Abstract

With the increasing severity of malware problem, this project aims at constructing computer models which are specialized to the defense against malware proliferation and providing some useful advice on what strategies should be deployed in different malware scenarios. The project is the combination of 70% theoretical investigation and 30% software implementation. We did much research on malware spread mechanism, epidemiology modeling with topology consideration, and game theory. Then, based on existing models and algorithms, we developed theoretical models to simulate malware propagation behavior and constructed cost-benefit models to carry out cost-benefit analysis. Finally, we developed computer models and analyzed real worms. We list several achievements as follows:

- We worked out the scale-free network parameter by using mean field approximation, see pages 21-22.
- We combined epidemiology models with strategy making analysis and the scale-free network parameter to construct game theoretical models, see pages 22-25.
- We deployed the same methods used in the "FLIPIT" game to construct cost-benefit models, see pages 25-28.
- By integrating all models, we found out the way to determine the best strategy, see pages 29-31.
- We developed computer models by implementing them in Matlab, see pages 30-32.
- Based on the analysis of real worms, we summarized some key observations which can help defenders choose the best strategy, see pages 33-50.

Contents

Chapter 1

Introduction

The malware problem has posed a great threat to the security of network systems nowadays. With the increasing complexity of malware, it has become much difficult to detect its existence. In order to prevent the infection and clear the malware, several deployments may exist. Each of them has the operational cost and effectiveness. A security action with great effectiveness often has a high cost. Therefore, when the infection begins to spread in the computer network, how to balance the cost and benefit and choose the best strategy is of vital importance.

1.1 Aims and Objectives

This project has two main aims. Firstly, we combine epidemiology modeling with game theory to construct computer models which are specialized to the defense against malware proliferation problem in the computer network. Secondly, we utilize models to analyze real malware and provide some useful advice on what security strategies should be deployed in different malware scenarios.

1.1.1 Model Objectives

Here, we list objectives of model development.

Investigate malware problem and epidemiology modeling.

Study the complex topology and work out the way of combining it with epidemiology models.

Construct theoretical models to simulate malware proliferation behavior when complex topology is taken into account and different security strategies are deployed.

Study game theory and have a clear understanding of the "FLIPIT" game.

Carry out the cost-benefit analysis and construct cost-benefit models.

Combine all models to work out the way to determine the best security strategy.

Implement all theoretical models.

1.1.2 Analysis Objectives

Here, we list objectives of data analysis.

Study several striking examples of malware in detail.

In each malware scenario, we address the following questions:

- For each security strategy, how do the overall benefit and the infected population change when defenders notice the infection and deploy this strategy at different time of the spread period?
- After defenders discover the infection, how do they choose the best strategy?

Give some helpful advice on what security strategies should be deployed in different malware scenarios.

1.2 Structure

This report consists of five chapters. To begin with, we review the background research of the project. This includes the study of malware problem, epidemiology modeling and game theory. As for game theory, we give a detailed description of "FLIPIT" game scenario which lays a solid foundation for the construction of game theoretical models. In the next chapter, we focus on explaining game theoretical models and cost-benefit models; then, we discuss how to determine the best strategy by utilizing the models and present a brief description of how to implement all models. The fourth chapter is the case study and analysis of results. We study three worms and fit real data into computer models. Based on the results, we summarize some key observations which may give useful guidance to the decision making problem. The evaluation chapter discusses whether this project has met the objectives and whether it has been successful. Finally, the future work chapter explains the further work that could be carried out to develop models and analyze more malware cases.

Chapter 2

Background and Context

This chapter reviews and summarizes the background research of the project. The project encompasses two main research areas including the epidemiology modeling and game theory. In order to investigate the relationship between them, we should at first study each area separately. Therefore, in the following sections, we focus on the fundamental concepts and methods which are necessary to the project. To begin with, we study some basic knowledge of malware in order to get a rough idea of malware proliferation mechanism. Secondly, we give a general description of two standard epidemiological models. In the third section, we investigate the complex topology. Finally, we aim at having a better understanding of game theoretical approaches and then give a detailed analysis of the "FLIPIT" game scenario.

2.1 Malware Background

Malware is the abbreviation of malicious software. It consists of computer viruses, worms, Trojan horses and other malicious programs. In this section, we focus on studying and understanding mechanisms of malware such that we are able to make countermeasures against it.

At first, we study the transmission and infection mechanism of the malware which is summarized in table 2.1 (taken from [15]). If malware wants to infect a computer successfully, it should gain access to the target by making use of some methods to travel to it. There are three ways in terms of the human cooperation. The malware can be inserted manually by an individual deliberately or triggered by someone unintentionally. In addition, it can also exploit the vulnerability which exists in the operating system and infect the computer without human intervention. Besides, the access to the target must be through some medium, either a storage medium or the network. From table 2.1, we can see that only network-based transmission does not need the human cooperation. In typical examples of this characteristic, the infection way of using security vulnerability does not require human intervention; using emails or sharing files still needs a human to trigger the infection by opening the email or the shared file.

Table 2.1: The transmission and infection mechanism of the malware (from [15]).

Characteristic	Typical Examples
Human Cooperation	Deliberate manual insertion
	Inadvertent insertion
	Autonomous insertion, no human cooperation
Through Removal Storage Medium	Floppy
	CD
Through Network	Security vulnerability in running service
	E-mail
	File shares

Then, we study the self-defense mechanism of malware. Table 2.2 (taken from [15]) shows four main ways that malware usually takes to protect itself. Protective strategies vary with different levels of effort that malware should take. The most common way is to keep a low profile. For the stealth strategy, the malware can hide itself in the system by taking the form of obscurity, or, it can mislead the administrator and users by pretending to act as the authorised program or running within the authorised program. Moreover, it is also able to subvert a system to confuse the detective software. In order to increase the detecting difficulty, another strategy has also been developed by changing the appearance of the malware, such as encryption and polymorphism. However, with the increasing number of well-designed antivirus software, it becomes much easier to identify different forms of secret malware. Thus, malware is designed to disable the antivirus software or developed into several variants to increase the chance of reinfection.

Table 2.2: The self-defense mechanism of the malware (from [15]).

Protective Strategy	Example Mechanisms
Stealth	Subverting system calls
	Mimicking authorised programs
	Piggybacking on authorised programs
	Obscurity
Evading Detection Software	Encryption
	Polymorphism
	Rapid development
Disabling Detection	Removing AV software
Redundancy	Re-infection

Finally, we study factors contributing to the malware proliferation. The spread of malware relies on two main factors: the population and the environment. They are shown in table 2.3 (taken from [15]).

As for the population, size is the first influencing factor. The number of individuals within the population determines the range of malware proliferation. Larger population is more likely to help malware spread and increase the infected population base. In addition, the intimacy between individuals affects the success ratio of infection. If two individuals contact each other frequently, the infection of one individual may probably transfer the malware to another individual by sending emails or sharing files. Finally, the common points shared by the whole population or parts of the population are also the main target exploited by malware. Suppose that all individuals use the same

Table 2.3: Factors that facilitate malware proliferation (from [15]).

	Size
Population	Intimacy
	Shared characteristics
	Number of features
Target Complexity	Level of feature integration
	Use of mobile code

office software. The malware can exploit vulnerabilities of this office software and carry out attacks on the whole population. If the software is used by a small portion of the total population, the infection range will reduce greatly.

The environment also contributes much to malware proliferation. Because malware exploits vulnerabilities of software, the security of software is the most important factor in preventing malware infection. The complexity of software determines that in what degree vulnerabilities may exist. The more codes the software has, the greater probability that more vulnerabilities exist and may be exploited by malware. Nowadays, more and more software is designed by the principle of compatibility to facilitate user experience. However, such design is also beneficial to malware propagation. Shared codes in different software provide malware with convenient channels to make use of vulnerabilities exposed in these codes and carry out attacks on many software simultaneously. The programming language also exerts great influence on software security. Some languages have built-in security mechanisms to prevent potential security failures. However, some languages, such as the scripting language, do not have such built-in mechanisms so that software with these languages may be exploited as vectors of malware.

2.2 General Description of Epidemiology Models

The epidemiology modeling of malware proliferation has been studied for decades. The spread of biological diseases and computer malware (viruses, worms etc.) is analogical, so the analysis of computer malware can benefit from investigating the behavior of biological diseases. Therefore, in macroscopic view, the epidemiology method provides an effective way to research on the prevalence of the malware.

In epidemiology area, there are two kinds of models for analyzing malware proliferation: stochastic models and deterministic models. Stochastic models are used to analyze small-scale network; while deterministic models are used to analyze large-scale network. In order to study the effect of mass action, we consider malware spread in the network with a large number of computers, so we will utilize deterministic models.

In general, individuals in the epidemic population have several states, including susceptible, infected, recovered, dead, etc. Some models have been studied on state transitions and named by the order of states. There are two commonly used models, *Susceptible Infected Susceptible* (SIS) model and *Susceptible Infected Recovered* (SIR) model. The simplest SIR model was created by Kermack and McKendrick [5] in 1927. Then, the SIS model was derived from the SIR model. Both of them are standard models and have been widely applied in many investigations of malware proliferation.

Both models assume that all individuals within a closed population (no births and deaths) are susceptible to the malware in the initial phase and an individual may go through each state sequentially. In the SIS model, the state transitions of an individual form a circulation. The individual may recover from the infection, but there is still a chance that it is reinfected. So an individual becomes susceptible to the malware again after it recovers. In the SIR model, the final state is a recovered state. It means that an individual may recover from the infection and become immune to the malware when the infection is cleared, and the vulnerability is patched.

Standard epidemiology models usually have two shortcomings. The first one is that the network should be homogeneous. It means that in a single network architecture, an individual can infect any other individuals. The second one is that the network should be symmetric. It means that there is no preferential direction of the malware transmission.

2.3 Topology Consideration

The topological analysis plays an important role in the epidemiological modeling. The prevalence of infection in the complex network is based on the connectivity of the network. Therefore, different topologies may significantly change the spreading way of malware. Placing malware in different kinds of social networks is an effective way to analyze malware proliferation comprehensively such that countermeasures are worked out in different situations.

Among many communication networks, the most valuable example is the Internet. Nowadays, the Internet has become the most essential tool in many fields including business, governance, communication and entertainment etc. The stability and availability of the Internet for us is of vital significance. However, because of the big importance, the Internet has also become the main attacking target of malware for dozens of years. Thus, the malware threat on the Internet becomes a very urgent and thorny problem.

The structure of the Internet can be well-described as the scale-free network. Nodes in the scale-free network are not evenly connected. There are several "very connected" nodes, called hubs with the highest degree. The most notable characteristic is that the degree distribution conforms to a power law distribution, which is shown in figure 2.1 (taken from [1]).

2.4 Game Theory in Malware Analysis

Game theory has been proposed to address the network security issues for decades. In this section, we apply game theory to investigate the malware proliferation. Firstly, we study why game theory can be used in malware

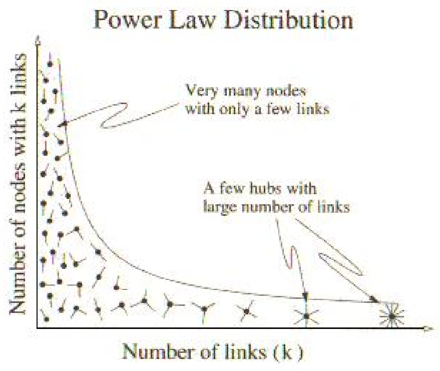


Figure 2.1: The power law distribution of the scale-free network (from [1]).

analysis and present some key concepts related to the project. Then, we analyze the "FLIPIT" game scenario.

2.4.1 Why Use Game Theory in Malware Analysis

Traditionally, the network security deployments are divided into two parts: "protective devices", like firewalls, or "detective and reactive devices", like intrusion detection systems (IDSs) [12]. Firewalls use a predesigned set of rules to deny unauthorized network communications. IDSs use two databases, attack signatures or baseline behaviors of legitimate users, to identify and alert attacks. Some IDSs may also make an active response to counter illegal activities that are detected.

These two security deployments have been used in combination to thwart malware attacks. However, both of them aim at analyzing the malware problem in micro level. They focus on collecting, dissecting and recording the structure and behavior of malware. So they are efficient for preventing well-known malware, because their techniques are based on experimental work. But the main weakness is that they do not take into account the decision making problem of attackers.

As traditional approaches are lack of exploring interactions between the attacker and the defender, researchers have been applying the game theory to the network security problem. The relationship between the attacker and defender is just like the relationship between two players in the game scenario. If the malware destroys the network or controls computers in the network, the attacker can benefit from such destruction. Meanwhile, people who own the computers suffer from the cost of losing control. So the attacker and the defender have contradictory objectives and share direct relationship of interests. Moreover, during the process, the attacker tries to find the most effective way to carry out attacks in the network; and the defender also plans to choose the best defensive approach to protect the network. As game theory is used for studying decision making problem in multiplayer scenarios, it can examine and evaluate the scenarios with all possible decisions made by players and compute the best strategy among the payoffs.

In brief, the game theoretical analysis is the best tool to study interactions between the attacker and the defender. It sophisticates the decision making process to the greatest extent and offers the defender an effective way to determine the best reaction against the attacker.

2.4.2 Brief Introduction to Game Theory

The background knowledge of game theory presented in [12, 16] provides us with a better understanding of game theoretical solutions and guidance to strategy making problem.

In general, games are divided into cooperative and non-cooperative games. In cooperative games, all players try to maximize the overall payoff. In non-cooperative games, all players just care about their own benefit and cost. In the field of network security, the research falls under the category of non-cooperative games, because the attacker aims at attacking the network and the defender aims at protecting the network; there is no cooperation between them.

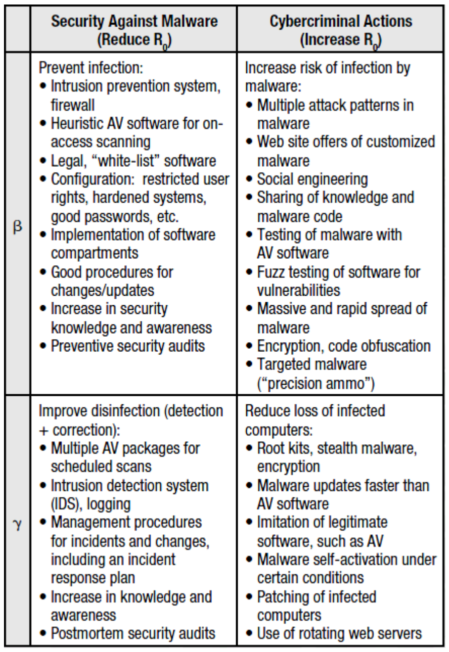
Under the category of non-cooperation, games are divided into static and dynamic games. In static games, all players make their decisions simultaneously such that they do not know others' strategies. Static game is a one-shot game. Each player has a pre-computed move list. In dynamic games, all players go through more than one stage. They can consider their strategies in each stage and then take actions sequentially. Static games can also be modified in a dynamic version if players repeat their decisions for many times.

There is another way to classify non-cooperative games based on the information received by players. Four subtypes exist: perfect and complete information, perfect and incomplete information, imperfect and complete information, imperfect and incomplete information. Complete information means that all players know the overall payoff. Incomplete information means that at least one player does not know the overall payoff. Perfect information means that the player with the move now knows all moves of what other players have taken. Imperfect information means that at least one player with the move does not know the full history of the game played so far. The static game is only of imperfect information.

One of the most commonly used types is the zero-sum game. It is the sub-type of the constant-sum game. In the zero-sum game, the gain (cost) of one player is completely balanced by the cost (gain) of other players; the total gain is equal to the total cost. In contrast, the sum of all payoffs in the non-zero-sum game is not equal to zero. The optimal solution, in the zero-sum game, is usually worked out by the Nash Equilibrium. A Nash Equilibrium computes a steady state which involves at least two players. Under this state, no player wants to change his strategy unilaterally, because no more gain is received when other players keep their strategies unchanged.

Strategies are divided into pure and mix strategies. Pure strategy is the single strategy made by the player. Mix strategy means that the player has a set of strategies with a probability distribution. In general, players make the mixed strategy and try to maximize their own benefit by finding out the most suitable probability distribution for their strategy profile. Figure 2.2 (taken from [23]) shows some detailed strategies made by the attacker and the defender. Here, β denotes the spread rate of malware and γ denotes the recovery rate of the computer; R_0 denotes the spread efficiency of malware in the network, which is shown in formula 2.1. When deciding what strategies to use, this figure is a good reference.

$$R_0 = \frac{\beta}{\gamma} \tag{2.1}$$



	Security Against Malware (Reduce R_0)	Cybercriminal Actions (Increase R_0)
β	Prevent infection: • Intrusion prevention system, firewall • Heuristic AV software for on-access scanning • Legal, "white-list" software • Configuration: restricted user rights, hardened systems, good passwords, etc. • Implementation of software compartments • Good procedures for changes/updates • Increase in security knowledge and awareness • Preventive security audits	Increase risk of infection by malware: • Multiple attack patterns in malware • Web site offers of customized malware • Social engineering • Sharing of knowledge and malware code • Testing of malware with AV software • Fuzz testing of software for vulnerabilities • Massive and rapid spread of malware • Encryption, code obfuscation • Targeted malware ("precision ammo")
γ	Improve disinfection (detection + correction): • Multiple AV packages for scheduled scans • Intrusion detection system (IDS), logging • Management procedures for incidents and changes, including an incident response plan • Increase in knowledge and awareness • Postmortem security audits	Reduce loss of infected computers: • Root kits, stealth malware, encryption • Malware updates faster than AV software • Imitation of legitimate software, such as AV • Malware self-activation under certain conditions • Patching of infected computers • Use of rotating web servers

Figure 2.2: Battle between the cyber attack and the cyber defense (from [23]).

2.4.3 "FLIPIT" Game

Combined with the cost-benefit analysis, Rivest and his colleagues investigated interactions between the attacker and the defender in the "FLIPIT" game scenario [17] and proposed some defensive strategies. Sometimes, the attacker may periodically control computers in the network or steal the important information without being noticed by the defender immediately. In such a situation, the "FLIPIT" game becomes the best model to study secret takeovers by the attacker and to find out effective strategies deployed by the defender.

In the "FLIPIT" game, there are two players, the attacker and the defender, and a shared resource. Two players compete to take control of the critical resource. The attacker tries to put the resource into a bad state; while the defender puts the resource into a good state. Two players take over the control by making moves sequentially. Sometimes, one player makes a move but does not take over the control, so the moves are not certainly effective. The objective of each player is to control the resource for a large fraction of time and minimize the total cost. Here, we assume that players do not know the current situation of the game when other players make a move; they learn the situation only when they make a move. Making a move pays cost and taking over the control gains benefit. Each player loses some points per move and gains some points per second when he is in control.

At first, we show a graphical example in figure 2.3 (taken from [17]). The blue area represents the period of defender's control on the critical resource; the red area represents the period of attacker's control on the resource. The blue circle represents the defender's move; the red circle represents the attacker's move. The arrow represents a takeover by the defender or attacker. We assume that the defender is in control at the beginning of the game.

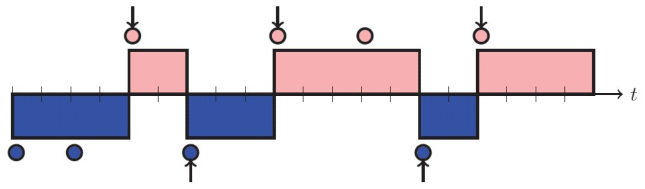


Figure 2.3: The FLIPIT Game (from [17]).

Then, we give the mathematical description taken from [17]. Here, we assume that the defender is player 0 and the attacker is player 1. For player i , he pays k_i points per move and gains one point per second in control. Formula 2.2 shows the gain rate $\gamma_i(t)$ for player i ; G_i denotes the number of seconds that player i is in control.

$$\gamma_i(t) = \frac{G_i(t)}{t} \tag{2.2}$$

Formula 2.3 shows that the whole period time t is the time controlled by the defender plus the time controlled by the attacker.

$$G_0(t) + G_1(t) = t \tag{2.3}$$

Thus, for each player, the gain rate $\gamma_i(t)$ also means the fraction of time in control, shown in formula 2.4.

$$\gamma_0(t) + \gamma_1(t) = 1 \tag{2.4}$$

Formula 2.5 shows the average rate of play $\alpha_i(t)$ for player i ; N_i denotes the number of moves for player i .

$$\alpha_i(t) = \frac{N_i(t)}{t} \tag{2.5}$$

In formula 2.6, the benefit $B_i(t)$ is denoted as the gain minus the cost.

$$B_i(t) = G_i(t) - k_i * N_i(t) \tag{2.6}$$

The benefit rate $\beta_i(t)$ is calculated based on Formula 2.2, 2.5, 2.6 and shown in formula 2.7. Each player aims at maximizing their benefit rate.

$$\begin{aligned} \beta_i(t) &= \frac{B_i(t)}{t} \Rightarrow \\ \beta_i(t) &= \frac{G_i(t) - k_i * N_i(t)}{t} \Rightarrow \\ \beta_i(t) &= \gamma_i(t) - k_i * \alpha_i(t) \end{aligned} \tag{2.7}$$

Strategies are divided into non-adaptive and adaptive strategies. In non-adaptive strategies, players can not get any feedback during the whole process. There are periodic play, exponential play and renewal play. In adaptive strategies, players are able to get certain forms of feedback. We summarize these strategies in the game scenario as follows:

- If both the attacker and the defender play periodically or exponentially, the optimal strategy for the defender is calculated by the Nash equilibrium.

- If the attacker plays renewal strategy, the optimal strategy for the defender is either periodic play or not playing (when the attacker plays very fast).
- If the attacker plays adaptively, the optimal strategy for the defender is to play exponentially. In this scenario, the defender cannot play periodically, because the attacker can learn the defender's strategy after the defender moves.

Above all, we can conclude two valuable lessons. Firstly, if the defender wants to win the game, the best strategy for him is to play faster than the attacker. Then, the attacker is unable to react immediately and forced to drop out. However, in order to play fast, the defender should make his cost of making a move much lower than that of the attacker. When the attacker takes over the control, the defender can retake control without taking much time and paying much cost. Secondly, the "FLIPIT" game provides guidance on when to implement the cost-driven defense. The analysis guides the defender in how often he should deploy security countermeasures; while it also helps the attacker with his strategies of how often to play in order to maximize the damage.

Chapter 3

Models and Methods

This chapter is divided into five sections. At first, we study the mathematical formulation of two standard epidemiology models (the SIR model and the SIS model). Secondly, we take the scale-free network parameter into consideration and show the way of combining it with epidemiology models. In the third section, we analyze the strategy making problem and construct game theoretical models based the content of previous two sections. In the fourth section, in order to determine the best security rate in game theoretical models, we work out cost-benefit models and present methods to determine the gain and cost when real data are provided. Finally, we discuss how to utilize all models to decide the best strategy and show the implementation of our models.

3.1 Mathematical Specification of Standard Epidemiology Models

In this section, we present the mathematical specification of two standard epidemiology models. Game theoretical models are extended from these two models. In general, epidemiology models are formulated over a fixed-size network. Nodes represent individuals. Links or edges between nodes represent contacts between individuals. The infection spreads along with the direct link between two nodes. Without the topology consideration, the epidemiology models assume that the infection of the virus evenly spreads over the simple network.

3.1.1 The SIR Model

The simplest SIR model is the "Kermack-McKendrick" model [22, 5] which computes the number of infected individuals in a fixed-size population over time. In the SIR model, the total population is divided into three parts: the susceptible individuals (denoted by S), the infected individuals (denoted by I) and the recovered individuals (denoted by R). Formula 3.1, 3.2 and 3.3 (from [22, 5]) show the increasing number of susceptible nodes, infected nodes and recovered nodes over time respectively. Here, β denotes the average number of transmissions possible from an infected node in each unit time, also regarded as the infection rate; γ denotes the average number of infected nodes who recover from an infection and become immune to the malware in each unit time, also regarded as the recovery rate.

$$\frac{dS}{dt} = -\beta IS \tag{3.1}$$

$$\frac{dI}{dt} = \beta IS - \gamma I \tag{3.2}$$

$$\frac{dR}{dt} = \gamma I \tag{3.3}$$

3.1.2 The SIS Model

The simplest SIS model is taken from [23]. In the SIS model, the total population is divided into two parts: the susceptible individuals (denoted by S) and the infected individuals (denoted by I). The model shows: at the beginning, the spread of the infection grows slowly, because the infected computers are few; then, with the increasing number of infected computers, the infection spreads fast; in the final phase, the infection grows slowly again and the number of infected computers reaches the maximum, because the chance of infecting a susceptible computer reduces. Formula 3.4 and 3.5 (from [23]) show the increasing number of susceptible nodes and infected nodes over time respectively. Here, β denotes the average number of transmissions possible from an infected node in each unit time, also regarded as the infection rate; γ denotes the average number of infected nodes who recover from an infection and become susceptible again in each unit time, also regarded as the recovery rate.

$$\frac{dS}{dt} = -\beta IS + \gamma I \tag{3.4}$$

$$\frac{dI}{dt} = \beta IS - \gamma I \tag{3.5}$$

3.2 Topology Consideration

We use the scale-free network as the topology. We make the assumption that the scale-free network is a homogeneous network in which all computers have similar configuration or run the same operating system. Although the scale-free network is considered as heterogeneity in most cases, malware proliferation behavior does not change in the homogeneous network. Then, we can use the average spread rate of malware in order to simply the situation.

At first, we construct a random scale-free network by using the "Barabasi-Albert" (BA) algorithm. The BA algorithm follows the "preferential attachment" procedure [14] which means that the new node is more likely to connect the highly connected nodes. By starting with a small number of disconnected nodes, it iterates the procedure for many times to get the network with the connectivity distribution of $P(k) \sim k^{-\gamma}$ in which $2 < \gamma < 3$. Secondly, after the construction of the scale-free network, we use the method of mean field approximation [25] to combine the parameter of scale-free network with epidemiology models. In formula 3.6, the parameter $< k >$ is "the average number of connections with the probability $P(k)$ of each degree k".

$$< k > = \sum\nolimits_k k * P(k) \tag{3.6}$$

By using the mean field method, we define $< k > I$ as the fraction of infected neighbours of a susceptible node. We take the standard SIR as the example. In formula 3.7, 3.8 and 3.9, we combine the scale-free network parameter with the SIR model.

$$\frac{dS}{dt} = -\beta < k > IS \tag{3.7}$$

$$\frac{dI}{dt} = \beta < k > IS - \gamma I \tag{3.8}$$

$$\frac{dR}{dt} = \gamma I \tag{3.9}$$

3.3 Game Theoretical Models

In this section, we consider the strategy making problem based on the analysis of malware. Then, we can determine what security strategies should be deployed and work out game theoretical models.

3.3.1 Strategy Making Problem

Before we make a decision about security strategies, we should decide the malware which we can utilize in game theoretical models.

With the study of malware classification, we have found that worms are self-replicating and able to spread without human intervention in the network [11]. Considering the limitation of models, we should analyze random-scanning and self-activated worms. As for random-scan, it means that worm spreads without topology constraint. Worms via emails have specific routes according to the email list of each infected computer; then, we cannot decide which susceptible computer becomes infected. Therefore, the random-scanning worm is the best choice. In terms of self-activation, it is the fastest activating way in which the worm usually exploits the vulnerability that exists in the computer system. Otherwise, we should determine the time that a human triggers the worm, which makes the situation more complicated.

Based on the analysis of worms, there are usually three security deployments: removal, patch, both patch and removal. When the computer is infected by a worm, we should download the removal tool to clear the worm and patch the vulnerability exploited by the worm in order to prevent re-infection; if we only remove the worm, the infected computer will become susceptible again. As for susceptible computers, we can patch the vulnerability to prevent infection. For each security deployment, we set up a game theoretical model with the scale-free network parameter to analyze the effect of this strategy.

3.3.2 Patch Strategy

When we use patch strategy, susceptible computers can become immune to the worm; but infected computers cannot recover from the infection. In this case, the worm and the defender seem to take part in a race. If the worm spreads very fast, it will infect most computers in a short time before defenders notice it; if people in the network can patch their computers much faster than the worm proliferation, the wide-range infection can be avoided.

We modify the standard SIR model and combine the scale-free network parameter to construct the model, shown in formula 3.10, 3.11 and 3.12. Here, we assume that the infected computer cannot be patched over the period of patch dissemination. We denote that R means recovered individuals from the susceptible state and the security rate γ is the patch rate.

$$\frac{dS}{dt} = -\beta < k > IS - \gamma IS \tag{3.10}$$

$$\frac{dI}{dt} = \beta < k > IS \tag{3.11}$$

$$\frac{dR}{dt} = \gamma IS \tag{3.12}$$

For recovered computers, the recovered process from the susceptible state is similar to the infected process [6]. At the beginning of the worm propagation, most people don't know the existence of the worm. Gradually, with the increasing infected computers, people notice the worm and begin to take some actions to defend against it. Finally, the patch rate decreases as the number of susceptible computers reduces to zero.

3.3.3 Removal Strategy

When we download the removal tool and clear the worm, infected computers can recover from the infection but become susceptible again. This conforms to the standard SIS model. We combine the scale-free network parameter with the SIS model to construct the model, shown in formula 3.13 and 3.14. Here, we denote that the security rate γ is the removal rate.

$$\frac{dS}{dt} = -\beta < k > IS + \gamma I \tag{3.13}$$

$$\frac{dI}{dt} = \beta < k > IS - \gamma I \tag{3.14}$$

3.3.4 Patch and Removal Strategy

The third strategy is to use both patch and removal. For susceptible computers, we patch the vulnerability to prevent the infection; for infected computers, we remove the worm at first and then patch the vulnerability to avoid reinfection. In this case, we have four states, susceptible (denoted by S), infected (denoted by I), recovered from the infection (denoted by R) and immunized from the susceptibility (denoted by Q). We have three state transitions: susceptible to infected, susceptible to immunized and infected to recovered and immunized. We can extend the standard SIR model and combine the scale-free network parameter to construct the model, shown in formula 3.15, 3.16, 3.17 and 3.18. Here, we assume that the patch rate for susceptible computers and the security rate (both patch and removal) for infected computers are the same.

$$\frac{dS}{dt} = -\beta < k > IS - \gamma(I + R)S \tag{3.15}$$

$$\frac{dI}{dt} = \beta < k > IS - \gamma I \tag{3.16}$$

$$\frac{dR}{dt} = \gamma I \tag{3.17}$$

$$\frac{dQ}{dt} = \gamma(I + R)S \tag{3.18}$$

3.4 Cost-Benefit Analysis

In game theoretical models, the worm spread rate is taken from the statistics of the real worm. Then, in order to determine the best security rate, we should analyze the payoff of worm behavior and security deployments. Thus, in this section, we work out cost-benefit models and present methods to determine the gain and cost of both worm and security actions when real data are fitted into our models.

3.4.1 General Description

Cost-benefit models are similar to the "FLIPIT" game models mentioned in the background chapter. As for the malware, it has the average spread rate and pays the cost for its propagating and infecting actions. As for the defender, each computer has defensive and responsive actions; then, there is a security rate in the network which means the average number of computers taking security actions to defend against the infection in unit time. In our models, we assume that players can not get any feedback during the worm spread period, because we use the average rate of play which stays constant.

3.4.2 Cost-Benefit Model

Now, we give the formulation of cost-benefit models. We suppose that there is a network consisting of a large number of computers, and the malware is player 0, the defender is player 1. All computers are the critical resource.

The malware tries to infect computers as many as possible; the defender tries to protect each computer against infection or restore it after its infection.

At any time of the worm spread period, a computer in the network is controlled by either the worm or the defender. Therefore, for each player i, we define the gain $G_i(t)$ as the amount of time for the fraction of the population in control from the start of the game up to time t. We show it in formula 3.19. Here, $P_i(t)$ denotes the fraction of population in control over time t.

$$G_i(t) = \int_0^t P_i(t) * t * dt \tag{3.19}$$

Thus, the total time of the spread period is the sum of each player's amount of time for their controlled population which is shown in formula 3.20.

$$G_0(t) + G_1(t) = t \tag{3.20}$$

Then, for each player i, we define the average gain rate $\lambda_i(t)$ in formula 3.21.

$$\lambda_i(t) = \frac{G_i(t)}{t} \tag{3.21}$$

A computer may have several state transitions during the whole spread period: a susceptible computer becomes infected at any time; a computer keeps disinfected state if it changes from susceptible state to immunized state by patch; an infected computer recovers from infection by patch and removal strategy; an infected computer becomes susceptible again by only removing the worm. Thus, the average gain rate can be approximated as the average fraction of population in control for each player. To sum up, in formula 3.22, we define the average gain rate $\lambda_i(t)$ for player i as the average fraction of population in control up to time t.

$$\lambda_0(t) + \lambda_1(t) = 1 \tag{3.22}$$

For each player, we define the cost $C_i(t)$ as total moves that player i has taken $n_i(t)$, times the cost of each move k_i, which is shown in formula 3.23.

$$C_i(t) = n_i(t) * k_i \tag{3.23}$$

In formula 3.24, the total moves for player i is denoted as the average rate of play $\alpha_i(t)$ times total time t.

$$n_i(t) = \alpha_i(t) * t \tag{3.24}$$

Thus, the average cost rate $\mu_i(t)$ is denoted as the average rate of play $\alpha_i(t)$ times the cost of each move k_i, shown in formula 3.25.

$$\mu_i(t) = \frac{C_i(t)}{t} = \alpha_i(t) * k_i \tag{3.25}$$

Players receive benefit which is equal to the total time for their controlled population, minus the cost of their total moves. We denote the benefit $B_i(t)$ for each player i in formula 3.26.

$$B_i(t) = G_i(t) - C_i(t) \tag{3.26}$$

Thus, the average benefit rate $b_i(t)$ is shown in formula 3.27.

$$\begin{aligned} b_i(t) &= \frac{B_i(t)}{t} \Rightarrow \\ b_i(t) &= \frac{G_i(t)}{t} - \frac{C_i(t)}{t} \Rightarrow \\ b_i(t) &= \lambda_i(t) - \mu_i(t) \Rightarrow \\ b_i(t) &= \lambda_i(t) - \alpha_i(t) * k_i \end{aligned} \tag{3.27}$$

Finally, in formula 3.28, we define the overall benefit rate of the network, which equals to the defender's benefit rate minus the attacker's benefit rate.

$$b_1(t) - b_0 t = (\lambda_1(t) - \alpha_1(t) * k_1) - (\lambda_0(t) - \alpha_0(t) * k_0) \tag{3.28}$$

3.4.3 Methods to Determine Gain and Cost

After the construction of cost-benefit models, we should find out the way to determine the gain and cost when real data are provided.

As for the gain, we should decide the fraction of population in control for each player. We focus on the analysis of the average infected population, then the average disinfected population is denoted by one minus the average infected population.

In patch strategy, the malware spreads in the susceptible population; and the patch can not make an infected computer recover. Thus, the average fraction of infected population controlled by malware, p, is determined by the initial infection and the spread efficiency in the susceptible population, which is shown in formula 3.29. Here, I_0 denotes the initial infection and S_0 denotes the initial susceptible population.

$$p = I_0 + \frac{S_0 < k > \beta}{2\gamma} \tag{3.29}$$

In removal strategy, the malware spreads in the susceptible population. The removal strategy clears the malware but can not make an infected computer become immune to the infection. Thus, the average fraction of infected population controlled by malware is determined by the average fraction of residual initial infection and the spread efficiency in the susceptible population, which is shown in formula 3.30.

$$p = \frac{I_0(1-\gamma)}{2} + \frac{(S_0 + I_0\gamma) < k > \beta}{2\gamma} \tag{3.30}$$

In the patch and removal strategy, all infected computers can recover from the infection and become immune to the malware at the end of the spread period. Therefore, the average fraction of the infected population is determined by the change of the initial infection and the spread efficiency in the susceptible population. In this case, the spread efficiency has some direct and proportional relation with the average fraction of residual infected neighbours of a susceptible computer, because the infected computer may recover from the infection and can not infect other computers. Thus, the average fraction of the infected population is shown in formula 3.31.

$$p = \frac{I_0}{2} + \frac{S_0(I_0 - I_0\gamma) < k > \beta}{2\gamma} \tag{3.31}$$

As for the cost, we utilize the operational quantity and complexity to determine it. For each player, they will take several actions to reach their goal; for each action, there is a complexity level. We set up three complexity levels, low, medium and high and give a score to each of them, 1, 2 and 3. We show an example in table 3.1. By using this standard, we can decide the cost of each player when they take an action.

Table 3.1: An example of calculating the cost.

Actions	Complexity			Total
	Low:1	Medium:2	High:3	
Action 1	1			m
Action 2			3	
...				
Action n		2		

3.5 Summary

In this section, we discuss how to use game theoretical models and cost-benefit models to find out the best security strategy and present the implementation of our models.

In general, we should take two things into consideration when we want to decide the best strategy: the initial fraction of infection and factors to determine the best strategy. As for the initial infection, defenders in the network can take actions in different stages of the spread period, because they may notice the infection at any time. Therefore, by analyzing each strategy in different initial fractions of infected computers, we can know what strategy should be deployed when we discover the worm and begin to take actions at any time. In order to determine the best strategy, we should analyze four factors including the overall benefit rate, the defender's benefit rate, the final infected population and the infection change; then, we combine these factors to analyze each strategy comprehensively and give the conclusion. We use Matlab to implement all theoretical models.

At first, we consider a medium-sized and homogeneous network which consists of computers running same operation system such that the worm has the equal chance to infect each computer. We assume that all computers are always online. Thus, the worm can infect the computer at any time. We download the source code of generating a scale-free network from the Internet. We use a random seed as the original network which consists of 5 nodes and produce a network of 1000 nodes in the scenario. Then, we download another source code of calculating the frequency of each degree and modify a small part of it to calculate the network parameter.

Secondly, we use Matlab to implement cost-benefit models. For each strategy, we have different formulas to find out the maximum of the overal benefit rate when the initial infection, the network parameter, the cost of both malware infection and security strategy, and the average spread rate of the worm are provided. By working out the best overall benefit rate, we can get the best security rate. For each strategy, we present the way to calculate the maximum of the overall benefit rate in formula 3.32, 3.33 and 3.34 respectively.

For patch strategy,

$$\begin{aligned}&(1 - I_0 - \frac{S_0 < k > \beta}{2\gamma} - k_1\gamma) - \\ &(I_0 + \frac{S_0 < k > \beta}{2\gamma} - k_0 < k > \beta).\end{aligned} \tag{3.32}$$

For removal strategy,

$$\begin{gathered}(1-\frac{I_0(1-\gamma)}{2}-\frac{(S_0+I_0\gamma)<k>\beta}{2\gamma}-k_1\gamma)- \\ (\frac{I_0(1-\gamma)}{2}+\frac{(S_0+I_0\gamma)<k>\beta}{2\gamma}-k_0<k>\beta).\end{gathered} \tag{3.33}$$

For patch and removal strategy,

$$\begin{gathered}(1-\frac{I_0}{2}-\frac{S_0(I_0-I_0\gamma)<k>\beta}{2\gamma}-k_1\gamma)- \\ (\frac{I_0}{2}+\frac{S_0(I_0-I_0\gamma)<k>\beta}{2\gamma}-k_0<k>\beta).\end{gathered} \tag{3.34}$$

Besides the best security rate and the overall benefit rate, we can also calculate the defender's benefit rate of each strategy.

Finally, we use Matlab to implement three game theoretical models. Then, for each strategy, we fit the real data into the corresponding game theoretical model and write a function to produce the final infected population (the population in the steady state of the network) when the initial infection, the network parameter, the average spread rate of the worm, the security rate, and the time of malware spread period are provided. Here, we set the time of malware spread period as the maximum value, which guarantees that the network goes into the stable state for each initial infection. We should also compute the change of infection, which is the final infected population minus the initial infected population.

The initial fraction of the infected population starts from 0.02 to 0.98 and increases 0.02 by each step. For each initial infection, we repeat the above procedure. Because we should analyze four factors, we produce a trend line of each factor by connecting the factor point of different initial infections. For each strategy, we produce four trend lines. For different strategies, we put the lines of the same factor in one graph to compare the difference between them.

In practical situations, we should decide the parameters before we conduct the analysis. At first, the network parameter can be calculated by collecting data from the network. Then, from the record of anti-virus software, we can compute the average worm spread rate and the initial infection, and estimate the cost of security deployments. Finally, we can produce the cost of worm infection and the worm spread period by approximation after doing some research on the worm infection mechanism.

Chapter 4

Case Study and Analysis

In this chapter, we fit real data into computer models and analyze the results. We study three worms, Code-Red, SQL Slammer and Conficker. They are all random-scanning worms with no topology constraints, and they are self-activated by exploiting the vulnerability which exists in the operating system or software. What's more, each of them has its striking features. Therefore, we can learn different things from each worm analysis.

4.1 Code-Red

Code-Red worm was discovered in July 2001. It exploits the "buffer-overflow" vulnerability in "Microsoft IIS Web Server". It produces a list of random IP addresses and launches 99 threads to search each computer in the list in order to infect as many vulnerable computers as possible. It has two versions: Code-Red v1 and Code-Red v2. Code-Red v1 spreads slowly because it generates an identical list by using a static seed. Code-Red v2 is the variant of Code-Red v1. It can infect new computers by using a random seed for its pseudo-random generator. Therefore, the new version has a greater impact on global resource because of its high spread speed. The greatest damage is that it can launch a massive DoS attack. This worm is memory resident. We can reboot computers to clear the worm. In order to prevent reinfection, we should also patch computers.

In this scenario, we suppose that the homogeneous scale-free network consists of 1000 computers, and each computer is running Microsoft Operating System which includes the Microsoft IIS Web Server. We take the average spread rate of Code-Red from [24] (0.00045). We set the maximum time of the worm spread period as 10000 seconds, 50000 seconds and 100000 seconds for patch strategy, removal strategy, patch and removal strategy respectively. We summarize the cost of worm proliferation and defender's security deployments in table 4.3 and 4.4 respectively.

Table 4.1: The cost of Code-Red.

Actions	Complexity			Total
	Low:1	Medium:2	High:3	
Exploit the buffer overflow vulnerability	1			3
Generate random IP addresses	1			
Launch 99 threads with generated IP addresses	1			

Table 4.2: The cost of security deployments.

Actions		Complexity			Total
		Low:1	Medium:2	High:3	
Patch	Detection	1			2
	Patch	1			
Removal	Detection	1			2
	Reboot	1			
Both	Detection	1			5
	Patch	1			
	Reboot	1			

By deploying the methods and models mentioned in the third chapter, we compute and analyze four factors of each strategy. We show the overall benefit rate and defender's benefit rate in figure 4.6 and 4.7 respectively, the spread rate and security rates in figure 4.8, the infected population and change of infection in figure 4.9 and 4.10 respectively.

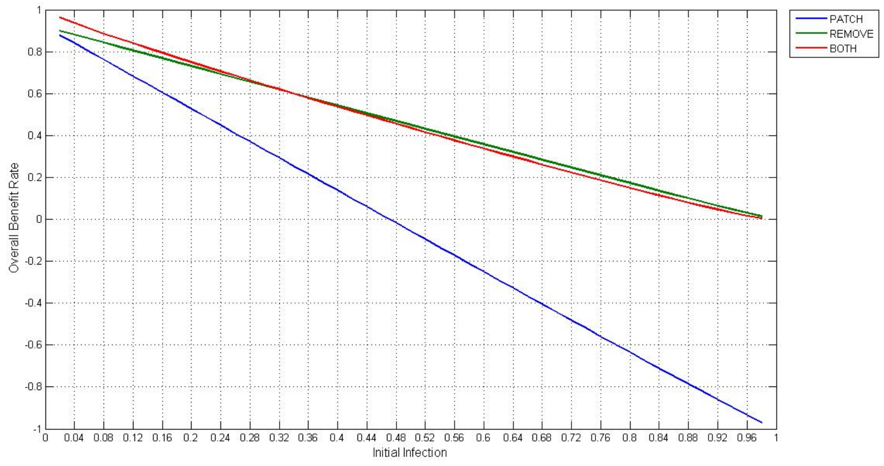


Figure 4.1: The overall benefit rate.

In patch strategy, the overall benefit rate is higher than zero when the fraction of initial infection is less than 0.48 and then becomes less than zero after that. The defender's benefit rate is higher than zero for each initial infection. Both benefit rates have the decreasing trend with the increasing initial infection, and they are lowest compared with benefit rates of the other two strategies. The patch rate reduces with the rising initial infection and it is higher than the worm propagation rate for each initial infection, because the cost of this strategy is lower than the cost of worm spread behavior. For each initial infection, the change of the infected population keeps stable around zero which means that the infected population does not increase much although patch cannot restore infected computers.

In removal strategy, the overall benefit rate and defender's benefit rate decrease steadily when the initial infection rises and keep much higher than zero. The removal rate has the similar trend with the patch rate. For each initial infection, the final infected population is zero at the end of the spread period.

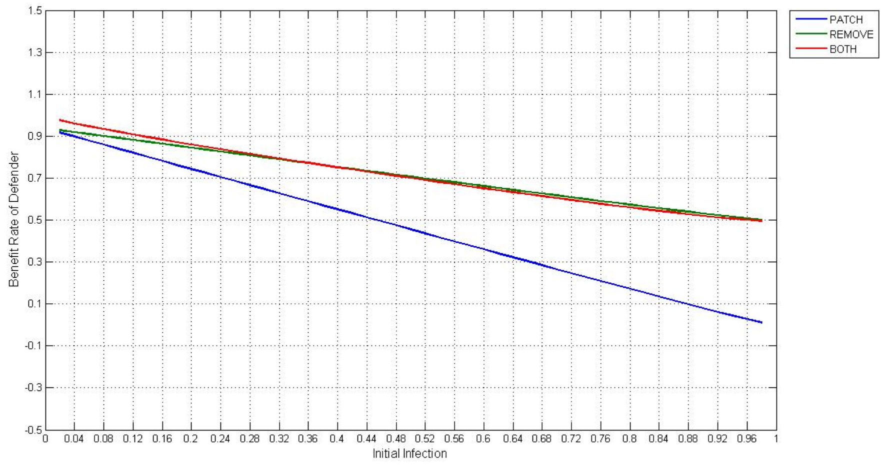


Figure 4.2: The defender's benefit rate.

In the patch and removal strategy, both benefit rates have similar trend with those of the removal strategy. The security rate is higher than the spread rate for each initial infection. It increases when the initial infection is less than 0.5 and decreases after that. For each initial infection, the final infected population keeps zero at the end of the spread period.

From the above analysis of the graphs, we can provide some suggestions about how to choose the best strategy when the following conditions are met: the worm spreads with low speed and low cost of infection; the cost of the patch is lower than the cost of infection; the cost of removal is lower than the cost of infection.

Because the worm spreads slowly, defenders in the network have enough time to notice the infection before the worm infects other computers. Therefore, even if we use patch strategy, the infection does not increase a lot and keeps the initial infection fraction.

If defenders use removal strategy, they can get highest benefit and all infected computers can restore from the infection. Because the cost of this strategy is lower than the cost of worm, defenders can play faster than the spread of the worm.

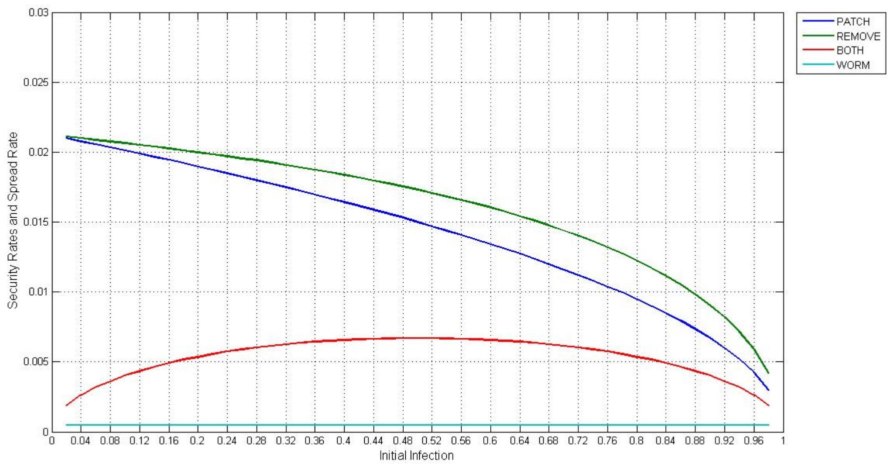


Figure 4.3: The spread rate of Code-Red and security rates of three strategies.

Defenders can also use patch and removal strategy. Deploying this strategy will pay more cost for each move, but the security rate is much lower than that of removal strategy; therefore, defenders can get high benefit just like in the removal strategy. What's more, defenders are able to play faster than the worm in this strategy, because the cost of this strategy is just a bit higher than the cost of worm and the worm spreads with low speed.

4.2 SQL Slammer

SQL Slammer is the first real-world worm with high spread speed. It started to propagate in January 2003. It exploits the “buffer-overflow” vulnerability in computers running ”Microsoft SQL Server” or “Microsoft SQL Server Desktop Engine (MSDE) 2000” [10]. SQL Slammer has no malicious payload. Its payload is a small piece of code which produces a list of random IP addresses and sends itself to those computers in the list. If the worm finds a vulnerable computer among the list of IP addresses, it infects the computer by exploiting the buffer overflow bug. After the infection, it starts to search other computers in order to infect more computers. It spreads very fast because its payload becomes a UDP (User Datagram Protocol) packet with the

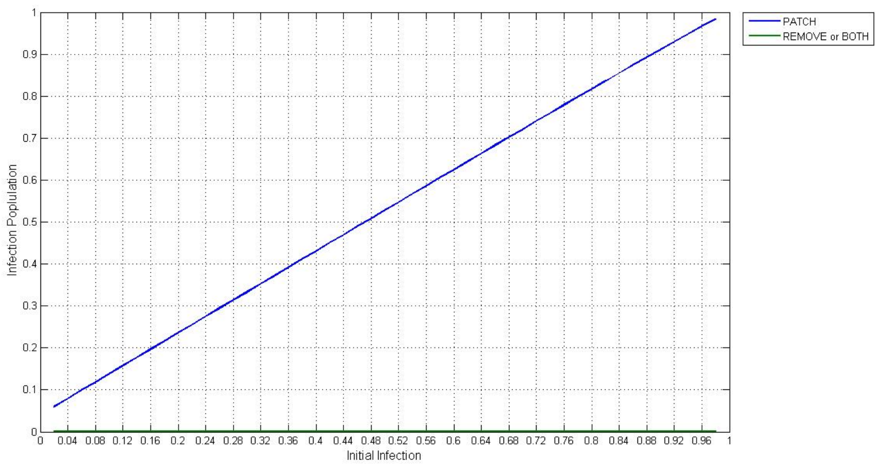


Figure 4.4: The final infected population.

header and goes to the UDP port 1434; then, it broadcasts scans and does not require responses from the destination computers. It can drastically slow down the internet traffic and disable data servers because of its high spread speed. This worm resides in memory, so we can remove it by simply rebooting the computer. However, rebooting cannot prevent reinfection when the computer is online again. We should patch the computer to avoid reinfection.

In this scenario, we suppose that the homogeneous scale-free network consists of 1000 computers, and each computer is running Microsoft Operating System which includes Microsoft SQL Server. We take the average spread rate of SQL Slammer from [24] (0.117). We set the maximum time of the worm spread period as 100 seconds, 500 seconds and 500 seconds for patch strategy, removal strategy, patch and removal strategy respectively. We summarize the cost of worm proliferation and defender's security deployments in table 4.1 and 4.2 respectively.

By using the methods and models mentioned in the third chapter, we compute and analyze four factors of each strategy. The overall benefit rate and defender's benefit rate are shown in figure 4.1 and 4.2 respectively; the spread rate and security rates are presented in figure 4.3; the infected population and change of infection are displayed in figure 4.4 and 4.5 respectively.

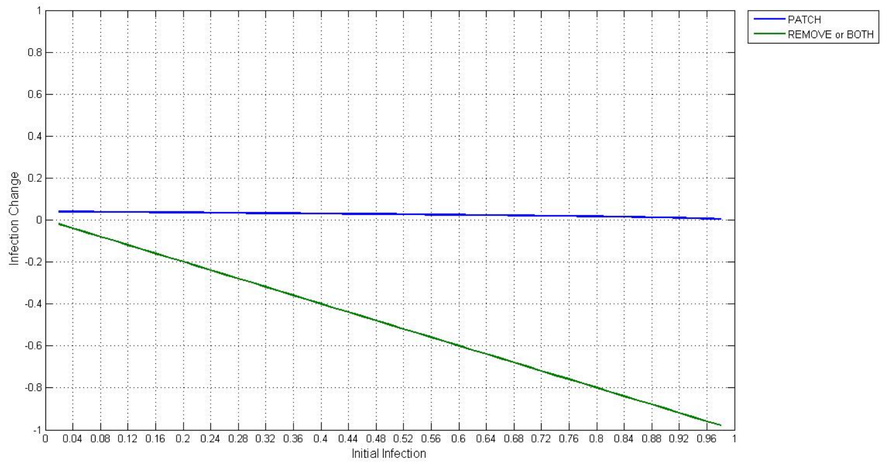


Figure 4.5: The change of infection.

Table 4.3: The cost of SQL Slammer.

Actions	Complexity			Total
	Low:1	Medium:2	High:3	
Exploit the buffer overflow vulnerability	1			2
Random scanning	1			

In patch strategy, both benefit rates are less than zero for most initial infections. The overall benefit rate decreases when the initial infection is lower than 0.9; the defender's benefit rate decreases when the initial infection is lower than 0.75. Both benefit rates are lowest compared with benefit rates of the other two strategies. The patch rate also reduces with the rising initial infection and it is higher than the worm propagation rate when the initial infection is less than 0.88. When the initial infection is high, the patch rate becomes lower than the worm spread rate in order to cut down the defender's cost and increase the overall benefit rate. For each initial infection, the change of the infected population is greater than zero which means that the final infected population goes up compared with the initial infected population, because patch cannot restore infected computers.

Table 4.4: The cost of security deployments.

Actions		Complexity			Total
		Low:1	Medium:2	High:3	
Patch	Detection	1			2
	Patch	1			
Removal	Detection	1			2
	Reboot	1			
Both	Detection	1			5
	Patch	1			
	Reboot	1			

In removal strategy, the overall benefit rate keeps stable above zero. The defender's benefit rate increases steadily when the initial infection rises and is higher than zero when the initial infection is greater than 0.2. The removal rate has the similar trend with the patch rate. It keeps higher than the worm spread rate for most initial infections such that the infection can not spread very fast. When the fraction of initial infection is less than 0.68, the final infected population keeps zero. It means that removal strategy clears the worm before it spreads. When the fraction is higher than 0.68, the infected population increases sharply. The infection can not be cleared but stays in a stable level, because the initial infection is very high. For each initial infection, the change of infected population is less than zero which means that the final infected population goes down compared with the initial infected population.

In the patch and removal strategy, the overall benefit rate decreases when the initial infection is less than 0.75. The defender's benefit rate decreases when the initial infection is less than 0.7. When the fraction of initial infection is less than 0.4, both benefit rates are highest compared with benefit rates of the other two strategies. The reason is that when the infected population is small, it's possible to control the infection spread effectively by clearing the worm and patching the vulnerability. With the increasing initial infection, the cost of the defender will increase, because the defender should pay more cost to restore the infected population. So the overall benefit rate and defender's benefit rate decrease below zero. When the initial infection is greater than 0.75, the overall benefit rate increases. When the initial in-

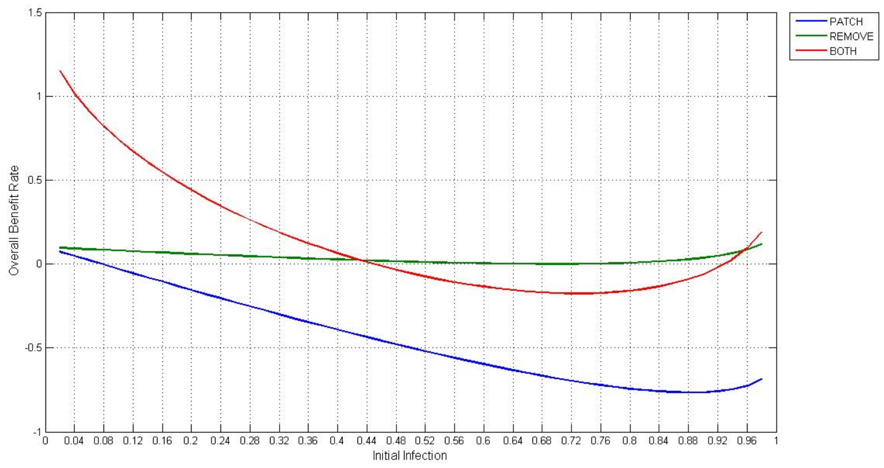


Figure 4.6: The overall benefit rate.

fection is greater than 0.7, the defender's benefit rate also increases. The security rate is lower than the spread rate for each initial infection, because the cost of this strategy is higher than the cost of worm spread behavior. It increases when the initial infection is less than 0.5 and decreases after that. At the beginning, the fraction of infection is small, so the security rate keeps low but can still clear the worm. When the initial infection grows higher, the security rate decreases in order to control the defender's cost and increase the overall benefit rate. For each initial infection, the infected population becomes zero no matter when defenders notice the worm and take actions.

From the above analysis of the graphs, we can provide several suggestions about how to choose the best strategy when the following conditions are met: the worm spreads with high speed and low cost; the cost of the patch is not greater than the cost of infection; the cost of removal is not greater than the cost of infection.

When the initial infection is small, if patch and removal strategy can be deployed, defenders in the network can get the highest benefit and all infection can be cleared; but they can not play faster than the worm, because the cost of this strategy is higher than the cost of worm spread behavior. When defenders notice the infection very late, they can also use this strategy,

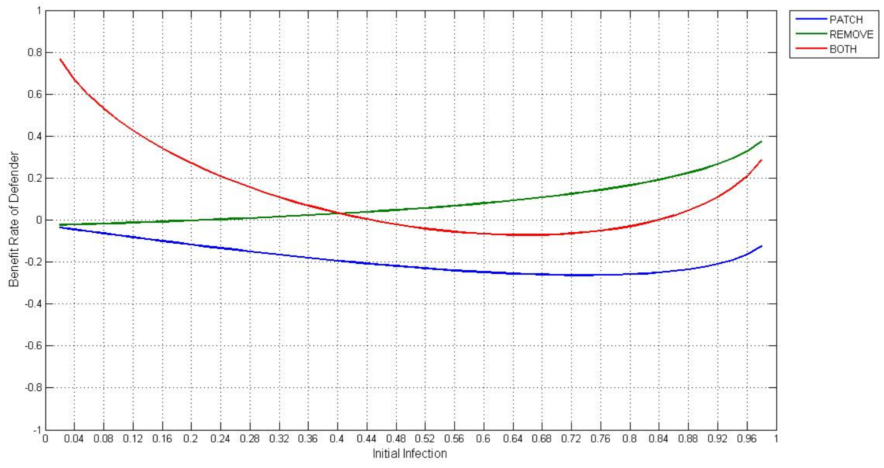


Figure 4.7: The defender's benefit rate.

because the benefit does not have large distance with the highest benefit. What's more, the worm can be cleared; otherwise, the infected population will remain in the network.

If the defender can discover the infection and clear the worm much faster than the worm proliferation, the worm is unable to spread its infection. Therefore, the defender can use the removal strategy to clear the worm when the number of infected computers is small.

Patch is the worst way to use when the worm begins to spread. However, if defenders form a good habit of patching their computers every time the patch is released, it will be the best way to prevent the worm.

4.3 Conficker

Conficker was first observed in November 2008. It exploits the "MS0–067" vulnerability in the Windows Operating System to inject executable code into Windows Server Service [19]. It probes new computers by generating a list of domain names and choosing a subset of all domain names which are

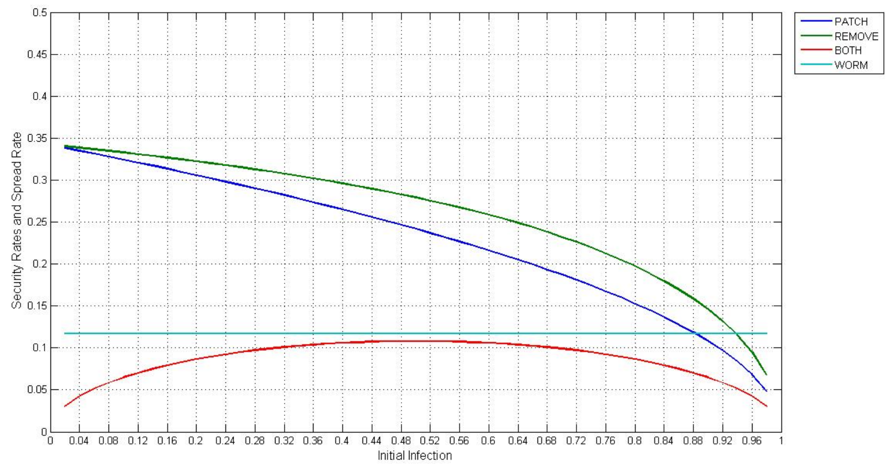


Figure 4.8: The spread rate of SQL Slammer and security rates of three strategies.

resolved to IP addresses by DNS resolvers. In order to prevent detection, it has developed into several variants with the improved self-defense ability, such as removing the antivirus software. It has a great impact on the network. It launches dictionary attacks on administrator passwords in order to control infected computers remotely and form a botnet. Because of the high complexity of its different variants, it's much more difficult to detect the infection. We should download the removal tool to clear the worm and patch the computer to avoid the reinfection; what's more, we need to restore our computer by modifying our passwords.

In this scenario, we suppose that the homogeneous scale-free network consists of 1000 computers, and each computer is running Microsoft Operating System. We take the statistics of Conficker detection rates from [2] which is shown in figure 4.11 and calculate the average spread rate (0.0741). We set the maximum time of the worm spread period as 1000 seconds, 5000 seconds and 5000 seconds for patch strategy, removal strategy, patch and removal strategy respectively. We summarize the cost of worm proliferation and defender's security deployments in table 4.5 and 4.6 respectively.

By using the methods and models mentioned in the third chapter, we

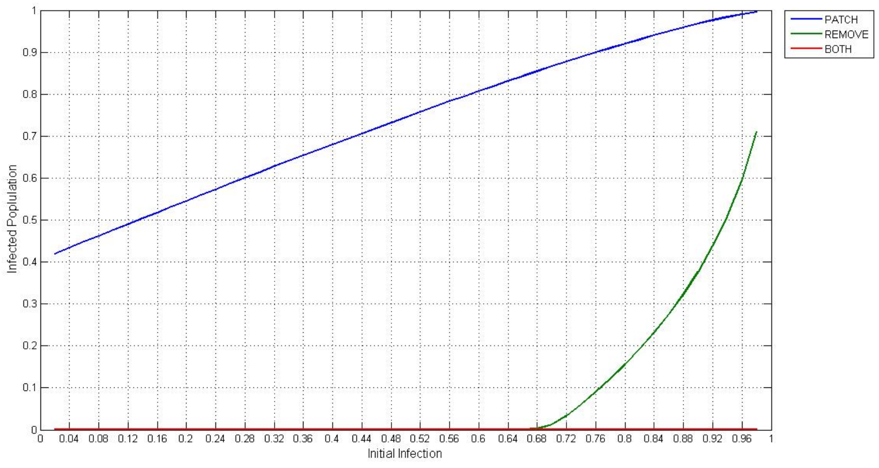


Figure 4.9: The final infected population.

compute and analyze four factors of each strategy. We present the overall benefit rate and defender's benefit rate in figure 4.12 and 4.13 respectively, the spread rate and security rates in Figure 4.14, the infected population and change of infection in figure 4.15 and 4.16 respectively.

In patch strategy, the overall benefit rate decreases when the initial infection is lower than 0.9. It is higher than zero when the fraction of initial infection is less than 0.32, and it becomes the lowest compared with overall benefit rates of the other two strategies when the fraction of initial infection is greater than 0.4. Defender's benefit rate keeps stable below zero for each initial infection and becomes the lowest when the fraction of initial infection is higher than 0.5. The patch rate also reduces with the rising initial infection and it is higher than the worm propagation rate when the initial infection is less than 0.85. Because the cost of this strategy is lower than the cost of worm and the initial infection is not high, the defender can play faster than the worm. When the initial infection is very high, the patch rate becomes lower than the worm spread rate in order to cut down the defender's cost and increase the overall benefit rate. For each initial infection, the change of the infected population is greater than zero which means that the final infected population goes up compared with the initial infected population.

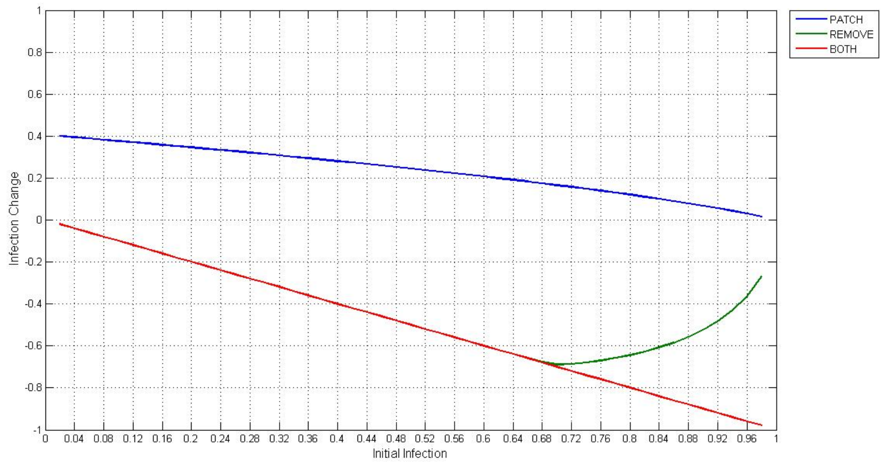


Figure 4.10: The change of infection.

In removal strategy, the overall benefit rate increases steadily with the increasing initial infection. It is the lowest when the fraction of initial infection is less than 0.4. The defender's benefit rate also increases when the initial infection rises, and it is the lowest when the fraction of initial infection is less than 0.5. The removal rate has the similar trend with the patch rate. For each initial infection, the change of the infected population is less than zero which means that the final infected population goes down compared with the initial infected population. However, the final infected population does not become zero and keeps a stable fraction for each initial infection.

In the patch and removal strategy, both benefit rates decrease when the initial infection is less than about 0.65. When the fraction of initial infection is less than 0.6, both benefit rates are highest compared with the other two strategies. Although the cost of this strategy is much higher than the cost of worm spread, it's possible to control the infection spread effectively when the infected population is small. The security rate is lower than the spread rate for each initial infection. It increases when the initial infection is less than 0.5 and decreases after that. For each initial infection, the final infected population keeps zero at the end of the spread period.

From the above analysis of the graphs, we can provide several suggestions

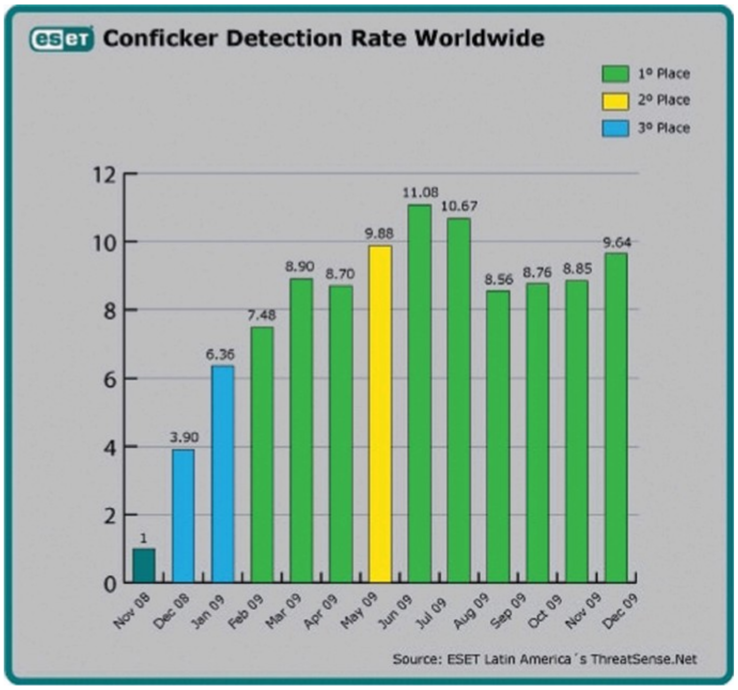


Figure 4.11: Conficker Detection Rate Worldwide (from [2]).

about how to choose the best strategy when the following conditions are met: the cost of worm spread is high; the cost of the patch is lower than the cost of infection; the cost of removal is higher than the cost of infection.

In this case, the worm spreads not fast. The patch and removal strategy is the best when the initial infection is not high. Defenders can gain highest benefit and all infection can be cleared. Because the cost of this strategy is much higher than the cost of worm, defenders can not play faster than the worm. When defenders notice the infection very late, they can also use this strategy if they want to clear the worm. Moreover, the benefit keeps above zero and is similar to the highest benefit.

The cost of removal is higher than the cost of the worm. To control the overall cost, the removal rate cannot be high. Therefore, the infected population will decrease but stay in a stable level higher than zero.

Patch is still the worst way to use when the worm begins to spread.

Table 4.5: The cost of Conficker.

Actions	Complexity			Total
	Low:1	Medium:2	High:3	
Load DDL and exploit the buffer overflow vulnerability		2		6
Dictionary attack	1			
Generate a list of domain names via TLDs	1			
Self-defense		2		

Table 4.6: The cost of security deployments.

Actions		Complexity			Total
		Low:1	Medium:2	High:3	
Patch	Detection			3	4
	Patch	1			
Removal	Detection			3	7
	Removal	1			
	Restoration			3	
Both	Detection			3	12
	Removal	1			
	Restoration			3	
	Patch	1			

4.4 Key Observations

From the above analysis, we have some key observations. In all cases, the cost of using both patch and removal is higher than the cost of worm infection.

When the cost of removal strategy is not greater than the cost of worm spread:

- If the worm spreads with high speed, the best strategy is to use both patch and removal. Defenders can clear all infection and get higher

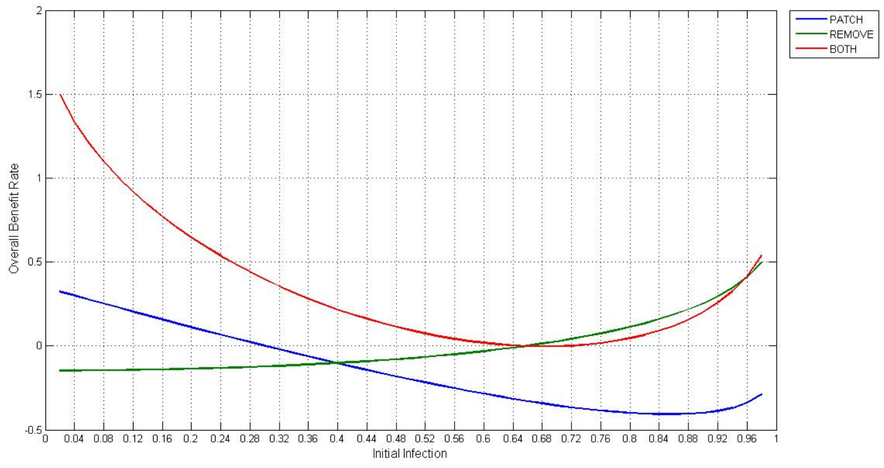


Figure 4.12: The overall benefit rate.

benefit rate; but, they can not play faster than the worm spread because the cost of this strategy is higher than the cost of worm spread. If the defender can discover the infection in the early spread period, the defender can also use the removal strategy. Because the cost of this strategy is lower than the cost of worm infection, the defender can play faster than the worm spread, then the worm is unable to spread its infection.

- If the worm spreads with low speed, the defender has enough time to notice the infection before the worm infects other computers. The best strategy is to use removal or both patch and removal.

When the cost of removal strategy is higher than the cost of worm:

- The best strategy is to use both patch and removal.

In both cases, which means that no matter what the cost of patch strategy is:

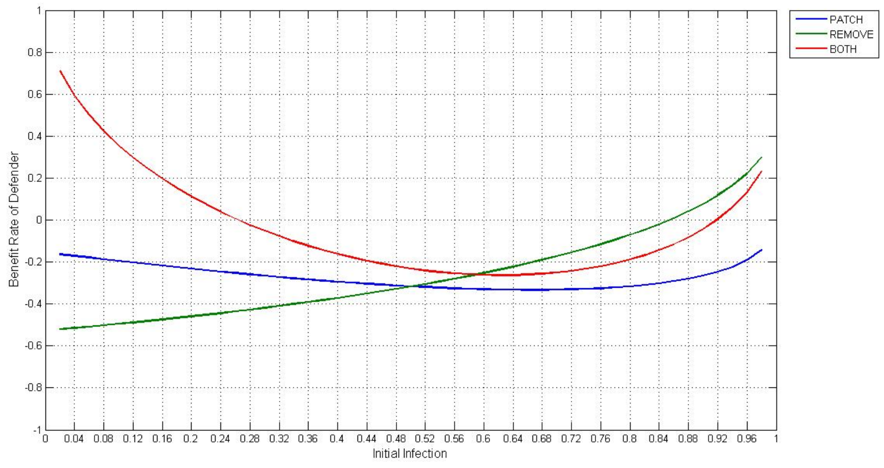


Figure 4.13: The defender's benefit rate.

- Patch is the worst way to use when the worm begins to spread, because infected computers cannot restore. However, if the defender forms a good habit of patching the computer every time the patch is released, it will be the best way to prevent the worm.

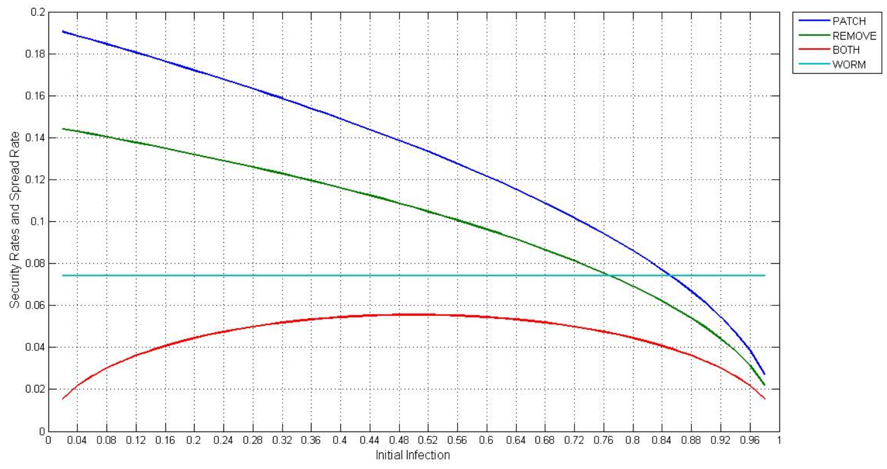


Figure 4.14: The spread rate of Conficker and security rates of three strategies.

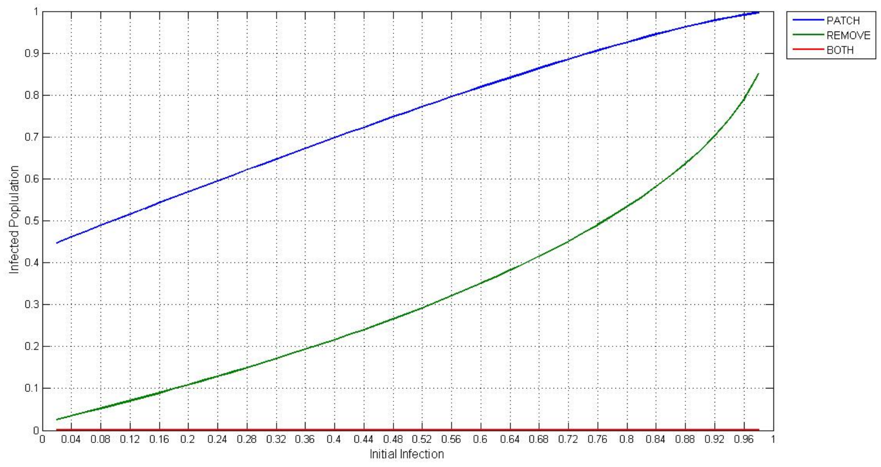


Figure 4.15: The final infected population.

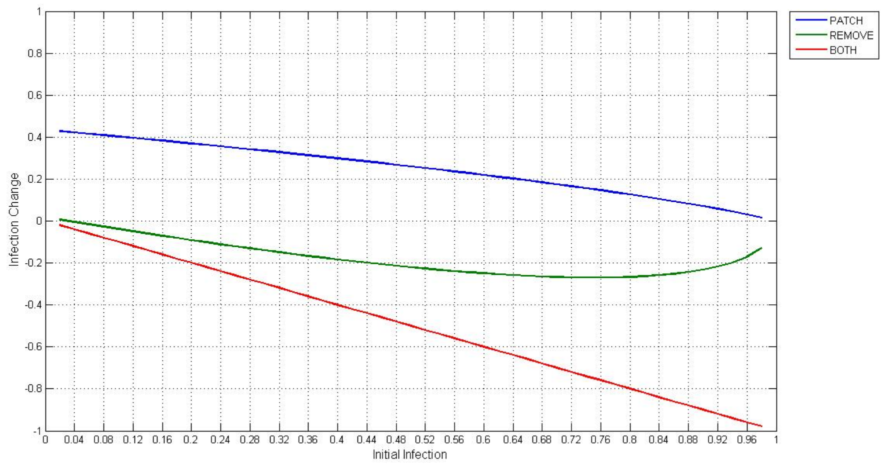


Figure 4.16: The change of infection.

Chapter 5

Evaluation

In this chapter, we review aims and objectives to discuss whether the project has been completed successfully.

The first aim is to combine epidemiology modeling with game theory to construct our computer models which are specialized to the defense against malware problem in the computer network. At first, we investigated the background knowledge of malware proliferation and epidemiology modeling. Then, we studied the complex topology. We found that the homogeneous scale-free network was the most suitable network, and we worked out the way to combine it with epidemiology models by utilizing mean field approximation. After that, we investigated the strategy making problem based on the research of malware classification. We decided to analyze the random-scanning and self-activated worms, and chose three security strategies, patch, removal and both patch and removal. Therefore, combined with all above analysis, game theoretical models were completed to simulate worm proliferation behavior when different strategies were deployed. After that, we studied the "FLIPIT" game scenario in which the cost-benefit method is used to determine the best defense strategy. We used the same method to work out cost-benefit models. Then, we were able to find out the best rate of play (the average number of computers with the security action) for each strategy. As for the best strategy, we considered different initial infections, and utilized game theoretical models and cost-benefit models to calculate four factors which can determine the best strategy, including the overall benefit rate, the defender's benefit rate, the final infection fraction and the infection change. Finally, we used Matlab to implement all theoretical models.

The second aim is to analyze real malware by utilizing computer models and provide some advice on what security strategies should be deployed in different malware scenarios. We chose three worms, SQL Slammer, Code-Red and Conficker. They are all random-scanning and self-activated worms, and they exploit the vulnerability which exists in the operating system to infect the computer. Thus, using patch to avoid reinfection and downloading removal tool to clear the worm are the most common way. Each worm has its striking feature. SQL Slammer spreads very fast, and Code-Red spreads slowly. For both worms, the cost of infection is low; defenders can use patch, reboot or both patch and reboot strategy; the cost of either patch or reboot is very low. Therefore, by comparing these two worms, we had some key observations. Conficker is much more complex such that the cost of infection is high. The cost of detection and restoration is also high. By analyzing this worm and comparing it with SQL Slammer and Code-Red, we had other useful findings.

A key factor to the success of this project was the reasonable and comprehensive workplan. For one thing, we allocated five weeks for the construction of models and three weeks for doing data analysis. Each time schedule was enough to guarantee the completion of the work. For another, we did the risk analysis and contingency plan before we started to do the project. We predicted several risks which could happen with high probability and worked out corresponding plans to deal with such terrible situations. Therefore, the project went smoothly during the whole development process.

The project dealt with the malware problem by using epidemiology modeling and game theory. It showed how we combined models and methods of these two areas to analyze malware proliferation behavior and to work out the way to determine the best security strategy. It showed how cost-benefit analysis could be valuable to guide the decision making. It also showed how we carried out data analysis and decided the best deployment in different malware scenarios.

Chapter 6

Future Work

When we designed theoretical models, we made some assumptions to simplify the situation. Thus, in this chapter, we list some further work which can be conducted to improve models and methods.

The first limitation is that we used the homogeneous network. In most cases, the scale-free network is considered as the heterogeneous network, because it consists of several highly connected hubs. In the real network, hubs are usually routers and different from computers. Therefore, we can develop game theoretical models by considering the heterogeneous network. If we utilize the heterogeneous network, we can analyze more reasonable network scenarios by dividing the network into the susceptible part and the immunized part.

Secondly, we assumed that the infection behavior does not change in the network during the whole spread period such that we can use the average spread rate of the worm in models. However, the spread rate of the worm changes during the whole propagation period. Different malware has different trend of the spread rate. In our models, we can modify the constant spread rate into dynamic one by adding parameters which vary with the time. Then, we need to do more research on how to carry out cost-benefit analysis to find out the best strategy.

In models, we analyzed random-scanning and self-activated worms and assumed that all computers in the network are always online. This simplified models, because we did not need to consider the route the worm spreads and the time a human triggers the infection. In order to analyze various types of malware, it is interesting to consider more factors which conform to the new malware behavior, and fit them into models.

Chapter 7

Conclusion

We have constructed computer models, which are specialized to the defense against random-scanning and self-activated worms. Three security strategies have been utilized: patch, removal, both patch and removal. We have carried out cost-benefit analysis and worked out the cost-driven method to determine the best strategy. We have investigated three different worms with striking features and got several useful findings. Although there is still much work to do for further improvement, this project gives valuable guidance to analyze malware proliferation problem.

Appendix A

We put key source code of the project here. Separate functions were written to optimize the performance.

1. Generate the scale-free network (downloaded from the Internet)

```
function SFNet = SFNG(Nodes, mlinks, seed)

seed = full(seed);
pos = length(seed);

rand(state,sum(100*clock));

Net = zeros(Nodes, Nodes, single);
Net(1:pos,1:pos) = seed;
sumlinks = sum(sum(Net));

while pos < Nodes
    pos = pos + 1;
    linkage = 0;
    while linkage ~= mlinks
        rnode = ceil(rand * pos);
        deg = sum(Net(:,rnode)) * 2;
        rlink = rand * 1;
        if rlink < deg / sumlinks && Net(pos,rnode) ~= 1
        && Net(rnode,pos) ~= 1
            Net(pos,rnode) = 1;
```

```
                Net(rnode,pos) = 1;
                linkage = linkage + 1;
                sumlinks = sumlinks + 2;
            end
        end
    end

    SFNet = Net;
```

2. Calculate the average degree distribution parameter (downloaded from the Internet)

```
function average = SF_FD(Net)

connections = single(sum(Net));

frequency = single(zeros(1,length(Net)));

plotvariables = zeros(2,length(Net));
P = [];

for T = 1:length(Net)
    P(1,T) = T;
    if connections(1,T) ~= 0
        frequency(1,connections(1,T)) =
        frequency(1,connections(1,T)) + 1;
    end
end

for c = 1:length(frequency)
    if frequency(1,c) ~= 0
       [X,Y] = find(plotvariables == 0);
       plotvariables(1,min(Y)) = P(1,c);
       plotvariables(2,min(Y)) = frequency(1,c);
    end
end

for d = 1:length(plotvariables)
```

```
    if plotvariables(1,d) == 0 & plotvariables(2,d) == 0
        break
    end
end

x = plotvariables(1,1:d-1);
y = plotvariables(2,1:d-1);
z = single(sum(y));
s = double(0);

% Calculate the average number of connections with
% the probability of different degrees
for i = 1:d-1
    k = double(x(1,i) * y (1, i) / z);
    s = s + k;
end

average = s;
```

3. Cost-benefit model in patch strategy

```
function [g, f, s] = COST_BENEFIT_PATCH(k, b, n, p, i, x, y)

k0 = x; k1 = y;
K = k; beta = b;
S0 = p * (1- i);
I0 = p * i;

% Calculate the maximum of the overall benefit rate
f = @(gamma)S0 * K * beta / gamma + 2 * I0 + k0 * gamma
    - k1 * K * beta - 1;

[gamma, fval] = fminbnd(f, 0, 1);

s = 1 - S0 * K * beta / (gamma * 2) - I0 - k0 * gamma;

f = fval;
g = gamma;
```

4. Calculate the final population when the network goes into the stable state

```
function n = FIND_POP(S, l)

len = l;
s = zeros(len, 1);
s(1, 1) = -1;
len = len -1;

for a = 1:len
    b = a + 1;
    d = abs(S(a, 1) - S(b, 1));

    if d < 0.0001
        s(b, 1) = 1;
    end
end

num = 0;
flag = 0;
j = 0;
a = 1;
while a <= len
    if s(a, 1) == 0
        num = num + 1;
        if s(a+1,1) == 1 && num > 5
            j = a;
            a = len;
        end
    end
    if s(a, 1) == 1
        num = 0;
    end
    a = a + 1;
end

n = j;
```

5. Game theoretical model in patch strategy

```
function [T, S, I, R] = PATCH(k, b, g, n, p, i, t)

beta = b; gamma = g;
tmax = t;
I0 = i * p; S0 = p * (1 - i);
S = S0; I = I0; R = 0;

[T, pop] = ode45(@Diff_SIR, [0 tmax], [S I R], [],
             [beta gamma], [k]);

S = pop(:, 1); I = pop(:, 2); R = pop(:, 3);

function dPop = Diff_SIR(T, pop, parameter, k)
beta = parameter(1); gamma = parameter(2);
K = k;
S = pop(1); I = pop(2); R = pop(3);
dPop = zeros(3,1);
dPop(1) = -beta * S * I * K - gamma * S * I;
dPop(2) = beta * S * I * K;
dPop(3) = gamma * S * I;
```

6. Calculate four factors (which determines the best strategy) for each initial infection in patch strategy

```
function sum = INITIAL_INFECTION_PATCH(k, b, n, p, t, x, y)

i = 0.02;
r = 1;

while i < 1
    [g, f, s] = COST_BENEFIT_PATCH(k, b, n, p, i, x, y);
    [T, S, I, R] = PATCH(k, b, g, n, p, i, t);

    sum(r, 1) = i;
    sum(r, 2) = g;
    sum(r, 3) = s;
    sum(r, 4) = -f;
    n = FIND_POP(S, length(T));
    sum(r, 5) = S(n, 1);
    sum(r, 6) = T(n, 1);

    n = FIND_POP(I, length(T));
    sum(r, 7) = I(n, 1);
    sum(r, 8) = T(n, 1);
    sum(r, 9) = I(n, 1) - p *  i;

    n = FIND_POP(R, length(T));
    sum(r, 10) = R(n, 1);
    sum(r, 11) = T(n, 1);

    i = i + 0.02;
    r = r + 1;
end
```

Bibliography

[1] The power law degree distribution of a scale-free network. `http://www.macs.hw.ac.uk/~pdw/topology/ScaleFree.html`.

[2] S. Bortnik. Conficker by the numbers. Technical report, ESET Latin America, February 2010.

[3] W Gong C. C. Zou and D. Towsley. Code red worm propagation modeling and analysis. In *9th ACM Conference on Computer and Communication Security*, November 2002.

[4] W. Gong C. C. Zou and D. Towsley. Worm propagation modeling and analysis under dynamic quarantine defense. In *The First ACM Workshop on Rapid Malcode (WORM)*, 2003.

[5] V. Capasso and G. Serio. A generalization of the kermack-mckendrick deterministic epidemic model. *Math. Biosci.*, 42:43–61, May 1978.

[6] W. Gong Ch. Zou and D. Towsley. Code red worm propagation modeling and analysis. In *CCS'02 Proceedings of the 9th ACM conference on Computer and communications security*, pages 138–147, Washington, DC, USA, November 2002.

[7] C. Shannon D. Moore and J. Brown. Code-red: A case study on the spread and victims of an internet worm. In *Proc. 2nd ACM Internet Measurement Workshop*, pages 273–284. ACM Press, 2002.

[8] C. Gkantsidis et al. Planet scale software updates. In *Proc.of ACM SIGCOMM 2006*, Pisa, Italy, 2006.

[9] C. Wong et al. Dynamic quarantine of internet worms. In *Proc. IEEE Int"l Conf. Dependable Systems and Networks*, pages 73–82, 2004.

[10] D. Moore et al. Inside the slammer worm. *Security & Privacy, IEEE*, 1(4):33–39, August 2003.

[11] M. Mohd Saudi et al. Edowa worm classification. In S. I. Ao et al., editor, *Proceedings of the World Congress on Engineering*, volume I, London, U.K., July 2008.

[12] S. Roy et al. A survey of game theory as applied to network security. In *Proceedings of the 43rd Hawaii International Conference on System Sciences*, pages 1–10, Hawaii, January 2010.

[13] I. Hamadeh G. Kesidis and S. Jiwasurat. Coupled kermack-mckendrick models for randomly scanning and bandwidth-saturating internet worms. In *Proceedings of 3rd International Workshop on QoS in Multiservice IP Networks (QoS-IP)*, February 2005.

[14] C. Griffin. A note on the spread of worms in scale-free networks. *IEEE Trans. Syst. Man Cybern. Part B*, 36(1):198–202, February 2006.

[15] Jay G. Heiser. Understanding todays malware. *Information Security Technical Report*, 9(2):47–64, 2004.

[16] J. Jormakka and J. V. E. Molsa. Modelling information warfare as a game. *Journal of Information Warfare*, 4(2):12–25, 2005.

[17] A. Oprea M. van Dijk, A. Juels and R. L. Rivest. Flipit: The game of stealthy takeover. Technical report, RSA Laboratories, February 2012.

[18] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86(14):3200–3203, April 2001.

[19] D. Piscitello. Conficker summary and review. Technical report, ICANN, May 2010.

[20] V. Paxson S. Staniford and N. Weaver. How to own the internet in your spare time. In *Proc. 11th Security Symp. (SEC 02)*, pages 149–167. Usenix Assoc., 2002.

[21] G. Serazzi and S. Zanero. Computer virus propagation models. *Performance Tools and Applications to Networked Systems*, pages 26–50, Spring 2004.

[22] T. Tassier. Sir model of epidemics. *Epidemics and Development Policy, Fordham University NY*, 2005.

[23] Henk-Jan van der Molen. Math on malware. *ISACA JOURNAL*, 3:40–47, June 2011.

[24] M. Vojnovic and A. Ganesh. On the race of worms, alerts and patches. *Networking, IEEE/ACM*, 16(5):1066–1079, October 2008.

[25] M. Minoura Y. Hayashi and J. Matsukubo. Oscillatory epidemic prevalence in growing scale-free networks. *Phys. Rev.*, E 69(016112), 2004.