of neighbors is still r. The average number of neighbors each node has is B.

Our infection simulation system is based on this randomly generated simulation network. The infection simulation system has three main procedures: infection, immunization, and recovering. When we start the simulation, one node will be randomly selected as the infected node. At each time step, all three events, infection, immunization and recovering, occur simultaneously. In the infection procedure, each infected machine infects its neighbors with rate β. Similarly, the recovering procedure checks all infected nodes and tries to recover it with rate γ, and the immunization procedure immunizes the susceptible machines with rate α.

3.3.2 The Random Number Generator

The random number generator provided by C++ library is a function *rand*( ). First, we need to initialize the random number generator by invoking *srand*(seed). Each initializing seed generates a different random number sequence. We get one random number from the sequence every time we call *rand*( ) function. Those random numbers returned by calling *rand*( ) function range from 0 to RAND_MAX, where RAND_MAX is the maximum number a machine could generate. If we want uniform random numbers in [a, b], we can use the expression x = a + (b – a) rand( )/(RAND_MAX+1.0). But the problem with the random number generator is that the same seed always generates the same random sequence. The random number generator we use to generate the random network is provided by [Vetterling 2002]. This generator avoids the problem we mentioned in *rand*( ) function given by the C++ library.

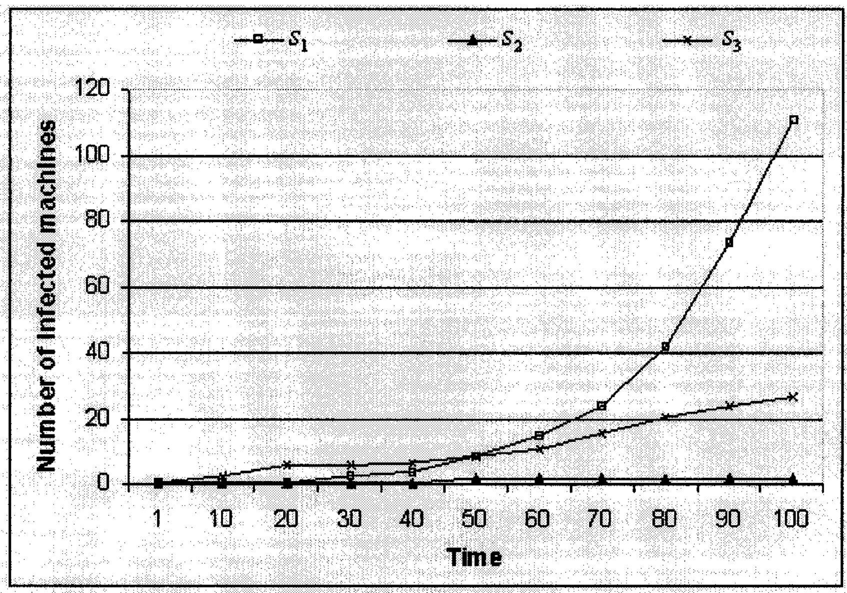
3.3.3 Simulation Results and Results Analysis

We run the infection simulation using different α, β and γ values with a same network of 1,000 nodes. Figure 3.2 - Figure 3.5 plot the simulation results. Each curve of the plots is the average of 100 simulation runs.

Figure 3.2 plots the simulated number of infected machines with different parameters. In Figure 3.2, s_1 is the simulation result of no immunization from healthy machines, while s_2 and s_3 show the simulation result when healthy machines are immunized before getting infected. The difference of s_2 and s_3 is that s_2 is the simulation result when the infection rate is less than the recover rate; s_3 is the simulation result when the infection rate is equal to the recover rate. The effect of immunization from healthy machines is not negligible since both s_2 and s_3 increase slower than s_1. We can see s_2 grows very slow since the infection rate is less than the recover rate.

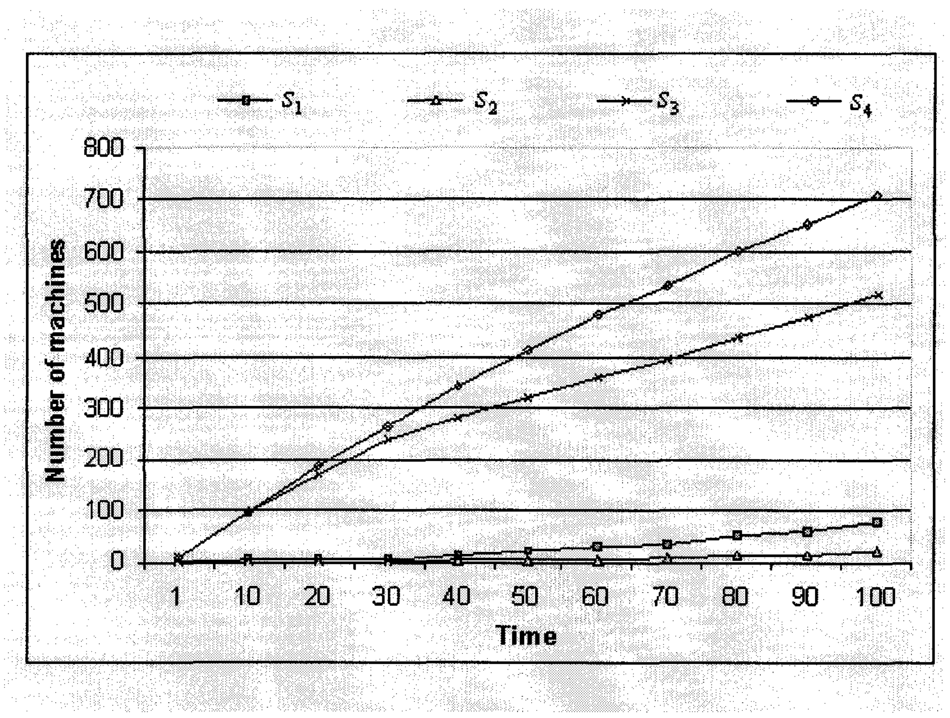
Figure 3.3, Figure 3.4, and Figure 3.5 give the simulation results and the expected values from INIM model. Figure 3.3 shows when $\lambda > \gamma$, i.e., infection rate is greater than the recover rate, the number of infected machine (s_1 is the simulation result, s_2 is the expected value from INIM model) is increasing as time goes on. Since healthy machines are immunized at the same time, the number of infected machines (s_1 and s_2) do not grow tremendously even though the infection rate is greater than the recover rate. With the same immunization rate, when infection rate is equal to the recover rate (see Figure 3.4), the expected number of infection shown by the model (s_2) is always a constant as its initial value, which is 1. The simulation result (s_1) shows that the number of infected machines is increasing very slowly instead of at a constant. The simulation result is reasonable since our model shows the variance of the expected number of infection is increasing as a function of time t (see Equation 3.2.11). If the infection rate is less than the recover rate, the expected number of infection is zero (see s_2 in Figure 3.6). The simulation result is not zero but very close to it (see s_1 in Figure 3.5).

In all three figures (Figure 3.3, Figure 3.4, and Figure 3.5), the simulation result (s_3) and theoretical result (s_4) of the number of immunized machine grows much faster than the number of infected machines. Also, we can see the theoretical results (s_4) fit the simulation results (s_3) better when infection just gets started. As time goes on, the difference between s_3 and s_4 in all three figures (Figure 3.3, Figure 3.4, and Figure 3.5) becomes bigger. This is because when we build the model, for simplicity we assume $(i+m)/N \to 0$, but as time goes on, the number of immunized node is increasing, which makes $(i+m)/N$ bigger and not ignorable. Therefore, we cannot directly apply this model to the whole life cycle of propagation of a malicious mobile code. In the future, we may use the exact value to build the stochastic model, but the solution of such a model will be much more difficult to obtain.



$s_1: \alpha = 0, \beta = 0.01, \gamma = 0.06$
$s_2: \alpha = 0.02, \beta = 0.01, \gamma = 0.06$
$s_3: \alpha = 0.02, \beta = 0.01, \gamma = 0.01$

Figure 3.2: The simulation results of the number of infected machines with different parameters.



$\alpha = 0.01, \beta = 0.01, \gamma = 0.02$

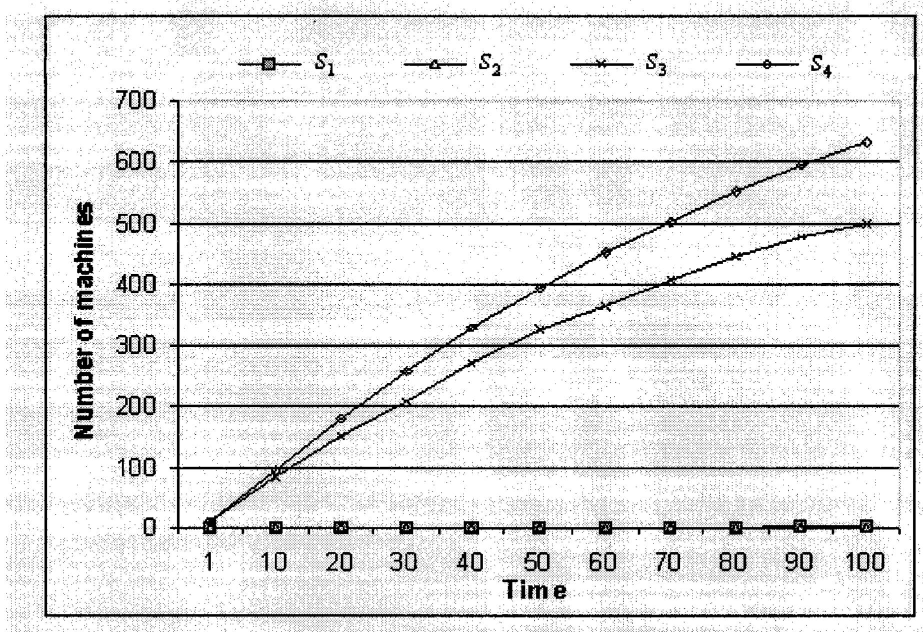
s_1 : Simulated number of infected machine

s_2 : Expected number of infected machine

s_3 : Simulated number of immunized machine

s_4 : Expected number of immunized machine

Figure 3.3: Simulation results and expected results when $B * \beta > \gamma$.



$\alpha = 0.01, \beta = 0.004, \gamma = 0.02$

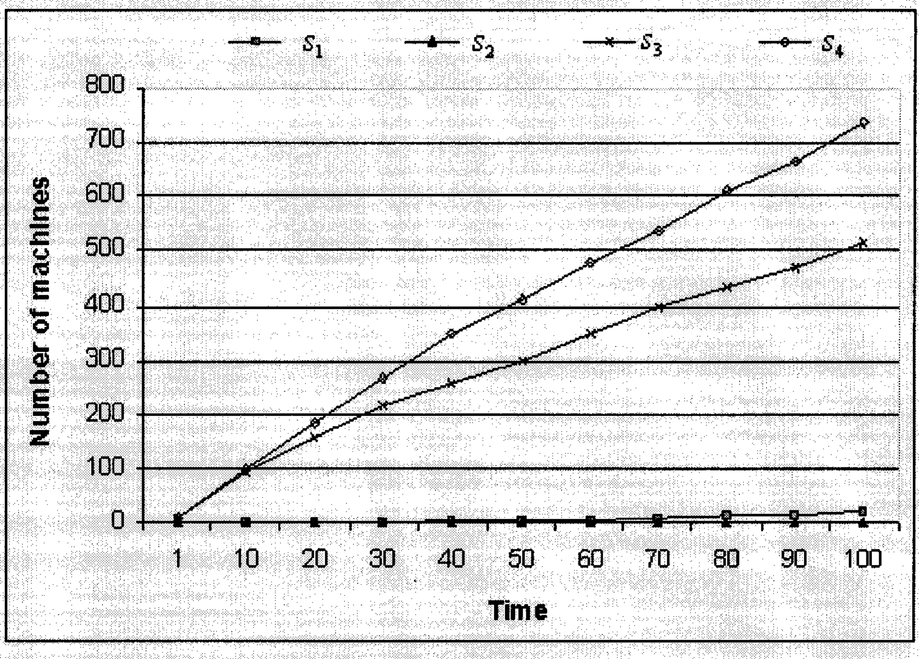
s_1 : Simulated number of infected machine

s_2 : Expected number of infected machine

s_3 : Simulated number of immunized machine

s_4 : Expected number of immunized machine

Figure 3.4: Simulation results and expected results when $B * \beta = \gamma$ **.**



$\alpha = 0.01, \beta = 0.01, \gamma = 0.06$

s_1 : Simulated number of infected machine

s_2 : Expected number of infected machine

s_3 : Simulated number of immunized machine

s_4 : Expected number of immunized machine

Figure 3.5: Simulation results and expected results when $B * \beta < \gamma$.

3.4 Summary

This chapter introduces the stochastic propagation modeling of a malicious mobile code. Instead of modeling the propagation in a homogeneous network, we introduce a new factor B which represents the average number of neighbors a machine could have. We proposed an INIM model propagation model which considers the immunization from healthy and infected machines. We then use the probability generation function method to obtain the expectation and variance of the number of infected machines at time t. The simulation result showed that it is effective to use our model to predict the propagation of malicious mobile programs, especially at the early stage. Later on, we may use the exact value instead of the approximation that $(i+m)/N \to 0$, so that the time parameter will have less effect on the prediction accuracy. Using the exact value of $(i+m)/N$ makes the equation much more complex and the solution becomes very difficult to obtain. We may explore other methods to solve the model in the future.

In the INIM model we proposed in this chapter, the infection rate, immunization rate, and recover rate are constant. This model could be refined by extending these parameters to be time dependent.

This chapter discusses only the propagation of malicious self-replicating programs. Similar models could be built to model the propagation of useful information through the network. For example, we could model the propagation of benign mobile code, also called "good virus". Analyzing the parameters that affect the propagation of benign mobile code could help us design future network that favors the propagation of such benign programs but throttle the propagation of those malicious programs.

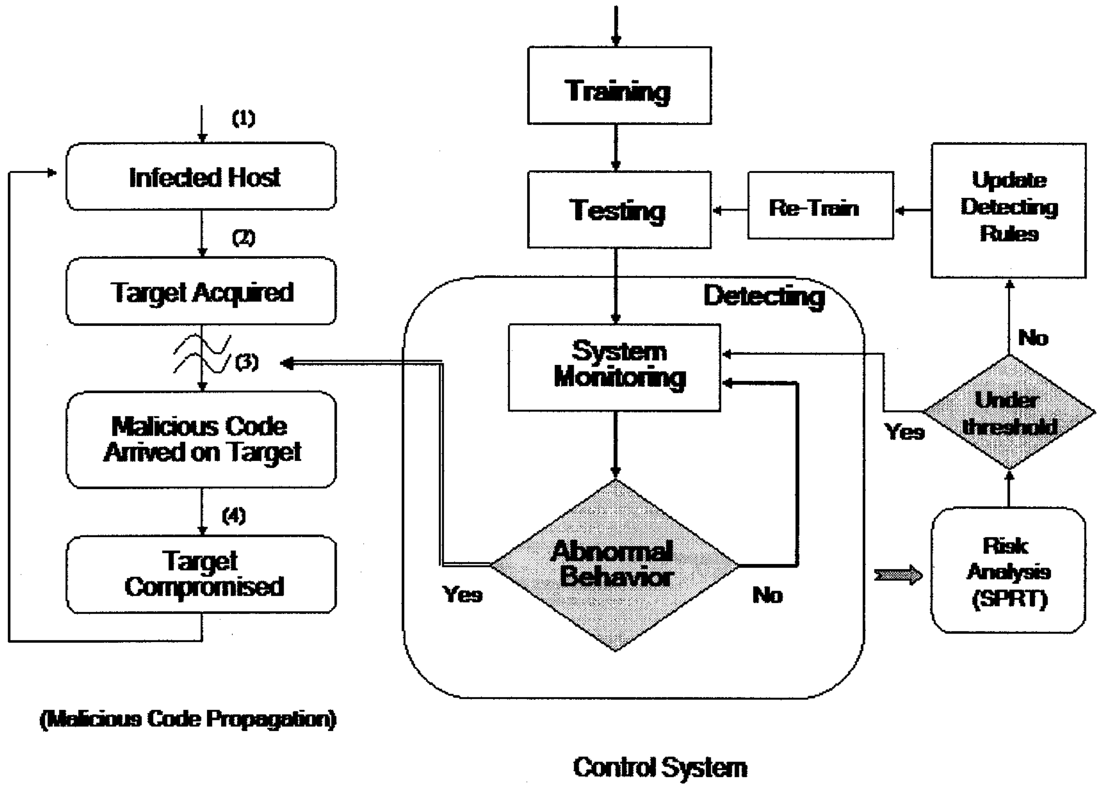
CHAPTER 4

EARLY DETECTION AND PROPAGATION MITIGATION OF MALICIOUS MOBILE CODE

This chapter proposes a control system to automatically detect and mitigate the propagation of malicious mobile programs such as computer worms at the early infection stage. The detection method is based on the observation that a worm always opens as many connections as possible in order to propagate as fast as possible. Therefore, we can monitor the connection rate to identify whether the status of a machine is normal or not. To develop the control system, we propose a detection algorithm, in which we provide an extension to the traditional statistical Process Control technique by introducing a sliding window. We apply sequential probability ration test to control the risk of the detection system so that the false positive rate is under certain threshold. We perform experiments to demonstrate the training phase and the testing phase of the control system using both real data and simulation data sets. Figure 4.1 shows the overall structure of the control system.

The experiment shows that by adjusting the tuning parameters appropriately, the control system can detect the propagation of malicious code with a zero false positive rate and less than 6% false negative rate, which asserts that our control system is effective.

We also analyze the propagation behavior of a network when the control system is applied to different proportions of the machines.



The steps of malicious code propagation: (1) Initial infection; (2) Acquire target; (3) Transfer malicious code; (4) Execute malicious code. When abnormal behavior detected, the control system will quarantine the infected host, therefore, no more machines will be infected.

Figure 4.1: The structure of the control system.

4.1 The Development of the Control System

Our control system includes a monitor and a controller. The monitor keeps track of the connection rate and reports it to the controller in real time. The controller makes the decision about whether or not there is an anomalous behavior, based on its knowledge from past experiences. This method belongs to *behavior blocking*, which is one of the anomaly detection methods. We extend the traditional Process Control technique to devise the detection algorithm for the controller. The general steps of building a control system are:

Step 1. Data collection:	Collect a normal data set for training
	Collect a normal data set for testing
	Collect an abnormal data set for testing

Step 2. Assumption checking: Check the normality assumption of the training data

Step 3. Training: Train the controller with the training data

Step 4. Testing: Test the control system using both the normal and the abnormal data.

Once these four steps are completed and the testing results are satisfying, the control system could be put into monitoring. The following subsections give the details of each step.

4.1.1 Data Collection

Since we are using the network connection behavior as an indicator of normal or anomalous behavior, we need to collect both normal and abnormal connection data, both from the same host. Under practical conditions, it is almost impossible to get connection data, both normal and abnormal, under the same circumstances, because we have no idea when there will be an outbreak of the malicious code. Fortunately, Goldsmith [Dave 2001] provided a connection request through TCP port 80 on July 18, 2001, the day when the network is normal (see Table 4.1), and on July 19, 2001, the day when Code Red broke out (see Table 4.2). The data was collected hourly.

The total connection rate T_c is defined as the total number of TCP connections that are built up in a given interval of time while the unique connection rate U_c is defined as the number of TCP connections built up to distinct destinations in a given interval of time. Thus, we have

$$T_c = \frac{c}{t}, \text{ where } c \text{ is the number of connections in time } t,$$

$$U_c = \frac{u}{t}, \text{ where } u \text{ is the unique number of connections in time } t.$$

For example, if a machine builds up *j* connections to the same destination in a given time interval *t*, the total connection rate is *j/t*, while the unique connection rate is *1/t*.

Table 4.1 gives the data from which we can calculate the average U_c, and it is about 17 connections per hour (*cph*) on July 18, 2001, while Table 2 gives the data from which we can calculate the average U_c, and it is about 37,549 *cph* on July 19, 2001. We plot the unique connection rate of both Table 4.1 and Table 4.2 in Figure 4.3. Here, we choose the unique connection rate instead of the total connection rate because a worm program always tries to connect to as many *new* hosts as possible. Connecting to a new host means opening a new connection from a local host to a remote machine. If we use total connection rate, sometimes the rate is high simply because we need to build more connections to the same remote machine, but not because of the propagation of malicious code. For example, when we browse a web page, we build up connections to the remote server through TCP port 80 of the local machine. If the web page contains more than one object, each object needs a TCP connection in order to make sure that the whole web page is viewed properly. Web pages are formatted in a markup language called HTML (*HyperText Markup Language*). Each picture file or audio file or video file is embedded as an object in the HTML file of that web page. It is common for a web page to contain more than one object and very few web pages contain only text. So when we open a web page with many objects, the total connection rate becomes high. However, since all these connections are to the same remote machine, the unique connection rate will not change unless we open new web pages that connect to another remote server.

Figure 4.2 plots the raw data from Table 4.1. We can see that the variation of T_c is much larger than U_c. Generally, we prefer sample data with smaller variations because small variation means that the data is more stable and hence the control system will give fewer false alarms.

From Figure 4.3, we can see that U_c was small on July 18, 2001, when the network was normal, then it grew tremendously when the malicious code started propagating at about 10:00 a.m. on July 19, 2001. Figure 4.3 reinforces the idea that we can detect the propagation of malicious code by monitoring the connection behavior, specifically, the unique connection behavior of a machine. We analyze the characteristics of the real data so that later on we can generate simulation data following the same distribution as that of the real data for training and testing purposes. Let Y be a discrete random variable that represents the number of the unique connections per hour, and $y_1, y_2, \cdots y_n$ be n observations. The mean value of the normal sample $\bar{y}$ is

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \qquad 4.1$$

The standard deviation of the sample s is calculated as

$$s = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \bar{y})^2}{n-1}} \qquad 4.2$$

We have 24 observations, so $n = 24$. Plugging the values of the unique connection rate from Table 4.1 into Equation 4.1 and 4.2, respectively, we get $\bar{y} = 17.62$ and $s = 3.23$.

Table 4.1: Connection attempts from a host on July 18, 2001

Hour	Total Connections	Unique Connections
0	143	20
1	148	15
2	89	15
3	96	18
4	144	22
5	127	16
6	98	15
7	111	16
8	116	15
9	149	22
10	143	18
11	175	24
12	134	22
13	146	20
14	118	21
15	95	17
16	133	22
17	104	17
18	78	17
19	76	15
20	67	15
21	85	15
22	62	12
23	105	14

Table 4.2: Connection attempts from a host on July 19, 2001

Hour	Total Connections	Unique Connections
0	120	17
1	81	12
2	62	11
3	97	20
4	85	18
5	128	20
6	140	20
7	212	34
8	645	137
9	5717	1281
10	36879	8186
11	150913	34361
12	362011	79789
13	519846	111148
14	556220	117946
15	547087	115193
16	540009	115983
17	519810	111290
18	499565	107106
19	390019	89331
20	14541	3493
21	9733	2233
22	9093	1882
23	8539	1672

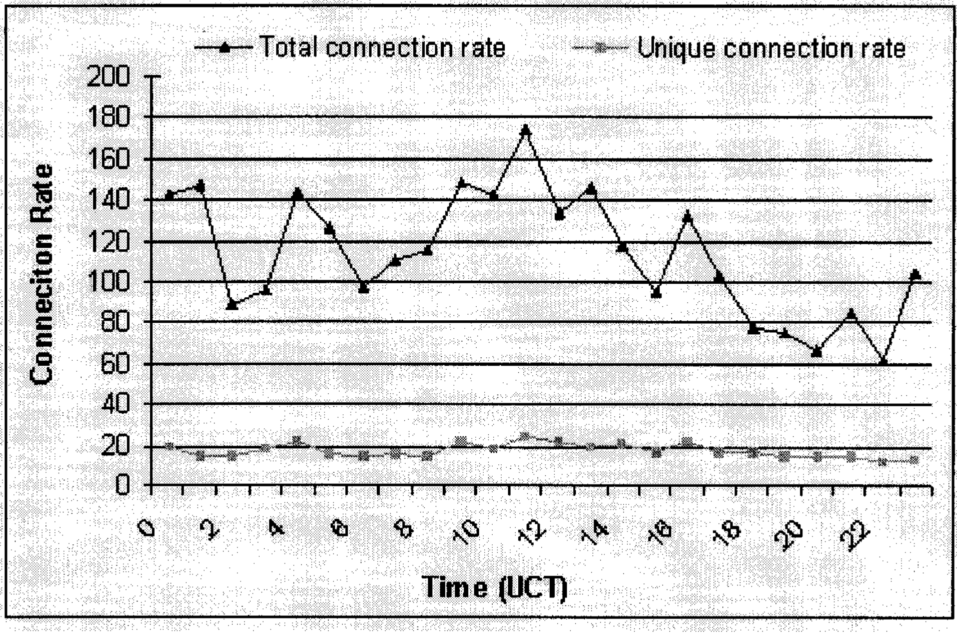


Figure 4.2: Total connection rate and unique connection rate of normal data. (UCT: United States Central time).

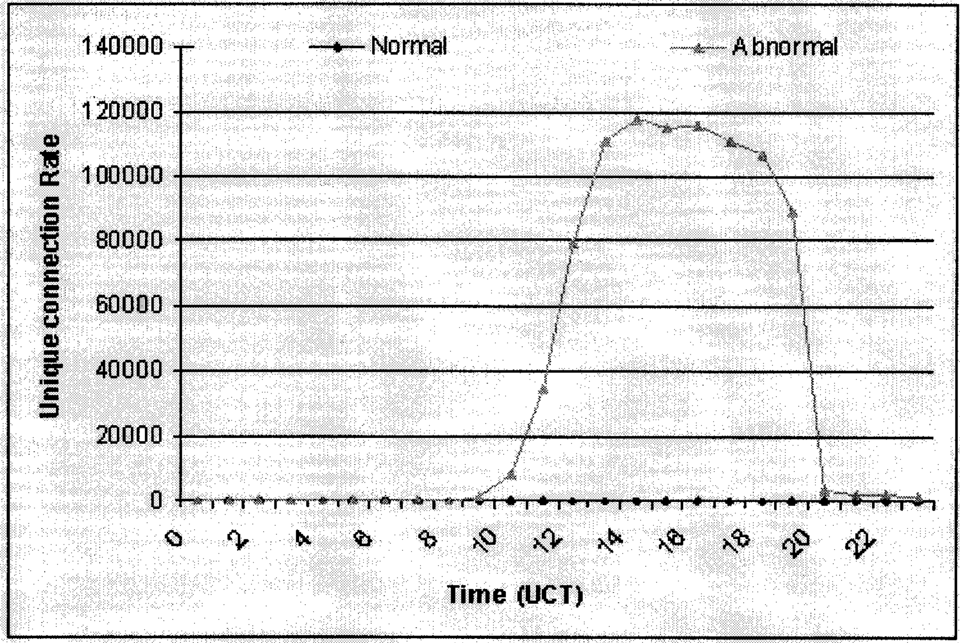


Figure 4.3: Normal and abnormal connection behavior. (UCT: United States Central Time).

4.1.2 Assumption Checking

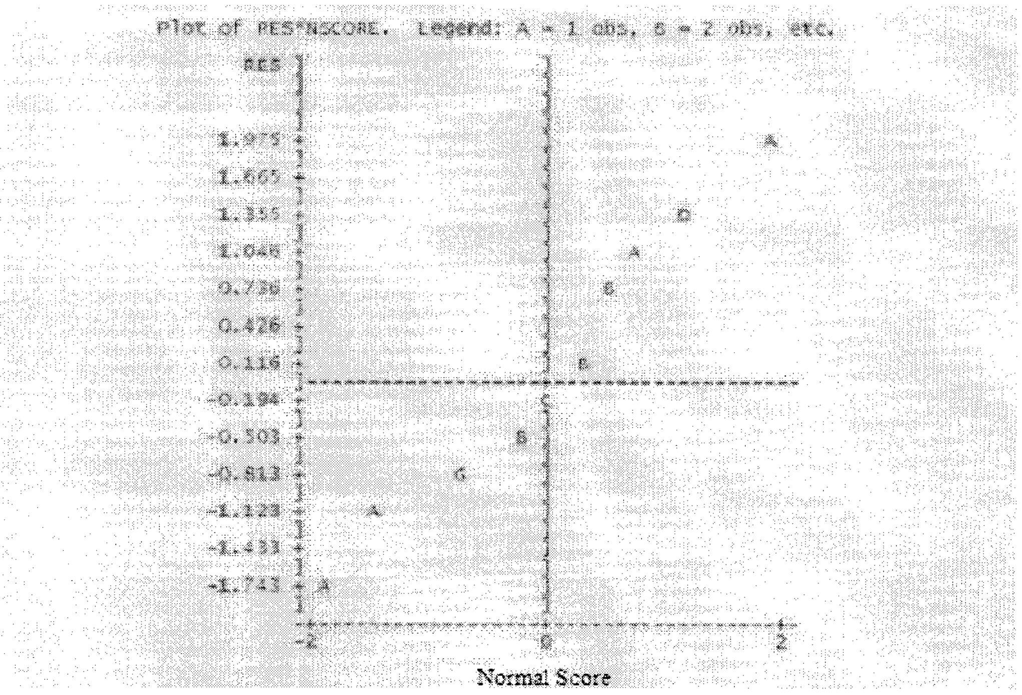
There are two reasons to check the normality assumption of the normal data: (1) To know the distribution of the real data so that we can generate a simulation data with the same distribution as that of the real data, to train the control system, and (2) To be able to apply the Process Control (or Quality Control) technique because the base data needs to satisfy the normality assumption.

We present the observed connection rate as $y_i = \bar{y} + \varepsilon_i$, where $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$ is the average of the sample data and ε_i is the error term. Thus, checking the normality assumption of y_i becomes checking the normality assumption of the residual ε_i. We use *normal probability plot,* which is a plot of standardized residual against their normal scores, to check the normality assumption of real data. Normal scores are the percentiles of the standard normal distribution. Statisticians [Dean 1999] found that if the normality assumption holds, a plot of the qth smallest standardized residual against the $100[(q-0.375)/(n+0.25)]$th percentile of the standard normal distribution for each $q = 1, 2, \cdots n$ would show points roughly on a straight line through the origin with a slope equal to 1.0. These percentiles are also called *Blom's normal scores*. Blom's qth percentile is the value ε_q for which

$$P(Z \leq \varepsilon_q) = (q-0.375)/(n+0.25)$$

where Z is a standard normal random variable. From the past experiences, statisticians conclude that normality plot is useful when sample size n is at least 15 [Dean 1999].
The normality plot generated by a SAS program is shown in Figure 4.4. In Figure 4.4, the Y-axis is the residual, and X-axis is Blom's normal score ε_q. The points shown in Figure

4.4 are roughly on a straight line through the origin, with slope equals to one. Although the line is not absolutely linear, it does not exhibit extremely heavy tails. Consequently, the normality assumption can be presumed to be approximately satisfied.



A means that there is one observation corresponding to that particular point;
B means that there are two observations corresponding to that particular point, and so on.

Figure 4.4: Normality plot of normal connection data on July 18, 2001.

<u>4.1.3 Simulation Data Generation</u>

The reason we use simulation data instead of collecting real data is that we can only collect normal connection behavior data, and it is almost impossible to get anomalous connection behavior data under the same circumstances, since the outbreak of malicious code does not happen very often and we have no idea when it will happen. The control system we propose is host-based, which means each user needs to install it on the local host to make it work. We cannot use the normal data of one host to train the control system and then put it to monitor the behavior of another host because the characteristics of the connection behavior are different for one host to another. The best we can do is to use the data provided by [Dave 2001], since this data is collected from the same host under the same circumstances.

To generate the simulation data with the normal distribution as that of the real data, we write a C++ program using the algorithm from Numerical Recipe [Vetterling 2002] which generates random numbers that follow a normal distribution. The function *Normal ( )* can generate random numbers with distribution $N(0, 1)$. We know that if a random variable X has a distribution of $N(\mu, \sigma^2)$, then $Y = \frac{X - \mu}{\sigma}$ has a distribution of $N(0, 1)$. So the random variable X can be written as $X = \sigma Y + \mu$. Therefore, we can generate sequences of random numbers with any normal distribution $N(\mu, \sigma^2)$ using function the *Normal* ( ) that generates random numbers with a distribution of $N(0, 1)$. Using this method, we generated a training data set and a testing data set.

4.1.3.1 Training data generation

We generate 30 training samples to represent the normal connection behavior of one month, each day with 24 elements, with each element representing the unique Internet connection rate (U_c) per hour. The elements of the training data set have a normal distribution of $N(17,\ 3^2)$, which is the same as the distribution that is obtained from the real data we presented in Table 4.1.

4.1.3.2 Testing data generation

Every time we run the data generation program, the program generates a testing data set with 1,000 samples that includes 600 normal samples and 400 abnormal samples.

Each sample has 24 randomly generated elements to represent the number of unique connections during each hour. For a better description, we number the samples from 0 to 999.

Sample 0–sample 599 simulate the normal connection behavior of a host.

The elements of sample 0–sample 199 follow the distribution of $N(17,\ 3^2)$.

To simulate a normal gradual increase of process mean,

the elements of sample 200–sample 249 follow $N(18.5,\ 3^2)$,

the elements of sample 250–sample 399 follow $N(20,\ 3^2)$,

the elements of samples 400–sample 449 follow $N(21.5,\ 3^2)$, and

the elements of samples 450–sample 599 follow $N(23,\ 3^2)$.

Sample 600–sample 999 simulate the abnormal connection behavior of a host. To simulate the change of the connection rate, in each of the abnormal samples, the first 8 elements still follow the distribution of normal connection behavior at $N(23,\ 3^2)$, and the rest of the elements have abnormal connection rates. To make the sample closer to real data, the value of the sample elements increase a little bit after the eighth element, and the value of the later elements keep increasing afterwards (as Figure 4.3 shows). To simulate the stealthy worm whose connection rate generally does not increase to an obvious high level, we generate random numbers from 29 to 39 as the abnormal elements of sample 600–sample 799. To simulate the connection behavior of most current worms, i.e., their attempt to connect to as many machines as possible, we generate random numbers from 40 to 100 as the abnormal elements of sample 800–sample 999. So in the simulation data, even the highest connection rate (which is 100) is much less than the connection rate of the real data (which is 1281, at 9 am, when the malicious code broke out) as seen in Table 4.2. But the connection rate is still high enough to demonstrate the efficiency of our control system. For each abnormal sample, when we generate the random numbers as abnormal elements, we sort them from small to large, so that each sample is similar to the real data shown in Table 4.2.

This kind of simulation is rough, but this is the best we can do based on the observation of connection behavior and the real data that we have. Figure 4.5 shows the overall distribution of the testing data sets by calculating the average of every 50 samples sequentially.

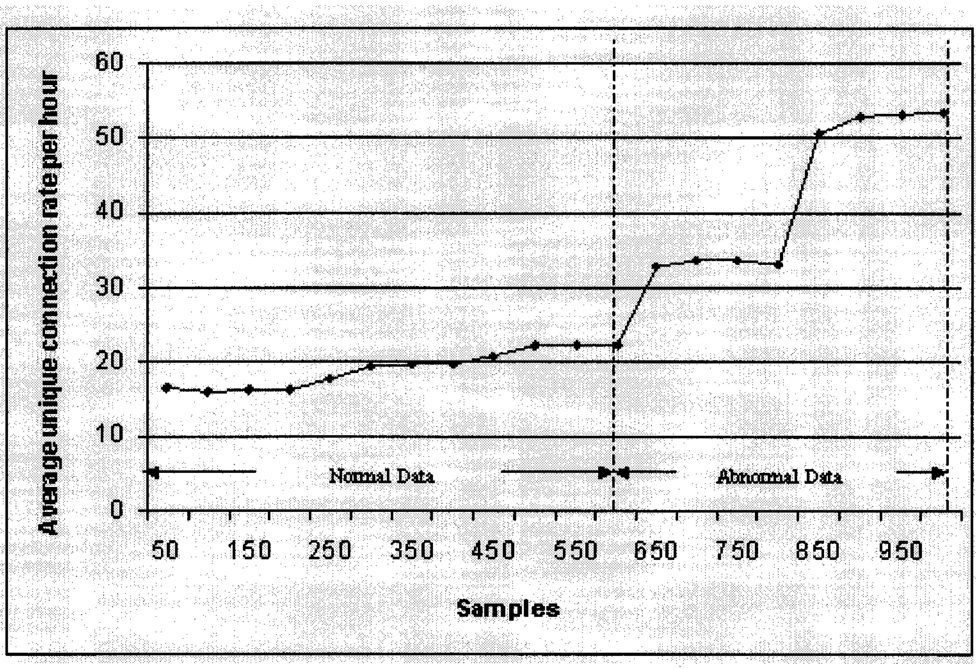


Figure 4.5: Distribution of testing data.

4.1.4 Detection Algorithm and Its Statistical Analysis

Traditional Process Control technique includes two stages. Stage one is called the *Base Period* during which the base data is collected and the normality assumption of the base data is checked as explained in Section 4.1.2. If the assumption is satisfied, we estimate the mean and variance and then calculate the control limit (CL) from the mean (μ) and variance (σ) as

$$CL = \mu \pm A\,\sigma$$

where A is a parameter determined by the control criteria. A bigger A makes the control limit wider. Consequently, the probability of making a type I error is smaller, but the probability of making a Type II error is greater.

Stage two is called the *Monitoring Stage* during which each new sample is collected and identified to see whether it is within the control limit or not. If the sample is beyond the control limit, a violation is detected. Figure 4.6 shows the procedure of the detection of the malicious mobile code using the traditional Process Control.

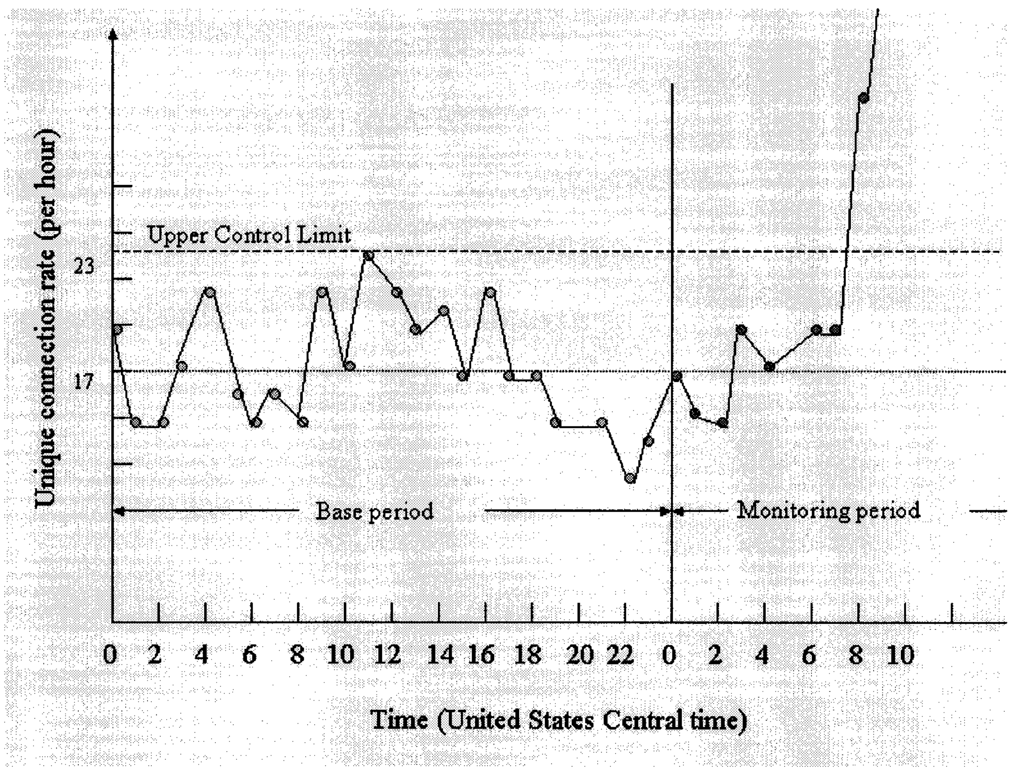


Figure 4.6: Traditional process control chart.

The traditional Process Control procedure makes the operation simple, but it is less adaptive to the changes developing in the process mean. Therefore, we extend the traditional Process Control technique by adding a sliding window with size w so that the base data always includes the most recent w observations. Whenever f new observations are collected, we move the window forward to include these f observations but still keep the size of the window the same. This way, the control system learns any change of the connection behavior and adapts to it by itself, and the false alarm rate is therefore reduced; f represents the updating frequency of the control limit. If f is set to be 1, the control limit is updated once a new observation is collected. High updating frequency (i.e. low value of f) adds unnecessary computation complexity to the system, while low updating frequency (i.e. high value of f) may lower the detection accuracy. Choosing an optimal frequency (f) is discussed in the experiment section.

Figure 4.7 gives the framework of building a control system for a given host using the extended Process Control technique. Figure 4.8 presents the monitoring procedure.

```
Algorithm: Early detection and propagation mitigation of malicious mobile code.
Procedure 1: Early Detection ( )
    Initialize $w$ and $f$; initialize i=0
    1. Collect $r$ samples as base data;
    2. Check the normality assumption of base data.
    3. If normality assumption is satisfied, continue to 4, otherwise stop.
    4. Estimate mean and variance of base data;
    5. Calculate the upper control limits;
    6. Set sliding window with the most $w$ samples of the base data.
```

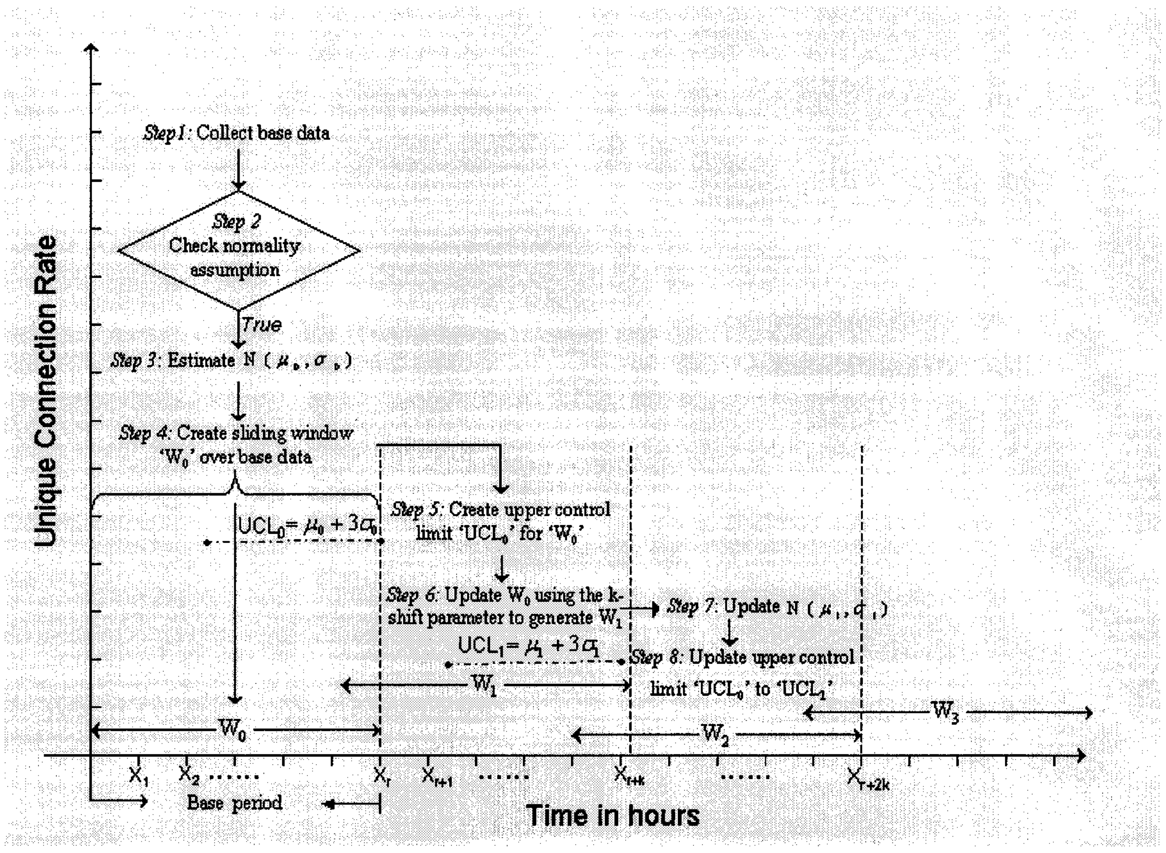
```
7. While system is online
  Do
  {
    Call Monitoring Procedure;
    If (i equals f)
    {
      Let i=0;
      Move the sliding window forward so that the base data includes the most recent
      f observations;
      Update mean, variance, and upper control limits;
    } //end if
    Otherwise i++;
  }
  End while
```

Figure 4.7: Early detection and propagation mitigation algorithm.

```
Procedure 2: Monitoring ( )
  Begin
    Collect new connection rate
    Check the control limit
    {
      If new observation is within control limit, back to while loop of Procedure 1.
      Otherwise, limit the outgoing connection and investigate the system
    }
  End.
```

Figure 4.8: Monitoring algorithm.

Figure 4.9 shows the control system using sliding window. We can see that the window with size *w* keeps moving forward as the monitoring period goes on. Each time when the sliding window has been moved forward, the mean and variance of the unique connection rate are updated.


Step1: Collect base data
Step 2 Check normality assumption
True
Step 3: Estimate N (μ0, σ0)
Step 4: Create sliding window 'W0' over base data
UCL0 = μ0 + 3σ0
Step 5: Create upper control limit 'UCL0' for 'W0'
Step 6: Update W0 using the k-shift parameter to generate W1
UCL1 = μ1 + 3σ1
Step 7: Update N (μ1, σ1)
Step 8: Update upper control limit 'UCL0' to 'UCL1'
W0
W1
W2
W3
X1 X2 …… Xr Xr+1 …… Xr+k …… Xr+2k
Base period
Time in hours
Unique Connection Rate

W_0, W_1 etc., are the sliding windows. Each time when f new observations are collected, the sliding window moves forward, and the upper control limit (UCL) is updated.

Figure 4.9: Flow chart of the detection algorithm.

4.1.4.1 Confidence interval for mean μ

Let $\overline{X}$ be the mean of a random sample of size n from a normal distribution with mean μ and standard deviation s, the random variable

$$T = \frac{\overline{X} - \mu}{s / \sqrt{n}} \qquad 4.3$$

has t distribution with $n-1$ degrees of freedom. The area between $-t_{\alpha/2,\,n-1}$ and $t_{\alpha/2,\,n-1}$ is $(1-\alpha)$ (area $\alpha/2$ lies in each tail), so probability

$$P(-t_{\alpha/2,\,n-1} < T < t_{\alpha/2,\,n-1}) = 1 - \alpha \qquad 4.4$$

Consequently, the $100(1-\alpha)\%$ confidence interval for μ is

$$(\overline{X}-t_{\alpha/2,n-1}\frac{s}{\sqrt{n}}, \overline{X}+t_{\alpha/2,n-1}\frac{s}{\sqrt{n}}). \tag{4.5}$$

Moreover, when the area of the upper tail is α , then probability

$$P(T<t_{\alpha,n-1})=1-\alpha. \tag{4.6}$$

So the upper confidence bound for μ is

$$\overline{X}+t_{\alpha,n-1}\frac{s}{\sqrt{n}}. \tag{4.7}$$

If we let α be 0.01, using Equation 4.5, the confidence interval for mean μ of normal unique connection rate is obtained as

$$(17.62-3.745\frac{3.23}{\sqrt{24}}, 17.62+3.745\frac{3.23}{\sqrt{24}}).$$

Simplifying it, we get the 99% confidence interval for mean μ as (15.77, 19.47), which means the probability that the mean value of the normal unique connection rate lies in the interval (15.77, 19.47) is 99%. Using Equation 4.7, the 99% upper confidence bound for μ is obtained as 19.27, which means the probability that the mean value of the normal unique connection rate is less than 19.27 is 99%.

4.1.4.2 Upper control limit

In the industry, if the random sample of a monitored process falls in the area of $\mu \pm 3\sigma$, we believe that the process is under control. In our case, we assume that the mean μ and standard deviation σ are known, and their values are 17.62 and 3.23, respectively. Hence, the control limit for the control system is $17.62 \pm 3 \times 3.23$. Since we know that when a malicious code propagates, it always increases the unique connection rate, then we only need to be concerned about the upper control limit. Therefore, when a random sample falls beyond $\mu + 3\sigma$ =27.31, an out of control signal will be given.

4.1.4.3 Level of significance

Let random variable Y denote the unique connection rate. It has normal distribution with mean $\hat{\mu}$ and variance $\hat{\sigma}$. $Y_1, Y_2, Y_3, \cdots, Y_n$ are the random samples of the unique connection rate. The monitoring period includes a hypothesis test at the given significant level α, that is:

Null hypothesis

$$H_0: \; Y_i = \hat{\mu}$$

Against alternative hypothesis

$$H_1: \; Y_i > \hat{\mu}$$

where $\hat{\mu}$ is the estimated mean value using the connection data in the sliding window.

The Test statistic TS is obtained as

$$TS = \frac{Y_i - \hat{\mu}}{s} \tag{4.8}$$

which has a Z distribution, so this hypothesis test is also called a Z test.

The rejection region for level α test is $TS \geq z_\alpha$, where z_α is defined as the point such that $P(Z > z_\alpha) = \alpha$; z_α is also called the critical value. An out-of-control signal occurs whenever a point falls in the rejection region, and an investigation for possible reasons should be initiated; α is the probability that we reject H_0 when H_0 is true, which means we identify a connection behavior to be an anomalous behavior since it is beyond the control limit when it actually is normal.

Traditionally, the control limit is defined as $\hat{\mu} \pm 3\hat{\sigma}$. But since malicious programs only increase but never decrease the connection rate, we are only concerned with the upper limit, which is defined as $\hat{\mu} + 3\hat{\sigma}$, and

$$P(Z > (\hat{\mu} + 3\hat{\sigma})) = 0.0013$$

Hence, the significance level α is 0.0013, which implies that the probability a normal connection rate falls beyond the UCL is 0.13%.

4.1.4.4 P-value of the hypothesis test

The P-value is the smallest level of significance at which H_0 would be rejected when a specified test procedure is based on a given data set. If the significance level is greater than P-value, the null hypothesis will be rejected; otherwise, the null hypothesis will be accepted. The P-value of a hypothesis test lets us know whether the null hypothesis is barely rejected or barely accepted by comparing the significance level α and the P-value. Figure 4.10 illustrates the P-value in a Z test where the P-value is greater than significance level α. So, TS does not lie in the rejection region and the null hypothesis will not be rejected in the example given in Figure 4.10.

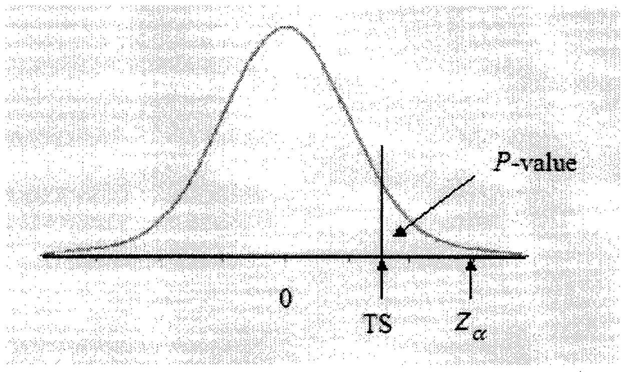


Figure 4.10: The *P*-Value of a *Z* test (TS: Test Statistic).

The P-value of a Z test can be obtained by checking a standard normal probability table. For example, if the connection rate is 24, using Equation 4.8, the test statistic is obtained as

$$TS = \frac{24-17.65}{3.23} = 1.966.$$

From the standard normal probability table, we approximately get

$$P(Z > 1.966) = 0.0247.$$

Hence, the P-value is 0.0247. Since the significance level we set is 0.0013, which is much less than 0.0247, we can strongly conclude that the null hypothesis H_0 should not be rejected.

4.1.4.5 Average run length

When a process is in control, we should observe many samples before we come across one sample that is beyond the control limit (a false alarm). Define a random variable *S*, such that $S =$ the first i for which Y_i falls outside the control limit.

If we think of each sample as a trial and an out-of-control sample as a success, then S is the number of trials necessary to observe a success. The expectation of S (*E(S)*) is called the Average Run Length (*ARL*), and *ARL* for a false alarm to appear could be obtained as

$$ARL = E(S) = \frac{1}{\alpha}. \quad 4.9$$

One reason why network users do not want to enable the intrusion detection systems is that the intrusion detection systems may give off false alarms and these false alarms are irritating to most people. Therefore, we need to make the false alarm rate as low as possible, i.e., to make the *ARL* value as large as possible.

For example, if the significance level of the hypothesis test is set at α=0.005, i.e., we take $\bar{Y}+2.575\hat{\sigma}$ as the upper control limit, then the false alarm rate is 0.5%. By calculating, we get

$$ARL = E(S) = \frac{1}{\alpha} = \frac{1}{0.005} = 200,$$

which means when the process is under control, one false alarm could happen for every 200 observations.

The control system we designed takes $\bar{Y}+3\hat{\sigma}$ as the upper control limit; therefore, the hypothesis test is at significance level $\alpha = 0.0013$. Hence, we could get the Average Run Length as

$$ARL = E(S) = \frac{1}{\alpha} = \frac{1}{0.0013} = 769.23,$$

which means when the process is under control, at the utmost, one false alarm could happen for every 769 observations if we set the upper control limit of the connection rate at 27.31.

4.2 Experiments

This section presents the details of building a control system using the simulation data. The purpose of the experiments is:

- To demonstrate the training process and the testing process of the control system.
- To test the reliability and the adaptability of the control system.
- To give a quantitative evaluation of how sliding window improves the performance.

- To evaluate the effect of tuning parameters like, the sliding window size (*w*) and the updating frequency (*f*).

4.2.1 Training

Before we train the control system, we should check whether or not the training data follows normal distribution. Since we already checked the normality assumption of the real data and the simulation data is generated following the same distribution as that of the real data, we know that the training data also satisfies the normality assumption. So we feed the training data to the controller and the controller learns the mean, variance, and upper control limit from it.

4.2.2 Testing

We build two controllers: controller one uses the traditional Process Control technique; controller two uses the extended Process Control technique with the sliding window. We perform experiments using both controllers. This section shows the reliability of the control system and the performance of each controller. We also discuss the choice of the optimal tuning parameters by testing their effects.

4.2.2.1 Performance analysis of the control system

The performance of the control system can be analyzed by comparing the false negative and the false positive rates of the two controllers (one using the traditional Process Control and the other using the extended Process Control).

False Negative:

We first train the two controllers with the training data set. Then we run the controller program 100 times to test the reliability, each time with a different testing data set. Both controllers detect all anomaly samples when malicious code starts spreading, so the false negative rate of both of them is zero.

False Positive:

Figure 4.11 plots the false positive rate of each experiment run. The false positive rate of controller one is about 30% to 35% and the false positive rate of controller two is lower than six percent. Therefore, we can demonstrate that the false positive rate is greatly reduced when we apply the sliding window.

From Figure 4.11, we can also see that the performance of controller two is quite reliable. It has an average false positive rate of 4.35% and no false positive rate higher than 6% in any single run. Therefore, we conclude that the control system using the extended Process Control technique is effective in detecting the propagation behavior of malicious codes. The sliding window plays an important role in reducing the false positive rate when the process behavior changes. The effectiveness of the extended Process Control can be attributed to this introduction of a sliding window, which causes the control system to be more adaptable to the changes of the connection behavior.

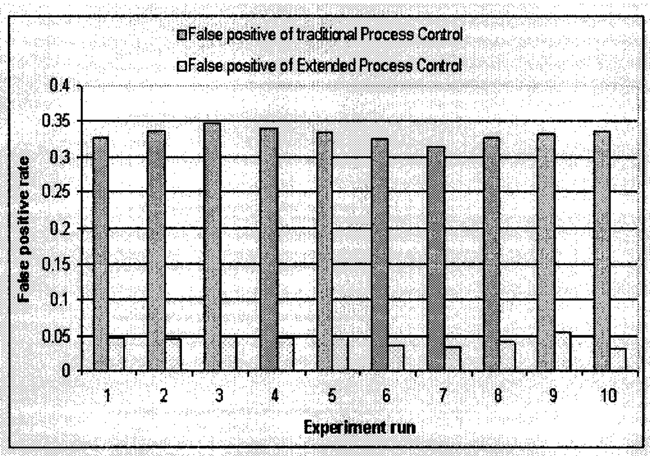


Figure 4.11: Reliability of the control system applying traditional and extended Process Control technique (w = 600, f = 20).

4.2.2.2 The effect of sliding window *w*

If the size of the sliding window *w* is too small, it will result in an inadequacy of the model to properly represent the system dynamics, and will therefore lead to a poor general performance. Conversely, if the window is too larger then the computational complexity is unnecessarily increased. To choose an optimal value for *w*, we perform an experiment with different sizes of the sliding window. For each size of the sliding window, we run the experiment ten times and get the average false negative rate and the false positive rate from these ten runs. Figure 4.12 and Figured 4.13 plot the average false negative rate and false positive rate respectively when the window size varies from 20 to 720.

Figure 4.12 shows that when *w* increases, the false negative rate decreases. The control system gets zero false negative rate when *w* is greater than 480. Figured 4.13 shows that when *w* is too small, the false positive rate is high, but it converges quickly to about 6% when *w* increases above 40. The overall performance analysis of the control system we presented in Figure 4.11 is given at *w* = 600.

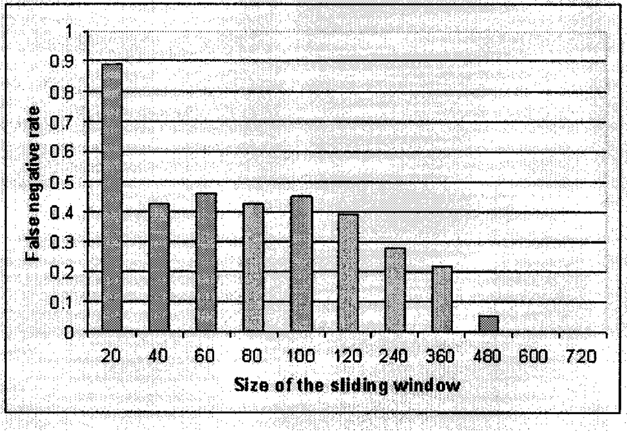


Figure 4.12: False negative rate of the control system when the size of sliding window *w* varies (The value of f is fixed at 1/ 20).

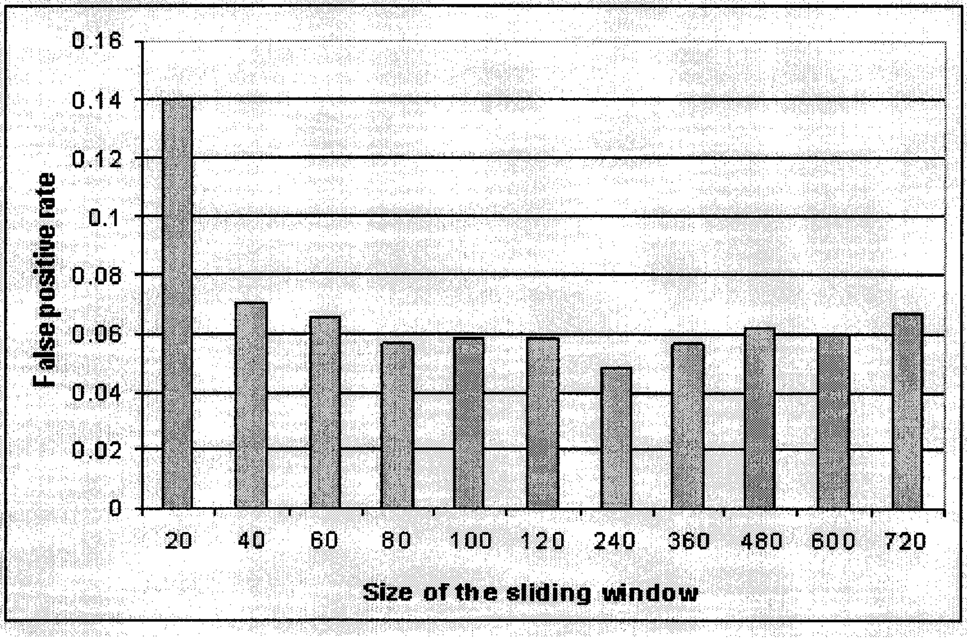


Figure 4.13: False positive rate of the control system when the size of Sliding window *w* varies (The value of f is fixed at 1/ 20).

4.2.2.3 The effect of updating frequency *f*

Let the number of new observations in the sliding window to be o_s, then the updating frequency *f* is defined as $f = \frac{1}{o_s}$. When *f* is larger, the control system moves the sliding window more frequently. Since the control rules are updated each time the sliding window is moved forward, the computation complexity is increased. If *f* is too small, the control system cannot catch the dynamic change of the connection behavior in real time. Hence, the performance of the control system is degraded.

Figure 4.14 and Figure 4.15 plot the average false positive rate and false negative rate respectively when *f* varies. Figure 4.14 shows that the false negative rate becomes zero when $1/f \geq 16$. Figure 4.15 shows that the false positive rate is stabilized when $f \leq 26$ but increased dramatically when $1/f > 26$. Overall, the optimal *f* value is between

1/16 and 1/26. The overall performance analysis of the control system we presented in Figure 4.11 is given at $1/f = 20$.

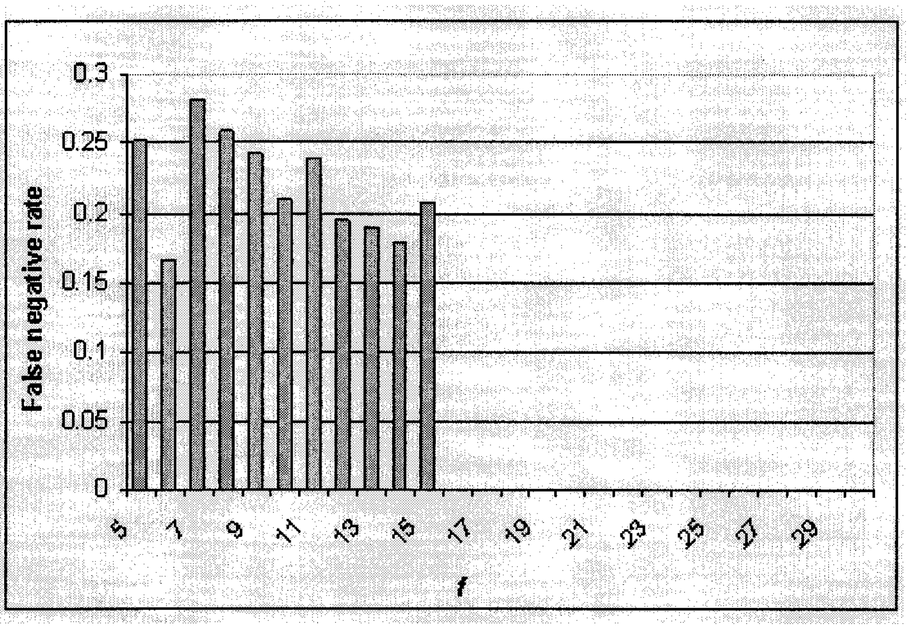


Figure 4.14: False negative rate when *f* varies (The value of w is fixed at 600).

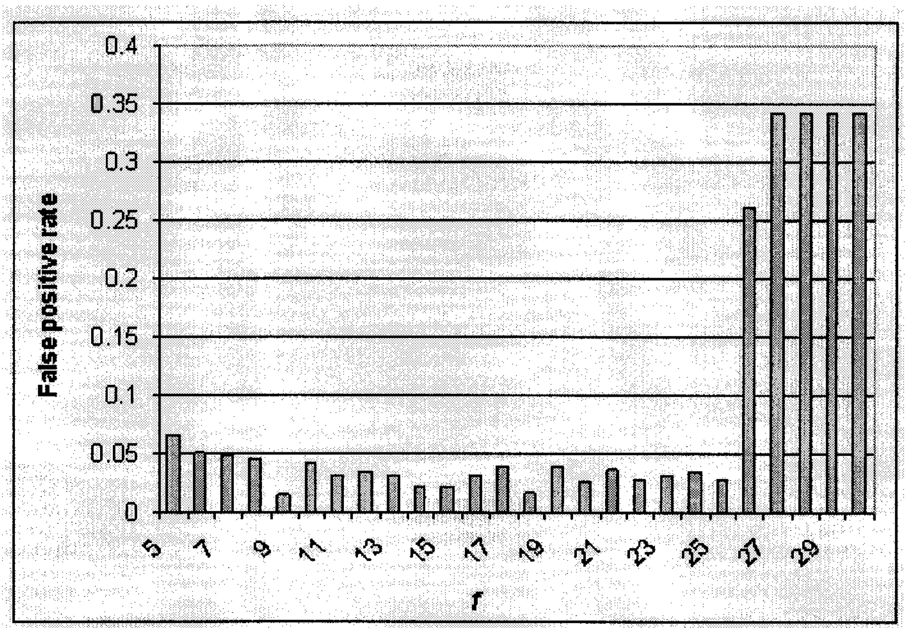


Figure 4.15: False positive rate when *f* varies (The value of w is fixed at 600).

4.3 Risk Analysis of the Control System

All intrusion detection systems have a false positive rate. Therefore, there is always a probability that a detection system detects a propagation that does not exist at all. In statistics, this is also called producer's risk. For an intrusion detection system, it is extremely important to reduce the producer's risk because it may lead to a complete rejection of usage. This section analyzes producer's risk and introduced a novel idea of using Sequential Probability Ratio Test (SPRT) to control the risk.

4.3.1 Risk Analysis Using SPRT

Let the inspection result of the ith unit be denoted as X_i, then X_i=1 if there is malicious code propagation detected; X_i= 0 otherwise.

Let f represents the probability function of X, then

$$f(1,p) = p \text{ and } f(0,p) = 1-p.$$

Here, p is interpreted as the false positive rate, that is, the probability of detecting a malicious behavior of a healthy machine.

To test the hypothesis of $H_0 : p = p_0$ against $H_1 : p = p_1$, let p_{0m} and p_{1m} be the probability of getting d_m false detection in the sample $(X_1, X_2, \cdots, X_m)$ of size m under H_0 and H_1, respectively. Then the Likelihood Ration

$$\lambda_m = \frac{p_{1m}}{p_{0m}} = \frac{\prod_{i=1}^{m} f(x_i, p_1)}{\prod_{i=1}^{m} f(x_i, p_0)} = \frac{p_1^{dm}(1-p_1)^{m-dm}}{p_0^{dm}(1-p_0)^{m-dm}} \tag{4.10}$$

$$\Rightarrow \log \lambda_m = d_m \log \frac{p_1}{p_0} + (m - d_m) \log\left(\frac{1-p_1}{1-p_0}\right) \tag{4.11}$$

The SPRT for a hypothesis $H_0 : p = p_0$ against its alternative $H_1 : p = p_1$ is carried as follows:

If $\log \lambda_m \geq A$, reject H_0 and terminate the process.

If $\log \lambda_m \leq B$, accept H_0 and terminate the process.

If $B \leq \log \lambda_m \leq A$, collect observation X_{m+1}, calculate $\log \lambda_{m+1}$ and compare the value of $\log \lambda_{m+1}$ with A and B again.

Where A and B are constants defined as

$$A = \log \frac{1-\beta}{\alpha} \text{ and } B = \log \frac{\beta}{1-\alpha}. \qquad 4.12$$

If we write

$$g_1 = \log \frac{p_1}{p_0}, \; g_2 = \log\left(\frac{1-p_0}{1-p_1}\right) = -\log\left(\frac{1-p_1}{1-p_0}\right), \qquad 4.13$$

then Equation 4.11 becomes

$$\log \lambda_m = d_m g_1 - (m - d_m) g_2 . \qquad 4.14$$

So, we reject H_0 if

$$d_m g_1 - (m - d_m) g_2 \geq A \Rightarrow d_m \geq \frac{A}{g_1 + g_2} + m \frac{g_2}{g_1 + g_2}. \qquad 4.15$$

We accept H_0 if

$$d_m g_1 - (m - d_m) g_2 \leq B \Rightarrow d_m \leq \frac{B}{g_1 + g_2} + m \frac{g_2}{g_1 + g_2}. \qquad 4.16$$

Then, the rejection line L_1 is

$$d_m = h_1 + sm , \qquad 4.17$$

and the acceptance line L_2 is

$$d_m = h_2 + sm . \qquad 4.\ 18$$

where $h_1 = \dfrac{A}{g_1 + g_2}$, $h_2 = \dfrac{B}{g_1 + g_2}$, and $s = \dfrac{g_2}{g_1 + g_2}$. 4.19

From Equation 4.17 and Equation 4. 18, we can see h_1 and h_2 are the interception of line L_1 and line L_2 on the d_m axis respectively, while s is the slope of both lines. Below is the Risk Control Algorithm using line L_1, L_2 and point (m,d_m). And Figure 4.16 is the control chart based on the risk control algorithm.

Risk Control Algorithm

1) Determine p_0, p_1, α, β;
2) Calculate A and B using Equation 4.12; Calculate g_1 and g_2 using Equation 4.13;
3) Get the value of h_1, h_2 and s using Equation 4.19;
4) Draw rejection line L_1 and acceptance line L_2 using s as slope and h_1, h_2 as interception for L_1 and L_2 respectively as Figure 4.16 shows;
5) Get the m^{th} sample, count d_m;
6) Plot point (m,d_m);
7) If point (m,d_m) lies between L_1 and L_2, back to 5; otherwise, stop.

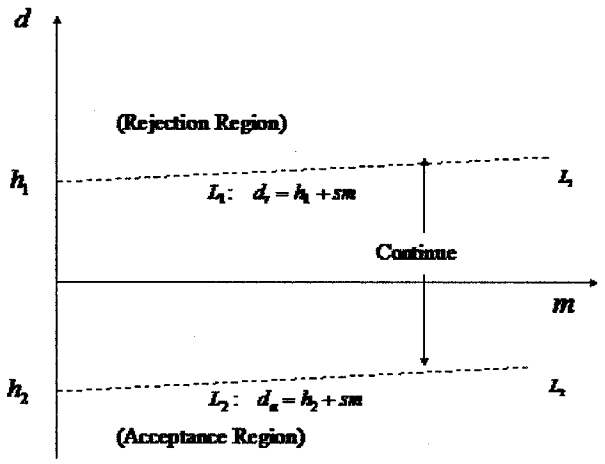


Figure 4.16: Risk control chart. If point (m,d_m) lies below L_2, accept H_0; if it lies above L_1, reject H_0; otherwise, keep sampling.

4.3.2 Examples of Risk Control Using SPRT

Let the false positive rate be p_0, since we are using a one-sided control process with 3σ as the control limit, p_0=0.001. To test the hypothesis that the false positive rate is p_0, we have

$$H_0 : p = p_0 \text{ against } H_1 : p = p_1$$

Let $p_1 = 0.002$, $\alpha = 0.01$, $\beta = 0.05$, we get

$$A = \log\frac{1-0.05}{0.01} = 1.978$$

$$B = \log\frac{0.05}{0.99} = -\ 1.297$$

$$g_1 = \log\frac{0.002}{0.001} = 0.301$$

$$g_2 = -\log\frac{1-0.002}{1-0.0013} = 4.349\times10^{-4}$$

$$h_1 = \frac{A}{g_1+g_2} = 6.563$$

$$h_2 = \frac{B}{g_1+g_2} = -\ 4.303$$

$$s = \frac{g_2}{g_1+g_2} = \ 0.0014.$$

Therefore, the rejection line L_1 is: $d_r = 6.563 + 0.0014m$; the acceptance line L_2 is

$$d_a = -4.303 + 0.0014m.$$

Table 4.3 lists two examples of using SPRT to control the risk of false positive. We plot the acceptance example on Figure 4.17. From Figure 4.17, we can see that points lies in the middle until m = 3800.lies acceptance region when, which means we should accept hypothesis H_0. In other words, the false positive rate is the same as we expected from the theory. We also take one sample from the stealthy malicious code whose propagation is hard to observe because of the low connection rate. Using the detection result of the stealthy malicious code, the sequential ratio test is processed as Table 4.3 Example 2 shows. Plot Example 2 on Figure 4.18, we can see when m = 760, the point lies in the rejection region, which means H_0 is rejected and we should accept the alternative. Therefore, we should adjust the control limit so that the false positive rate could be reduced to what we expected.

Table 4.3: Risk analysis using SPRT

Example 1: An Acceptance Example

m	d_m	d_a	d_r
0	0	-4.303	6.563
1	0	-4.3016	6.5644
10	0	-4.289	6.577
100	0	-4.163	6.703
1000	0	-2.903	7.963
1500	1	-2.203	8.663
2000	1	-1.503	9.363
3000	1	-0.103	10.763
3600	1	0.737	11.603
3700	1	0.877	11.743
3800	1	1.017	11.883

Example 2: A Rejection Example

m	d_m	d_a	d_r
0	0	-4.303	6.563
1	0	-4.3016	6.5644
10	0	-4.289	6.577
100	1	-4.163	6.703
200	2	-4.023	6.843
300	3	-3.883	6.983
400	3	-3.743	7.123
500	4	-3.603	7.263
600	5	-3.463	7.403
700	7	-3.323	7.543
760	8	-3.183	7.683

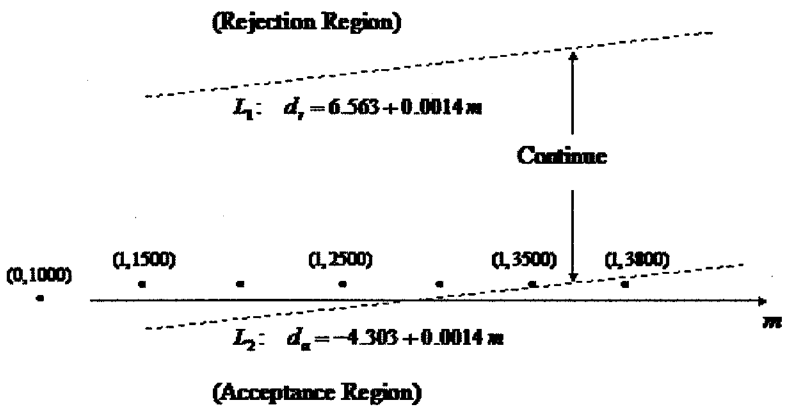


Figure 4.17: An acceptance example of risk control (Since point (1, 3800) falls in the acceptance region, we should accept the hull hypothesis).

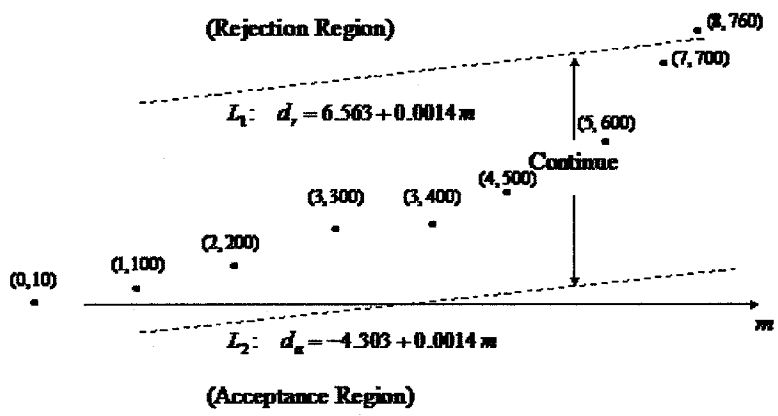


Figure 4.18: A rejection example of risk control (Since point (8, 760) falls in the rejection region, we should reject the hull hypothesis).

4.4 Network Performance Analysis

This section analyzes the propagation behavior of malicious code when the control system applied. We compare the performance of a network when different number of machines applied the control system. The performance is evaluated by the mathematical propagation models. We extend the propagation models introduced in Chapter 2 by introducing a new factor, which we will give the detail in the following section. We also design a serial of propagation simulations in a network with control system. The simulation result is consistent with the theoretical result, which implies the success of the control system. The notation we use in this section follows the same definition given in Chapter 2.

4.4.1 Extended SI Model

In Chapter 2, we have presented the standard SI model that assumes the population is homogeneously mixed, which is not true in the real world. In Chapter 3, we have introduced a new factor B, the average number of contactors an individual has in a given time period, into the stochastic propagation model. Here, we will present the deterministic SI model with factor B.

Assuming we have only one machine infected at time zero, the total number of infected individuals can be modeled as:

$$\frac{dI(t)}{dt} = B\frac{N - I(t)}{N}\beta I(t) \quad \text{with } I(0) = 1 \tag{4.20}$$

Let $i = I/N$ and $k = \beta \cdot B$, dividing both sides of Equation 4.20 by N, we get

$$di(t)/dt = k(1 - i(t))i(t) \tag{4.21}$$

The solution of the general epidemic model given as Equation 4.21 is

$$i(t) = \frac{e^{kt}}{N - 1 + e^{kt}} \tag{4.22}$$

where k is the infection rate. Infection rate is the number of machines that could be infected by one infectious machine in one time unit.

4.4.1.1 Fitting the observed data with extended SI model

The computer worm Code Red exploits the buffer-overflow vulnerability in Microsoft's IIS web server [Moore 2002]. The propagation of Code Red is noticed at about 10:00 am (central time) on July 19. The propagation stops at 12:00 midnight by the designing of Code Red. It infected more than 359,000 machines during approximately 13 hours of propagation [Moore 2002] [Caida 2001]. Code Red randomly generates 100 threads; each thread randomly chooses one IP address and connects to the machines with corresponding IP addresses through port 80 [Zou 2002]. It installs the mechanism for remote, administrator-level access to the infected machine so that the machine could be used to execute any code [Moore 2002]. Therefore, it is highly dangerous. Figure 4.19

shows the observed Code Red propagation. The infection came to saturating around 20:00, and the propagation stops at 12:00 midnight, so the number of infected host in Figure 4.19 does not change after 12:00 midnight.

Assume that the total number of vulnerable hosts is 400,000. If we fit the model with $k = 0.8$, we get the theoretical result of Code Red propagation in Figure 4.20.. Compare Figure 4.19 and Figure 4.20, we conclude that even the simple SI epidemic model provides a reasonably good approximation of malicious mobile code propagation. The observed value does not grow as smooth as the theoretical model because the network bandwidth is exhausted by the malicious code so that it cannot connect and infect the target machine as it does at the early infection stage.

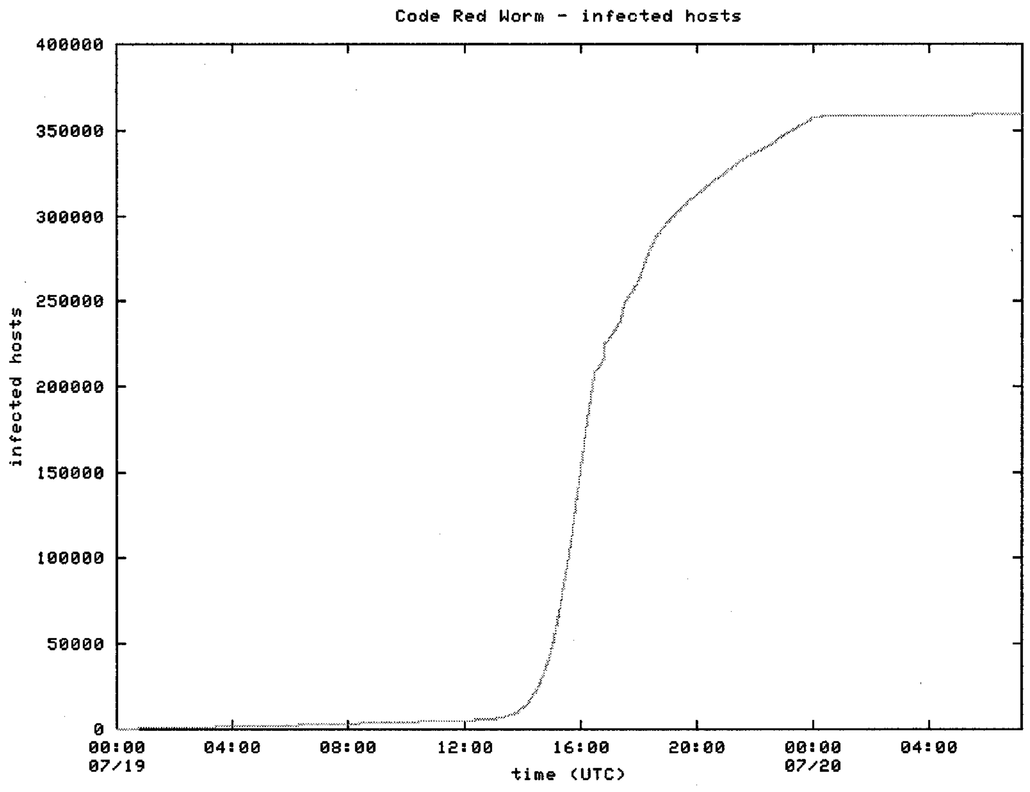


Figure 4.19: Observed Code Red propagation (From www.Caida.org).

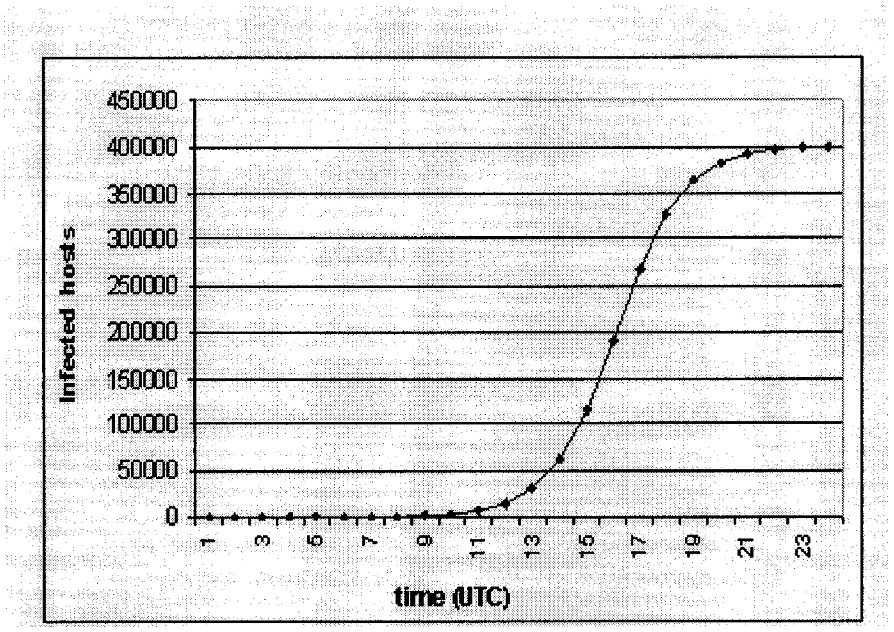


Figure 4.20: Theoretical result of Code Red propagation.

4.4.2 Propagation Modeling with Control System

As we discussed in Chapter 2, when propagation starts, the speed is slow at the very beginning, then it comes to the fast spreading stage during which the number of infected nodes grows tremendously, and finally the infection speed slows down since very few susceptible nodes are still available. At the first and the second stages, the number of infected nodes grows almost exponentially. If all hosts use the control system, the infection speed could be greatly reduced and the infection may never get to the third stage because countermeasures taken by humans could immunize nodes when they are healthy. In the following subsection, we present the propagation model with the control system and give a quantitative evaluation of the effects of the control system. We do not consider immunization and recover in this model.

When a malicious code detection system is applied in a network, a healthy host might be flagged as infected (False Positive), and an infected host might not be detected and therefore declared as healthy (False Negative) (See Figure 4.21). A false positive will not affect the propagation of a malicious code, so we do not need to consider it in the propagation model. Suppose p percent of the hosts installed the control system and the detection rate of the control system is d. We know that the false negative rate is $1-d$.

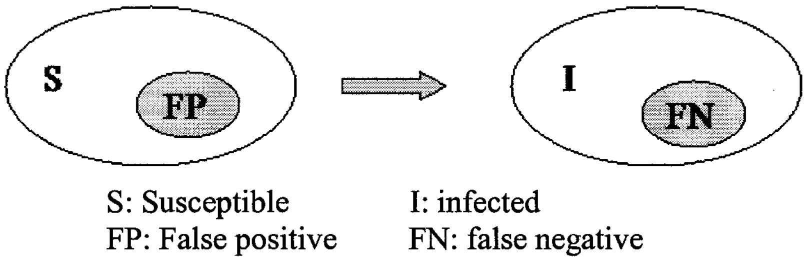


Figure 4.21: State transition of extended SI model with detection system.

When a machine is detected with malicious code, its connection rate is limited. Ignoring the one unit detection delay, we have

$$\frac{dI(t)}{dt} = I_1 \beta B_1 \frac{N - I(t)}{N} + I_2 \beta B_2 \frac{N - I(t)}{N} \qquad 4.23$$

where $I_1 = I \cdot p \cdot d$, $I_2 = I \cdot (1-p) + I \cdot p \cdot (1-d)$; B_1 is the limited number of contactors a machine could have when infection is detected; B_2 is the number of contactors a machine could have when infection is not detected, and $B_2 >> B_1$.

Let $k_1 = \beta \cdot B_1$, $k_2 = \beta \cdot B_2$, $i = I/N$, $i_1 = I_1/N$, $i_2 = I_2/N$, then from Equation 4.23, we obtain

$$di/dt = k_1(1-i)i_1 + k_2(1-i)i_2 \qquad 4.24$$

The solution of Equation 4.24 is

$$i(t) = \frac{e^{\lambda t}}{N - 1 + e^{\lambda t}} \quad 4.25$$

where $\lambda = k_1 \cdot p \cdot d + k_2(1 - p \cdot d)$.

Figure 4.22 shows the infection delay when a different percentage of hosts install the control system. Obviously, the more hosts installing the control system, the better the results are. If only 20% of the hosts adopt the control system, the overall effect is very limited. When p is about 80% or 90%, the difference is huge.

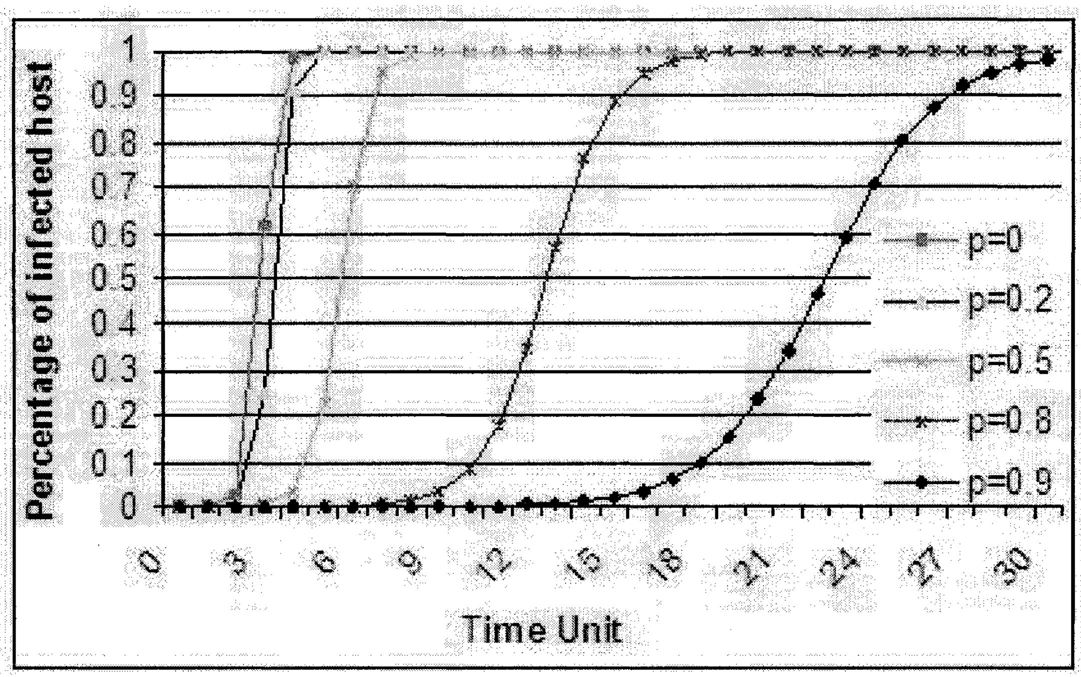


Figure 4.22: Infection evolution with different p values (N = 10,000, β = 0.8, d = 1.0).

4.4.3 Simulation

This section conducts simulation experiments to verify the prediction of the spreading speed and scale given by the models. We first simulate a simple epidemic propagation model, and then we simulate the propagation model with the control system.

4.4.3.1 Simulation setup

Our simulation program includes a network generator and an infection process. Every time we run the simulation, the network generator will generate a random network with 5,000 nodes. Each node has a switch. If the switch is on, the control system is turned on; otherwise, the control system is off. The number of neighbors has a uniform distribution of (1, 20). The neighbors of each node are randomly generated. Neighbors of a given node are defined as the nodes that the given node will contact in an infection process. Figure 4.23 gives the general structure of each node in C++ style. Once node *j* is randomly chosen as the neighbor of node *i*, we also add node *i* as node *j*'s neighbor.

General Structure of Each Node in the Random Network

```
struct node {
  // declare the Unique ID of the node
  int id;

  //the number of neighbors allows; this number has uniform distribution
  int egNo;

  //node status; 1 = healthy; 0 = infected; -1 = immunized
  int status;

  // switch of the control system,
  bool control; // control=1, control system on; control = 0, no control system

  //to specify the node is infected or not in current dt or before
  bool newInf;

  // store the id of its neighbors
  int array nborID[ ];

  // the total number of neighbors
  int nborNo;

  } End node
```

Figure 4.23: General structure of each node.

One node is randomly selected as the initially infected node. We use time step to represent the time unit dt in the propagation models. During each time step, each infected node tries to infect its neighbors with pair wise infection rate β.

4.4.3.2 Propagation simulation of SI model

To simulate the simple epidemic model without the control system, we just turn off the switch. We run the simulation 100 times. Figure 4.24 plots the infection process by taking the average of all simulation runs. The simulation is a little slower than the theoretical model at the early infection stage. Overall, the simulation results are close to the theoretical results, so we conclude that simple epidemic model given by Equation 4.20 matches the general propagation phenomena. Since the real data have validated the accuracy of the theoretical model as shown in Section 4.4.1, and the simulation result is close to the theoretical model, we conclude that the simulation does approximate the real infection phenomenon. Therefore, we are confident that our further simulations approximate the real propagation scenarios.

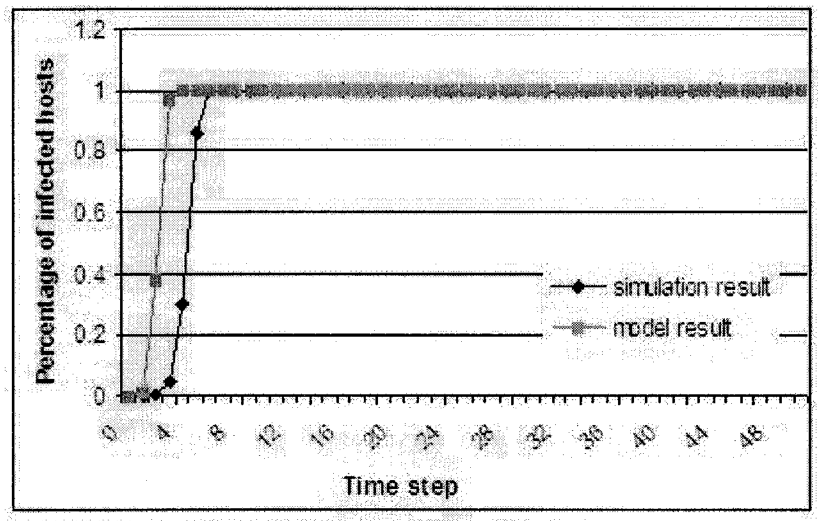


Figure 4.24: Simulation results and theoretical results of SI model ($k = 0.4, p = 0$).

4.4.3.3 Propagation simulation with control system

We simulate four cases of worm propagation with control system. In each case, the percentage of nodes with control system on is different. Every time we run the infection simulation, we randomly choose p percent of the nodes with control system turned on. During propagation, an infected node tries to infect its neighbors with a lower infection rate (k value) if the switch is on; otherwise, we keep using the same infection rate (k value) as in Section 4.4.3.2. We run the simulation 100 times for each case. Table 4.4 shows the average time units needed to infect certain percentage of the network nodes under different cases.

Table 4.4: Simulation results of four different p values

Case No.	p	T1	T2
1	20%	5.5	68.7
2	50%	9.9	124.9
3	80%	12.1	161.4
4	90%	25.2	192.4

***p* is the percentage of hosts with control system on.**
T1 is the time steps needed to infect half of the nodes.
T2 is the time steps needed to infect all the nodes.

Figure 4.25 and Figure 4.26 show the infection evolution when $p = 0.8$ and $p = 0.9$, respectively. We also plot the infection results of the theoretical model. We can see the theoretical results match the simulation results. Furthermore, the average time needed to infect the population decreases dramatically when increased from 0.8 to 0.9.

From Table 4.4, we know it takes about 12 time steps to infect half of the nodes when $p = 0.8$, but 25 time steps when $p = 0.9$.

Table 4.4, Figure 4.25, and Figure 4.26 illustrate that when p is less than 20%, the propagation limiting effect is very small, but when p is more than 50%, the slowing down effect is obvious. In summary, both simulation and theoretical model show the effectiveness of applying the control system. To fight the malicious mobile programs efficiently and to minimize the overall damages, we should not just think about protecting ourselves from the outside world, or just depend on a few hosts to do the good deeds. In order to get the satisfactory result of propagation restriction, we need to have at least 50% of the hosts of an organization or community to install the control system. This is also the limitation of the control system.

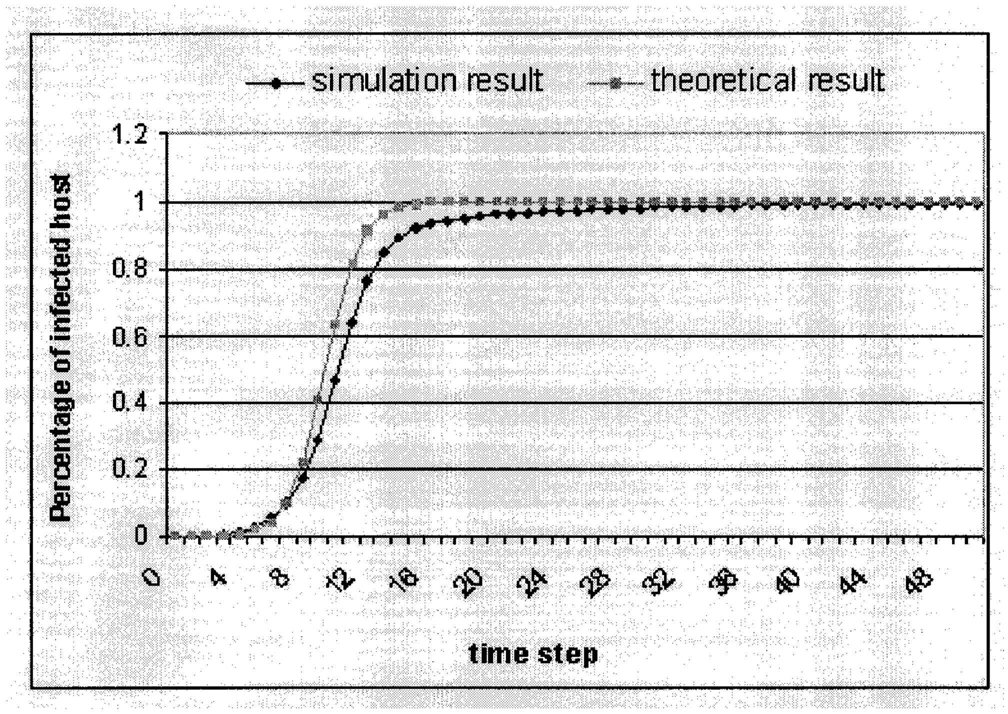


Figure 4.25: Simulation result and theoretical result of malicious code propagation when $p = 0.8$ ($N = 5000$).

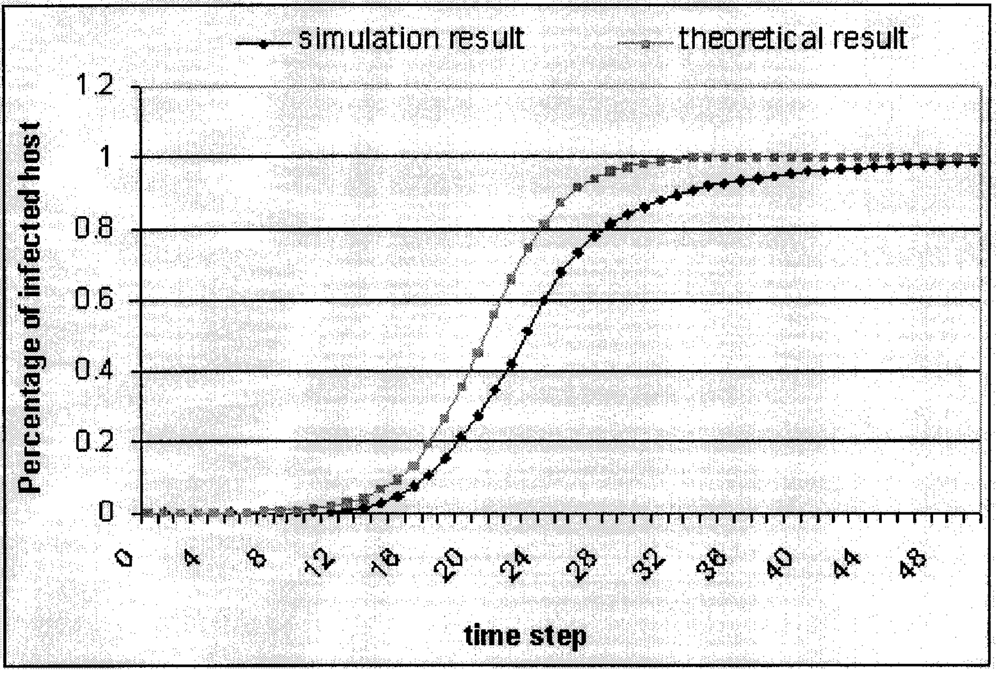


Figure 4.26: Simulation result and theoretical result of malicious code propagation when $p = 0.9$ ($N = 5000$).

4.4.4 Conclusion

In this section, we modify the standard SI model by introducing a new factor B. We present the propagation models with and without the control system. We fit the observed Code Red propagation data into the theoretical model and the result is satisfactory. The simulation of Code Red propagation is close to the observed data. Therefore, we conclude that the propagation simulation approximates the real propagation of malicious mobile code, and the conclusions we draw from the simulation experiments are reasonable.

The mathematical analysis of modified SI models (Equation 4.20) and the model with control system (Equation 4.23) shows that the control system helps reducing the spread of malicious code. The propagation scale is reduced more when more other hosts are using the control system. Further simulation verifies the prediction of the number of infected hosts using the theoretical models.

4.5 Discussion

The proposed control system is host-based and it is adaptive to the characteristics of the local host. Once installed, it learns the local host's connection behavior and blocks the anomalous behavior based on the host's own normal behavior. Besides, when the host's behavior changes, the detection system learns the changes and makes the corresponding changes in its operating parameters, and thus ensures that the false alarm rate is reduced.

4.5.1 Detection Delay

The detection delay of the control system depends on the time interval between two observations. In this experiment, since the unique connection rate data is recorded per hour, the detection delay is one hour. However, when we put the system into real use, the one-hour monitoring interval is too long when malicious code really exists. We should set the monitor interval smaller: for instance, one minute, or even one second depending on the security requirement. In the experiment, we use one-hour interval just because the real data we have is collected per hour. Besides, the main purpose of the experiment is to demonstrate the effectiveness of the control system.

4.5.2 The Advantages of the Control System

Normally, the side effect of high false positive rates in intrusion detection systems is that when an anomaly is detected, the machine is isolated from the network by the intrusion detection system. This may lead to the annoyance of a network user whose work will be hampered and ultimately may cause the user to discard the intrusion detection system. In order to decrease this side effect, the control system does not isolate the machine from the network when anomaly is detected but just limits the connection rate to a lower level. The advantage of doing this is that a user will not get annoyed and discard the control system because of the false alarms. The propagation of malicious code is automatically reduced, though not completely blocked, if hosts have the control system installed. Therefore, the overall damage to the machine and the network and thereby our society, caused by the malicious code is reduced, and the degree of reduction highly depends on the percentage of hosts that adopt the control system in the community. In the next chapter, we will discuss the relationship between the propagation scale and the percentage of hosts that have the control system installed, using some mathematical models.

4.5.3 The Limitation of the Control System

One limitation of the control system is that it cannot detect malicious code that does not propagate through Internet connections. For example, some malicious code may propagate through e-mail, usually as e-mail attachment. The machine gets infected when people open the attachment. The malicious code will scan the address list of the victim and sends e-mails with the same malicious attachment to everybody in the address list automatically. In this case, we can define the normal behavior as the traffic size or the number of e-mails sent/received by a machine in a certain period of time. The same framework can be applied to build and train the control system. The only difference is that the monitored behavior is not the number of connection requests but the number of e-mails sent/received in a given period of time. Similarly, we can extend the control system and apply it to monitor the behavior of the server system. The important thing is that we need to define the normal behavior that can differentiate the normal and abnormal status of a system.

Another limitation is that the network performance relies on the portion of machines with control system. The performance improvement evaluated by infection delay does not have a linear relationship with the portion of machines with control system. In fact, the control system has little effect if only a few machines of the whole network are using it.

4.6 Summary

This chapter presented the development of the control system using Process Control technique. We checked the normality assumption of the real data and generated the simulation data with the same characteristics of that of the real data. Then, we showed the training and the testing process of the control system. The test results showed that the control system achieved zero false negative rate and less than 6% false positive rate when we used the optimal tuning parameters. Therefore, the control system is reliable in detecting the propagation of malicious mobile code. We also discussed the detection delay, the advantages and limitations of the control system. The uniqueness of this approach includes:

- It is novel to apply the Process Control theory in the early detection of malicious mobile code propagation.
- The addition of a sliding window to the traditional Process Control algorithm is very aboriginal, and this makes the system adaptive to the changes of the connection behavior.

- The hypothesis test underlying the monitoring period gives a statistical explanation and quantitative measurement of the detection accuracy.
- The Sequential Probability Ratio Test ensures the quality of the detection system.
- The mathematical models validate the efficiency of the control system in a network environment.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This dissertation discusses propagation modeling of malicious mobile code, and proposes a control system to detect and mitigate the propagation of malicious code automatically. The goals of our work are: (1) Propagation evaluation and prediction of malicious code using stochastic models; (2) Early detection and propagation mitigation of malicious code. We have been successful in achieving the goals since the simulation results match the theoretical results from the propagation models, and the control system we proposed detects the propagation of malicious code with zero false negative rate and less than 6% false positive rate.

Chapter 1 is an overview of this dissertation. Chapter 2 introduces the backgrounds and related research.

Chapter 3 presents stochastic propagation modeling of malicious mobile code. We build a propagation model, INIM model, considering passive immunization from both healthy machines and infected machines. Probability generation function technique is used to get the explicit solution of the stochastic propagation models. The propagation results from the solution match the simulation results, which implies that it is reliable to use the propagation model to evaluate and predict the propagation of a malicious mobile code. To detect and mitigate the propagation of malicious mobile code automatically, we propose a control system using statistical process control techniques in Chapter 4. We extend the traditional process control by adding a sliding window so that the changes of process mean will not affect the detection result. We present the general steps of building a control system and give a statistical analysis of the control system. We also present the details of data collection, assumption checking, training, and testing. The simulation data we used in training and testing are generated based on the real data we have. In the simulation experiments, we discuss the effects of tuning parameters like the size of the sliding window and the updating frequency. The testing results are satisfying and the false positive rate is reduced from more than 30% to less than 6% when the sliding window is applied. We also used sequential probability ratio test to control the false positive rate so that it will never exceed the threshold. Network performance analysis shows that the relationship between propagation mitigation effect and the portion of machines applied control system is not linear. Experiments show that if less than 30% of the machines in a network applied the control system, the effect is negligible, but if more than 90% of the machines applied control system, the propagation delay is significant, which gained us precious time to fight for it.

5.2 Future Work

5.2.1 Network Immunization

Malicious computer mobile codes have been considered as a form of artificial life [Spafford 1994] [Thimbleby 1998] since (1) it exists in space and time; (2) it has the characteristic self-reproduction; (3) information is stored when malicious mobile code replicates itself; (4) it interacts with the environment and damages are caused by these interactions; (5) it has interdependent parts as live organism has; and (6) it mutates. People also argue that malicious mobile code has a kind of metabolism because it takes electrical energy to disseminate its patterns of instructions and infect other systems [Spafford 1994]. Biologically inspired immune systems are developed for computer systems [Forrest 1997] [Harmer 2002] [D'haeseleer 1996] [Kephart 1994]. These immune systems profile the normal activities of a machine as *self* and detect intrusions as *non-self*. Immune systems proposed in [Forrest 1997] [Harmer 2002] [D'haeseleer 1996] [Kephart 1994] are basically intrusion detection systems using misuse detection method. The immunization we discuss in this chapter is different from the immune systems. The immunization of malicious mobile code is more like the immunization of epidemic diseases of human beings.

Immunization has been very successful in controlling epidemic diseases of human beings. Small pox, the disease that originated the research of epidemic modeling, has been eradicated since vaccination is available to everyone. This chapter defines two immunization strategy terms for immunization of malicious mobile code. One is called passive immunization; the other is called active immunization. Propagation models of malicious mobile code considering the effect of passive immunization and active immunization are presented, respectively. In active immunization, we present the idea of using beneficial mobile code to fight against malicious code.

5.2.1.1 Propagation modeling with passive immunization

When an active malicious code is found propagating along the network system, a security expert will analyze its signature. The way to remove the malicious and fix the system will be released to the public once it is available. Users whose machines have been infected will take action to immunize the machine.

Definition: An immunization is called passive immunization if the immunization action is taken by human beings.

We divide the lifetime of a malicious code T into two stages. Stage one (T_1) is the period during which the immunization of certain malicious code is not available. Stage two (T_2) is the period during which the immunization method has been announced. The propagation models we presented in Network Performance Analysis of Chapter 4 describe the propagation of malicious code in T_1. Here, we present the propagation model in T_2.

Suppose the immunization rate of infected machine is γ. Following the same notations given in previous chapters, in a homogeneous network, the deterministic propagation model in T_2 is

$$\begin{cases} \dfrac{dS(t)}{dt} = -\beta S(t)I(t) \\ \dfrac{dI(t)}{dt} = \beta S(t)I(t) - \gamma I(t) \\ \dfrac{dM(t)}{dt} = \gamma I(t) \\ S(t) + I(t) + M(t) = N \end{cases} \qquad t > t_1 \qquad 5.1$$

with initial conditions $S(0) = N - I_0$, $I(0) = I_0$, and $M(0) = 0$; t_1 is the moment that immunization method is available.

Usually, when a computer system is infected, the user of that system may tell this to his/her friends. Also, people may get the message from the media or the Internet. The warning message about the malicious code will be disseminated, and people may take action to prevent the machine from getting infected. Therefore, healthy machines could also be immunized. The warning message traverses through the social network of human beings. The network topology has little effect on passive immunization rate since the spread of warning message does not go through the computer network. Chapter 3 gives the model which considers passive immunization from both healthy and infected machines. The model we present below considers the control system we proposed in Chapter 4.

Let α denote the immunization rate of a healthy machine and γ denote the immunization rate of the infected machine. Suppose p percent of the machines in our network installed the control system, and the average number of neighbors a machine has is B, then the propagation model of the malicious code is given as

$$\begin{cases} \frac{dS(t)}{dt} = -\alpha S(t) - I_1\beta B_1\frac{S(t)}{N} - I_2\beta B_2\frac{S(t)}{N} \\ \frac{dI(t)}{dt} = I_1\beta B_1\frac{S(t)}{N} + I_2\beta B_2\frac{S(t)}{N} - \gamma I(t) \qquad t > t_1 \\ \frac{dM(t)}{dt} = \alpha S(t) + \gamma I(t) \\ S(t) + I(t) + M(t) = N \end{cases} \qquad 5.2$$

with initial conditions $S(0) = N - I_0$, $I(0) = I_0$, $M(0) = 0$. I_1, I_2, B_1 and B_2 follow the same notations we gave in Chapter 4.

5.2.1.2 Propagation modeling with active immunization

Mobile programs are also called self-replicating programs because it has a self-reproducing mechanism. Self-replicating mobile codes are considered to be malicious by most people since the earliest and most prevalent self-replicating mobile program is malicious. However a self-replicating mobile program does not have to be malicious; it can be designed to be beneficial [Chen 2004] [Eugster 2004] [Thimbleby 1999] [Bontchev 1994]. For example, mobile `code can be designed to travel from machine to machine and do useful work in a distributed environment [Levis 2002] [Eugster 2004]; mobile code can be used to fight against the malicious programs [Bontchev 1994]. The designed network should favor the dissemination of benign mobile code but throttle the spread of malicious mobile code. If a beneficial mobile code that is designed to fight against the malicious one spreads faster than the malicious one, the overall network system will become less vulnerable.

Definition: An immunization is called active immunization if the immunization action is automatically taken by benign mobile code.

Suppose a benign mobile code immunizes the healthy machines with rate α and the infected machines with rate γ. In a homogeneous network, the propagation model of the malicious code becomes

$$\begin{cases} \dfrac{dS(t)}{dt} = -\beta S(t)I(t) - \alpha S(t)M(t) \\ \dfrac{dI(t)}{dt} = \beta S(t)I(t) - \gamma I(t)M(t) \\ \dfrac{dM(t)}{dt} = \alpha S(t)M(t) + \gamma I(t)M(t) \\ S(t) + I(t) + M(t) = N \end{cases} \quad t > t_1 \qquad 5.3$$

with $S(0) = N - I_0$, $I(0) = I_0$, $M(0) = 0$, and $M(t_1) = M_0$; *M(t)* still denotes the number of machines that has been immunized. The big difference between these immunized machines and the ones in previous models are that these immunized machines automatically disseminate a copy of the benign mobile code to its neighbors so that the neighbors of this machine could also be immunized.

Suppose the average number of neighbors a machine has is B, and the benign mobile code propagates through exactly the same network as the malicious one does. In the network that p percent of the machines applied the control system, then the propagation model of malicious mobile code is

$$\begin{cases} \frac{dS(t)}{dt} = -\alpha B \frac{S(t)}{N} M(t) - I_1 \beta B_1 \frac{S(t)}{N} - I_2 \beta B_2 \frac{S(t)}{N} \\ \frac{dI(t)}{dt} = I_1 \beta B_1 \frac{S(t)}{N} + I_2 \beta B_2 \frac{S(t)}{N} - \gamma B \frac{I(t)}{N} M(t) \\ \frac{dM(t)}{dt} = \alpha B \frac{S(t)}{N} M(t) + \gamma B \frac{I(t)}{N} M(t) \\ S(t) + I(t) + M(t) = N \end{cases} \quad t > t_1 \qquad 5.4$$

with $S(0) = N - I_0$, $I(0) = I_0$, $M(0) = 0$, $M(t_1) = M_0$

The idea of using benign mobile code to fight against the malicious code has been implemented in current commercial anti-virus tools. When fixing method of a malicious code is available, the anti-virus companies will automatically update its users' virus definition database. But the anti-virus companies will do this only if people pay them. Furthermore, a user's machine cannot disseminate the updating to other machines.

5.2.1.3 Simulation

The immunization simulation is an extension of the propagation simulation with the control system. An immunization procedure is added to the simulation program we used in Chapter 4. Each time we run the simulation, when t is less than t_1, it follows exactly the same infection evolution we have done in Chapter 4; when t is larger than t_1, which means immunization method becomes available, both infection and immunization procedures are running and they are running independently. In the simulation, we assume the immunization method is available when t = 15. A node cannot be infected if it has been immunized. The simulation results demonstrate the efficiency of the immunization strategies. Each result is obtained by running the simulation 10 times based on the same random network with 5,000 nodes.

Simulation of Passive Immunization

In passive immunization, the immunization procedure randomly immunizes the infected node with rate gamma, and the healthy node with rate alpha. It is more likely that an infected machine becomes immunized because a user with an infected machine is more likely to seek out available methods to fix the problem. In the simulation, we set alpha = 0.01, and gamma = 0.1. Figure 5.1 shows the infection evolution of propagation model given by Equation 5.2 in which the effect of passive immunization is modeled.

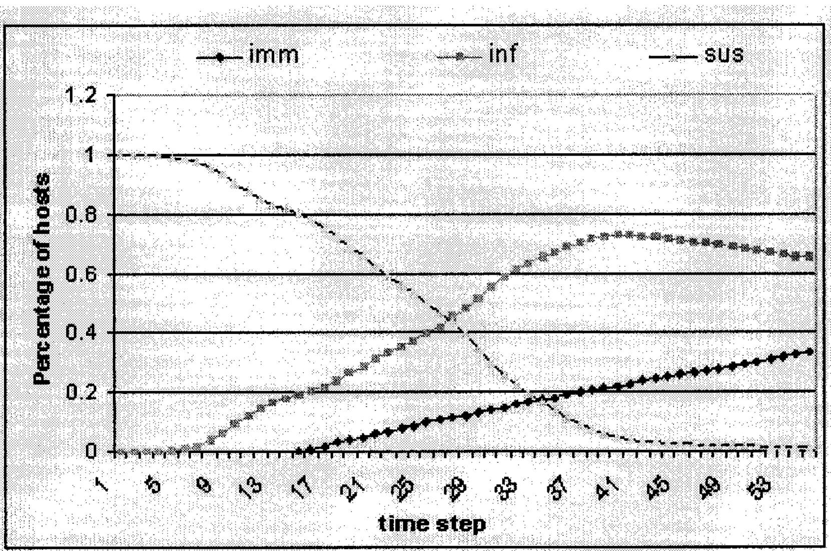


Figure 5.1: Infection evolution with passive immunization ($N = 5000$, $\alpha = 0.01$, $\gamma = 0.1$, $p = 0.8$, $t_1 = 15$).

Simulation of Active Immunization

In active immunization, it is easier for a benign mobile code to immunize a healthy machine than an infected machine because for a healthy machine, the benign program just needs to fix the flaw that has been exploited by the certain malicious code, while for an infected machine, it has to remove the malicious program as an additional work. Therefore, we set alpha = 0.1, and gamma = 0.05 in the simulation. Figure 6.2 shows the infection evolution of model given by Equation 5.4 in which the effect of active immunization is modeled. Comparing Figure 5.2 to Figure 5.1, we can see active immunization is slower than passive immunization at the beginning, but once there are have enough "good" seeds in the network, the machines are immunized at a dramatic speed. After 50 time units, almost all machines are immunized if active immunization applied, and only one third are immunized if passive immunization applied. Overall, active immunization has better performance. However, the implementation of such benign mobile code is not easy. If not properly designed, the benign mobile code may bring another disaster.

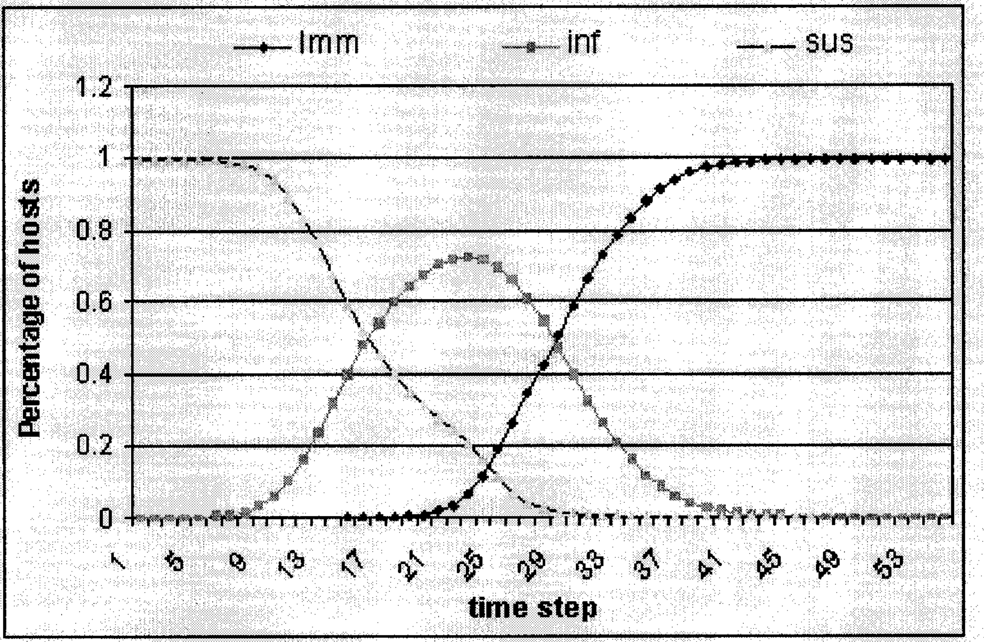


Figure 5.2: Infection evolution with active immunization ($N = 5000$, $\alpha = 0.1$, $\gamma = 0.05$, $p = 0.8$, $t_1 = 15$).

5.2.2 Passive Immunization and Network Topology

The immunization strategies we used to prevent infectious diseases for human beings include:

- Random immunization.
- Target immunization, for example, immunization by group.

We could have similar strategies when we apply passive immunization to malicious mobile code. The immunization we discuss in Section 5.2.1 is purely random, and we have no idea who will and who will not immunize the machine. In the future, we could have the security experts apply passive immunization according to the priority of the machines. For example, we immunize the Internet routers or network backbones first, then we immunize the nodes with a higher number of neighbors. The information of network topology helps us to decide the target nodes with more than an average number of neighbors. We should analyze the effect of random immunization and target immunization under network topologies so that we can apply the optimal immunization strategy for a network with certain topology.

5.2.3 Active Immunization and Network Topology

If we use a graph to represent a subnet in which malicious code traverse, we get a graph whose nodes are machines infected by the malicious code, and the lines between nodes are the ways malicious code propagate. The propagation model we give as Equation 5.4 assumes that benign mobile code propagates through the same subnet a malicious one does. In reality, these two may not propagate through the same subnet. Also the spreading rate of benign mobile code must be higher than the malicious one to prevent the outbreak of malicious code. Previous research has shown that the topology of underlying subnet affects the spread of mobile code dramatically [Chen04]. We will do more research on how topology affects the propagation speed of mobile code. In the future, we will build propagation models that could catch the effect of topology and design networks that makes a benign mobile code spread faster.

5.2.4 Distributed Malicious Mobile Code Detection

The control system we propose in this dissertation is host based. The effect of propagation mitigation is not good if the number of machines in a network that applied the control system is small. It takes much time and effort to make sure that every machine in the network applied the control system properly. A better way to achieve the same effect is to build a control center. Monitors are distributed in the network and each monitor reports to the control center periodically. When a host is identified with abnormal behavior, its outgoing Internet connections will be limited. At the same time, the control center will send a message to other machines in the network. We do not want to disable the function of the whole network. One way to defend against the possible malicious code is choosing some nodes in the network and limiting the outgoing connection of these nodes so that the propagation of malicious code will be slowed down. More research needs to be done on how to decide the number of nodes we should choose, and the policy of choosing specific nodes so that the propagation could be slowed down maximally with minimum effect on the normal function of the network.

REFERENCES

[Anderson 1992] Roy M. Anderson, Robert M. May, B. Anderson, *Infectious Diseases of Humans: Dynamics and Control*, Oxford University Press; (December 1, 1992) ISBN: 019854040X.

[Andersson 2000] Hakan Andersson, Tom Britton, *Stochastic Epidemic Models and Their Statistical Analysis*, Springer-Verlag 2000, ISBN 0-387-95050-8.

[Bailey 1975] N. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*, Oxford University Press, New York, 1975.

[Bertino 2004] Elisa Bertino, Krithi Ramamritham, Data Dissemination on the Web, *IEEE Computer*, Vol. 8, No. 3, p27-28, May/June 2004

[Boguna 2002] Marian Boguna, Romualdo Pastor-Satorras, Epidemic Spreading in Correlated Complex Networks, *Physical Review E*, Volume 66, 047104, 2002.

[Bontchev 1994] Vesselin Bontchev, Are 'Good' Computer Viruses Still a Bad Idea, in *Proceeding of. EICAR'94 Conference*, pp. 25-47.

[Caida 2001] http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml, last accessed April 21, 2005.

[CERT01-19] http://www.cert.org/advisories/CA-2001-19.html, last accessed April 9, 2005.

[CERT01-26] http://www.cert.org/advisories/CA-2001-26.html, last accessed April 9, 2005.

[CERT03-04] http://www.cert.org/advisories/CA-2003-04.html, last accessed April 9, 2005.

[CERT03-20] http://www.cert.org/advisories/CA-2003-20.html, last accessed April 9, 2005.

[CERT04-02] http://www.cert.org/incident_notes/IN-2004-02.html, last accessed April 9, 2005.

[CERT04-05] http://www.us-cert.gov/current/archive/2004/05/11/archive.html, last accessed April 9, 2005.

[Chen 2003] Z.Chen, L.Gao, K.Kwiat, Modeling the Spread of Active Worms, in *Proceedings of IEEE INFOCOM 2003*, San Francisco, CA, April 2003.

[Chen 2004] Li-Chiou Chen, Kathleen M Carley, The Impact of Countermeasure Propagation on the Prevalence of Computer Viruses, *IEEE Trans Syst Man Cybern B Cybern,* 2004 Apr; 34(2); 823-33.

[Chiang 1978] Chin Long Chiang, *An Introduction to Stochastic Processes and Their Applications*, Krieger Publish Company, 1978.

[Cohen 1985] Fred Cohen, *Computer Viruses*, PhD thesis, University of Southern California, 1985.

[Cohen 1987] F. Cohen. *Computer Viruses: Theory and Experiments*. Computers & Security, Vol.6 22-35, 1987.

[Cohen 1994] Frederick B. Cohen. *A Short Course on Computer Viruses*, 2 edition Wiley; New York, 1994, ISBN 0471007684 .

[Csrc 2005] http://csrc.nist.gov/publications/nistir/threats/threats.html, last accessed April 9, 2005 [link2]

[Daley 2001] D. J. Daley and J. Gani, *Epidemic Modeling: An Introduction*, Cambridge University Press, May 2001, ISBN 0-521-01467-0.

[Dave 2001] Dave Goldsmith, "*Possible Code Red Connection Attempts*", available at: http://lists.jammed.com/incidents/2001/07/0158.html last accessed February 9, 2005.

[Dean 1999] Angela Dean and Daniel Voss, *Design and Analysis of Experiments*, Springer-Verlag, New York, 1999, ISBN 0-387-98561-1.

[Denning 1990] Peter Dinning, *Computers under attack : intruders, worms, and viruses,* Addison-Wesley, New York, N.Y., 1990, ISBN 0201530678

[D'haeseleer 1996] P. D'haeseleer, S. Forrest and P. Helman. An immunological approach to change detection: algorithms, analysis, and implications. In *Proceedings of the IEEE Symposium on Computer Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, 1996, 110—119, 1996.

[Eckmann 2002] Steven T. Eckmann, Giovanni Vigna and Richard A. Kemmerer, STATL: An Attack Language for State-based Intrusion Detection System, *Journal of Computer Security,* 10(1/2): 71-104 (2002).

[Eugster 2004] Patrick T. Eugster, Richid Guerraoui, Anne-Marie Kermarrec, Laurent Massoulié, Epidemic Information Dissemination in Distributed Systems, *IEEE Computer* 37(5): 60-67 (2004).

[Fites 1992] Philip Fites, Peter Johnston, Martin Kratz, *The Computer Virus Crisis*, 2nd ed, Van Nostrand Reinhold, New York, 1992, ISBN 0442006497.

[Forrest 1994] Stephanie Forrest, Alan S.Perlelson, Lawrence Allen, Rajesh Cherukuri, Self-Noneself discrimination in a Computer, in *Proceedings of 1994 IEEE Symposium on Research in Security and Privacy.*

[Forrest 1997] S. Forrest, S. Hofmeyr, and A. Somayaji, Computer Immunology, *Communications of the ACM* Vol. 40, No. 10, pp. 88-96 (1997).

[Grimes 2001] Roger Grimes, *Malicious Mobile Code : Virus Protection for Windows*, O'Reilly & Associates, Sebastopol, CA , 2001, ISBN 15659268X.

[Hansen 1987] Bertrand L. Hansen and Proabhakar M. Ghare, *Quality Control and Application*, Prentice-Hall, 1987, ISBN 0-13-745225-X.

[Harmer 2002] Paul K. Harmer, Paul D. Williams, Gregg H. Gunsch, Gary B. Lamont, An artificial immune system architecture for computer security applications, *IEEE Transactions on Evolutionary Computation*, Volume 6, Number 3, June 2002 252-280.

[Heberlein 1990] L.Heberlein, G.Dias, K.Levitt, B. Mukherjee. J. Wood, and D. Wolber, "A Network Security Monitor", in *proceedings of the IEEE symposium on Security and Privacy*, May 1990 pp.296-304.

[Toyoizumi 2002] Hiroshi Toyoizumi, Atsuhi Kara, Predators: Good Will Mobile Codes Combat against Computer Viruses, *in Proceeding of the New Security. Paradigms Workshop*, Virginia Beach, Virginia, 2002.

[Hofmeyr 1999] S.A. Hofmeyr, *A Immunological Model of Distributed Detection and Its Application to Computer Security,* PhD thesis, Department of Computer Science, University of New Mexico, Apr. 1999.

[Kephart 1991] Jeffrey O Kephart and Steve R White, Directed-Graph Epidemiological Models of Computer Viruses, in *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 343-359, May 1991.

[Kephart 1993] Jeffrey O Kephart and Steve R White. Measuring and Modeling Computer Virus Prevalence, in *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1993.

[Kephart 1994] Jeffrey O. Kephart, a Biologically Inspired Immune System for Computers, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 1994.

[Kumar 1992] S. Kumar and E. H. Spafford (1992). A generic virus scanner in C++, in *Proceedings of the 8th Computer Security Applications Conference*, IEEE Press.

[Levin 1990] Richard B. Levin, *The Computer Virus Handbook*, Osborne McGraw-Hill, Berkeley, 1990, ISBN: 0078816475.

[Levis 2002] P. Levis and D. Culler, *Mate:* A Tiny Virtual Machine for Sensor Networks, *the International Conference on Architectural Support for Programming Languages and Operating Systems*, San Jose, CA, USA, 2002.

[Miller 1995] Barton P Miller, David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, Jeff Steidl, *Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Service,* Computer Sciences Department, University of Wisconsin, 1995. (NOT IN ENDNOTE)

[Moore 2002] David Moore, Colleen Shannon, Jeffery Brown, Code-Red: a case study on the spread and victims of an Internet worm, in *Proceeding of the Internet Measurement Workshop (IMW)*, Marseille, France, November, 2002.

[NISTIR4939] Lawrence E. Bassham and W. Timothy Polk, *Threat Assessment of Malicious Code and Human Threats*, the United States National Institute of Standards and Technology, 1992.

[Phoha 2002] Vir V. Phoha, *The Springer Dictionary of Internet Security*, Springer-Verlag, New York, (Forthcoming) January 2002.

[Phoha 2003] V. Phoha, X. Xu, A. Ray and S. Phoha, Supervisory Control Automata Paradigm to Make Malicious Executables Ineffectual, in *Proceedings of the 5TH IFAC Symposium on Fault Detection, Supervision and Safety*, Washington D.C., 2003, pp. 1167-1172.

[Satorras 2001] Romualdo Pastor-Satorras and Alessandro Vespigani, Epidemic Dynamics and Endemic States in Complex Networks. *Physical Review E*, Volume 63, 066117, 2001.

[Satorras 2002] Romualdo Pastor-Satorras. Epidemics and Immunization in Scale-Free Networks, in S.Bornholdt and H.G. Schuster, editors, *Handbook of Graphs and networks: From the Genome to the Internet,* Wiley-VCH, Berlin, May 2002.

[Serazzi 2003] Giuseppe Serazzi and Stefano Zanero. Computer Virus Propagation Models. In M. C. Calzarossa, E. Gelenbe, editor, *Tutorials of the 11th IEEE/ACM Int'l Symp. On Modeling, Analysis and Simulation of Computer and Telecom. Systems - MASCOTS 2003,* Springer-Verlag, 2003.

[Spafford 1989a] Eugene H. Spafford, the Internet Worm: Crisis and Aftermath, *Communications of the ACM*, 32(6):678-687, June 1989.

[Spafford 1989b] Eugene Spafford, The Internet Worm Program: An analysis, *Computer Communication Review*, 19(1), January 1989.

[Spafford 1994] Eugene H. Spafford, Computer Viruses as Artificial Life, *Journal of Artificial Life*, volume XI, pages 371-408.

[Staniford 2002] Stuart Staniford, Vern Paxson, Nicholas Weaver. How to Own the Internet in You Spare Time. In *Proceedings of the 11th USENIX Security Symposium*, 2002.

[SYMANTEC03]http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html, last accessed April 9, 2005.

[Thimbleby 1998] Harold W. Thimbleby, Ian H. Witten, and David J. Pullinger, Concepts of Cooperation in Artificial Life, *IEEE Transactions on Systems, Man & Cybernetics*, 25(7), pp1166—117, 1998.

[Thimbleby 1999] H. Thimbleby, S. Anderson and P. Cairns, *A Framework for Modeling Trojans and Computer Virus Infection*, The Computer Journal, 41 (1999), pp. 444-458.

[Usa 2001] USA today news. The cost of Code Red: $1.2 billion. http://www.usatoday.com/tech/news/2001-08-01-code-red-costs.htm, last accessed February 9, 2005.

[Vetterling 2002] William T. Vetterling, Brian P. Flannery, *Numerical Recipes in C++*, Cambridge University Press, 2 edition (February, 2002), ISBN 0521750334.

[Wang 2000] Yongge Wang, Using Mobile Agent Result to Create Hard-to-detect Computer Viruses, *Information Security for Global Information Infrastructures, the 16th IFIP SEC* (2000), pages 161—170.

[Wang 2003] Yang Wang, Chenxi Wang. Modeling the Effects of Timing Parameters on Virus. In *Proceedings of the ACM CCS Workshop on Rapid Malcode*, 2003.

[White 1998] Steve R. White. Open Problems in Computer Virus Research. *Virus Bulletin Conference*, Oct 22, 1998, Munich Germany.

[Williamson 2002] Matthew M. Williamson, "Throttling Viruses: Restricting propagation to defeat malicious mobile code", in *Proceedings of the 18th Annual Computer Security Applications Conference*, 2002, p 61.

[Wong 2004] Cynthia Wong, Chenxi Wang, Dawn Song, Stan Bielski, Gregory R. Ganger, Dynamic Quarantine of Internet Worms, in *Proceedings of the International Conference on Dependable Systems and Networks (DSN-2004)*, Italy. June 28th - July 1, 2004.

[Xu 2002] Xin Xu, Supervisory Control Automata as a New Paradigm to Make Malicious Executables Ineffectual, M.S. thesis, Louisiana Tech University, 2002

[Xu 2004] X. Xu, V. V. Phoha, A. Ray and S. P. Phoha, "Supervisory Control of Malicious Executables in Software Processes", in A. Ray, V. V. Phoha and S. P. Phoha, eds., *Quantitative Automata-Based Supervisory Decision and Control*, Springer-Verlag, New York, 2004

[Ye 2000] Nong Ye, Mingming Xu and Syed Masum Emran, Probabilistic Networks with Undirected Links for Anomaly Detection, in *Proceedings of the 2000 IEEE Workshop on Infromation Assurance and Security*, June, 2000.

[Zou 2002] Cliff Changchun Zou, Weibo Gong, and Don Towsley, Code Red Worm Propagation Modeling and Analysis, in *Proceedings of the 9th ACM Conference on Computer and Communication Security*, November 2002.

[Zou 2003] C.C.Zou, L.Gao, W. Gong, and D.Towsley, Monitoring and Early Warning for Internet Worms, in *Proceedings of the 10th ACM conference on Computer and communication security*, 2003.