Thwarting Network Stealth Worms in Computer Networks through Biological Epidemiology

Kristopher Joseph Hall

Dissertation submitted to the Faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY
in
Electrical Engineering

Dr. Nathaniel Davis, IV, Co-Chairman
Dr. Lynn Abbott, Co-Chairman
Dr. Jung-Min Park
Dr. Thomas Hou
Dr. Charles Bostian
Dr. James Arthur

May 11, 2006
Blacksburg, VA

Keywords: Epidemiology, Bio-mathematical Modeling, Demographic Analysis, Network Stealth Worms, Network Security

Thwarting Network Stealth Worms in Computer Networks through Biological Epidemiology

Kristopher J. Hall

(ABSTRACT)

This research developed a system, Rx, to provide early identification and effective control of network stealth worms in digital networks through techniques based on biological epidemiology. Network stealth worms comprise a class of surreptitious, self-propagating code that spread over network connections by exploiting security vulnerabilities in hosts. Past outbreaks due to traditional worms subverted hundreds of thousands of machines. Network stealth worms exacerbate that threat by using clandestine methods to maintain a persistent presence in the network.

Biological epidemiology was shown to support the real-time detection, characterization, forecasting, and containment of network stealth worms. Epidemiology describes a scientific methodology in biology that seeks to understand, explain, and control disease. Bio-mathematical modeling led to the development of a mechanism for digital networks to identify worm infection behavior buried in anomaly data, to characterize a worm, and to forecast the temporal spread of a worm. Demographic analysis of the infected hosts revealed the subset of vulnerable machines within the population. The automated response of advanced quarantine used this information to control the spread of an identified worm by isolating both infected and vulnerable machines.

The novel contributions of this research included the identification of a network stealth worm at the network-level based on end-host reports while simultaneously characterizing and forecasting the spread of the worm. Additionally, this task offered the technique of advanced quarantine through demographic analysis of the population. This work resulted in a scalable, fault-tolerant strategy that dramatically enhanced the survival rate of network hosts under attack by a stealth worm. Moreover, this approach did not require new hardware, changes to existing protocols, or participation outside the implementing organization.

This research showed application to a wider range of challenges. The bio-mathematical models are extensible, allowing Rx to respond to variations on the self-propagating code presented here. The approach is applicable to other forms of malware beyond self-propagating code by interchanging the epidemic model with one more appropriate. Lastly, the strategy allowed anomaly detectors to be sensitive to lower reporting thresholds and a variety of often benign yet potentially useful events.

Dedication

This work is dedicated to my parents, Don and Carolyn Hall. Their love, support, and encouragement allowed me to overcome the many challenges I encountered during this journey and throughout life.

Acknowledgments

I wish to acknowledge the many people who contributed to this research. I foremost owe a great debt of thanks to my adviser, Dr. Nathaniel J. Davis, IV. His guidance and motivation allowed me to stay the course through this endeavor and enjoy and benefit from the pursuit of knowledge. I would like to thank Dr. Lynn Abbott for accepting the position and added responsibility of Co-Chairman, as well as the other members of my committee–Dr. Jung-Min Park, Dr. Thomas Hou, Dr. Charles Bostian, and Dr. James Arthur–for their direction of this research.

The IT Security Officer, Mr. Wayne Donald, and the Director of the IT Security Lab at Virginia Tech, Mr. Randy Marchany, deserve a sincere thank-you for their dedicated support of this research. The resources of this laboratory proved indispensable for the completion of this effort. Mr. Marchany provided significant technical expertise and guidance throughout the duration of this endeavor.

I must thank my colleague, Dr. Bob Morris, for his insightful comments and technical reviews through the course of this research. My mentors, Mr. Chuck Nemec, Dr. Doug Woods, and Mr. Phil Gollucci, also deserve my thanks for the direction they provided. I greatly appreciate the efforts of Mr. Ruiliang Chen who took time from his own busy schedule to construct the test-network used in this research. Thanks go to my mother, Mrs. Carolyn Hall, for diligently proof-reading the entire dissertation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computers have become an indispensable part of the modern world. These machines provided the driving force behind business, entertainment, communications, dissemination of information, and infrastructure control. Society embraced these machines as an indispensable resource to the point that intangible products such as communications, information processing, data storage and retrieval, network services, and connectivity carried value. Threats of many varieties emerged to exploit these assets [1].

Within the last five years, the subversion of millions of Internet hosts by attackers no longer remained a theory confined to hypothetical simulations. The surreptitious control of large numbers of systems allowed attackers to destroy data, alter information, distribute illicit material, and even disrupt the communications infrastructure itself. An emerging threat in cyberspace offered attackers an effective method to not only *gain* control but also to *maintain* control: network stealth worms.

This document presents an innovative and biologically-inspired cyber security system that identifies, characterizes, forecasts, and controls network stealth worms. This chapter

presents an overview of this research. Section 1.1 begins by providing the problem statement. A concise background of the problem and the motivation for pursuing a solution appears in Section 1.2. Next, Section 1.3 describes the research contributions followed by Section 1.4 that outlines the approach and validation methodology. This chapter concludes with a discussion of the results and conclusions in Section 1.5.

1.1 Problem Statement

This research developed a cyber security system, Rx, to mitigate the threat of network stealth worms. Rx defined a reporting and response framework to identify, characterize, forecast, and control stealth worms at the network-level based on end-host incidence and anomaly reports. The system depended on detectors to be sensitive, directly or indirectly, to a network stealth worm. Rx incorporated concepts from biological epidemiology–specifically, bio-mathematical modeling and demographic analysis–in the defense against digital disease. Epidemiology describes a scientific methodology in biology used to study the nature, prevalence, and causal factors of disease [2]. The system did not require new hardware, changes to existing protocols, or participation outside the implementing organization.

1.2 Background and Motivation

Network stealth worms provided attackers with a powerful and surreptitious network intrusion mechanism that posed a serious threat to the Internet. Stealth worms capitalized on

the success of classic worms by adding obfuscation techniques to resist detection and remain persistent in the network.

Worms were stand-alone, autonomous programs that spread by replicating themselves in remote hosts over network connections, penetrating systems through security vulnerabilities. Worms provided an automated and configurable delivery vehicle for the insertion of malicious payloads into Internet hosts on global scales. The Code Red worm marked the dawn of modern worms. In July 2001, the worm infected 359,000 hosts world-wide within 14 hours [3] [4]. The Slammer worm later spread at impressive rates by infecting 90% of the vulnerable hosts within 10 minutes [5].

Attackers proved they possessed the capability to control large numbers of machines and are now using these armies of subverted systems to destroy and steal data, alter information, establish illicit distribution points, harvest personal identities, and disrupt communications and services. This supported the changing motivation of attackers from media attention and notoriety to profit, spurring the development of surreptitious malcode [6].

Network stealth worms thus offered attackers a mechanism to maintain a persistent presence in the network. Researchers warned of malware advances including obfuscation techniques, new spreading strategies, control structures, and authentication and encryption mechanisms [5] [7]. The evolution of malicious code suggested the deadly merger of proven exploits and methods; to include network worms, DDoS tools, root and kernel kits, IRC Bots; and academic research in peer-to-peer networking and intelligent agents. Stealth worms presented a sobering reality to traditional network defenses.

The cyber world created an environment richly suited to sustaining such deadly and epidemic growth through its lack of diversity, insecure software, unpatched systems, open Internet communications model, and reactive defense mechanisms [8]. The elimination of a new worm was very tedious, requiring manual detection, signature generation, and containment. The manual, reactive approach for countering worms proved dangerously insufficient [9].

The Internet remained perpetually vulnerable to self-propagating programs. Network security research explored new methods to counter worms. Network-centric work largely addressed TCP-based, random scanning and fast spreading worms. To be effective, these mitigations consequently depended on a worm to generate a significant network imprint. Unfortunately, the approaches of these techniques specifically neglected slow spreading worms and stealth worms. Host-based approaches to worm detection included anomaly detection, heuristics, behavioral techniques, and artificial immune systems. However, these methods predominately based detection and decision metrics on a single host and lacked a reporting framework to share anomaly information and an approach to identify and respond to work behavior from aggregated reports.

1.3 Research Contributions

The primary contribution of this research was the early identification and effective control of network stealth worms in digital networks through concepts based on biological epidemiology. Previous use of epidemiology on computer networks largely used mathematical models to

study the propagation of classic and hypothetical viruses and worms. Research presented in this document significantly extended that work by using epidemiology as an approach to real-time network security.

Techniques used in biological epidemiology were shown to support the real-time detection, characterization, forecasting, and containment of network stealth worms. Bio-mathematical models served as a network-level mechanism to identify worm infection behavior buried in end-host anomaly data. The models further parameterized and forecasted the temporal spread of the worm early in the infection cycle. Additionally, demographic analysis was demonstrated to have a profound impact on the survival rate of computers under attack by a network worm. This technique used in epidemiology analyzed the infected hosts to determine and then quarantine the subset of vulnerable machines within the population, a response termed *advanced quarantine*.

This research also resulted in the definition of a scalable, fault-tolerant reporting and response framework for epidemic events. This strategy was valid on relatively small wireless networks through medium-sized corporate or campus networks. The framework showed its ability to enhance the survival rate of hosts under worm attack with Internet-scale participation. This approach did not require new hardware, changes to existing protocols, or participation outside the implementing organization.

Finally, this work showed application to a wider range of challenges. The bio-mathematical model is inherently extensible, allowing augmentations of this to respond to variations of self-propagating code. The overall approach was applicable to other forms of malware by

interchanging the epidemic model with one more appropriate. The strategy also allowed anomaly detectors to be sensitive to lower reporting thresholds and a variety of often benign yet potentially useful events.

1.4 Methodology Overview

The key elements of the approach were verified and then the resultant system, Rx, was evaluated through trials using simulation and implementation. The bio-mathematical model was constructed in the mathematics package Octave while the simulator was written in C++ [10]. Both were verified against the biological disease Hong Kong Flu and the digital worm Code Red I v2. The model and simulation were also qualitatively and quantitatively compared.

The bio-mathematical model and the demographic analysis module, both written in Octave, formed the foundation for the Rx system and were exercised through simulation. The simulated trials included corporate or campus-sized networks of 20,000 machines. Input data to Rx consisted of either incidence-detection of pure worm infection alerts or anomaly-based reports that included both worm spread and false-alarm anomalies.

A custom worm validated Rx on a test-network. The network stealth worm was created with client-server Java software installed on each machine in the test network. Five physical machines composed the test-network. Using VMware, each system executed an additional machine image thus yielding a network of 10 systems [11]. The Java worm emulated the behavior of a network stealth worm which specifically (i) provided complete control over

the parameters of the worm, (ii) abstracted the exploited vulnerability of the machines, (iii) emulated the presence of end-host detectors with test-defined sensitivity or detection probability to the worm, and (iv) provided ground-truth data for the evaluation of Rx.

The efficacy of Rx was evaluated based on the ability of the system to (i) accurately identify a worm from end-host reports, (ii) detect a worm early in its propagation cycle, (iii) correctly parameterize and forecast worm spread, and (iv) enhance the average survival rate of computers in the network. The bio-mathematical model was expected to identify worm infection behavior from end-host anomaly reports. The model also held the responsibility of parameterizing and forecasting worm spread. Demographic analysis was charged with containing the spread of network stealth worms by computing and quarantining the subset of vulnerable hosts within the population.

1.5 Summary

This research contributed a method to identify network stealth worms at the network-level based on anomaly reports generated at the host-level. The approach also characterized and controlled this form of self-propagating code. This work, implemented in a system named Rx, incorporated and adapted concepts from biological epidemiology. The results demonstrated that epidemiology can not only study digital worm propagation but also serve as an effective mechanism in the real-time defense of computer networks.

Simulated and live trials showed that a bio-mathematical model from epidemiology possessed the capacity to identify worm behavior from both pure host-based incidence alerts

and from similar data polluted by false-alarm anomaly reports. The same model was able to parameterize and forecast the spread of the worm.

Demographic analysis profoundly demonstrated its ability to contain the spread of a network stealth worm. This technique identified similarities among infected hosts to determine and quarantine the subset of machines vulnerable to a spreading worm.

The combined Rx system defined a scalable and fault-tolerant framework for reporting and responding to epidemic events. Simulations showed that medium-sized networks of 20,000 machines common to a corporation or campus to Internet-scale participation benefited from the deployment of Rx. Rx improved the survival rate of hosts within the network under attack by a network stealth worm. The system also caused the worm to experience a temporal lag in its effective spreading rate.

The following chapter defines network stealth worms and reviews traditional and novel approaches to mitigating the threat of this surreptitious, self-propagating code. The design of the epidemiological security modules and the composite system, Rx, receive attention in Chapter 3. Next, Chapter 4 verifies the bio-mathematical model incorporated by Rx as well as the simulator and implemented Java-based worm both used to exercise the system. Analysis of Rx, presented by Chapter 5, demonstrates the efficacy of the approach and the system. Lastly, Chapter 6 concludes the document and, in doing so, offers final thoughts on extending this research.

Chapter 2

Background

This chapter introduces the background required for this research. It provides an understanding of traditional network worms and their stealthed counterparts. The chapter then presents a review of the literature on classic and emerging methods to avert the worm menace and a discussion on biological concepts this research will apply to countering network stealth worms.

Section 2.1 establishes the theory and presents the history of network worms. This chapter continues with Section 2.2 which provides a description of the unique attributes of stealth worms and the evolutionary path from which they continue to develop. Approaches and specific work applied to thwarting network worms appear in Section 2.3. Next, Section 2.4 describes bio-mathematical models and demographic analysis techniques used in biological epidemiology to control the spread of disease in living populations. This is followed by Section 2.5 which discusses the application of these biological principles to digital networks for the purpose of this research. This chapter concludes with a summary in Section 2.6.

2.1 Network Worms

This section discusses the traditional network worm. It presents a concise history, several examples, and the basic theory behind this form of self-propagating code.

2.1.1 Worm History

The forerunners of network worms began as distributed computing research conducted at Xerox's Palo Alto Research Center in the late 1970's [12]. John Shoch and Jon Hupp explained their program as a computation that resided on multiple machines and later coined the term "worm" to describe their creation, borrowing the term from John Brunner's 1972 novel entitled *The Shockwave Rider* [13]. Researchers regard the worm created by Shoch and Hupp as an early realization of mobile agents.

The first true worm observed in the modern Internet, the Morris Worm written by Cornell University student Robert Morris, Jr., emerged on the second day of November of 1988 [14]. Interestingly, the worm did not purposely pursue any malicious activity yet managed to cause $10 to $100 million in damage on the young Internet of 60,000 computers [14] [15]. Though not purposely harmful, the Morris Worm could infect the same computer multiple times, stealing resources from the machine until it became unusable [14].

2.1.2 Modern Worms

Teachings from the 1988 Morris Worm quickly faded. Internet security in the years following the Morris Worm addressed the threats posed by viruses and hackers and ignored the dangers

of self-propagating code. The Internet community received a painful reminder in the summer of 2001 with the emergence of the Code Red worm that compromised 359,000 machines in 14 hours [3] [4].

Subsequent worms demonstrated increasingly unique, sophisticated, and virulent techniques. Code Red II utilized an effective localized scanning strategy and carried a payload that established a backdoor [5]. The multi-exploit worm Nimda passed through many firewalls and other defenses by using five separate mechanisms by which to spread and propagate [5]. Slapper controlled infected machines by establishing a Peer-to-Peer (p2p) network [5]. Slammer, sometimes called Sapphire, spread at unbelievable rates by infecting 90% of the vulnerable hosts within 10 minutes, resulting in a population doubling time of 8.5 seconds [5].

The Leaves worm was arguably one of the first stealth worms. It infected machines stricken with the SubSeven Trojan, assumed control of the machine and the Trojan, and pointed the computer to an IRC channel to await commands [16]. The Leaves worm recruited an army of $20,000$ machines in the summer of 2001 [16]. Leaves proved the validity of a network stealth worm and showed the potential danger involved with the merger of self-propagating code and DDoS tools.

2.1.3 Worm Theory

This section describes the constituent elements of a network worm. A worm, at a minimum, consists of a spreading mechanism and an exploit. Many worms carry a malicious payload and

may implement other functionality such as obfuscation techniques and learning abilities. The next section discusses network stealth worms and expounds on other capabilities available to both traditional and stealthed worms.

Body

Network worms were stand-alone, autonomous programs. Unlike legitimate software, worms exploited security flaws to gain unauthorized access in order to replicate themselves on susceptible hosts via network connections. Once inside a host, a worm executed its malicious payload, if any. Worms, in effect, implemented an autonomous intrusion agent [17].

The body of a worm commonly existed as a single piece of code which often resided in the file system or memory of a host. However, worm code could have divided itself into multiple segments on a single host or set of remote hosts. As an example, the combined exploit and propagation code segments could have infected a susceptible host and then copied the remainder of the worm body from the launching point.

Several elements composed the body of a worm. The body contained a spreading mechanism and an exploit. The spreading mechanism transported a worm across a medium and the exploit penetrated the machine at the end of that medium. The worm often contained a malicious payload that might corrupt host data, establish a backdoor, or conduct a DDoS attack. The worm could have implemented other capabilities such as obfuscation techniques and learning abilities.

Spreading Mechanism

The spreading mechanism selected and transported the worm over a medium to the next target. The spreading mechanism acted autonomously and did not necessarily require any outside action for it to operate. Worms utilized a variety of techniques to propagate through network connections and may have contained one or more spreading mechanisms. The worm propagation engine was often characterized by its spreading pattern in both space and time.

A worm exhibited either a topological or overlay spreading pattern. A topologically spreading worm propagated based on physical network topology, often tied to the IP addressing scheme. It commonly chose network addresses to probe and potentially infect in a sequential or random manner. An overlay spreading worm propagated based on logical connections, relationships, or knowledge that may already exist in the network, application-layer data, or user data. This worm, in effect, spread through a logical network. The worm propagated through existing p2p connections, in open network shares, by taking advantage of trust relationships, by bulk e-mailing itself to those in the address book of the compromised machine, through instant messenger contacts, and so on. Such a worm may have also used a pre-compiled target list for increased speed [7]. A worm could have implemented a hybrid of these two scanning strategies.

The spreading mechanism could have run at varying speeds. For simplicity, these were regarded as either fast or slow. A fast spreading worm typically infected a host and immediately tried to probe other machines at the quickest available rate, often consuming all available computer and bandwidth resources. A slow spreading worm, by contrast, lay

dormant in a host for a set period of time or until a certain trigger occurred at which point it activated and probed a small number of machines, often such that its actions went unobserved. A hybrid spreading rate strategy could have forced the worm to spread as quickly as possible to distribute and establish itself in the Internet and then switch to a less aggressive and more subtle slow spreading method.

Exploit

A worm penetrated a machine through an exploit. The machine must have first been susceptible to the exploit–the vulnerability–used by the worm in order for it to gain access to the system. Multi-exploit worms implemented two or more exploits.

Worms abused a variety of security flaws in order to compromise a machine. Buffer overflows proved to be a common method of system ingress used by a worm. For this exploit, the worm transmitted a copy of itself plus benign filler data to a program with a known overflow vulnerability [18]. The size of the worm plus the filler data was greater than the size of the receiving buffer. The receiving buffer failed to properly check the input size and, as a result, the error caused program control to transfer to the worm. Worms also took advantage of vulnerable software, system services, established trust relationships, weak passwords, and open network shares.

Payload

Once inside a host, a worm executed its malicious payload, if any, at its discretion. The malicious payload might have attacked the user by stealing personal information, attacked

the machine by corrupting data or denying services, attacked the network infrastructure by consuming the available network bandwidth, or served the hacker by establishing an illicit content distribution point. In some cases, the mere presence of the worm itself constituted an attack as with the case of Slammer. This worm spread so successfully that its spreading mechanism effectively conducted a DDoS attack on the Internet [3]. The payload could have been viewed as a module in the worm that could have been replaced with different functions as desired.

2.2 Network Stealth Worms

A network stealth worm, a dangerous extension of a network worm discussed in Section 2.1, obfuscated its existence both in the host and on the network to avoid detection. This implied that the worm employed measures to hide its body on the host machine, concealed its communications on the network, and possibly masked the effects of its malicious payload. The emergence of such insidious malware stems from the financial motivations of attackers.

2.2.1 Motivation

Past worm outbreaks proved that attackers possessed the capability to control large numbers of machines. Attackers used advanced forms of this malware to destroy and steal data, alter information, establish illicit distribution points, gather personal identities, and disrupt communications and services. This supported the changing motivation of attackers from media attention and notoriety to profit, spurring the development of surreptitious malcode

[6]. Key logging software quickly emerged as a prevalent and highly virulent cyber threat [19]. Over half of the viruses, worms, and other malicious code tracked by Symantec clandestinely harvested personal user data rather than harming the computer itself [19]. Nearly 10 million households owned a computer infected with key logging software [19], which placed a significant amount of personal information at risk of being compromised by attackers.

2.2.2 Surreptitious Techniques

A network stealth worm avoided detection by concealing its code and behavior both at the host and in the network. The code comprising the worm could have been compressed and encrypted. This obscured the purpose of the code and complicated its disassembly, thus delaying a targeted response to the worm. A worm could have used separate software such as root and kernel kits, discussed below, to subvert the system into concealing the presence of the worm from security tools and users.

A stealth worm also moderated its activities to thwart identification by network and host-based malware sensors as well as concern or discovery by the user. A stealth worm rate-limited its probing attempts to infect new hosts as well as the actions of payloads carried by the worm to include, for example, data harvesting, key logging, screen capture, and content distribution software. Stealth worms could have cooperated to achieve goals, thereby reducing the amount of work and thus exposure to detection required by a single worm instance. Inter-worm communications could have applied functions proven in DDoS tools, which receive attention below, to authenticate worm network members and messages,

encrypt traffic, and transfer information in covert channels.

2.2.3 Evolution to Network Stealth Worms

The plague of stealth worms emerged from the combination of classic network worms with DDoS tools, root and kernel kits, and IRC Bots as well as concepts from network research to include peer-to-peer communications and intelligent agents. Researchers warned of malware advances such as obfuscation techniques, new spreading strategies, control structures, and authentication and encryption mechanisms [5] [7]. Other work provided detailed descriptions of potential developments for virulent and effective malicious code which included learning capabilities, environmental awareness, cooperation, and autonomy [20] [17].

Network worms, DDoS tools, root and kernel kits, and IRC Bots represented proven tools and techniques for penetrating networks and systems, conducting attacks, and controlling large-scale overlay networks. Peer-to-Peer networking and intelligent agents stood to contribute unique methods for malcode to communicate, cooperate, adapt, and, ultimately, achieve virulent goals.

Classic Network Worms

Classic network worms implemented a relentless autonomous intrusion agent that exhibited Internet-scale, epidemic growth. Network worms provided the perfect delivery vehicle for automatically moving a malicious payload into a host. Classic network worms formed the foundation for network stealth worms.

DDoS Tools

DDoS tools stood to contribute significantly to the creation of stealth worms [21]. These tools showed impressive sophistication in their command and control capabilities over large networks, thus providing the attacker with a means of harnessing the power of vast numbers of compromised machines. DDoS tools also implemented effective obfuscation techniques.

Root and Kernel Kits

Root and kernel kits concealed the presence of network worms and other malware from both the system and the user. Root kits altered user-level binaries such that these normally trusted applications hid the exploit and provided false information to other programs and operators attempting to uncover the malicious code. Kernel kits protected the exploit through direct modification of the kernel by intercepting and re-directing system calls.

IRC Bots

Internet Relay Chat (IRC) Bots offered a simple yet formidable method to control large numbers of compromised machines [22]. IRC Bots were subverted hosts that connected to an IRC channel to await attack instructions from the controlling hacker. This provided a simple, effective, and largely anonymous method of commanding large numbers of compromised machines. IRC Bots also allowed the remote upgrade of compromised machines with new vulnerabilities and payloads, thus remaining ahead of security efforts [1].

IRC Bots were prevalent on the Internet. Symantec discovered a bot network of 400,000

nodes; a network administrator later detected another bot network composed of 1 to 2 million unique IP addresses [21]. Bots performed a variety of malicious functions like identity theft, key logging, traffic sniffing, spamming, and DDoS attacks [22].

Peer-to-Peer

Peer-to-peer (p2p) played a role in the management of large-scale, covert overlay networks formed by malware. The Slapper worm established a p2p network through infected machines [5]. Curious Yellow, the first coordinated worm design, used AChord, which was similar to peer-to-peer protocols, as the foundation for its communications [23]. The distinct advantages of a p2p framework included decentralized routing, communications, and control resulting in scalability and fault-tolerance.

Intelligent Agents

Intelligent agent concepts provided malware with enhanced effectiveness, stealth, and communications. Intelligent agents referred to self-contained computer programs that represented the user [24]. These principles suggested methods to create more flexible malcode that could have inferred awareness of its environment and adapt to changes to solve problems. Such malcode also could have cooperated in social groups to achieve complex goals. This intelligence could have been accomplished through emergent behavior and swarm intelligence as demonstrated with an Intrusion Detection System (IDS) built on these concepts [25] [26].

Intelligence could have conferred advanced stealth capabilities in malware not yet seen

through typical obfuscation techniques alone. Such worms could weigh the risk of detection against the reward of achieving a goal by executing a task. Malware could cooperatively divide tasks that signal an infection among many nodes, thereby forcing security mechanisms to correlate data over many nodes and large time windows. Intelligent malware can enable free-flowing information between all nodes, thus facilitating cooperation, while making the covert overlay network scalable, distributed, and fault-tolerant.

2.3 Thwarting Network Worms

This section discusses the traditional methods and emerging research techniques to battle network worms. These approaches fall into five categories consisting of prevention, detection, characterization, containment, and eradication and recovery. These schemes often meld into multiple categories but receive attention within the most applicable area.

2.3.1 Prevention

Prevention suppresses the availability of the exploit used by a worm, thereby impeding or blocking a worm from initially spreading.

The Root of the Problem

Network worms and the vast assortment of malcode existed because of user-interface, system design, and implementation flaws in hardware, protocols, software, and operating systems. Addressing the vulnerabilities in this paradigm stood as the ultimate form of prevention as

it addressed the root of the worm problem. These vulnerabilities continued because there was little to no business reason to expend development time and money on securing and testing products. This fueled the emergence of an entire security industry that included anti-virus, firewall, spyware removal, and intrusion detection tools and systems. Moreover, there was neither regulation nor significant end-user recourse for damage resulting from product insecurity. Thus, fundamental security flaws–some known for more than 25 years–remained, and security professionals and researchers continued to develop and deploy defense mechanisms based on an insecure computing foundation.

Traditional Mitigations

Cyber security was, truly, a proverbial fortress built upon the sand. Many researchers maintained that security using insecure systems is not possible [27]. Despite this, security professionals achieved reasonable, albeit imperfect, security by layering defense mechanisms. Network filtering devices acted to purge malicious content from the network before it reached the intended targets. Network-based IDSs and other devices logged and understood the network in order to aid in identifying and thwarting an attack. Users and administrators protected individual machines from contracting and harboring network worms by closing unnecessary services, keeping the system patched and up-to-date, using strong passwords, running a virus scanner, and enabling a host-based firewall.

Novel Approaches

Network worms, much like other malicious exploits, often exploited the most common vulnerability such that the worm would have enjoyed widespread success. The uniformity in computer networks ensured that a single vulnerability resulted in a vast number of susceptible machines [28] [29] [30]. A system should have avoided any unnecessary consistency which included varying the location of code execution, the order in which instructions were executed, and access to external routines [28].

Researchers demonstrated the blocking of a buffer overflow attack by randomizing the amount of memory allocated on a stack frame [28]. Stack monitoring, implemented by products like ProPolice and StackGaurd, contributed to the prevention of buffer overflow-based attacks, however, the approach incurred a system resource penalty and may prevented legitimate programs from executing by incorrectly detecting a buffer overflow [31]. Other work proposed an architecture to identify a piece of infected software and automatically generate a patch for the vulnerable code [31]. This technique required that the patch-production machine received a copy of the worm early in the infection cycle and implied compromised machines remain infected for the time needed to generate a patch. Furthermore, the patch applied to a specific, well-specified machine and could not be used generically.

2.3.2 Detection

Detection identifies the presence of a spreading worm in order to alert security professionals and to direct focused countermeasures.

Traditional Mitigations

Traditional detection and response to self-propagating code largely depended upon the use of signatures to accurately identify malcode and to avoid falsely detecting legitimate software or data. Companies producing signature-based security products required copies of worm code often either captured by their own Honeypot networks, discussed below, or submitted by astute system administrators from other organizations. Signature generation used automated tools that typically required human intervention and finalization. The challenge to this approach was capturing a worm early in the infection cycle, centralization of patch creation and availability, and timely distribution of the patches. Detection of novel worms on end-hosts or network products using heuristics or similar methods proved rather ineffective.

Honeypots and network telescopes detected worms and other malicious activities through the use of non-production, monitored machines for the former and unused IP addresses for the latter [32] [33]. Incoming traffic was immediately considered suspicious since the these end-points were otherwise unused. Honeypots and network telescopes, however, primarily proved effective against random-scanning, topologically-spreading worms and required a worm to reach the end-point early in the infection cycle. Signature generation from a captured worm typically involved human intervention.

IDSs and anti-virus scanners used signatures to detect known worms and heuristics to detect novel worms [34] [35]. The same held true for many distributed intrusion detection systems (DIDs) [36] [37]. These security tools enjoyed high success with signature-based detection of worms but struggled to identify a novel worm. Many IDSs used centralized

and intelligent network-based analysis and control with network and host-based sensors, resulting in a single-point of failure. Furthermore, a worm on a compromised machine could have subverted the virus scanner, thus providing a false sense of security.

Worms, or the effect of a worm infection, were sometimes detected by looking for changes between known versions of a file, hash values of information, or complete install image. Logging and auditing helped identify the presence of the worm, however this required tedious, manual human analysis and did not scale well to large networks. The security tool Tripwire automated this approach [38]. Tripwire, however, required the maintenance and inspection of old configurations against new configurations, relied upon the malcode to make a change to a monitored entity, and entrusted the system to correctly report its configuration and possibly to perform the comparison.

Novel Approaches

Researchers proposed and evaluated a network-level worm detection scheme by recording and processing the number of failed worm infection attempts over time. Large occurrences of Internet Control Message Protocol (ICMP) error messages indicated these failed worm scans [39]. This method, however, did not sufficiently detect those worms that spread slowly, use pre-compiled hitlists, or left little imprint in the network.

Host-based behavioral techniques offered a promising method by which to detect network worms by identifying suspicious or malicious patterns of system calls [8] [40]. Other research identified abnormal behavior by modeling the detection technique after the human

immune system [41]. Researchers developed a worm detection tool for network worms by analyzing the data entering a network interface [42]. However, this method was limited to buffer overflow attacks on the network interface. Another technique used existing detectors to identify the exploit used by a worm at the end-host and propagated this information to other hosts in order to support the defense of other machines [43]. This approach, like the others, offered possible methods to identify and control worm behavior at the host but only when it was possible to exactly identify a worm at a single host. They neglected statistical information that arose from the overall population.

2.3.3 Characterization

Characterization analyzes a new worm in order to produce targeted countermeasures.

Traditional Mitigations

Characterization methods involved the detection of network worms with a signature that identified the worm itself or generic exploits. Signature-based detection proved effective against the specific malware for which it was developed but struggled in the defense against novel worms. Other techniques, instead, identified and blocked common vulnerabilities used by worms. The IDSs and Bro, among others, used both methods to characterize network worms and launch countermeasures [34] [35]. The Shield was a highly successful vulnerability guard that focused on host machines rather than the network [8]. However, such techniques were only effective against previously known vulnerabilities that received significant attention

from human operators.

Novel Approaches

Kephart and Arnold proposed and implemented one of the earliest and successful automated systems for signature extraction, which found commercial use by IBM [44]. Their system isolated an instance of a virus, encouraged it to spread within the device, and heuristically generated invariant signatures. This system proved highly effective but required a malicious pathogen to infect the controlled analysis machine early in the spreading cycle. Honeycomb extended this work by moving to a host-based approach [8]. Honeycomb correlated data seen in message exchanges and searched for the longest common string among them. Honeycomb, though, proved effective only when defending against a sizable and reasonably fast-spreading worm population.

Autograph and Earlybird dynamically generated signatures for novel exploits by observing network traffic [45] [46] [8]. These systems, developed independently, identified network worms and created signatures based on two principles of the behavior of network worms: content invariance and address dispersion [8]. Both systems successfully characterized many traditional network worms and deployed countermeasures, such as a Snort signature. However, these systems depended upon a sizable worm population and resultant network imprint in order to detect a worm.

2.3.4 Containment

Containment slows or stops a worm from spreading further.

Traditional Mitigations

Traditional approaches to worm containment typically involved the automatic response of IDSs or the manual deployment of countermeasures by human operators. IDSs, discussed in 2.3.2, detected and potentially deployed firewall and routing rules to block the spread of a worm but depended largely on signatures to trigger the response. Human-deployed containment countermeasures were typically delayed and often complemented or replaced automated responses. Such actions were sometimes used to deny connectivity to healthy machines or subnets, thereby preventing their infection while sacrificing availability.

Novel Approaches

Research contained worm spread by significantly slowing the average worm propagation rate at the network and hosts levels. A tar pit, like LeBrea, slowed the worm scanning engine by accepting moving incoming TCP connection requests from 'connected' to 'established' [47]. The worm believe it had reached a legitimate host and waited for further communications. This approach only addressed TCP-based, random-scanning worms and required significant participation from the Internet population in order to prove effective.

A non-representational approach to throttling a spreading worm identified suspicious activity but refrained from taking immutable actions [30]. A host-based anomaly detector

analyzed system calls, communications, and resource usage for suspicious behavior and restricted the offending process. If the perceived behavior of the suspicious process improved, the detector restored the resources or else it further reduced or even usurped resources from the process. As a result, this technique could have resulted in a higher false-positive rate with less adverse impact to the system or user. This approach, however, required near-universal deployment and may have only slowed a worm instead of stopping it completely.

2.3.5 Eradication and Recovery

Eradication eliminates the active worm, and recovery restores and immunizes the infected hosts.

Traditional Mitigations

Several methods availed themselves to rid infected hosts of network worms. Memory resident worms could have been removed from the infected system by simply cycling the power. Virus scanners and specialized removal tools identified and deleted those worms for which the tools possessed signatures. Re-formatting the computer was often the only way to rid the machine of certain worms. This was the safest method to use as it guaranteed the removal of the worm and any malicious payload it deposited.

Novel Approaches

Researchers extended the idea of propagating dynamically generated worm signatures among machines to a worm predator that actively hunted and consumed spreading worms [48]. A

different researcher realized two worms to destroy Code Red but did not release them due to ethical and legal concerns [49]. The Welchia worm in 2003 attacked the Blaster worm by infecting machines compromised by Blaster, removing the Blaster worm, and patching the system. Some versions of the Welchia worm added a backdoor [50]. Ethical and legal concerns, stability of the network, and control of the predator worm were challenges to this approach to worm eradication.

2.4 Biological Epidemiology

Epidemiology literally means the study of that which befalls mankind [2]. Epidemiology describes a scientific methodology in biology used to study the nature, prevalence, and causal factors of disease [2]. This science provides a method for understanding and responding to a disease as it spreads through a population. Epidemiology uses mathematical models to quantify, characterize, and predict the spread and impact of disease. Demographic analysis determines the relationship between the disease and the population. The role of the epidemiologist is often to damage or destroy this relationship, thereby preserving the population from infection. The ultimate goal of epidemiology is to curb the spread of a disease and prevent its further recurrence.

2.4.1 Bio-mathematical Modeling

Epidemiology uses mathematical models to understand the past events contributing to a disease and make qualitative and quantitative forecasts for the disease in the population.

This facilitates measures to mitigate the spread of disease. Many types of models exist to describe the spread of different diseases.

Epidemiological models often fall into three categories: box or compartmental, continuous space, and network models [51]. Compartmental models divide the population into distinct groups and consider the homogeneous spread of a disease between each set. Compartmental models readily apply to human and animal populations. Continuous space models describe the habitat of a population as a surface and provide qualitative insight into the spatial spread of a disease. Network models form a hybrid of compartmental and continuous space models and contribute to establishing a more correct human contact structure. They provide insight into the spatial spread of disease in human epidemics, but are considerably the most computationally expensive and analytically complex of the three models. The majority of the models allow the extension of groups in order to investigate the relationships between population, disease, environment, and time [52] [53]. Generalized, deterministic, compartmental models sufficiently describe many events. The discussion below develops a deterministic, compartmental Susceptible-Infected-Recovered (SIR) model.

The SIR model, as the name implies, defines three compartments within the population: susceptible, infected, and recovered [53]. The susceptible individuals, S, are those who may contract the disease. Infected individuals, I, constitute those who carry the disease and transfer it to others. The recovered individuals, R, include those who overcame the disease and now have immunity and those who succumbed to the disease. The only allowable transitions consist of S to I and I to R. Figure 2.1 shows this relationship.

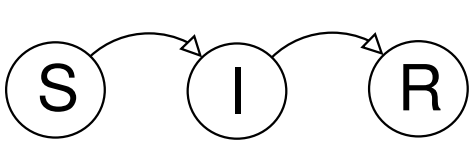

Figure 2.1: SIR Diagram

The SIR model assumes the population, P, remains constant through the course of the analysis. Equation 2.1 defines the quiescent relationship among the four groups [53]. At any given instant in time, Equation 2.2 holds true [53].

$$P = S + I + R \tag{2.1}$$

$$P(t) = S(t) + I(t) + R(t) \tag{2.2}$$

Differential equations provide the dynamics between these compartments shown in Equations 2.3 to 2.5 [53].

$$\frac{dS}{dt} = f_1(S, I, R) \tag{2.3}$$

$$\frac{dI}{dt} = f_2(S, I, R) \tag{2.4}$$

$$\frac{dR}{dt} = f_3(S, I, R) \tag{2.5}$$

The definition of these functions establishes the rate of transfer of individuals between compartments. A law of mass action describes the infection process, defined as the transfer

from S to I, and is controlled by the positive constant α [53]. The dependence upon both S and I is reasonable given that the number of infected individuals in the next time step depends upon the interactions between both the infected and susceptible individuals in the current time step. Pure exponential decay governs the recovery or death process, shown as the transfer from I to R as set by the positive constant λ. Equations 2.6 to 2.8 establish the basic equations for the SIR model.

$$\frac{dS}{dt} = -\alpha IS \tag{2.6}$$

$$\frac{dI}{dt} = \alpha IS - \lambda I \tag{2.7}$$

$$\frac{dR}{dt} = \lambda I \tag{2.8}$$

The transmission coefficient, also known as transmission probability or infection rate, α, defines the probability that a pathogen will successfully transfer from an infected individual to an uninfected yet susceptible individual [52] [53]. Characteristics of the infective source, the disease, the susceptible host, and the type of contact affect the transmission coefficient [54]. This value further depends on the average number of sufficient contacts and the average probability of transmission per contact. Individuals recover or succumb at a rate λ which thus implies that the duration of the infectious period is $\frac{1}{\lambda}$ [54].

Table 2.1 enumerates the six parameters used in the SIR model. The parameters S, I, and R define the three distinct compartments while the value P represents the sum of

Table 2.1: Parameters in SIR Model

Parameter	**Name**	**Explanation**
P	Population	Total set of individuals under study
S	Susceptible	Individual who may contract the disease
I	Infected	Individual who carries the disease and may infect others
R	Recovered	Individual who is no longer infective due to recovery or death
α	Transmission coefficient	The rate or probability at which susceptible individuals become infected and therefore may infect others
λ	Recovery rate	The rate or probability at which an infected individual recovers or succumbs to a disease and is no longer infective

all individuals within the compartments. The inputs α and λ set the rates of infection and recovery for the underlying dynamics established by differential equations.

Figure 2.2 shows example output from an SIR model. The initial population was set at 10,000 with one infected individual and no recovered individuals. An infected member of the population made sufficient contact to transmit the disease to one vulnerable individual every two days. One trial in the graph assumed that members of the population did not recover from the disease once infected. The last trial included recovery, allowing an infected individual to transmit the disease for five days.

The number of infected individuals over time for the two different trials is given by Figure 2.2. The population without recovery showed a more drastic increase in the number infected over time versus that for the trial with recovery since infected individuals remained infected and continued to fuel the law of mass action dynamic that governed the infection of

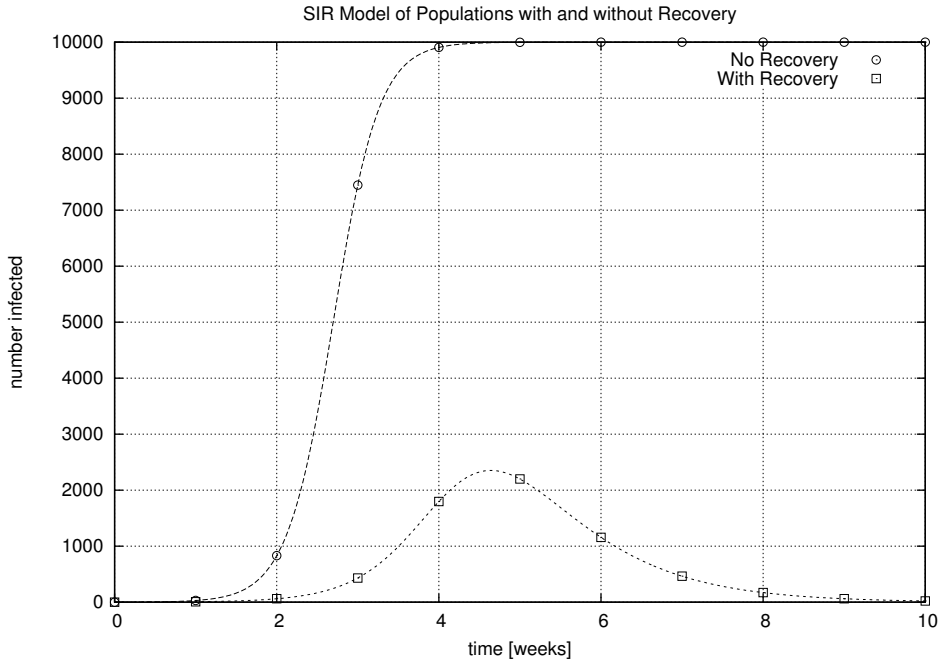

Figure 2.2: SIR Model of Populations with and without Recovery

susceptible individuals in the first trial. The scenario that included recovery still experienced a significant number of infected individuals, however the recovery process in this case curbed the spread of the disease before it saturated the population. The role of the epidemiologist is to identify disease, especially potentially epidemic disease, like the first trial, early in the infection cycle and issue a response such as quarantine to curb the spread of disease as with the second trial.

Other models quickly extend from the basic SIR model [52] [53]. An example is the SEIR model [53] which adds the transition "exposed" in order to incorporate a latent period for a disease into the spreading dynamics.

Epidemiology defines other important parameters beyond the infection and recovery

rates that contribute to understanding and controlling the spread of disease. The basic reproductive number, R_o, gives the average number of individuals directly infected by an infectious individual when the latter enters a completely susceptible population [54]. It ignores recursive cases. R_o is a dimensionless number such that an R_o of five indicates that each infective introduced into a population induces five new secondary cases. Equation 2.9 defines R_o for the SIR model [54].

$$R_o = \frac{\alpha}{\lambda} \tag{2.9}$$

The power of the R_o parameter lies with the capability to quickly assess the threat of a spreading disease [54]. An R_o equal to one means that each infective person infects exactly one new susceptible individual and the disease will remain pandemic in the population. An R_o less than one suggests the disease will naturally die on its own while a value greater than one says the disease will establish itself and become epidemic. As an epidemic progresses, R_o, will eventually fall below unity and the disease will disappear.

2.4.2 Demographic Analysis

Epidemiology employs demographic analysis of the population to identify and address the causal factors of a disease in order to arrest the spread of the latter. Demography is the study of the characteristics of a population to include size, growth, and vital statistics as well as any other defined feature [55]. Attributes divide the population into demographic groups based on a variety of traits such as age, gender, social status, and physical location.

These groups may also specify characteristics unique to a disease or situation such as defining groups of those who consumed potato salad versus those who did not when investigating a case of food poisoning at a picnic.

The qualitative side of epidemiology seeks to curb and prevent the spread of disease through a population by understanding and disrupting the relationships between population, disease, and environment with respect to time [56]. An example would be fighting the spread of malaria in humans commonly transmitted through mosquitoes. The disease-environment relationship can be damaged by spraying marshland with insecticide to kill mosquitoes and their offspring in their habitat, thus curbing the transmission of malaria to humans by mosquitoes. An understanding of the interactions between host, environment, and disease can suggest an effective strategy to curb the spread of disease.

Central to accuracy and success of epidemiological techniques is the availability of vital statistics on the population as well as numbers and characteristics concerning infected individuals. The document *International Classification of Disease, 9th edition, Clinical Modification* (ICD-9-CM) was designed for the purpose of classifying morbidity and mortality data for statistical uses [2]. The United States passed federal laws that require the reporting of 49 notifiable diseases [2]. The United States Center for Disease Control and Prevention (CDC) executes this legal mandate using a hierarchical reporting and response system such that reports from local health care providers propagate to local, regional, and, finally, national organizations [2].

2.5 Digital Epidemiology

Digital epidemiology applied biological epidemiology to the cyber world. Network administrators and researchers realized that the security posture of a single machine depended upon the security of the overall population which may have included the subnetwork, campus or corporate network, and even the Internet. Techniques from biological epidemiology offered methods to understand and counter security issues that threaten the health of the population.

2.5.1 Digital Bio-mathematical Modeling

Epidemiological modeling in digital networks dated back to the dawn of the computer virus in 1984. Fred Cohen established the fundamentals for understanding computer viruses and discussed elemental mechanisms of movement for the virus [57].

Years later, Kephart and White first applied formal epidemiological models in 1991 to hypothetical viruses [58]. They used a Susceptible-Infected-Susceptible (SIS) model in deterministic, stochastic, and simulated trials with the defined transitions between groups as susceptible to infected to susceptible. Using this, the researchers established the conditions that contributed to a potential viral epidemic.

The researchers then applied an SIR model to explore the spread of a virus in which users installed anti-virus software on an infected machine, ridding the machine of the virus and conferring it with immunity [59]. They sampled and graphed viral incidence rates which led them to a new concept: the kill signal. Once infected nodes identified a virus and cleaned

themselves, they issued a kill signal to neighboring nodes to take appropriate action. This explains the deviation of observed viral incidences from theory at the time of high media exposure covering an outbreak. Later work showed that network topology affects the spread of the worm and investigated the epidemic threshold parameter in virus spread [60].

This early research grounded the application of epidemiology to computer networks by using this field of biology to understand the spread of digital contagions. This technique provided a method to test and evaluate new anti-virus technologies, to determine the importance of vulnerable systems, and to estimate the cost of potential outbreaks [61].

The prevalence and threat of network worms sparked new interest in the area of cyber epidemiology, largely due to the overwhelming success of the Code Red worm in the summer of 2001. Researchers used a logistic curve equation as a means to model the historic spread of the Code Red worm using stored infection data [7]. They found such simple models provided reasonable initial insight into the characteristics and threat of network worms [7]. Researchers closely matched the epidemic curve of Code Red by applying a sigmoid curve to recorded data [39]. They reported that the sigmoid curve gave reasonable results and proved more efficient to compute over the SIR differential equations. Other work showed that techniques from plant epidemiology, mainly spatial autocorrelation, successfully identified worm propagation that exhibited dependence on physical geography or network topology [51].

Others maintained that basic models from biology were not suitable for describing worm spread in computer networks [62]. Researchers thus constructed a two-factor worm

model to accurately describe the spread of Code Red I v2 by incorporating aspects of dynamic countermeasures and self-congestion into the model [62]. Similar work resulted in an Analytic Active Worm Propagation Model (AAWP) to, again, more accurately model the spread of Code Red I v2 [63]. This work showed the AAWP model to more closely follow the spread of an actual worm than an epidemiological model [63]. Others used the SIR epidemiological model as an initial basis for simulations that studied the requirements for defense using dynamic quarantine [64].

Epidemiological models received further application beyond the direct study of known worm propagation. Researchers used these biological models to analyze the spread and impact of hypothetical worms [7]. Deterministic models from biology were also used to speed large-scale worm propagation simulations [65]. The epidemiological model emulated the spread of the worm at the Internet-level while the simulator computed the spread of a worm within the smaller subset network under study. The free and open-source network simulator *ns2* included a module to emulate the spread of a worm using the SIR epidemiological model [66].

Research established the validity of using biological models from epidemiology in digital networks. However, this work focused on studying the spread of historic worms from captured data and analyzing the propagation and impact of hypothetical, virulent worms. Bio-mathematical models are capable of providing further contribution to the network security field. These models possess the unrealized capability to parameterize, predict, and even identify worm spread in *real-time* based on infection and anomaly data. This can con-

tribute to early detection, understanding, response, and ultimately controlling the spread of self-propagating code.

2.5.2 Digital Demographic Analysis

Traditional approaches most similar to demographics were largely limited to static policies that controlled the procurement, deployment, and maintenance of hardware and software and the connectivity available to end-hosts. This was common to organizations and businesses. The purchase and deployment of information technology items could have been constrained to discrete categories of system hardware and software. End-host scripts or software or network-based scanners may have assessed the security posture of a machine. If the host failed to meet an established security policy, the machine could have been updated automatically or removed from the network until it met the security guidelines.

Demographic analysis performed in real-time for systems under attack by self-propagating code substantially contributed to the field of network security. The application of static policies on digital networks proved effective given significant deployment and rules that covered the vulnerabilities used by malicious code. Such traditional approaches failed when malcode exploited a novel vulnerability and, furthermore, the rules were very slow and manual to update. Demographic analysis showed the potential to determine the causal factors contributing to the spread of a worm by analyzing attributes of the infected population. This information was used to isolate infected and vulnerable machines, thereby curbing the spread of the worm.

2.6 Summary

This chapter provided an overview of the information and literature that formed the foundation for this research. Network worms implemented autonomous intrusion agents that propagated via network connections and penetrated computers by exploiting vulnerabilities. Network stealth worms employed surreptitious techniques to conceal signs of their presence, allowing them to spread and execute their malicious actions in relative obscurity. Security research to thwart worms made substantial gains, but largely addressed fast-scanning, topologically-spreading worms which specifically neglected stealth worms.

Biological epidemiology contributed to cyber security. Epidemiology describes a scientific methodology in biology used to study the nature, prevalence, and causal factors of disease [2]. Epidemiology received significant application to the digital world as a method to study the spread of hypothetical worms, model the spread of historic worms, and speed large-scale network simulations. This work neglected the *real-time* use of epidemiology on computer networks as a *proactive* approach to security against self-propagating code.

The next chapter discusses the design and evaluation metrics for a proposed solution to these challenges. Concepts from biological epidemiology–specifically, bio-mathematical modeling and demographic analysis–are adapted and applied to digital networks as a mechanism to identify, characterize, forecast, and control the spread of network stealth worms through end-host incidence and anomaly reports.

Chapter 3

System Design

This chapter presents the design of a system developed through this research, Rx, to address the threat of network stealth worms by applying concepts from biological epidemiology. The purpose of this effort, discussed in Section 1.3, was to develop a strategy to nullify the threat of this surreptitious, self-propagating code. The system adapted biological concepts–specifically, bio-mathematical modeling and demographic analysis–to characterize and control stealth worm propagation in digital networks.

This chapter begins with Section 3.1 that provides an overview for the approach and evaluation methods used in this research. Section 3.2 discusses the parameters that guided the design of Rx. Section 3.3 maps biological terminology and techniques into the digital problem space. The unit continues by developing the constituent modules of Rx, showing the integration of the modules into a system, and discussing the hierarchical reporting and response framework of Rx.

Next, Sections 3.4 and 3.5 cover the realization of Rx in simulation and then implementation on a test-network. Section 3.6 explains the construction of the simulator used to

exercise Rx followed by Section 3.7 that defines metrics for measuring the effectiveness of Rx. This chapter ends with a summary in Section 3.8.

3.1 Overview

Rx implemented a cyber security system that mitigated the threat of stealth worms through the real-time application of concepts from biological epidemiology. Simulation and test-network trials evaluated Rx.

3.1.1 Approach

Biological epidemiology motivated the development of Rx. Epidemiology describes a scientific methodology in biology used to study the nature, prevalence, and causal factors of disease with the goal of controlling disease spread [2]. This research mapped these biological concepts to computer networks and evaluated them through Rx.

Six modules implemented the functionality of Rx. The main operation of Rx and consequently the primary contribution of this research–bio-mathematical modeling and demographic analysis–were performed by two modules. These received the most attention in the subsequent discussion. The other four modules acted in support of the two primary modules. Rx also defined a hierarchical reporting and response framework for identifying and controlling epidemic events inspired by the United States Center for Disease Control and Prevention [67]. This gave Rx a scalable and fault-tolerant deployment mechanism. Rx operated on detection and anomaly data reported by existing malware sensors that were

sensitive to a network stealth worm.

Rx was implemented in the mathematics package Octave [10] for the simulations; a Java-based worm exercised Rx on a test-network for implementation trials. Octave was chosen to simulate Rx due to the ability of the former to quickly realize the system functionality and allowed expedient changes to the system configuration through a high-level scripting language. Java provided reasonable processing speed and efficient network programming for constructing a worm and recording the infection pattern of the malcode.

3.1.2 Evaluation

Rx was realized and evaluated as a cohesive system in simulation and through implementation on a test network. The simulator, written in C++, modeled the spread of network stealth worms as a time-series of probabilistic events divided over infinitesimally small time steps. The simulator recorded the number of susceptible, infected, and recovered individuals within the population at each time step. A custom simulator was written in place of using existing network simulation software, as the former provided the necessary worm propagation data without a learning curve or unnecessary processing overhead often associated with the latter. C++ was chosen as the language for the simulator for speed execution.

The implemented version of Rx was evaluated on a test-network against a Java-based program that emulated the spread of a network stealth worm. The Java worm consisted of software installed on each machine in the test network that possessed the ability to receive incoming probe attempts and, if successful, initiate its own probes. In this manner, the

software emulated the spread of a network worm. This software was used as an evaluation tool instead of actual self-propagating code because Rx did not detect individual worm instances on single systems. Instead, it operated at the network level on aggregated end-host anomaly reports, explicitly depending on detectors to be sensitive to worm instances. The usage of a worm emulator also provided complete control over the experiment and ground-truth results with which to evaluate Rx.

These trials measured the efficacy of Rx by evaluating the ability of the system to identify a worm, the confidence measure computed by Rx when identifying worm spread, the estimated and actual probing rates, the temporal-lag in worm propagation time caused by Rx, and the percentage of the population infected with and without Rx. Further analysis also investigated the scalability and effects of partial deployment.

3.2 Design Parameters

Two parameters guided the design and evaluation of Rx. The system assumed that existing malware sensors were sensitive to a network stealth worm in order to provide Rx with host-oriented detection or anomaly data to support network-level inference and control of stealth worms. Rx was targeted for deployment on medium to large-sized networks consisting of around 10,000 machines common to campuses, businesses, and Internet Service Providers (ISPs).

3.2.1 Support from Malware Sensors

Rx depended on malware sensors that were sensitive to the effects of a network stealth worm. The system aggregated and processed the data submitted by these existing sensors. Rx specifically did **not** require malware sensors that formally detected malicious code. Instead, Rx enhanced the effectiveness of these security tools by accepting their detection and anomaly data based on individual machines and providing network-level identification, characterization, and control of malcode.

Rx processed two types of information from malware sensors: incidence-based and symptom-based data. Incidence-based detection occurred when network or host security tools positively detected the presence of malicious code but did not possess the ability to identify, characterize, disable, or remove the malware or to repair the exploit used by the surreptitious code. Cases where signatures existed but are not yet fully deployed, machines remained unpatched and vulnerable to the exploit, the malcode re-used pieces of previous malicious code, or worms that triggered a serious event in anomaly detectors such as the escalation of privileges made this a valid and important trial. Incidents of infection could have been gathered by installed software such as a virus scanner, configuration change-monitoring software like Tripwire, or by passive and active network tools like Snort and Nessus [38] [34] [68].

Symptom-based detection was fundamentally equivalent to traditional anomaly detection. Reported host anomalies, like symptoms of a cold in the biological paradigm, only indicated a deviation from an established normal profile and may not have immediately

suggested the individual machine was compromised. Rx aggregated and processed anomaly data from individual machines in order to identify, characterize, and control *novel* worms at the population-level. This step proved critical for thwarting new worms and to ultimately generate signatures for host and network-based security tools such as anti-virus and network filtering products. The reporting of anomalies was reasonable given the research in artificial immune systems and behavioral detection [8] [40] [41]. Other research specifically targeted network worms at the host-level [42] [43].

3.2.2 Target Scenario

Rx was intended for use at campus and corporate-sized networks or by Internet Service Providers (ISPs). These environments ensured the availability of host and network-based malware sensors capable of identifying a diverse set of detection and anomaly events. Moreover, the system was not designed to prevent the infection of a single machine, rather to ensure the survival of the overall population. A sizable population ensured that Rx received sufficient data on which to operate while simultaneously improving the survival rate of the population under attack. Experiments gauged the effectiveness of Rx on networks with significantly less or more machines, but the primary focus fell on campus and corporate-sized networks.

The defined framework of the system called for Rx nodes to be positioned at main subnets or at core routers within the network of an organization. Additionally, one Rx node was located at the edge or core router and was designated the Rx root node. Rx nodes

communicated infection and anomaly data up the hierarchy, which allowed higher levels to produce a broader view of network health and to affect an expedient, localized response as appropriate. Rx root nodes of different organizations exchanged infection and anomaly data to facilitate early-warning and prevention across organizational boundaries.

The downsides to this deployment strategy were the centralization of reporting, processing, and intelligence as well as trust between Rx root nodes of different organizations. These machines represented points of failure within the network. The system did implement an amount of redundancy in that Rx nodes continue to operate within their subnet with a gradual degradation in effectiveness as other Rx nodes or even the Rx root node was eliminated. Additionally, manipulated or falsified data shared between organizations may weaken the effectiveness of Rx for a given entity.

This deployment scenario was sufficient for this research as the system contributed to the goals of identifying, characterizing, and controlling actively spreading network stealth worms. The hierarchical reporting structure created central repositories for information exchange and processing and resulted in a minimum of required hardware and management. The issue of trusting data sets from different organizations was a common challenge among Distributed Intrusion Detection Systems (DIDS) that was not considered in this research. This effort assumed data sets were only accepted from trusted, verified outside organizations and were transferred through a secure medium that authenticated both the sender and receiver such as SSL with certificates. Data sets not meeting this criteria could still have been used to signal potential alerts. Alternate implementations, deployment strategies, distribu-

tion, and redundancy could have been considered using the foundations established by this research for identifying, characterizing, forecasting, and controlling network stealth worms through an biological approach.

3.3 Rx Design

Rx implemented a scalable and voluntary security system, grounded in biological concepts, to thwart network stealth worms. Others discussed similar mechanisms for defense against self-propagating code, including Staniford et al. who envisioned a world-wide Cyber Center for Disease Control [7]. Rx bears similarity to this yet takes a more grass-roots approach in order to overcome limiting problems with other strategies such as ownership, scalability, graceful degradation, and effectiveness under partial deployment.

This effort focused on the piece of Rx that performed the functionality of identification, characterization, and forecasting of worm propagation: the Bio-mathematical Model Module (BMM). The Demographic Analysis Module (DAM) also received significant attention as it shouldered the responsibility for curbing worm spread by determining and damaging the disease-host relationship through a directed, dynamic quarantine response. The four remaining modules acted in support of the DMM and the DAM.

Section 3.3.1 justifies the epidemiological approach chosen by this research by comparing and contrasting the spread of biological disease and digital malware. It then draws parallels between biological techniques and the digital paradigm for use by Rx. The constituent system modules are presented in detail in Section 3.3.2. The integration of these

modules into the Rx system is discussed in Section 3.3.3. Finally, Section 3.3.4 defines the deployment framework for Rx.

3.3.1 Biological Influence

This section explains and justifies the biological approach to network stealth worm mitigation chosen by this research and incorporated by Rx. The pages below compare and contrast the spread of biological disease and digital malcode. The final part of this section maps the biological techniques into the digital paradigm for use by Rx.

Similarities Between Biological and Digital Diseases

Worms propagated through computer networks by penetrating systems and then spreading to other hosts. This situation paralleled the transmission of infectious disease in biology where a foreign entity propagated through a defined group of individuals by infecting a member of the group which, in turn, spread it to others.

The use of this scientific technique focused on epidemiological principles concerned with the transmission of disease in a living population. The adaption of this approach initiated with the mapping of biological terminology into the digital paradigm as outlined in Table 3.1. The first five terms–population, organism, disease, environment, and transport medium–discuss the fundamentals of disease propagation explained in Section 2.4.2. Health, illness, and symptom describe manifestations of the disease and facilitate detection and identification. Diagnosis, prognosis, and treatment, the final three terms in the table, define

a process to detect, understand, predict, and control a disease.

Epidemiology is concerned with the health of individuals, specifically living organisms; just as the field of network security worked to maintain the normal operation and availability of computing devices. Groups of such devices formed a network while biology regards sets of individuals as a population. A biological disease adversely affects the health of individuals just as digital malicious code alters the normal operation of computing devices. The digital disease, in the case of a network stealth worm, was transported by network connections whereas biological disease may be spread by air, surfaces, or by other organisms. A biological disease and an individual share an environment characterized by temperature, moisture, social interactions of the individual, and so forth. Prevalence of vulnerabilities, reactive defense mechanisms, open Internet communications model, and lack of diversity established the digital malware environment [8].

The normal state and behavior for an individual is known as health while a computing device was said to be under normal operation. Such a device was under abnormal operation when it adversely deviated from normal operation, a condition known as illness for biological organisms. A living organism exhibited a symptom, similar to an anomaly in a digital computing device, which indicated an illness or abnormal operation.

The identification and determination of causal factors of disease came as a result of examination and data study. This was parallel to detection and characterization of malcode in an abnormally operating computing device. Analysis and prediction contributed to understanding the effect and outcome of malcode on the system much like prognosis does for a

Table 3.1: Mapping of Terminology Between Biological and Digital Paradigms

Biological Paradigm	**Digital Paradigm**	**Explanation**
Population	Network	Total set of individuals under study
Organism	Computing device	Individual within set under study
Disease	Malicious code or hacker	Response of entity to invasion or influence of foreign substance that affects normal state or behavior [2]
Transport medium: air, surfaces, or organisms, etc.	Transport medium: network connections	Method or mechanism of propagation of a disease
Environment: temperature, moisture, organism interactions, etc.	Environment: prevalence of vulnerabilities, reactive defenses, etc. per Section 2.3.1	Geographical, physical, social, etc. surroundings where disease and host coexist
Health	Normal operation	Condition of normal state and behavior for an organism [2]
Illness	Abnormal operation	Adverse deviation from healthy state or behavior as a result of disease [2]
Symptom	Anomaly	Sign or indication of a disease especially when it signals an adverse deviation from normal state or behavior [2]
Diagnosis	Detection and characterization	Identifying or determining cause and nature of disease through evaluation of patient characteristics and data [55]
Prognosis	Analysis and prediction	Prediction of likely course and outcome for a disease [55]
Treatment	Response	Providing remedies to an ill individual with the intent to return it to a healthy state [55]

biological organism infected with a disease. The objective of prognosis is to ultimately identify a treatment for the disease in order to return the organism to a healthy state. The digital paradigm issued a response to eradicate and recover from the effects of digital malware.

Differences Between Biological and Digital Diseases

Several key differences existed between biological disease and digital malicious code. These points included the durations of the latent and incubation periods and the speed of the transmission medium which affected the total duration of the infection cycle. Biological disease typically affects a population over days and weeks through tens of years; digital malcode, however, often propagated on the order of seconds and hours with a few strains stretching into months and years. Such differences placed additional constraints on the effective defense against such malcode which required an expedient and accurate response.

Epidemiological Concepts in Digital Networks

Table 3.2 maps epidemiological techniques to the cyber paradigm, shown in italics, by presenting the three main stages in countering sickness: diagnosis, prognosis, and treatment. Rx incorporated those techniques that are printed in boldface. The foundation of Rx and primary contribution of this research addressed the second phase in thwarting illness known as prognosis. Rx adapted a bio-mathematical model and used demographic analysis from biology in order to identify, parameterize, and forecast an active stealth worm in a computer network.

Rx accepted data from other devices performing the diagnosis phase. Rx, however,

Table 3.2: Mapping of Techniques Between Biological and Digital Paradigms

	Biological Paradigm	**Digital Paradigm**
Diagnosis	*Examination* to identify disease in an individual	*Anomaly detection* to identify malicious code or hacker subversion of computing device
	Classification of disease with ICD-9-CM document [2]	*Classification of malicious code and exploits* with CVE database
	Disease reporting structure for hierarchically reporting and analyzing incidents of disease for timely and effective response at higher levels as established by United States Center for Disease Control [2] [67]	***Malcode reporting structure*** for hierarchically reporting and analyzing incidents of malicious code for timely and effective response at higher levels
Prognosis	***Bio-mathematical modeling*** to parameterize disease and forecast propagation	***Bio-mathematical modeling*** to identify and parameterize disease and forecast spread
	Demographic Analysis to determine nature and cause of disease in living organisms	***Demographic Analysis*** to determine nature and cause of malicious code in computer networks
Treatment	***Quarantine*** infective organisms, *cure* those infected, and *immunize* those uninfected	***Isolate*** compromised computing devices, *repair* those compromised, and ***isolate*** and *update* those yet compromised

contributed at the diagnosis phase. As the two paragraphs below explain, Rx also performed a population-level diagnosis based on aggregated anomaly data.

Bio-mathematical modeling served the system in several ways. First, the model offered a mechanism to confirm network stealth worm infection behavior given pure infection data or, additionally, a technique to infer worm infection behavior from aggregated reports that contained both worm incidence and false-alarm anomaly reports. Bio-mathematical modeling also allowed Rx to parameterize the malcode and provide a forecast for worm propagation.

Note that Rx, unlike epidemiology, used bio-mathematical modeling as a technique to initially *identify* a disease, a step typically reserved for the diagnosis stage in biology. This resulted from the differences in the reported data among the two disciplines. Medical professionals primarily submit incidents of specified diseases as set forth by federal law. Digital malware sensors, however, sent reports of detected malware that was known, but generated anomalies when presented with uncharacterized malcode. Rx applied the bio-mathematical model as a real-time tool to identify self-propagating code from anomaly reports.

Demographic analysis of the digital population by Rx continued the prognosis phase. This allowed the system to understand and specifically identify the relationship between the malware and the host. The result of this function determined, for example, that a particular piece of malicious code affected only machines with a certain operating system and specified service active on a set port.

Rx made a further contribution to the field of network security by extending the result

of demographic analysis into a treatment–the third stage of Table 3.2–for infected machines. Rx used information computed by the demographic analysis step to quarantine infected and vulnerable machines in *advance* of a spreading worm. This dramatically improved the survival rate for networked machines under attack by a surreptitious worm.

3.3.2 Modules

Six distinct modules, listed below, formed Rx. These encapsulated the functionality of the overall system. This allowed for the testing of individual modules and provided for the straightforward extensibility of Rx.

1. Bio-mathematical Model Module
2. Demographic Analysis Module
3. Analysis and Response Module
4. Storage Module
5. Input Transformation Module
6. User Interface Module

This research focused on the Bio-mathematical Model Module (BMM) and the Demographics Analysis Module (DAM). These modules comprised the core of the epidemiological approach that motivated this effort. The BMM used an epidemiological model to identify the presence of an active worm based on end-host worm infection data. The BMM further estimated the effective probing rate for the malcode and provided a forecast for worm propagation through the population. Accurate modeling of worm spread through a network was

achieved with population characteristics provided by the DAM. The DAM held the responsibility for determining and damaging the disease-host relationship through directed, dynamic quarantine of infected and vulnerable individuals.

Four additional modules supported the BMM and the DAM. The Analysis and Response Module (ARM) provided intelligence through administrator-configured metrics to trigger the BMM to search anomaly data sets for signs of network stealth worm propagation. The ARM also executed the defensive responses of the DAM. The Storage Module (SM) archived demographic data from the DAM and infection and anomaly data for the BMM. The Input Transformation Module (ITM) was a mechanism to translate data from commercial off-the-shelf security products into reports that Rx, specifically the ARM, understood. Finally, the User Interface Module (UIM) provided an interface for system administrators to configure settings for Rx and to allow users to view network health reports.

Bio-mathematical Model Module (BMM)

The Bio-mathematical Model Module (BMM) identified a network stealth worm by using an epidemiological model to perform statistical inference on aggregated incidence and anomaly reports generated by end-hosts. The BMM, triggered by the ARM, required model parameters as inputs and returned a confidence measure indicating the fitness of the data as an epidemic curve. The BMM, in effect, performed real-time comparisons to measure how well aggregated anomaly reports matched epidemic curves. In so doing, the BMM also estimated the effective probing rate and forecasted the propagation of a worm.

This research used a BMM that implemented an SIR model from biological epidemiology as discussed in Section 2.4.1. Both biological and digital paradigms, covered in Sections 2.4 and 2.5, employed techniques from epidemiology to study the spread of disease.

The BMM realized the SIR model defined by Equations 2.6 through 2.8 as a set of difference equations. Difference equations provide an efficient and reasonable approximation to differential equations. First, the four sets of individuals–population, susceptible, infected, and recovered–were normalized by the total population. This is shown by Equations 3.1 through 3.4. Note that the normalized population, p, always equaled unity.

$$p = \frac{P}{P} = 1 \tag{3.1}$$

$$s = \frac{S}{P} \tag{3.2}$$

$$i = \frac{I}{P} \tag{3.3}$$

$$r = \frac{R}{P} \tag{3.4}$$

The parameters α and λ that described the average infection and recovery rates were divided by the supplied number of time steps, *numsteps*. This resulted in α_s and λ_s as shown by Equations 3.5 and 3.6.

$$\alpha_s = \frac{\alpha}{numsteps} \tag{3.5}$$

$$\lambda_s = \frac{\lambda}{numsteps} \tag{3.6}$$

The BMM implemented the difference equations given by 3.7 through 3.9 which follow from Equations 2.6 through 2.8 to construct the SIR model. Unit time, t is divided by *numsteps*, each of duration $\frac{t}{numsteps} = z$.

$$s_z = s_{z-1} - \alpha_s i_{z-1} s_{z-1} \tag{3.7}$$

$$i_z = i_{z-1} + \alpha_s i_{z-1} s_{z-1} - \lambda_s i_{z-1} \tag{3.8}$$

$$r_z = r_{z-1} + \lambda_s i_{z-1} \tag{3.9}$$

The difference equations come as a straightforward solution using the numerical Euler method. This is appropriate as Equations 3.7 through 3.9 result in smooth curves with rapid convergence without using the modified Euler method. The BMM could have used a standard ordinary differential equation (ODE) solver. However, direct implementation with difference equations proved accurate for capturing worm behavior, sufficient for processing speed, and was more convenient than interfacing to a third-party software package.

The BMM conducted an exhaustive search over the probing rate, α, to determine the best-fit epidemic curve computed by the model that matched the observed propagation data. The following list enumerates the input parameters required by the BMM. The module expected the number of the total population potentially affected by the worm as well as the

initial values of the number susceptible, infected, and recovered. The BMM also needed the ranges and step values for α and λ in addition to the number of steps per unit time for the model. Finally, the BMM required aggregated incidence or anomaly data from the population.

1. P
2. $S(0)$
3. $I(0)$
4. $R(0)$
5. α_{start}
6. α_{stop}
7. α_{step}
8. λ_{start}
9. λ_{stop}
10. λ_{step}
11. $numsteps$
12. propagation data

The BMM used the normalized weighted least squares error, given by Equation 3.11, to measure the accuracy between the spread computed by the model and the supplied propagation data. Note that the division by $Model_i$ in Equation 3.11 gives preference to the early part of the infection cycle where more accuracy was desired. The error measurement was scaled by $\log_{10}$ as shown by Equation 3.12 for convenience in discussion. Equation 3.12 serves as the confidence measure.

$$error_{raw} = \sum_{i=1}^{numsteps} \frac{(Actual_i - Model_i)^2}{Model_i} \tag{3.10}$$

$$error_{normalized} = \frac{error_{raw}}{numsteps} \tag{3.11}$$

$$error_{scaled} = \log\left(error_{normalized}\right) \tag{3.12}$$

Returned data from the BMM included the scaled error, the effective value of the probing rate, and forecasted spreading data. These are given in the list below. A lower scaled error more strongly indicated the presence of a network stealth worm. The effective probing rate was the network-average probing rate computed by the BMM. A worm spreading in a population with a significant number of immune machines experienced an effective probing rate lower than its actual probing rate. The forecasted spreading data computed by the BMM predicted the propagation of the worm given current network conditions, specifically without a defensive response. The BMM could have been tasked to consider different responses with alternate effective probing rates at different points in the infection cycle.

1. $error_{scaled}$
2. $\alpha_{effective}$
3. forecasted spreading data

The BMM depended on sensors that were sensitive to a network stealth worm and were able to generate incidence or anomaly reports. The BMM further required specific population information, namely the number of machines in each compartment that were infected by the network stealth worm under consideration. The DAM provided this population data.

The BMM implemented a three-state SIR model. Such models are readily extensible to describe different or more complex behavior [52] [53]. Epidemiological models can also describe spatial worm propagation [51]. The BMM could have simultaneously implemented a variety of models to search and identify worm propagation from aggregated end-host reports.

Demographic Analysis Module (DAM)

The Demographic Analysis Module (DAM) gathered and stored vital statistics on the network hosts and derived population-level information. During a potential worm attack, the DAM supplied the BMM with population data to facilitate mathematical modeling and identified the disease-host relationship to enable directed defenses.

The DAM viewed network entities as possessing degrees of sickness. Digital pathogens, even when not directly detected, commonly caused compromised hosts to deviate from a normal or healthy state. Possible symptoms included consumption of disk space, memory, processor time, and communications bandwidth. More subtle perturbations potentially materialized as rogue processes, file system changes, abnormal sequences of system calls, open ports, timed jobs, and sniffers on the communications channels, such as, keyboard loggers and network interfaces. Furthermore, otherwise healthy hosts may have been affected by the sickness of others. Consumption of bandwidth, malicious packets, and the denial of services stood as examples of this. The DAM measured and evaluated this perceived sickness in order to detect and respond to a disease.

The DAM characterized the population by recording demographic data through active

scanning, passive monitoring, or trusted host-based software. Tools such as Nessus scanned a target machine and determined the operating system, open ports, running services, and security warnings and vulnerabilities [68]. The physical device itself could have been fingerprinted to continue to track the device as it moved or changed configuration [69]. More information would have been available with cooperative, trusted software running on the host much like traditional anti-virus and firewall software. Such programs could have provided detailed information such as the presence, type, and configuration of a virus scanner and firewall, the configuration of the OS, installed software, user habits, and even observations or problems submitted by the user. This information was gathered and stored in the SM by the DAM.

Empirical trials suggested that a single scanner like Nessus required about 16 seconds to scan a single host for open ports and services; an unused IP address required about four seconds. Given this, the DAM would have needed about four days to record demographic data for a campus-sized network like that of Virginia Tech with about 20,000 active machines and a total IP space of 30,000 addresses. Deployment of additional scanners distributed throughout the network and scanning machines in parallel would have dramatically reduced this time.

Table 3.3 showed the storage space required for demographic information for a single host. The MAC address, IP address, and open ports were expressed as unsigned characters. The operating system was enumerated and then recorded as an unsigned character. The patch level for the operating system was expressed as a string of 15 characters.

Table 3.3: Required Demographic Storage Space for the DAM

Characteristic	**Representation**	**Disk Space (Bytes)**
MAC Address	6 1-byte unsigned characters	6
IP Address	4 1-byte unsigned characters	4
Operating System	1 1-byte unsigned character	1
Patch Level	15 1-byte character strings	15
Open Ports (0 - 65,535)	2 1-byte unsigned characters	2
TOTAL for 1 host		28

According to Table 3.3, demographic data for one machine occupied 28 bytes of disk space. A campus-sized network like that of Virginia Tech with about 20,000 active machines would have needed 560 kB of disk space. This was both technically and financially reasonable and easily allowed room for network growth and for recording other demographic data such as active services.

This research implemented a DAM that collected the operating system type, open ports, and active services. This information was sufficient to model diversity in a campus or corporate-sized network and thus measure the effectiveness of the DAM.

Systems in the network periodically reported their perceived sickness to the DAM. This included symptoms they exhibited, those observed in their neighbors, and those experienced in the network. The DAM assessed those systems not reporting based on a security and privacy policy established by the implementing organization.

The ARM triggered the DAM during a potential worm attack. The ARM provided a list of machines known or believed infected where a compromised machine was identified by its Medium Access Control (MAC) address. Device fingerprinting may have offered a

way to track a system as it moved or counter a system that used multiple MAC addresses [69]. The DAM analyzed the vital statistics of the population, searching for common features among the compromised hosts. This yielded a result that indicated all infected machines, for example, used a particular service with a known open port. This could have been checked for a known exploit in the vulnerability database contained in the Nessus Vulnerability Scanner [68].

The DAM returned two pieces of valuable information. First, the module provided population data to the BMM to support bio-mathematical modeling. The information specifically included the total number in the population as well as the number susceptible, infected, and recovered. The DAM also determined the disease-host relationship by identifying the common features among the infected individuals that also differed from the unaffected machines. This finding was reported to the ARM to facilitate quarantine response.

Analysis and Response Module (ARM)

The Analysis and Response Module (ARM) served as the prime mover for Rx. It set the analysis and response functions of Rx in motion when the module received an detection report or upon suspicion of a worm based on excessive numbers of anomaly reports. The threshold for each of these decisions could have been established by a set value or taken from the mean and variance of a normal distribution based on *healthy* network observations to estimate and control false-positives.

The ADM specifically triggered the BMM and the DAM to execute during potential

worm attacks. The ARM created instances of the BMM to test different hypotheses concerning worm spread. The ARM identified a network stealth worm based on the confidence measure returned by the BMM as compared to the threshold value of the confidence measure and the minimum number of infected individuals set by administrators to indicate a worm event.

The ARM issued a containment response to control the propagation of the identified network stealth worm. The DAM determined and reported the specific machines *infected* by the worm and *vulnerable* to the worm based on demographic analysis of the population. The DAM effectively identified the disease-host relationship that included features such as the operating system type, open ports, and active services. The ARM quarantined the infected machines and isolated vulnerable machines in *advance* of a spreading worm. This assumed the ARM had permissions to update switch, router, firewall, and network filtering rules.

Storage Module (SM)

The Storage Module (SM) held information gathered by Rx. This included vital statistics gathered by the DAM and incidence and anomaly reports received by the ARM.

Input Transformation Module (ITM)

The Input Transformation Module (ITM) translated data from external sensors into the format expected by Rx. The ITM was easily updated with augmented or new input data formats. This allowed Rx to utilize information from a variety of sources to include current and future commercial off-the-shelf security products. These could have been either network

or host-based and could have collected data through active scanning, passive monitoring, or cooperative end-host software reporting. Rx expected data consisting of *time*, *machine*, and *event* where an event described either a *confirmed infection* or an *anomaly*.

User Interface Module (UIM)

The User Interface Module (UIM) provided a means of interacting with Rx and for viewing network health status. The module offered administrators a means to configure system settings, to access low-level or protected information, and to view and moderate automated responses. The UIM allowed administrators and possibly users, as determined by the administrators, to view network health status.

3.3.3 Node: Module Integration

The six modules combined to form an Rx node as shown by Figure 3.1. The research focused on the creation and evaluation of the BMM and the DAM. The BMM performed bio-mathematical modeling based on models from biological epidemiology. Demographic analysis, conducted by the DAM, provided population data to support the BMM and determined the disease-host relationship for a directed containment response.

The ARM principally drove the operation of an Rx node. The module used incidence alerts and processed anomaly reports to initiate the identification, characterization, and control of network stealth worms. The ARM created and tasked instances of the BMM to test hypotheses related to confirmed or suspected worm spread. The BMM identified an

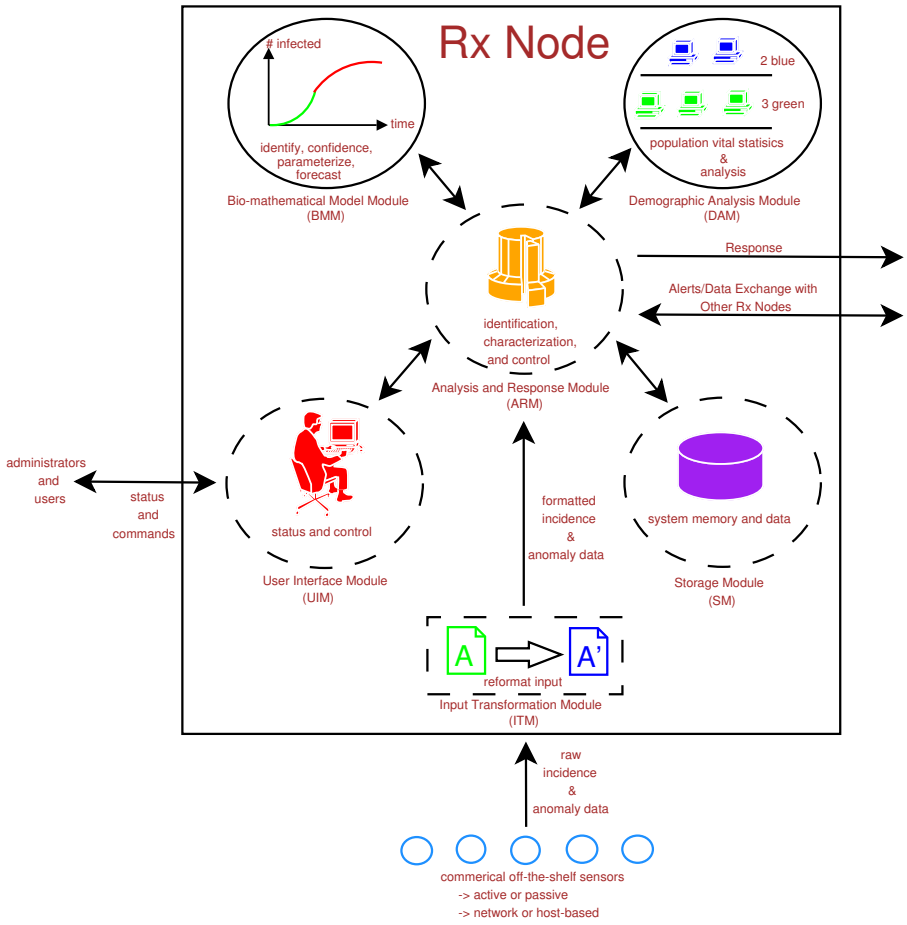

Figure 3.1: Rx Node

epidemic spreading pattern contained in anomaly data and computed a confidence measure for this detection. The BMM also parameterized the worm and forecasted the expected propagation of the malcode. The DAM supplied population vital statistics to the BMM to support mathematical modeling and performed demographic analysis to determine the set of machines within the population that lie vulnerable to the worm. The ARM used this population information to contain the worm propagation by quarantining infected hosts and isolating vulnerable machines in advance of a spreading worm.

The ARM, described above, and the SM, the ITM, and the UIM supported the BMM and the DAM. The SM stored demographic data from the DAM and infection and anomaly data for the BMM. The ITM processed raw input data from commercial off-the-shelf security products into reports that Rx understood. Finally, the UIM provided an interface for node configuration by administrators and for users to monitor the current network health status.

An Rx node used three inputs and three outputs. Incidence and anomaly data entered the node from local malcode sensors, data exchanges with higher-level Rx nodes, or Rx root nodes of different organizations. An Rx node issued a containment response by deploying new switch, router, firewall, and network filtering rules. The user interface accepted incoming commands and settings into the system and sent status back to the user.

3.3.4 Framework: Node Configuration

Rx defined a hierarchical reporting and response framework for epidemic events as illustrated in Figure 3.2. Under this framework, Rx nodes were positioned at key routers and in large subnetworks with a single Rx root node at the core or edge router. Hosts reported malcode incidence and anomaly data to the nearest Rx node which, in turn, relayed this information to the next higher-level Rx node with the data eventually reaching the Rx root node. Organizations optionally peered their Rx root nodes to exchange data. This framework was modeled after the successful reporting and response framework for biological disease used by the United States Center for Disease Control and Prevention discussed in Section 2.4.2.

The CDC, through federal law and mandates, manages the reporting, collection, and

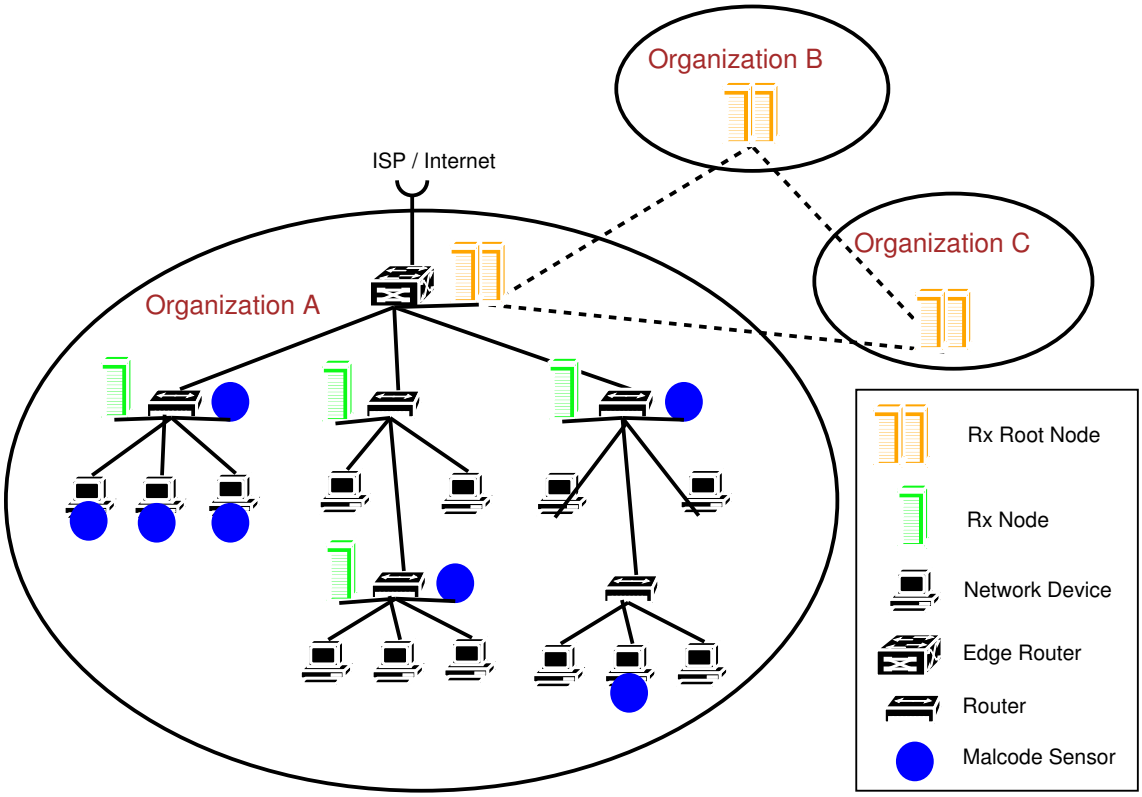

Figure 3.2: Rx Framework

processing of population vital statistics and morbidity and mortality statistical analysis. The CDC achieves this by specifying the hierarchical activities and report flows at the local, state, and national levels.

In the same manner, Rx aggregated and processed incidence and anomaly reports pertaining to individual end-hosts. The role of the health care provider or doctor in the biological world was played by active or passive sensors based either in the network or installed on the end-host. Rx required sensors to be sensitive to the effects of a network stealth worm. Security professionals positioned Rx nodes at any level within the network. Rx was designed for one Rx root node at the edge or core router with supporting Rx nodes

positioned in large subnetwork segments. The hierarchical reporting allowed for higher levels in the network to gain a broader view of the health status of the population and to more quickly issue a defensive response to impede or block further worm propagation without having to wait for the Rx root node to respond. Organizations peered their top-level Rx root nodes in order to share information. Rx overcame the failure of a node by instructing hosts to report to the next nearest Rx node. Rx nodes continued to operate even if severed from other Rx nodes or even the Rx root node.

This framework ensured a scalable deployment and fault-tolerant, graceful degradation in effectiveness given the failure of Rx nodes or communications links. Rx did not require Internet-wide participation or even organizational-wide involvement. Small and medium-sized networks and subnetworks made for excellent deployment scenarios for Rx. As the grass-roots movement towards such security systems grew, more Rx nodes could have been added into the scalable architecture.

Rx was intended for use at ISP, campus, and corporate-sized networks. These networks typically had a sizable population of hosts that provided the necessary infection data for Rx to operate while still benefiting from the dynamic response of the system. Rx was not designed to absolutely prevent the infection of a single machine, rather, in agreement with biology, to ensure the survival of the overall population. The primary concerns involved the ethical, legal, and privacy problems associated with actively or passively monitoring machines and the installation of cooperative software on end-systems.

A company offered the best deployment model. It typically owned the network as

well as most, if not all, the connected equipment. The company could have mandated the deployment of Rx nodes among subnets, required the use of host-based software to communicate with Rx nodes, and established a firm security and privacy policy for its users and system administrators.

A campus environment was similar to that of the corporate one except a substantial population of machines would have been privately owned, mainly by students living on campus and those connecting into university machines from home. It might have been difficult, if not unreasonable, to require custom, host-based software installed on personal machines. Additionally, this would have potentially resulted in legal liability for the software. In addition, actively scanning private machines especially over a network not owned by the university may have been construed as an invasion of privacy and theft of Internet service. In this case, it might have been possible to passively fingerprint machines although this would certainly result in a loss of granularity.

An Internet Service Provider (ISP) could have deployed the full Rx system within its intranet but would have suffered the same challenges met by the university since machines connecting to the ISP belong to a customer. Like the university, the ISP could have passively monitored machines with some loss of precision. The ISP could have advertised Rx as a service and made the host-based software available as an option to the subscriber.

3.4 Rx Simulation

Rx was realized in simulation with the mathematics package Octave [10]. Simulation allowed for the evaluation of Rx under a controlled environment and with large-scale network deployment. Both of these scenarios would have proved difficult if not impossible to obtain through physical implementation. Octave was chosen to simulate Rx due to the ability of the former to quickly realize the system functionality and allowed expedient changes to the system configuration through a high-level scripting language.

Rx was challenged with worm propagation data computed by the Probabilistic-based Simulator. This simulator processed worm spread as a time-series of probabilistic events as discussed in Section 3.6. This simulator was used in the course of analysis as it captured the stochastic propagation of a worm in large networks and the independent relationship between the worm and the defensive response.

The BMM and the DAM, both written in Octave, formed the foundation for Rx. The simulated trials used corporate or campus-sized networks of 20,000 machines. Input data to Rx consisted of either incidence-detection of pure worm infection alerts or anomaly-based reports that included both worm spread and false-alarm anomalies.

3.5 Rx Implementation

After the completion of simulation trials, Rx was evaluated against a Java-based worm. This language provided for efficient network programming and offered reasonable processing

speed. The Java worm consisted of software installed on each machine in the test network that possessed the ability to receive incoming probe attempts and, if successful, initiate its own probes. In this manner, the software emulated the spread of a network worm. The software also reported the infection of the machine to Rx based on parameters such as detection probability and detection time.

A test network of 10 machines was constructed using VMware on five physical hosts [11]. The Java-based worm software was installed on these VMware system images. Per Section 3.3.4, the design of Rx targeted campus or corporate-sized networks that numbered in the hundreds to upwards of tens of thousands of machines. This gave Rx a sizable population of hosts to provide an amount of infection data sufficient for Rx to operate and issue a response that would benefit the network. Rx was not designed to absolutely prevent the infection of a single machine, rather, in agreement with biology, to ensure the survival of the overall population. The number of machines used during implementation was consequently low but still demonstrated the efficacy of Rx.

The Java-based worm was used as an evaluation tool instead of an actual piece of self-propagating code because Rx did not detect individual worm instances on single systems. Instead, it operated at the network level on aggregated end-host anomaly reports, explicitly depending on detectors to be sensitive to worm instances. The Java worm emulated the behavior of a network stealth worm which specifically (i) provided complete control over the parameters of the worm, (ii) abstracted the exploited vulnerability of the machines, (iii) emulated the presence of end-host detectors with test-defined sensitivity or detection

probability to the worm, and (iv) provided ground-truth data for the evaluation of Rx.

3.6 Simulator

This research implemented a Probabilistic-based Simulator to compute the spread of a network stealth worm. The simulator considered activities within the simulation as probabilistic events over infinitesimally small time steps. The simulators were verified against biological and digital diseases.

The Probabilistic-based Simulator modeled the spread of network stealth worms as a time-series of probabilistic events divided over infinitesimally small time steps. The simulator recorded the number of susceptible, infected, and recovered individuals within the population at regular intervals. The simulator was written in C++.

The simulator processed events as occurring with a small yet finite chance at each time interval. This is commensurate with the biological view of the parameters as probabilities [52]. Specifically, the probing rate α and the recovery rate λ were supplied by the user for a given time period then translated to a probability by dividing the values by the number of specified time steps. The user set the *sampleinterval* to instruct the simulator to record population counts at regular intervals. The simulator used the pseudo-code algorithm listed below.

1. Initialize *S(0)* susceptible machines, *I(0)* infected machines, and *R(0)* recovered machines
2. Initialize *current time* = 0 and set *sample time* = *current time* + *sample interval*
3. Divide *time* by the number of *time steps* from the *start time* to the *stop time*

4. Divide the probing rate α and the recovery rate λ by the number of *time steps* for α_{step} and λ_{step}
5. For all machines, if the current machine, M, is infected
   (a) M has a α_{step} chance to send a probe. If it wins, pick a random target from the population and infect that target if it is susceptible.
   (b) M has a λ_{step} chance to recover. If it wins, set its status to 'recovered'.
6. If *current time* = *sample time*, then record S, I, and R and set *sample time* = *current time* + *sample interval*
7. Return to 5 until *current time* > *stop time*

The simulator was written in C++. This language was chosen for the advantage of speed in execution. A custom simulator was written in place of using existing network simulation software as the former provided the necessary worm propagation data without a learning curve or unnecessary processing overhead often associated with the latter. The Probabilistic-based simulator was verified against the biological disease Hong Kong Flu and the digital worm Code Red I v2.

3.7 Evaluation Metrics

Rx was evaluated through simulation and test-network implementation trials. Simulation provided a controlled environment and large-scale network topology for the study of the efficacy of Rx. Implementation demonstrated the validity of the approach and the effectiveness of the system on physical networks.

This research investigated two types of network stealth worm detection and response: incidence-based and symptom-based. Section 3.2.1 explained this data. Incidence data resulted from absolute identification of malware by network or host-based sensors through

pre-defined signatures. Symptom data was fundamentally equivalent to anomaly detection reports in which network or host-based sensors reported deviations from an established normal profile.

The research defined five metrics, given in Equations 3.13 through 3.17, which measured the effectiveness of the approach and thus Rx. The scaled error follows directly from Equation 3.12. The term *incidence* refers to incidence detection or incidence data while *anomaly* means anomaly-based detection or anomaly data. The number of infected individuals is given by the parameter I. The presence of the subscript Rx indicates the modification of that value due to the usage of Rx on the network; otherwise, the subscript reads *none*. The $effective$ probing rate is the network-average probing rate achieved by the worm which is less than or equal to the *actual* probing rate. The parameter, t, of course, indicates time.

$$M_{1A} = error_{scaled,incidence}(incidence) <= -1 \tag{3.13}$$

$$M_{1B} = error_{scaled,anomaly}(incidence + anomaly) << error_{scaled}(anomaly) \tag{3.14}$$

$$M_2 = \alpha_{effective,estimated} = \alpha_{effective,actual} \tag{3.15}$$

$$M_3 = t_{Rx}(I) << t_{none}(I) \tag{3.16}$$

$$M_4 = I_{Rx} << I_{none} \tag{3.17}$$

These metrics gauged the efficacy of the approach and the ability of the overall Rx system to (i) accurately identify a worm from either incidence or symptom reports, (ii)

correctly parameterize and forecast worm spread, (iii) impede if not stop worm propagation, and (iv) enhance the average survival rate of computers in the network. The following list specifically describes factors and maps them to the metrics. Rx should thus have:

1. **Incidence-based Detection**

   (a) identified a network stealth worm with a strong confidence measure (Equation 3.13)

   (b) provided an accurate estimate of the effective probing rate and forecasted spread (Equation 3.15)

   (c) created a temporal lag in worm propagation with Rx versus without Rx (Equation 3.16)

   (d) increased in the average survival rate of the population with Rx versus without Rx (Equation 3.17)

2. **Symptom-based Detection**

   (a) identified a network stealth worm with a strong confidence measure over that for pure false-alarm anomaly data (Equation 3.14)

   (b) provided an accurate estimate of the effective probing rate and forecasted spread (Equation 3.15)

   (c) created a temporal lag in worm propagation with Rx versus without Rx (Equation 3.16)

(d) increased in the average survival rate of the population with Rx versus without Rx (Equation 3.17)

3.8 Summary

This chapter presented the design of Rx, a system developed through this research, to address the threat of network stealth worms. The system applied concepts from epidemiology to identify, characterize, and control stealth worm propagation in digital networks. Epidemiology describes a scientific methodology in biology used to study the nature, prevalence, and causal factors of disease with the goal of controlling disease spread [2].

This research mapped epidemiological concepts into the digital problem space for use by Rx in the real-time defense against surreptitious, self-propagating code. The system used bio-mathematical modeling to identify, parameterize, and forecast worm propagation and demographic analysis to control worm spread. These functions were implemented in two modules with another four modules serving in support.

The six modules formed an Rx node that possessed the capability to identify, characterize, forecast, and control network stealth worms. Rx depended on sensors to be sensitive to the effects of this malware in order to generate incidence or anomaly reports. The system required access to managed switches, routers, firewalls, and network filtering devices in order to dynamically update rules to contain worm spread. Rx nodes in a campus or corporate-sized network were hierarchically deployed over subnetworks in order to establish a reporting and response framework for epidemic events. This was modeled after the reporting and re-

sponse strategy for biological disease used by the United States Center for Disease Control and Prevention.

This research developed two simulators: a Probabilistic-based simulator that assigned events to occur with a certain probability and an Event-based simulator that scheduled events at set times. The Probabilistic-based Simulator was used in the course of analysis as this simulator better captured the stochastic propagation of a worm in large networks and the independent relationship between the worm and the defensive response.

Rx was realized both in simulation and implemented on a test-network. Simulation provided a controlled environment and large-scale network topology for the study of the efficacy of Rx. Implementation demonstrated the validity of the approach and the effectiveness of the system on physical networks. The system was developed in Octave for the simulation. A Java-based worm emulated the spread of a network worm to exercise Rx for the implementation trials.

Five metrics were defined to evaluate the effectiveness of Rx. These gauged the ability of the approach and thus Rx to (i) accurately identify a worm from either incidence or symptom reports, (ii) correctly parameterize and forecast worm spread, (iii) impede if not stop worm propagation, and (iv) enhance the average survival rate of computers in the network.

The next chapter presents verification results for worm propagation components used to exercise Rx. Collected data from the spread of the Hong Kong Flu and the Code Red I v2 worm verify the bio-mathematical model implemented by Rx and the simulator used to

generate worm propagation data to exercise the system. A stochastic, epidemiological model verifies the implemented Java-based worm for trials of Rx on a test-network.

Chapter 4

Verification

This chapter provides results that verify the bio-mathematical model incorporated into the Rx system as well as the simulator and Java-based worm used to exercise Rx. Section 4.1 discusses the objectives for the verification trials.

Section 4.2 describes two cases for verification testing: the Hong Kong Flu and the Code Red I v2 worm. Sections 4.3.1 and 4.3.2 present the verification results for the model and the simulator as evaluated with the flu and the worm. These show the correctness of design and implementation for the model and the simulator in both the biological and digital paradigms.

Verification of the Java-based worm used to exercise Rx on a test-network appears in Section 4.3.3. The propagation data computed by a stochastic, epidemiological model tested the Java-based worm. This chapter closes with a summary in Section 4.4.

4.1 Verification Objectives

Verification showed that entities describing worm propagation produced accurate and precise data. Propagation components included the epidemiological model incorporated within Rx and the simulators and the Java-based worm used to exercise the system. Such testing isolated each of these components and executed them with parameters corresponding to known and established epidemic spreading data. The propagation results of the entities and the known spreading data were then compared. The data generated by the worm propagation components was expected to match that of the known spreading data in both number infected or dead over time.

The verification phase served a crucial role in establishing the validity of Rx. The results from these trials specifically showed that the bio-mathematical and probabilistic-based simulator correctly described the spread of both biological and digital diseases. This supported the approach of this research in mapping the problem of controlling biological infectious agents in living organisms to controlling the spread of self-propagating code in digital networks. The results also showed the model and simulator accurately computed the spread for both slow and fast-spreading diseases. The verification of the Java-based worm ensured correct implementation of the worm as well as the accurate operation of the reporting and recording software within the test-network.

4.2 Test Cases

Propagation data from the biological contagion Hong Kong Flu and the digital worm Code Red I v2 verified the bio-mathematical model and the probabilistic-based simulator. This established the correctness of the cross-disciplinary approach, mapping, and implementations of the model and simulator.

4.2.1 Hong Kong Flu

The Hong Kong Flu of 1968 resulted in a relatively mild pandemic in the United States during the winter of 1968 to 1969 [70]. The verification focused on the spread of this flu in New York City in order to compare the simulated results to data recorded by the Center for Disease Control and Prevention (CDC) [67] gathered by [71]. CDC mortality data bears a relationship to the number of individuals infected by a given disease [72]. Table 4.1 enumerates the parameters used for verification of the model with the Hong Kong Flu. The table provides a listing of the parameters and their meanings used in the SIR model as well as the values and units of these variables as set for this investigation.

The new strain of flu meant that nearly the entire population of New York City, about 7.9 million people, was vulnerable [72]. The calculations below assumed 500 infectious individuals, representative of mass population flux, initially brought the disease into the city. The value of α was first estimated at 1/3, indicating a reasonable assumption that each infected individual would make sufficient contact to transmit the disease every three days [72]. After a few iterations of comparison with the actual CDC data, α was tuned slightly to

Table 4.1: Parameters for Model Validation with Hong Kong Flu

Parameter	**Name**	**Value**	**Unit**
P	Population	7.9E6 [72]	individuals
S	Susceptible	7.85E6	individuals
I	Infected	500	individuals
R	Recovered	0	individuals
α	Transmission coefficient	0.3571 [72]	$\frac{contacts}{day}$
λ	Recovery rate	1/4.5 [73]	$\frac{1}{daysinfective}$
cfr	Case fatality rate	0.3 per 1,000 [74]	$\frac{fatalities}{individuals}$

0.3571. The period of infectiousness commonly lasted for 4 to 5 days, so λ was set to 1/4.5 [73].

The output data was compared to the CDC fatality data by converting the percentage of the population infected to number of deaths. A case fatality rate of 0.3 per 1,000 reasonably fit the CDC mortality data. This value agrees with that common to pandemic influenza [74].

4.2.2 Code Red I v2 Worm

The Code Red I v2 worm provided for an excellent validation trial because the worm was well documented and studied. The worm spread by exploiting the .ida vulnerability found in machines running Microsoft IIS [7]. Code Red began propagation on 19 July 2001 at 1000 hours UTC, infecting 359,000 machines within 14 hours before shutting itself off at midnight of the next day [4] [62]. Table 4.2 gives the parameters fed to the model for verification against the Code Red worm along with parameter names, explanations, and units.

Researchers estimated the average effective scan rate for the TCP-based Code Red worm between 2 and 11 scans per second [63] [50]. The experiment below selected a scan

Table 4.2: Parameters for Model Validation with Code Red Worm

Parameter	**Name**	**Value**	**Unit**
P	Population	575,000 [62]	individuals
S	Susceptible	574,999	individuals
I	Infected	1	individual
R	Recovered	0	individuals
$\alpha(t(i < 0.5))$	Transmission coefficient	5 [63] [50]	$\frac{scans}{sec}$
$\alpha(t(i >= 0.5))$	Transmission coefficient	0.5	$\frac{scans}{sec}$
$\lambda(t < 5)$	Recovery rate	0	$\frac{1}{daysinfective}$
$\lambda(t >= 5)$	Recovery rate	0.03 [63]	$\frac{1}{daysinfective}$

rate of 5 scans per second. The number of infected machines represented 60% of the on-line and vulnerable population, suggesting a total vulnerable population of 575,000 machines [62].

Code Red presented two challenges to mathematical modeling. First, the worm enjoyed a recovery rate near zero early in the spreading cycle since reactive defenses failed to significantly stop or impede the worm [62]. Human-deployed countermeasures eventually slowed the spread of the worm. A recovery rate, λ, of 0.03 five hours after the release of the worm captured this effect. This agreed with the disruption in the graph showing the spread of the worm from Cooperative Association for Internet Data Analysis (CAIDA) data [4] and bore similarity, though less aggressive, than the value chosen in [63]. Additionally, Code Red slowed its own propagation due to bandwidth limitations, a result of the enormous number of probes from the worm population later in the infection cycle [62]. To capture this phenomenon, the trials lowered the scan rate to 0.5 scans per second when 50% of the population became infected [62].

4.3 Worm Propagation Generators

Worm propagation generators, verified in the subsequent discussion, described the spread of network worms. Section 4.3.1 presents the verification of the model while Section 4.3.2 does the same for the simulator. Finally, Section 4.3.3 provides verification results for the Java-based worm used to exercise Rx on a test-network.

4.3.1 Deterministic SIR Model

The SIR model within Rx, discussed in Section 2.4.1 and defined in Section 3.3.2, described the spread of disease through a population by establishing the dynamics of spread for three groups from susceptible to infected to recovered. The system depended upon this model to identify, characterize, and forecast the propagation of network stealth worms.

The spread of the Hong Kong Flu first verified the model. The model parameters were set according to Section 4.2.1 with a total population of 7.9 million individuals: a susceptible population of 7.85 million individuals, 500 infected individuals, and no recovered individuals. The model assigned the flu an average effective spreading rate of 0.3571 and a recovery rate of 1/4.5. Table 4.1 shows these parameters.

Figure 4.1 compares the result of the model with recorded CDC fatality data for the Hong Kong Flu. The model returned the percentage of the population infected, i, by the disease. This value was converted to number of fatalities as discussed in Section 4.2.1.

The model accurately described the spread of the biological contagion as recorded by the CDC. The former captured both the temporal trend in the fatality data as well as

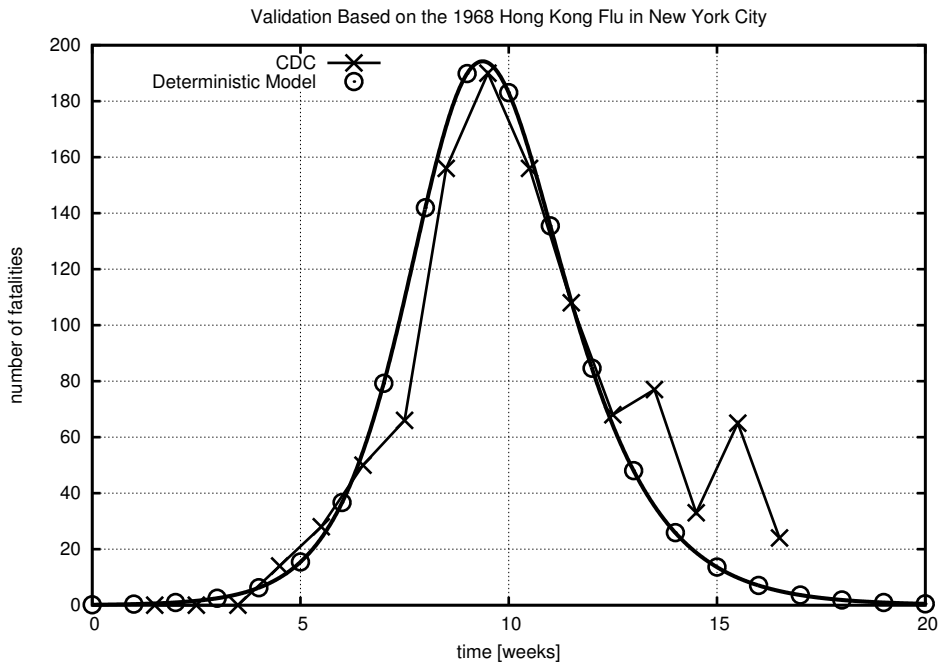

Figure 4.1: SIR Model vs. CDC Data for Number of Fatalities

the actual number of individuals who succumbed to the disease over time. Perturbations occurred in the CDC fatality data toward the end of the infection cycle which the model did not exhibit. The results also matched those of [72], derived from [71] and [75].

Next, the spread of the digital worm Code Red I v2 was used to exercise the model. A total population of 575,000 machines with one among them infected, leaving a susceptible population of 574,999, and no recovered or immune machines initialized the population demographics for the model. The transmission coefficient was set to 5 scans per second until the worm infected half the population at which point the value was lowered to 0.5 scans per second. The model assigned a recovery rate of zero to the worm from the start of the infection cycle through the first five hours, increasing the recovery rate at the time to 0.03

$\frac{1}{daysinfective}$. These values, shown by Table 4.2, follow from Section 4.2.2.

Figure 4.2 plots the data from the model describing the spread of the Code Red I v2 worm. The graph appears on the same time scale as the plot in Figure 4.3[1] showing the actual spread of the Code Red worm as recorded by CAIDA [76]. Several efforts were made to obtain this data from CAIDA over the course of three years. Data for public use was expected to be available in six to nine months. The time of '0' on Figures 4.2 and 4.3 indicates the initial release of the worm at 1000 UTC on 19 July 2001.

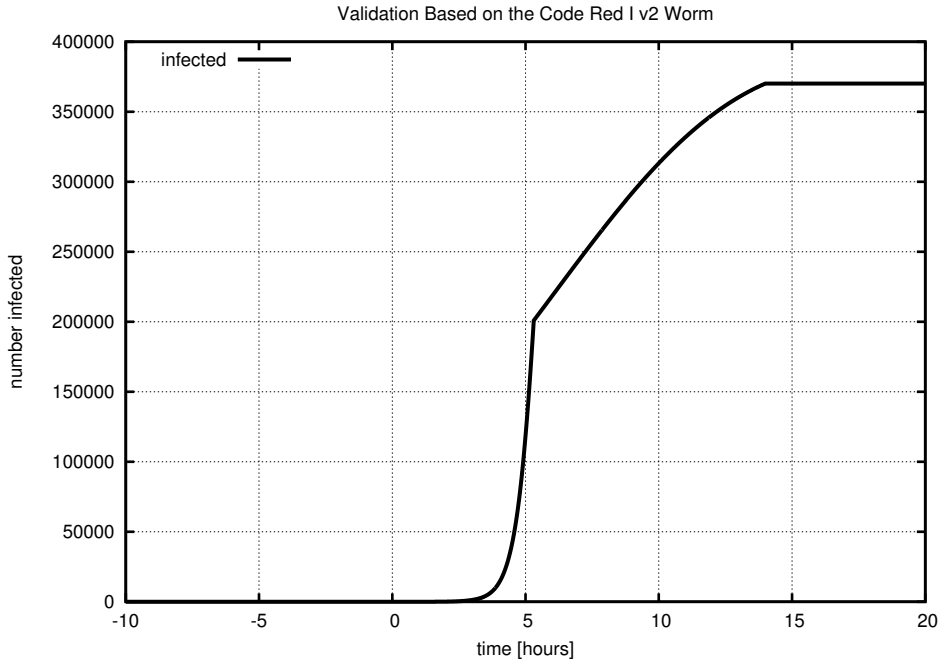

Figure 4.2: SIR Model Predicting Number of Compromised Systems

The expected propagation of the worm computed by the epidemiological model matched the recorded spread of the worm in terms of number infected over time as shown by Figures

[1] ©2001 The Regents of the University of California. Courtesy University of California. All Rights Reserved.

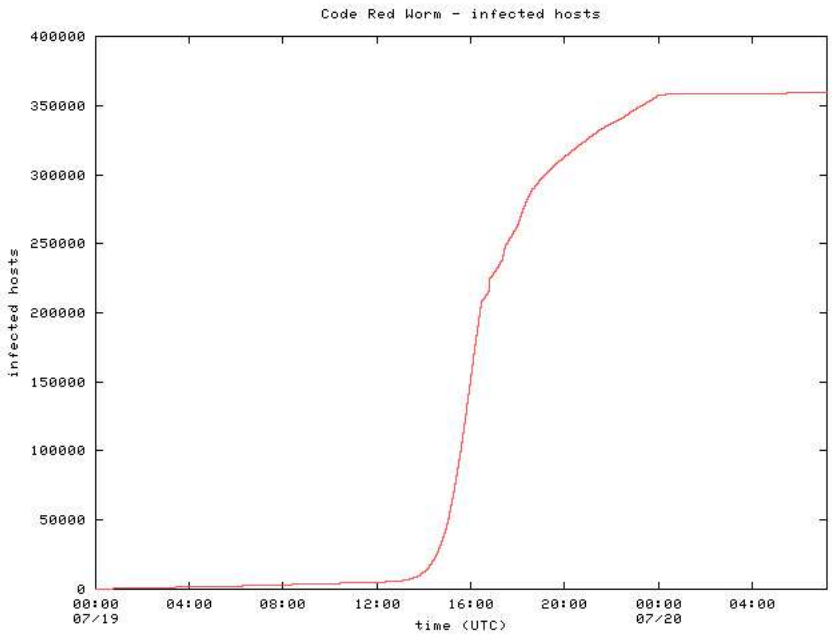

Figure 4.3: CAIDA, Code Red Visualization, 2001

4.2 and 4.3. The model exhibited a sharp discontinuity not immediately observed in Figure 4.3. This was due to varying the parameters α and λ to the SIR epidemiological model during the course of the simulation. Such models are designed to operate with parameters that remain constant for the entire infection cycle. It was likely that gradually varying the model parameters would have yielded a better result had this data been available. However, the epidemiological model fit the observed data shown in Figure 4.3 as collected by [4]. Furthermore, the model proved accurate during the early period of the infection cycle which was important for the early detection of network worms.

4.3.2 Probabilistic-based Simulator

Verification of the probabilistic-based simulator initiated with the comparison against the Hong Kong Flu. Table 4.1 provides the parameters used for this trial according to Section 4.2.1. Figure 4.4 shows the predicted spread of the Hong Kong Flu compared with a trace of actual fatality data recorded by the CDC. Like the model in the previous section, the simulator also returned the percentage of the population infected, i, by the disease. This value was converted to number of fatalities per Section 4.2.1.

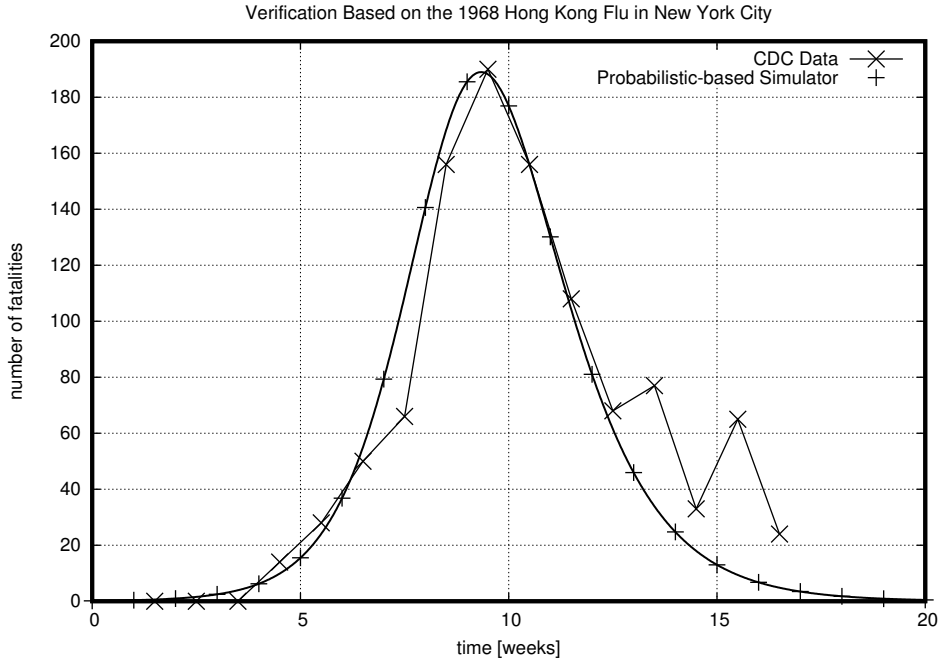

Figure 4.4: Probabilistic-based Simulator vs. CDC Data for Number of Fatalities

The simulator accurately computed the propagation of the biological contagion Hong Kong Flu as compared to the data recorded by the CDC. The former captured both the temporal trend in the fatality data as well as the actual number of individuals who succumbed

to the disease over time. Perturbations occurred in the CDC fatality data toward the end of the infection cycle which the model did not exhibit. The results also matched those of [72], derived from [71] and [75].

The spread of the digital worm Code Red I v2 verified the simulator. The simulation settings were applied according to the discussion in Section 4.2.2 and the information given in Table 4.2. Figure 4.5 gives the expected propagation of Code Red I v2 as predicted by the simulator. The graph appears on the same time scale as the plot generated by CAIDA, shown in Figure 4.6[2], based on collected data from the recorded spread of Code Red [4]. The time of '0' indicates the initial release of the worm at 1000 UTC on 19 July 2001.

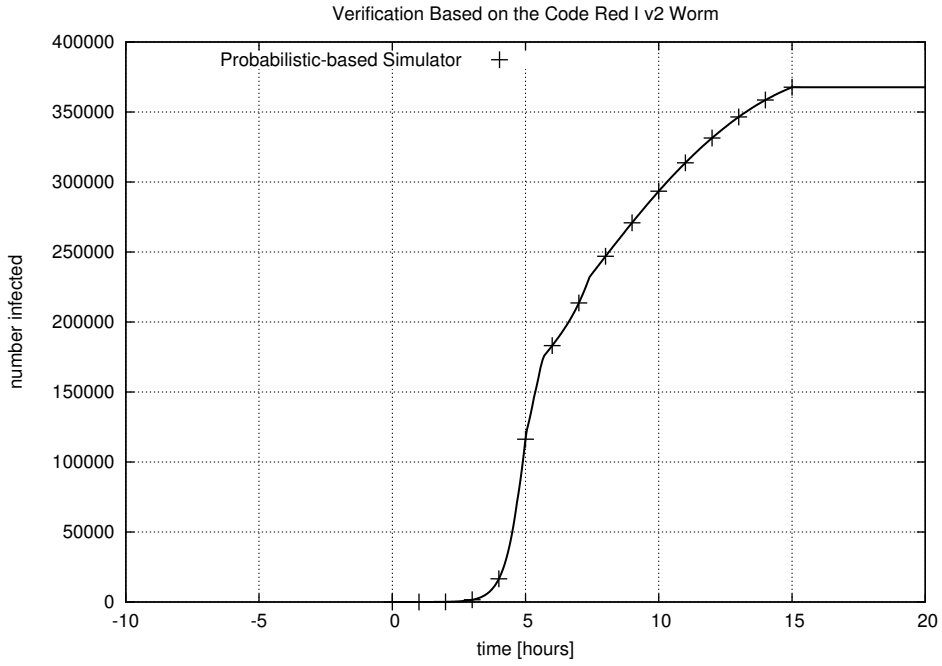

Figure 4.5: Probabilistic-based Simulator Predicting Number of Compromised Systems

[2]©2001 The Regents of the University of California. Courtesy University of California. All Rights Reserved.

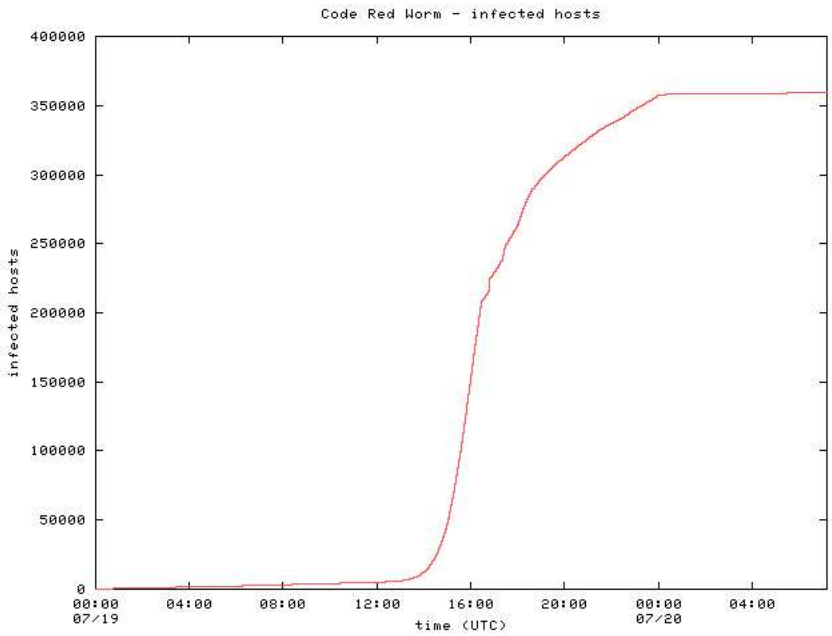

Figure 4.6: CAIDA, Code Red Visualization, 2001

The result of the probabilistic-base simulator, like the epidemiological model before it, matched the recorded data plotted in Figure 4.6 from [4] in terms of the number of infected machines over time. The simulator exhibited a sharp discontinuity not immediately observed in Figure 4.6. This was due to abruptly varying the parameters α and λ during the course of the simulation. It was likely that gradually varying the parameters would have yielded a better result had this data been available. However, the simulated data fit recorded data from the propagation of the Code Red worm [4]. Furthermore, the simulator proved accurate during the early period of the infection cycle which was important for the early detection of network worms.

Comparisons of the epidemiological model to the probabilistic-based simulator were made to further verify the simulator. Figure 4.7 shows the propagation of the Hong Kong

Flu as computed by the model and the simulator along with actual CDC data. The results from the model and simulator match precisely with a small perturbation at the height of the infection cycle where the former predicted the deaths of 5.28 individuals more than the latter. The difference proved very small compared to the peak fatalities of 190 individuals per the CDC data in a population of roughly 7.9 million people.

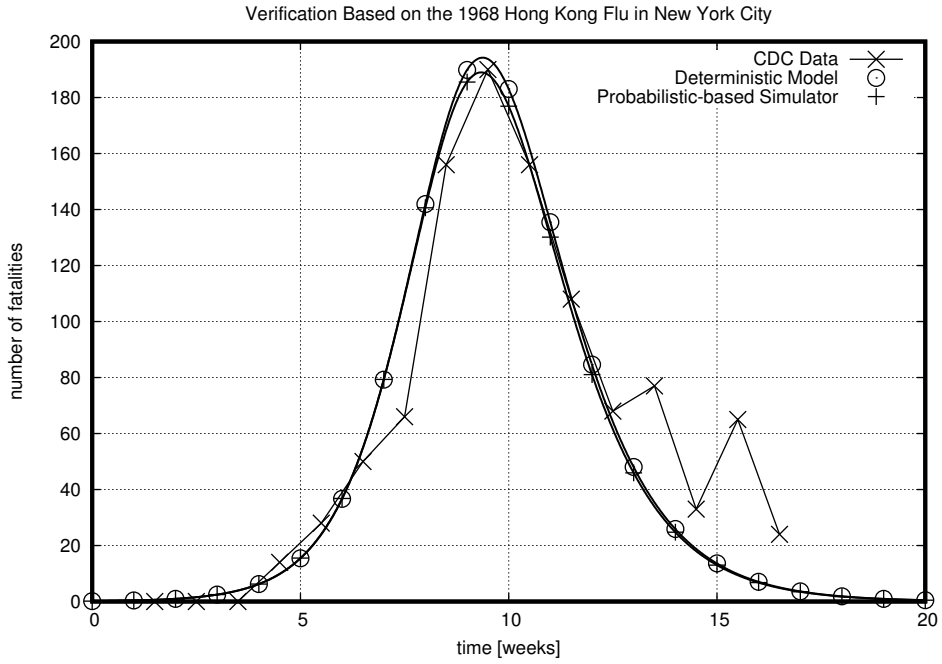

Figure 4.7: Model and Probabilistic-based Simulator vs. CDC Data

Figure 4.8 gives the spread generated by the model and the simulator for the Code Red I v2 worm. The data from both the model and the simulator matched exactly through the first five hours of propagation whereupon the simulated spread lagged that computed by the model by 54.6 minutes. The temporal difference in propagation resulted from abruptly varying the parameters in the simulation and the model; a more gradual change in the parameters

would have likely provided smoother and matching curves had such been available. The lag, however, resulted in an error of only 6.5% out of the total infection cycle duration of 14 hours. Moreover, the model and simulator remained in agreement through the early part of the infection cycle which proved important for the early identification and response to network stealth worms.

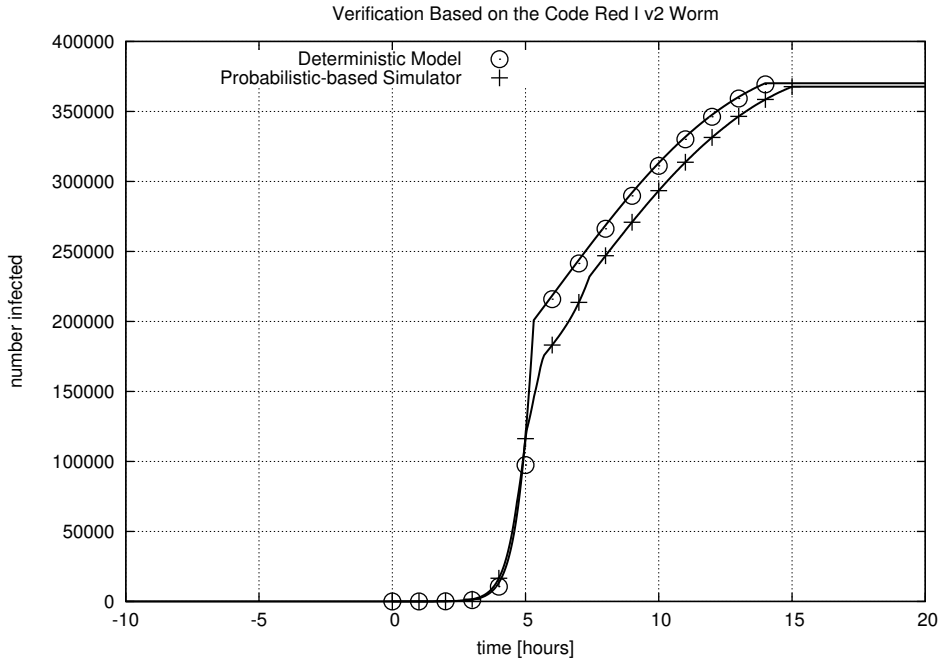

Figure 4.8: Model and Probabilistic-based Simulator for Code Red Worm

Further investigations compared the performances of the model and the simulator on campus and corporate-sized networks. These networks represented the target of this research and deployment for Rx. The model and the simulator set the population demographics with a total of 20,000 machines: 19,999 susceptible machines, 1 infected machine, and no recovered or immune machines. The model and the simulator added a network stealth worm into the

population with a probing rate of 1 scan per minute and a population recovery rate of zero. Figure 4.9 presents the result of this trial.

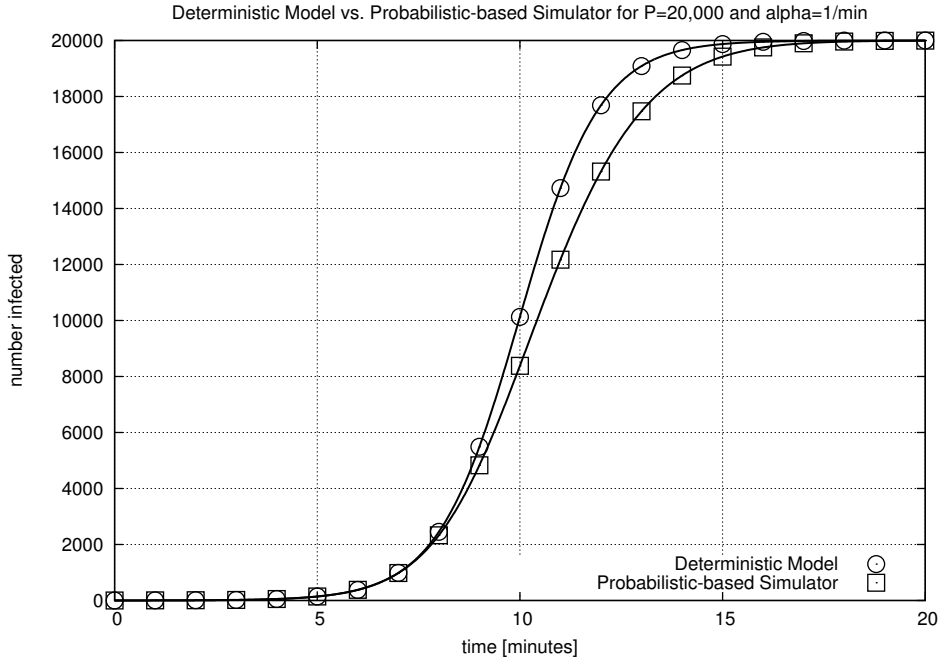

Figure 4.9: Comparison of Model and Simulator

Figure 4.9 shows that the simulation and model were in agreement with the former lagging slightly toward the end of the infection cycle by less than 1.2 minutes. This represents a 7.4% error based on the duration of the infection cycle at 16 minutes. Additionally, the traces for the model and simulator lay in agreement for the first part of the infection cycle which proved important for the early detection and reponse to network stealth worms. The simulated data was processed by the BMM described in Section 3.3.2. The BMM estimated the effective probing rate of the simulated worm as 0.96 probes per minute which was reasonably close to the actual probing rate of 1 probe per minute.

The minute differences between the epidemiological model and the probabilistic-based simulator resulted from subtleties in underlying dynamics that drove the disease propagation process. This motivated the subsequent figures. These trials used a total population of 1,000 individuals: 999 infected individuals, 1 infected individual, and no recovered or immune individuals. The investigations considered a worm with a transmission coefficient of 2 probes per hour and a recovery rate of zero.

Figure 4.10 shows the 2-sigma error bars to indicate the top 95th percentile of trials for the probabilistic-based simulator. The figures show that low variation occurs at the beginning and end of the infection cycle with high variability in between. Such behavior resulted from the treatment of the transmission coefficient as a probability.

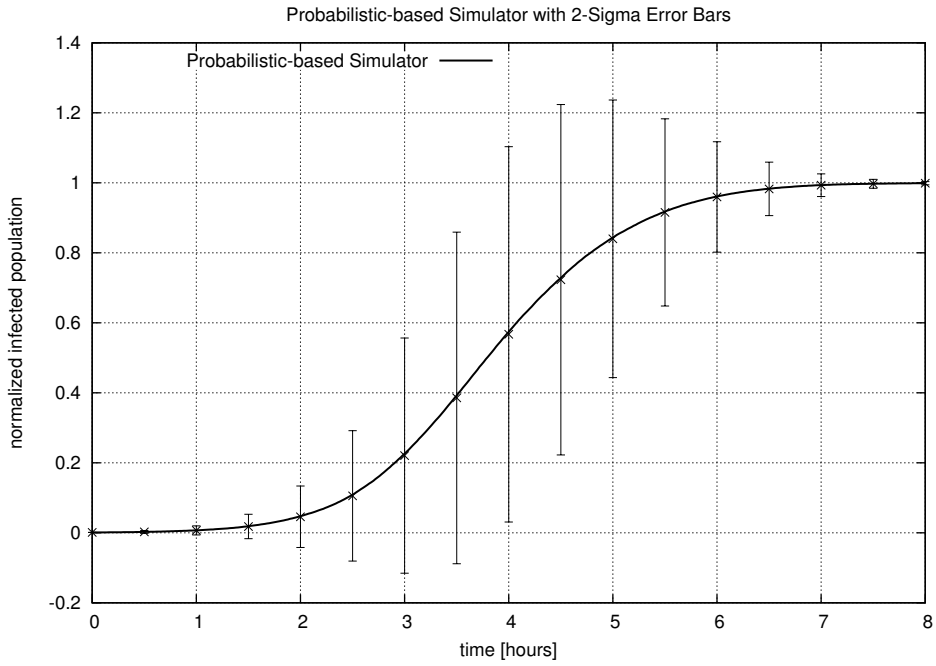

Figure 4.10: Probabilistic-based Simulator with 2-Sigma Error Bars

Low variability occurred early in the infection cycle because nearly all trials had a low number of machines infected at this time and thus could not produce a significant amount of change from trial to trial. The same argument held for the latter stage of the infection cycle where nearly all trials had few vulnerable machines remaining, preventing the mass of infected machines from creating significant change. The opposite occurred for the middle portion of the infection cycle where trials began the period with varying numbers of infected and vulnerable machines which resulted in significant changes leading to the end of the infection cycle.

Figure 4.11 demonstrates the variability of worm propagation at different points in the infection cycles. It shows one hundred individual trials for the probabilistic-based simulator, the simulation mean, and the result from the epidemiological model. The simulator mean lagged the model result as in previous investigations. More interestingly, the graph shows significant outlier trials that serve to lower the simulation mean. For example, several trials resulted in complete saturation of the population in 5 hours while a few others only managed to infect about 15% of the population by the same point in time.

The previous graphs indicate that the variability during the early portion of the infection cycle played a dominate role in the overall infection behavior. Figure 4.12 shows this for the probabilistic-based simulator. The experiment changed the number of initially infected individuals to 20% of the total populalation. This investigation thus used 200 infected individuals and 800 vulnerable individuals. The figure shows 100 individuals trials of the probabilistic-based simulator, the mean of the simulator, and the result of the epidemiological

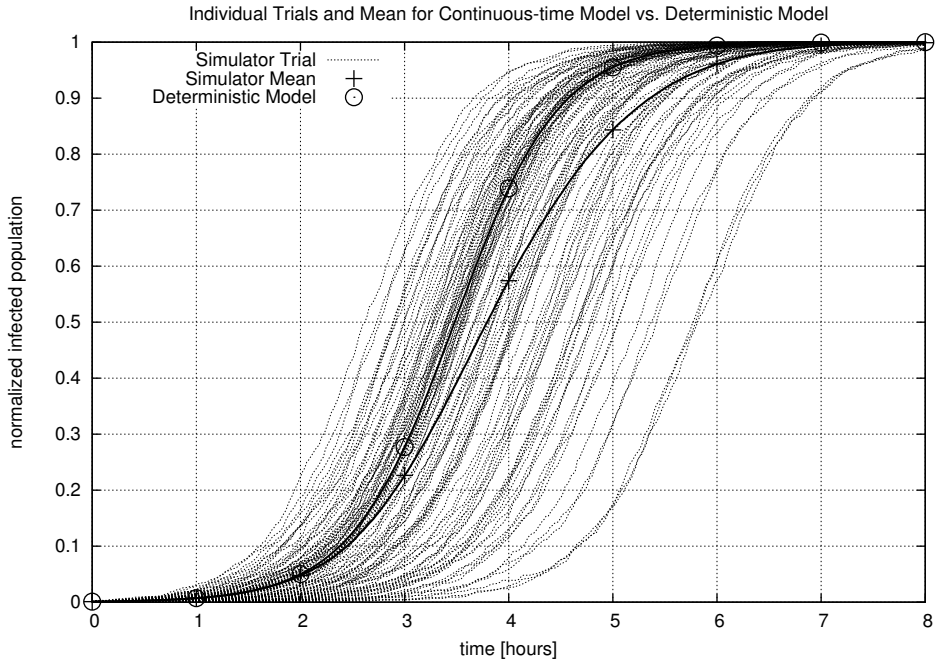

Figure 4.11: Individual Trials and Mean for Probabilistic-based Simulator

model.

Figure 4.12 demonstrates that the early part of the infection cycle dominates the behavior and duration of the overall infection cycle. The figure shows the 100 individual trials of the simulator to lie within a tight band with the largest part of the band centered about the middle of the infection cycle by time. This resulted in a mean for the probabilistic-based simulator that exactly matched the result from the epidemiological model.

The previous figures and discussion show that the variation in the probabilistic-based simulator compared to the epidemiological model was reasonable. The fundamental differences in the model resulted from variations early the infection cycle that produced profound impact later in the infection cycle. The subtle differences in propagation generation dy-

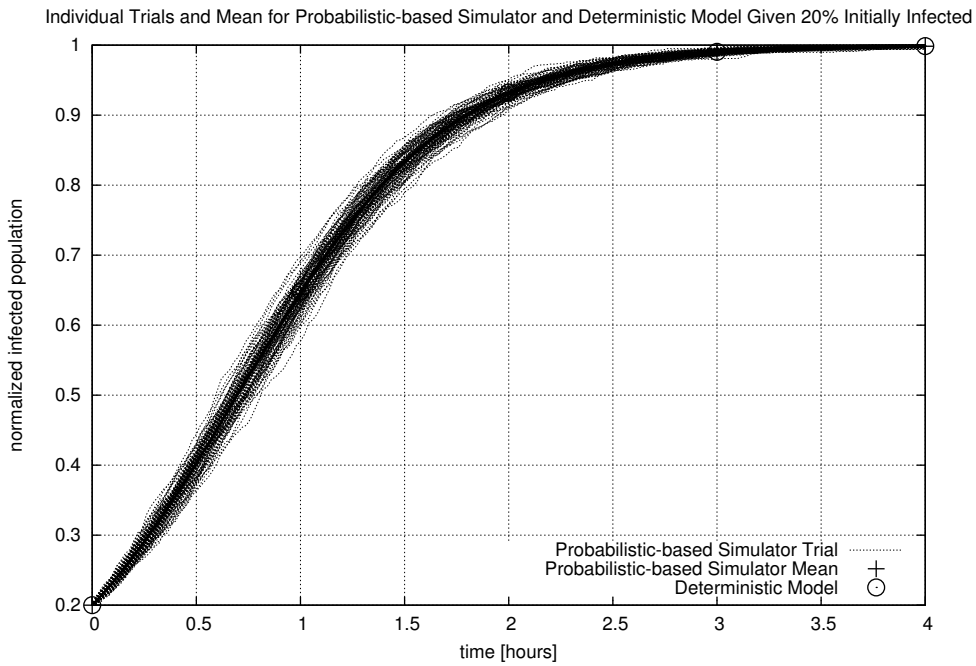

Figure 4.12: Probabilistic-based Simulator for 20% Initially Infected

namics, deterministic equations for the model and probability of events for the simulator, accounted for these variations. Deterministic models use a set mathematical structure to approximate a the random process of disease propagation [77] [78]. Stochastic approaches depend on chance variation of the individual, or conditional probability, to generate disease spread data [79] [77]. Figures 4.11 and 4.12 suggest that the success of a disease early in the infection cycle strongly influences its behavior later in the infection cycle. These results agreed with findings in worm propagation in computer security that stated the time to compromise the first 10,000 hosts throughout the Internet governed the total infection duration for the worm [7].

4.3.3 Java-based Worm

An epidemiological model verified the implementation of the Java-based worm. This process considered the spread of a worm through a population of 10 machines in order to evaluate the implemented worm for the testing scenario described by Section 3.5 and used in Section 5.5. At the start of the test, one system was infected while the remaining nine machines were vulnerable. A network worm with a probing rate of 1 probe every thirty seconds spread through the population.

The propagation of the Java-based worm was recorded using the test-network described in Section 3.5; 10 machines formed the isolated network through which the worm spread. An additional machine collected reports of infection due to the Java-based worm. The propagation data from the implemented worm was compared to a stochastic, compartmental SIR model that treats the processes of infection and recovery as binomial events. Stochastic models are used for relatively small populations, especially those of household size in the biological paradigm [78].

Equations 4.1 through 4.3 provide the definition of a stochastic SIR model as defined and used in [65] to generate realistic network worm traffic. These follow from the deterministic SIR model given by Equations 2.6 to 2.8 in Section 3.3.2. The number of infected machines in a time interval $\frac{1}{numsteps}$ is given by the binomial distribution X where $X = Bin(a, b)$. The parameter a is the number of susceptible machines while $b = I * \alpha * \frac{1}{numsteps}$. The number of recovered machines in a time interval is similarly defined with $Y = Bin(c, d)$ where c is the number of infected machines and $d = \lambda * \frac{1}{numsteps}$.

$$s_z = s_{z-1} - X_{z-1} \tag{4.1}$$

$$i_z = i_{z-1} + X_{z-1} - Y_{z-1} \tag{4.2}$$

$$r_z = r_{z-1} + Y_{z-1} \tag{4.3}$$

Figure 4.13 gives the propagation data for the implemented Java-based worm and the spread computed by the epidemiological model. The stair-step propagation pattern of the worm resulted from the spread of a malcode with a set probing interval of 30 seconds on a small, homogeneous population of computers. The pre-determined probing interval caused the number of infected individuals to change very close to that interval. A small, and homogeneous population did not create much variation to shift the probing intervals of different worm instances. The actual and expected propagation patterns matched exactly with very minute differences occurring in the middle of the infection cycle. The graph shows that two data sets indicated the complete infection of the population of 10 machines occurred about four minutes after the release of the worm.

4.4 Summary

This chapter verified the tools used to exercise Rx as well as the bio-mathematical model incorporated into the system itself. Section 4.1 reviewed the objectives for the verification of Rx and the tools used to evaluate the system.

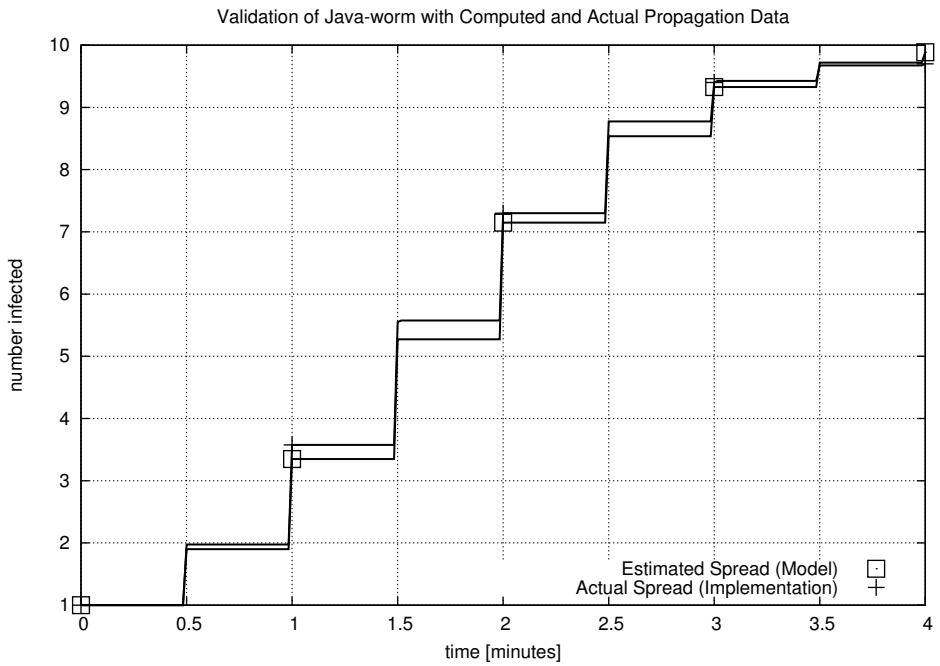

Figure 4.13: Binomial Model vs. Observed Test-Network Worm Spread

Section 4.2 presented detailed descriptions of the biological disease Hong Kong Flu and the digital contagion Code Red I v2 worm for use in verification testing. Section 4.3 presented the results of these trials. The SIR model implemented by Rx and the simulator used to generate worm propagation data to investigate the system were verified against the Hong Kong Flu and the Code Red I v2 worm. The results for the model and the simulator appeared in Sections 4.3.1 and 4.3.2. Both entities accurately estimated the propagation of the epidemic events under test given known parameters and recorded spreading data. This verified the model and the simulator in both the biological and digital paradigms.

Section 4.3.3 verified the implementation of the Java-based worm used to exercise Rx on a test-network. The Java-based worm was tested against spreading data computed by

a stochastic epidemiological model. The observed propagation of the Java-based worm on the test-network and the worm spread estimated by the model matched with only minute differences.

The next chapter presents results on the efficacy of Rx. It uses the worm propagation components verified in this chapter to analyze the effectiveness of Rx given incidence or symptom-based detection data given varying network sizes and malware probing rates. The discussion ultimately shows that both the approach and Rx satisfy the metrics established in the previous chapter.

Chapter 5

Validation

This chapter presents the results and analysis which establish the efficacy of the Rx system and the underlying conceptual approach. Section 5.1 briefly discusses the validation objectives while Section 5.2 establishes the evaluation scenario used to investigate the effectiveness of Rx.

Three parts comprise the complete validation results of Rx consisting of node, framework, and implementation testing. Node testing in Section 5.3 evaluates the sensitivity of Rx to malware with a wide range of probing rates within varying sizes of computer networks. The discussion continues to evaluate the system given incidence or anomaly data. Framework testing in Section 5.4 investigates the Rx framework through the simulated spread of the Code Red I v2 worm. Finally, Section 5.5 presents the results from the implementation testing of Rx on a test-network in response to a Java-based worm. These sections show that Rx to met the evaluation metrics set by Section 3.7.

Section 5.6 ends this chapter by summarizing and analyzing the key results and implications. The findings ultimately demonstrate the validity of the Rx system in identifying,

characterizing, forecasting, and controlling network stealth worms on computer networks. Moreover, the chapter shows the validity and advantages of applying epidemiological techniques, specifically bio-mathematical modeling and demographic analysis, to the *real-time* and *proactive* defense of computer networks against surreptitious, self-propagating code.

5.1 Validation Objectives

Validation demonstrated the efficacy of the approach and of Rx in defending computer networks against surreptitious, self-propagating code. Specifically, this process ensured that the system achieved the problem statement defined in Section 1.1 by satisfying the metrics established in Section 3.7. Validation testing separately evaluated an individual Rx node and then an Rx framework composed of a group of nodes. Finally, an implementation of Rx on a small test-network received attention to substantiate both the system and the approach on a physical network.

5.2 Evaluation Scenario

The validation trials focused on the effectiveness of Rx as deployed on campus or corporate-sized networks of 20,000 machines. This represented the target deployment scenario for Rx per Section 3.2.2. The quarantine response of Rx was assumed to have been enabled by the automated manipulation of access control rules on routers, firewalls, and network filtering devices. The system accepted incidence alerts and anomaly reports on worm propagation

from malware sensors that were sensitive to the spread of the network stealth worm under test as discussed in Section 3.2.1.

The simulated and implemented network stealth worms that exercised Rx used a slow, random-scanning spreading strategy. Other propagation methods were available to network stealth worms, but the slow, random-scanning spreading mechanism posed the most immediate threat. This established the correctness and effectiveness of epidemiological concepts as a real-time network defense tool. Epidemiological models are readily extensible to describe more complex contact structures for other network stealth worm propagation methods [52] [53]. The probability of detection and partial deployment were considered to capture the surreptitious nature of the stealth worm.

Tests validated the system in Section 5.3 followed by the framework in Section 5.4 and finally culminated with implementation on a test-network in Section 5.5. Simulation allowed for the evaluation of Rx under a controlled environment and with large-scale network deployment. Both of these scenarios would have proved, difficult if not impossible, to obtain through physical implementation. The implementation of Rx validated the approach and the system on a test-network.

5.3 Node Testing

This section validates Rx by considering a single node deployed on a network. Section 3.3.3 defined an Rx node. The following analysis explores the sensitivity of an Rx node to broad ranges in the population size and malware rates and evaluates the ability of the

system to detect a worm early in the propagation cycle. The discussion continues with investigations that establish the capability of an Rx node to achieve the validation metrics given both incidence and symptom-based data and, additionally, explores the effectiveness of the system given partial deployment and low probability of detection.

5.3.1 Sensitivity over Network Size and Malware Scan Rate

Rx effectively identified and characterized epidemic trends in incidence data generated by broad ranges of both network size and malware probing rates. This specifically achieved the metrics M_{1A} and M_2. The results presented below validated the sensitivity of the approach and Rx for a wide range of parameters while subsequent sections explore these and other metrics by considering the propagation of a set worm in a defined network size.

The probabilistic-based simulator generated networks consisting of ten, one hundred, a thousand, ten-thousand, and one-hundred thousand vulnerable machines and computed the propagations of stealth worms with probing rates of 1, 2, 5, and 10 probes per unit time. Rx received this propagation data at the end of the infection cycle and was evaluated in its ability to identify an epidemic trend in the data with a confidence measure at or below -1 and to accurately estimate the probing rate. Figure 5.1 shows the confidence measures computed by Rx as a function of population size versus probing rate.

Rx demonstrated its ability to identify the presence of surreptitious, self-propagating code over a broad range of network sizes and malware probing rates as shown by Figure 5.1. The graph indicates that Rx satisfied metric M_{1A} by identifying worm propagation patterns

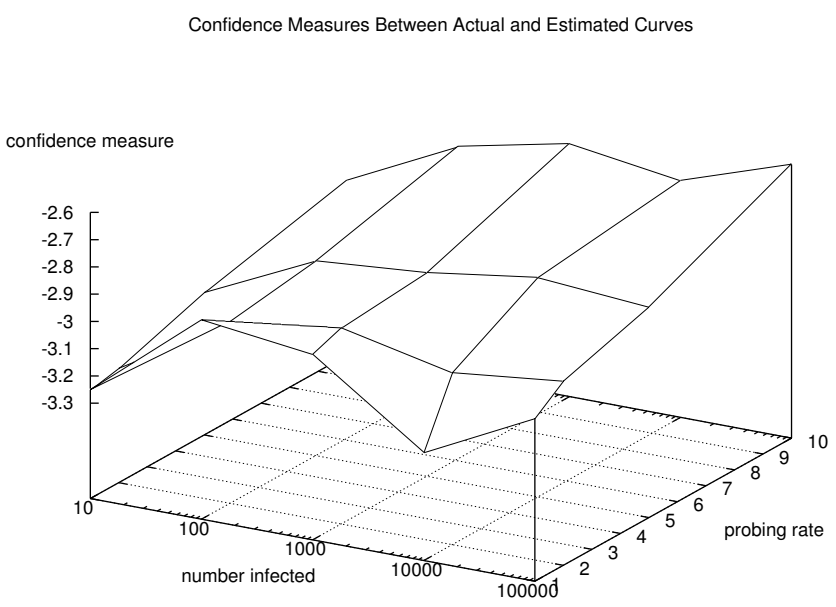

Figure 5.1: Confidence Measures Between Actual and Estimated Curves

contained in incidence data; the system did so by computing confidence measures below -1. The system produced the strongest confidence measure for epidemic spread at lower probing rates. This resulted from a static window in time: a faster probing rate generated a sharp, nearly vertical propagation pattern which proved more difficult to fit as compared to more slowly increasing curves from lower probing rates. This could have been addressed by re-scaling the time window.

Figure 5.2 shows the ability of Rx to compute the probing rate from incidence data as a function of population size and malware probing rates. The graph gives the magnitude of the percent error between the actual and estimated probing rates. The worst estimates resulted in a magnitude percent error of 8% at a probing rate of 1 probe per unit time

and populations of 10 and 100. This yielded computed probing rates of 0.92 probes per unit time which were both reasonably close to the actual probing rates of 1 probe per unit time. Equation 3.12 weighted the early portion of the infection cycle which contributed to the magnitude percent errors that considered the entire duration of malware spread. The system effectively estimated the probing rate of the instances of self-propagating code commensurate with metric M_2.

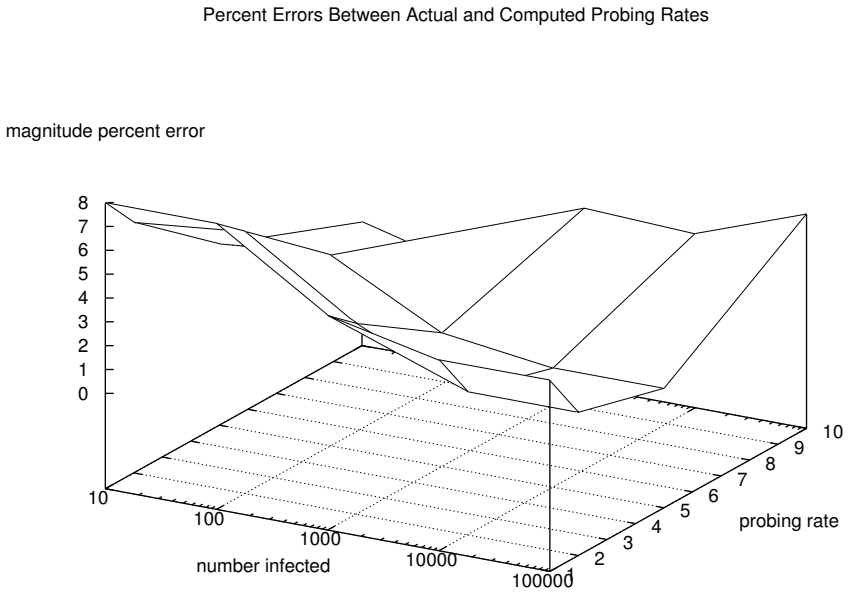

Figure 5.2: Percent Errors Between Actual and Computed Probing Rates

5.3.2 Early Warning and Identification

The expedient detection and response of self-propagating code is crucial for containing the malware. Researchers have shown that the first portion of the infection cycle heavily dom-

inates the overall time to saturate the population [7]. The results given below established the effective response of Rx to the initial propagation of a network stealth worm.

The probabilistic-based simulator created networks with ten, one hundred, a thousand, ten-thousand, and one-hundred thousand vulnerable machines and added a network stealth worm with a probing rate of 2 probes per unit time. The simulator triggered Rx at predetermined points in the infection cycle given by the number of machines infected. The simulator triggered Rx to specifically investigate the performance of the system at these points. The trigger set consisted of one through 10 infected machines in steps of one in addition to 10 through 50 machines infected in steps of five.

Rx was evaluated in its ability to identify a worm. The system did so by finding an epidemic curve that matched the propagation pattern of the worm as indicated by a confidence measure at or below -1. Rx was also measured in its capability to accurately estimate the probing rate. Figure 5.3 shows the confidence measure estimated by Rx as a function of the population size versus the number of infected machines.

Rx established its ability to identify epidemic trends with a confidence measure at or below -1 as shown in Figure 5.3. This temporally validated metric M_{1A} since probing rate and time were directly related. The value of the confidence measure fell as the population increased indicating a stronger correlation to an epidemic curve. This was very true for the trials involving populations of 10 and 100. The spread of the worm through the relatively small populations did not generate a law of mass action effect and, consequently, did not fit an epidemic curve as well as those trials with larger populations. Despite the small

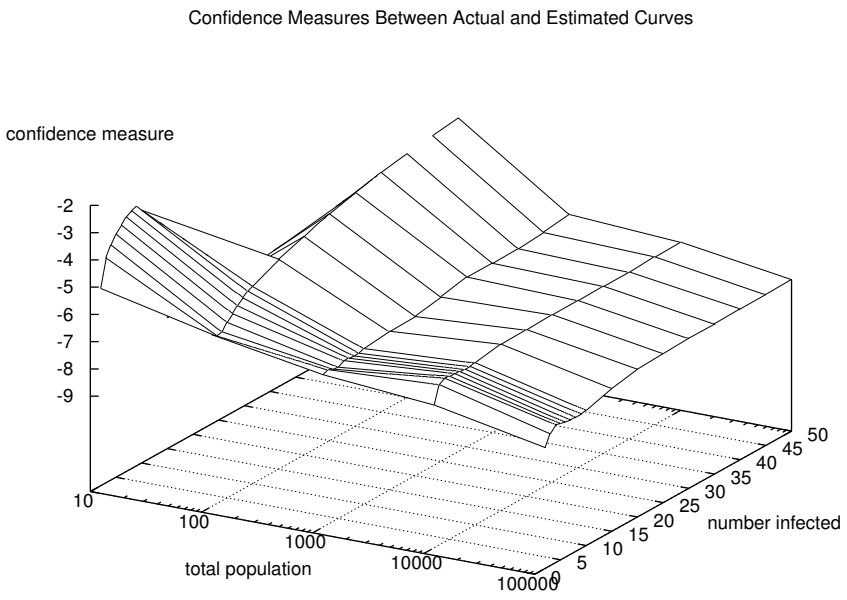

Figure 5.3: Confidence Measures Between Actual and Estimated Curves

populations of 10 to 100 machines, Rx still identified epidemic propagation trends with a confidence measure under -2. Also, the confidence measure tends to increase slightly as Rx received more infection data; this resulted from weighting placed on the early portion of the infection cycle in Equation 3.12.

The system produced reasonable estimates of the probing rate at different times during the infection cycle. Figure 5.4 shows the ability of Rx to compute the probing rate from incidence data as a function of the population size and the number of infected machines. The plot gives the magnitude of the percent error between the actual and estimated probing rates. The worst estimates occurred with only two to three infected machines. The poorest probing rate computation resulted in a magnitude percent error of 12.5% for the population

of 100,000 with two infected individuals. This resulted because two machines in this case were insufficient to generate a propagation pattern that closely matched a spreading pattern computed by the bio-mathematical model. The estimated probing rate for this was 1.75 probes per unit time which was reasonably close to the actual probing rate of 2 probes per unit time.

Ignoring the data points for the infection of the first three machines, the average magnitude percent error was 2.37% for an average estimated probing rate of 2.05 probes per unit time. This was very close to the actual probing rate of 2 probes per unit time. The results indicated that Rx identified the presence of an active worm and produced reasonably accurate estimations of the probing rate when as few as two machines were infected. The system resulted in more accurate probing rates with percent errors around 2% once four machines were infected. The increase in percent error as the number of infected inviduals rose resulted from the weighting given to the initial portion of the infection cycle from Equation 3.12.

5.3.3 Incidence-based Response

The effectiveness of Rx with incidence-based data for malware detectors was investigated through simulation. The simulator computed the spread of a stealth worm with a probing rate of one probe per minute through a campus or corporate-sized network of 20,000 vulnerable machines. The experiment also simulated the spread of a jumping executable and generated false-alarm data on the same network to compare the sensitivity of Rx to network

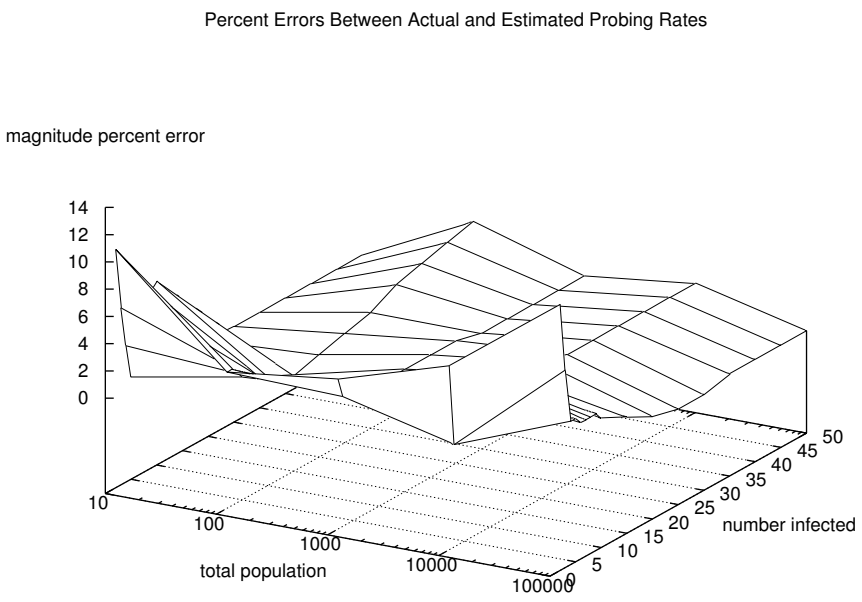

Figure 5.4: Percent Errors Between Actual and Estimated Probing Rates

stealth worms versus the other data sources. A jumping executable was a special case of a network worm in which the malcode migrated from one machine to another instead of spawning new copies [23]. The jumping executable was simulated with a probing rate of one probe per minute to match the stealth worm. The false-positive anomalies were generated from a uniform distribution at the same probing rate as the stealth worm and jumping executable. A simulated Rx node received separate incidence data on these three sources.

The simulated environment afforded Rx the ability to manipulate rules for switches, routers, and network filtering devices in order to remove a specific machine from the network. This facilitated the quarantine response. The analysis evaluates the sensitivity of Rx as the worm achieves greater saturation within the population. However, the simulation set the

trigger point for an automated response at a minimum of 50 infected individuals out of the total population of 20,000. Section 5.3.2 suggested that Rx could have triggered earlier, but the choice of 50 infected individuals accounted for implementation margin and the reluctance of security professionals to aggressively deploy automated defenses.

M_{1A}: Identify Malware Through Incidence Data

Figure 5.5 shows the identification confidence values for (i) a network stealth worm, (ii) a jumping executable, and (iii) false-positive anomaly data. The differences in temporal characteristics between the three data sources late in the infection cycle were accounted for by searching over a large range of possible probing rates.

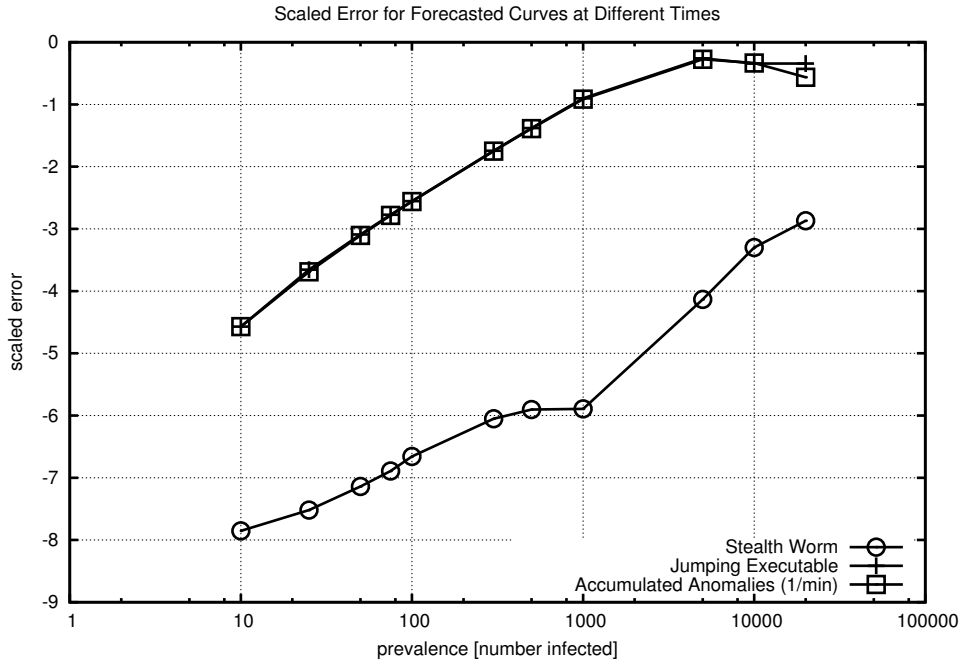

Figure 5.5: Scaled Error for Forecasted Curves at Different Times

Figure 5.5 demonstrated the ability of Rx to identify a network worm and distinguish

it from the other traffic sources. The confidence measures between the network worm and the other traffic sources maintained a consistent and substantial separation as the worm spread through the population. The confidence measure increased as more of the population became infected suggesting less correlation with epidemic behavior. This resulted from the natural differences in dynamics between the model and simulator as well as the weighting assigned to the early part of the infection cycle per Equation 3.12.

Moreover, Figure 5.5 showed the ability to detect the worm *early* in the spreading cycle with as few as 10 infected machines. Thus, this early detection enabled a rapid response to the network worm for the effective control of the malcode.

M_2: Characterize Malware

Next, Rx sought to characterize the worm. Figure 5.6 gives the estimated probing rate, α, as a function of the number of hosts that became infected over time. Rx accurately estimated the probing rate to be about 1 probe per minute which matched the actual simulation value. The probing rate was used to judge the virulence of the worm, the saturation of the worm in the network, and the necessary response time for containment [2].

The estimated probing rate along with demographic data facilitated the forecasted spread of the worm. Rx accurately identified the worm and estimated the probing rate with as few as 10 infected individuals. Figure 5.7 compares the actual infection data from simulation and the forecasted curve generated once 50 machines were infected. This occurred in just under four minutes from the release of the worm.

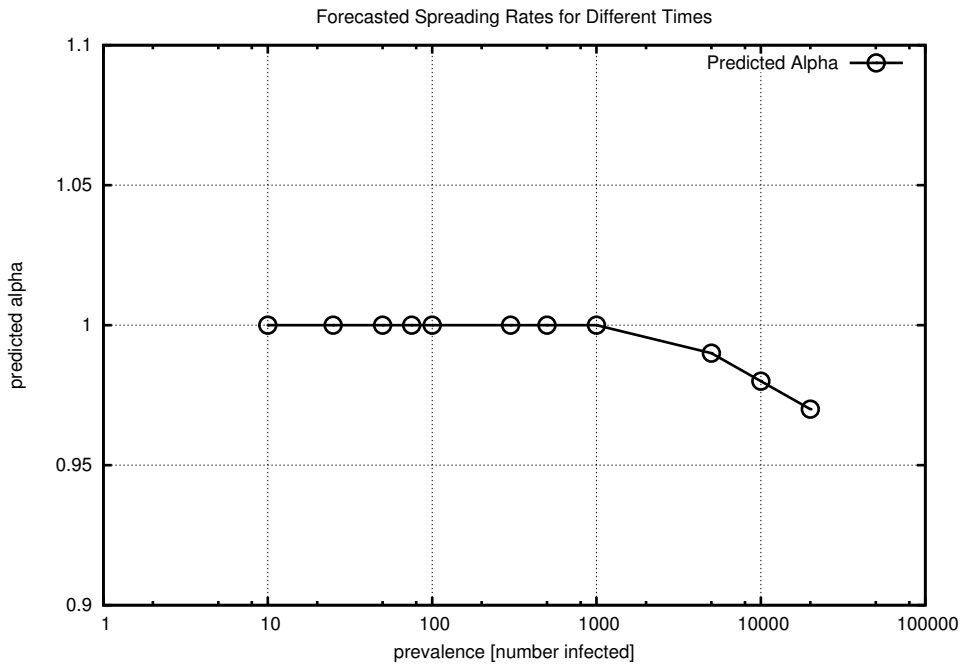

Figure 5.6: Forecasted Spreading Rates for Different Times

The forecasted curve in Figure 5.7 gives a reasonable approximation of the temporal spread of the worm. The forecasted curve matched the simulated data through the first 15 minutes of the infection and then lead the simulated curve during the latter portion of the infection cycle. The discrepancies in the curves resulted from the differences in the underlying dynamics between the simulation and model and the weighting placed upon the early portion of the infection cycle per Equation 3.12. Such prediction on the temporal spread and saturation of the worm could have helped security professionals understand the virulence of the worm and direct further response and recovery resources.

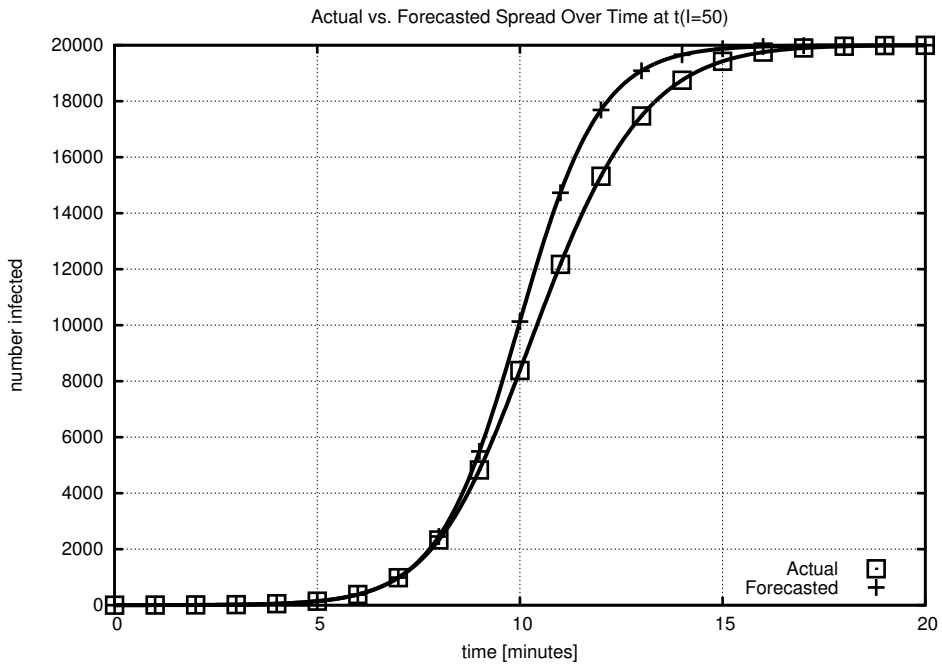

Figure 5.7: Actual vs. Forecasted Spread Over Time at t(I=50)

M_3: Effect Temporal Lag on Malware Spread

Rx identified the worm in under four minutes after the worm began propagation as the malcode exceeded the defined trigger point of 50 infected machines and a confidence measure under -1. Rx then manipulated network-device rules to quarantine infected and vulnerable machines in *advance* of the spreading worm. Figure 5.8 shows the spread of a network worm given (i) no response by Rx, (ii) forecasted spread assuming no response by Rx, (iii) a perfect advanced quarantine response using Rx, and (iv) an imperfect advanced quarantine response with a 20% error using Rx.

Figure 5.8 indicates that a quarantine response through demographic analysis profoundly contained the spread of the network stealth worm. A perfect demographic response,

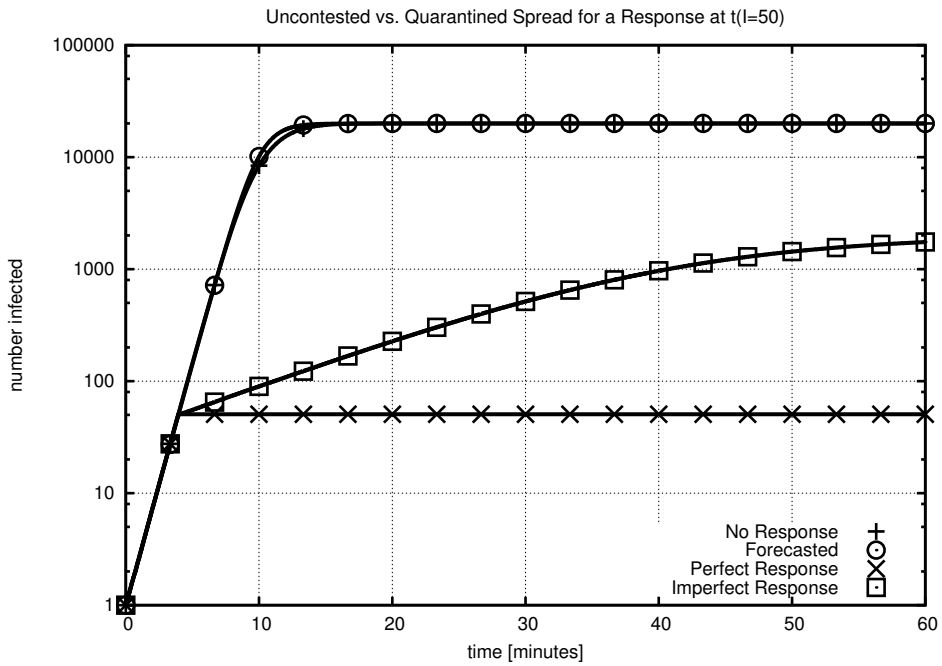

Figure 5.8: Uncontested vs. Quarantined Spread for a Response at t(I=50)

of course, completely and instantly curbed worm propagation. Errors in demographic data resulted from the addition of new machines or frequent changes in system configurations.

Rx introduced a significant amount of delay into the effective propagation rate of the network stealth worm. Table 5.1 shows that the system caused the worm to experience a temporal lag of 6.5 minutes to infect 100 machines and over 1.5 hours to infect 2,000

Table 5.1: Delay in Number of Compromised Hosts from Epidemiological Response

Number Compromised	**Delay (mins)**
100	6.50
500	23.33
1,000	33.60
2,000	92.84

machines. Such delays could have provided additional time to broadcast security alerts and data, manually deploy countermeasures, and trigger other defensive responses.

M_4: Enhance Average Survival Rate of Individuals

Rx ultimately resulted in a significant increase in the survival of individual machines under attack by a network stealth worm versus a scenario lacking a response. Figure 5.8 demonstrates the ability of Rx to control the spread of a network stealth worm through advanced quarantine. Rx isolated infected and vulnerable machines in advance of the spreading worm. The worm achieved full saturation of the population of 20,000 machines in about 15 minutes without any response. A perfect demographic response prevented the compromise of over two orders of magnitude of machines while even a demographic response with 20% error preserved about an order of magnitude of machines.

Partial Deployment and Probability of Detection

The simulated scenario continued to consider the effects of partial deployment and the reduced probability of incidence detection on Rx. The simulation defined partial deployment as the probability of a networked host having malware detection sensor. The sensor software at each host was given a probabilistic chance to either detect the worm or else failed to do so through the life of the simulation. In partial deployment, a machine without a detector never identified itself as infected. The total probability of detection was given as the product of the probabilities of incidence detection and partial deployment. Figure 5.9 shows the effect of partial deployment and probability of detection on the survival rate of the population.

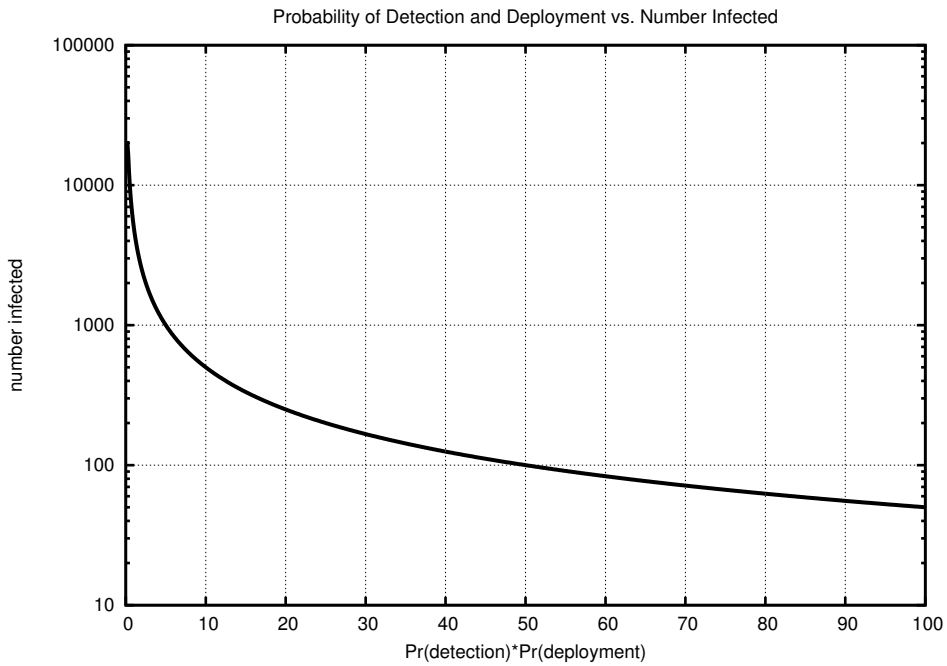

Figure 5.9: Probability of Detection and Deployment vs. Number Infected

Figure 5.9 shows that Rx experienced graceful degradation under times of reduced deployment and detection. A combined detection and deployment probability of 50% allowed the worm to double the number of compromised machines as compared to the perfect detection case. Probabilities less than 50% quickly resulted in increasingly higher levels of infection. A total probability below 5% indicated an ineffective defensive response. This data could have been used as a risk-based assessment in order to evaluate the cost of deploying and maintaining malware sensors versus the perceived risk of the threat of similar virulence. An implementing organization could have selected an acceptable number of machine losses due to worm compromise and used Figure 5.9 to determine the corresponding combined probability of detection for malware sensors and Rx deployment. This level of detection and

deployment would have come at a monetary cost to the organization based on the price and number of malware sensors and Rx nodes and the total cost of time required by employees to implement and maintain the readiness level.

5.3.4 Symptom-based Response

Reported host anomalies, like symptoms of a cold in the biological paradigm, only indicated the deviation from an established normal profile and often did not immediately suggest the subversion of the individual machine. Rx identified epidemic propagation trends from the reported symptoms by performing statistical inference on the anomaly data. This allowed Rx to detect and characterize a worm at the network-level and, through demographic analysis, issue a containment response.

The sensitivity of Rx to anomaly-based data was evaluated through the same simulation steps as in Section 5.3.3. The simulator computed the spread of a stealth worm with a probing rate of one probe per minute through a campus or corporate-sized network of 20,000 vulnerable machines. A simulated Rx node processed the propagation data. This trial directly assumed that host and network-based anomaly detectors were sensitive to the effects of a network stealth worm.

An anomaly generator, added to the network simulator, created anomalies using a uniform distribution with a mean of 2400 reports per hour. This number was commensurate with that observed by DShield at Virginia Tech. DShield was a distributed intrusion detection system and that collected and summarized reports of potentially malicious activity to

search for trends in the data [80].

As before, Rx possessed the ability to alter rules for switches, routers, and network filtering devices. This allowed the system to remove a specific machine from the network, thus facilitating the quarantine response. Fifty infected individuals and a confidence measure at or below -1 remained as the trigger point.

The analysis evaluated the sensitivity of Rx as the worm achieved greater saturation within the population. However, the simulation set the trigger point for an automated response at a minimum of 50 infected individuals out of the total population of 20,000. Rx could have triggered earlier per Section 5.3.2; however, the chosen threshold represented implementation margin and the reluctance of security professionals to aggressively deploy automated defenses.

M_{1B}: Identify Malware Through Symptom Data

Rx identified a stealth worm from symptom-based reports in the presence of false-alarm anomaly data. The simulator generated nearly two days worth of anomaly data; half the data was marked for training, the other half was used to exercise Rx. Rx processed over 23 hours of training data to establish a mean and standard error for the anomaly data. Figure 5.10 shows a one-hour sample window of the anomaly data.

The test anomaly data was divided into 20-minute windows corresponding to the infection cycle of the worm. Rx received the sum of the anomalies and worm infection data. Figure 5.11 shows the cumulative number of (i) worm infections only, (ii) reported anomalies

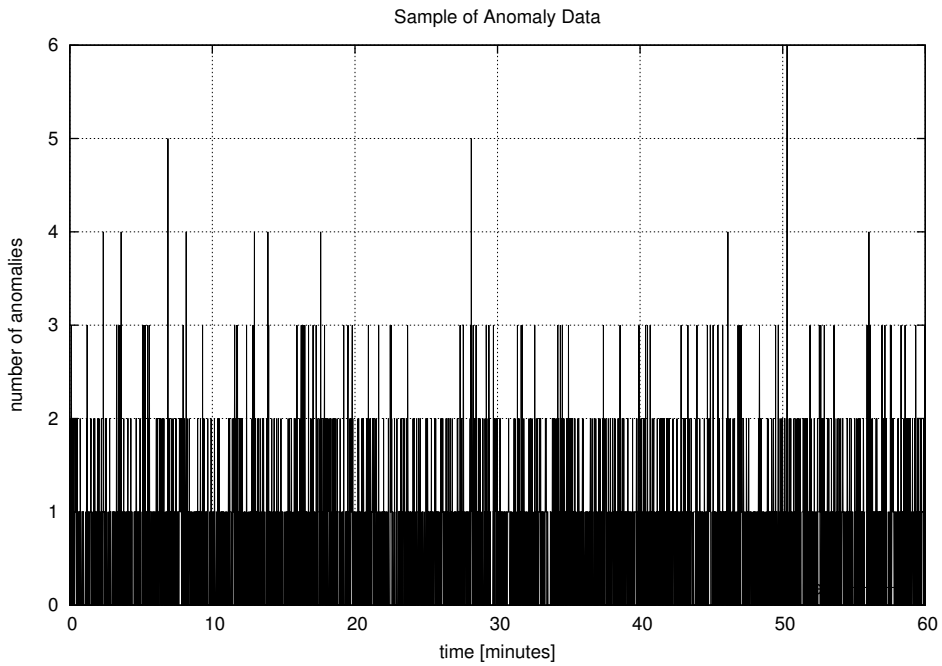

Figure 5.10: Sample of Anomaly Data

only, and (iii) the sum of the two. The infection curve for the worm lies substantially below the trace for the *total* cumulative number of reports.

Figure 5.12 gives the confidence measure for detection of a network worm using end-host infection data. The top curve corresponded to detection of the worm in the presence of anomaly data while the lower curve indicated the correlation values without anomaly data.

Figure 5.12 demonstrates the ability of Rx to infer active worm behavior in the presence of anomaly data. The values of the confidence measure for the detection in anomaly data was clearly higher than that without anomaly data due to the addition of noise to the bio-mathematical modeling process. However, Rx still detected the network stealth worm with a strong confidence value in the presence of anomalies. Moreover, Rx rejected pure anomaly

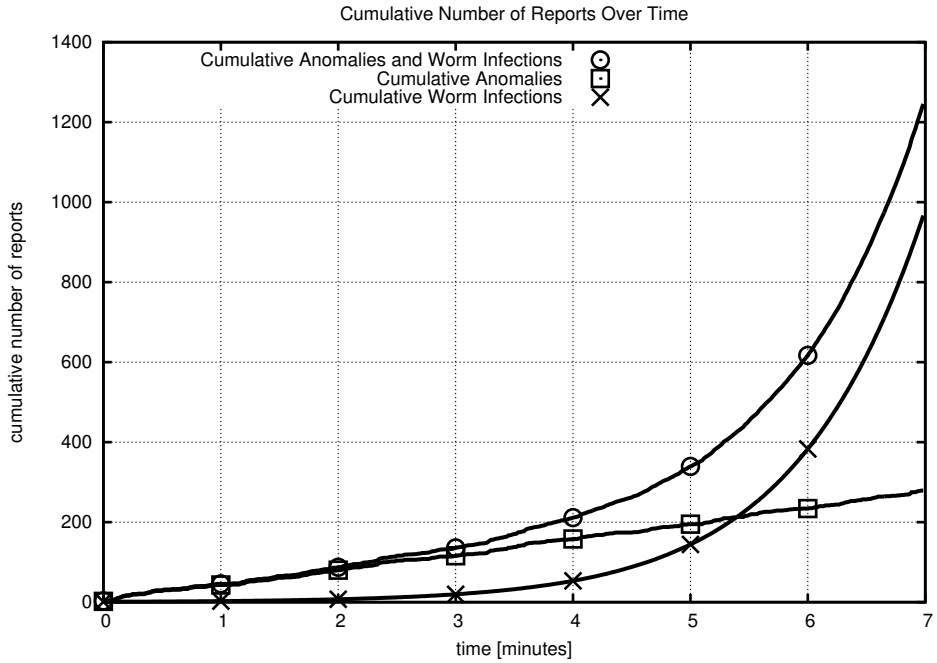

Figure 5.11: Cumulative Number of Reports Over Time

data from consideration by subtracting the cumulative average computed with the training data from the test data set.

M_2: Characterize Malware

Rx continued to predict the probing rate for the worm at different intervals in time given by the prevalence of the worm. This is shown in Figure 5.13. The results indicated that Rx predicted a probing rate from just over 1.12 probes per minute at the start of the infection with 50 machines compromised to under 0.96 probes per minute at full saturation with 20,000 machines infected. The estimate for the trigger point of 50 infected machines was computed at 1.06, closely approximating the actual probing rate of 1 probe per second. This confirmed the ability of Rx to accurately identify and characterize a worm in the presence of anomalies.

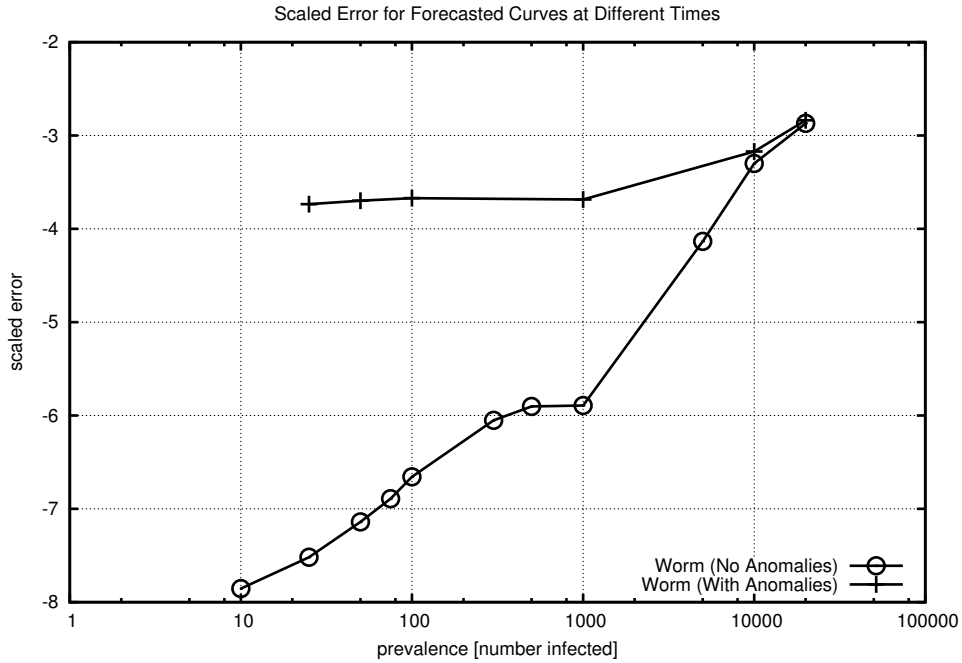

Figure 5.12: Scaled Error for Forecasted Curves at Different Times

This also substantiated the data from Figure 5.12 by showing the bio-mathematical model correctly correlated the worm infection data despite the addition of false-alarm anomaly reports.

The forescast of worm spread followed from the estimated probing rate and demographic data. Figure 5.14 compares the simulated infection curve against the projected worm propagation once 50 machines were infected. Rx identified this trigger condition in under four minutes from the release of the worm.

The predicted propagation of the worm shown in Figure 5.14 matched the trend of the actual, simulated curve. The actual and forecasted curves matched for the first six minutes of worm spread until the latter began to lead the former. The differences in dynamics between

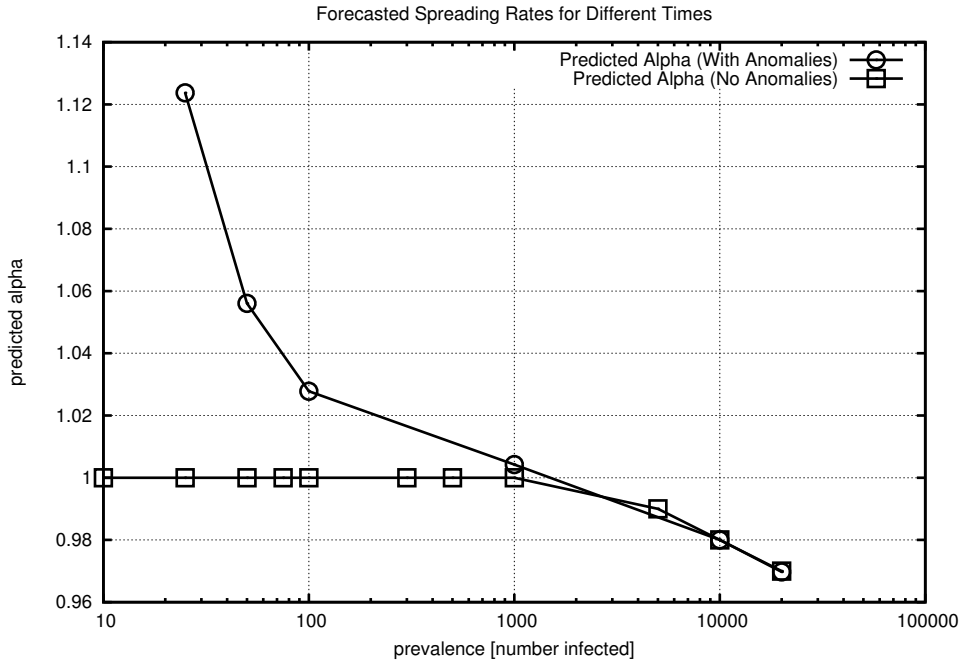

Figure 5.13: Forecasted Spreading Rates for Different Times

the model and the simulator accounted for some of the discrepancies in the two curves: the modeled data typically led the simulated data for a set probing rate. Additionally, Rx estimated a probing rate higher than the actual probing rate which contributed more error to the forecasted spread. The predicted curve still provided reasonable initial insight into the temporal spread of the worm which could have aided security professionals in understanding the virulence of the worm and to direct further response and recovery resources.

Rx identified and characterized the worm in the presence of anomaly data in the same time as the worm with incidence data. Metrics M_3 and M_4 for the symptom-based response investigating temporal lag and enhanced survival rate appear identical to that shown in Figure of 5.8.

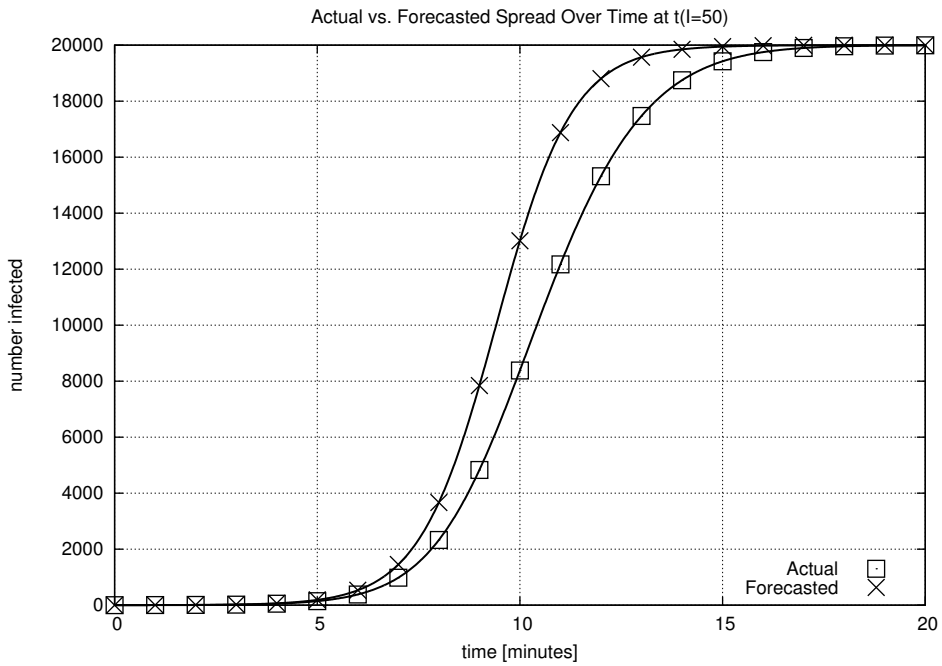

Figure 5.14: Actual vs. Forecasted Spread Over Time at t(I=50)

5.4 Framework Testing

The following discussion validates the framework defined by Rx as established by Section 3.3.4. The analysis below evaluates the capability of the Rx framework to achieve the validation metrics set forth in Section 3.7. In so doing, this section also considered the realistic challenges of large-scale operation to include partial deployment and reporting delays.

The investigations evaluate the effectiveness of the Rx framework by using simulated data for the spread of the Code Red I v2 worm. Section 4.2.2 explained this worm and Table 4.2 defined the parameters for Code Red. The simulator computed the spread of the worm with an initial probing rate of 5 scans per second in a vulnerable population of 575,000 machines in an IPv4 address space. The machines exhibited a recovery rate of zero at the

start of the simulation and then were assigned a recovery rate 0.03 five hours after the release of the worm. The delay in worm spread caused by bandwidth consumption due to probes was modeled by lowering the scan rate to 0.5 scans per second when 50% of the population became infected.

The simulator deployed Rx nodes across the address space at a rate of 50%; that is, Rx populated about 50% of the used address space. Data from top-level Rx root nodes managed by these organizations was shared with peer nodes with a reporting delay of 15 minutes. This allowed for the time required for Rx nodes to aggregate 10-minute windows of data and then five minutes for the data to propagate between all Rx root nodes. An automated response by Rx occurred when 1,000 machines were globally detected as compromised with a confidence measure equal to or less than -1. Though Rx could have responded earlier per Section 5.3.2, the higher trigger point was chosen to represent implementation margin and the reluctance of system administrators to aggressively deploy automated defenses.

M_{1A}: Identify Malware Through Incidence Data

Rx identified the presence of the Code Red I v2 worm in accordance with metric M_{1A}. The system detected the worm early with a strong confidence measure despite the reduced deployment of sensors. Figure 5.15 shows the actual, simulated spread of the Code Red worm along with the worm propagation actually observed by Rx.

Figure 5.15 indicates that the *observed* spread of the worm significantly differed both in number infected and trend as compared to the *actual* trace of worm propagation. Even

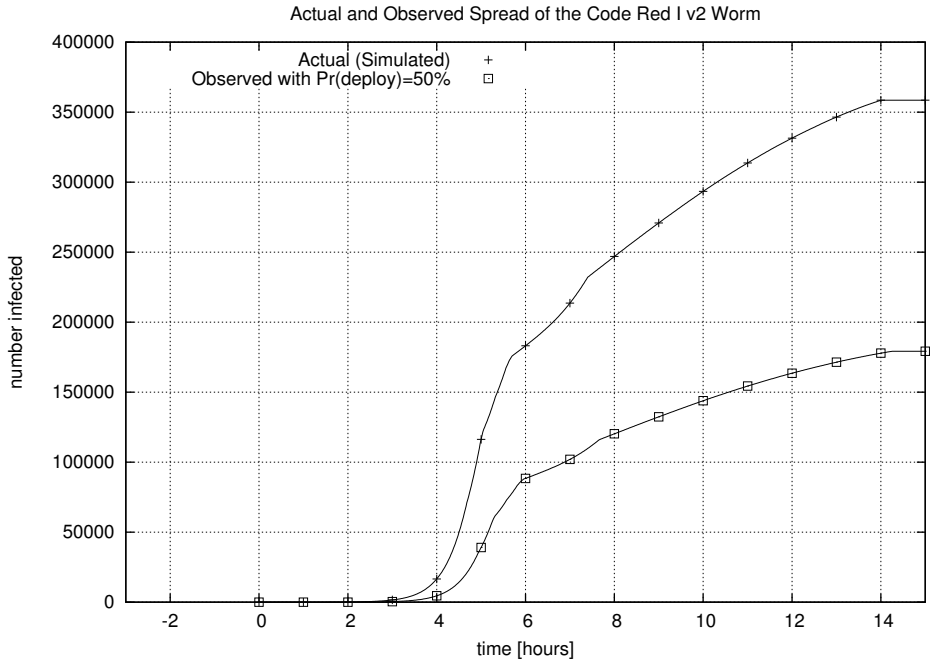

Figure 5.15: Actual and Observed Spread of the Code Red I v2 Worm

with this error in incidence data due to partial deployment, Rx identified epidemic trends in the infection reports with high confidence measures as shown in Figure 5.16. The graph indicates the confidence measure at the trigger point of 1,000 observed infected along with worm prevalence above and below that value.

Rx strongly identified the presence of epidemically propagating code per Figure 5.16. The system generated alerts with high confidence measures as early as 50 infected individuals when manually executed by the simulator. Rx returned a confidence measure below -7 for the desired trigger point of 1,000 infected. The alerts would have warned security professionals about a potential worm very early in the infection cycle allowing administrators to assess the risk to their networks, take steps to lessen the impact of the worm, prepare a recovery

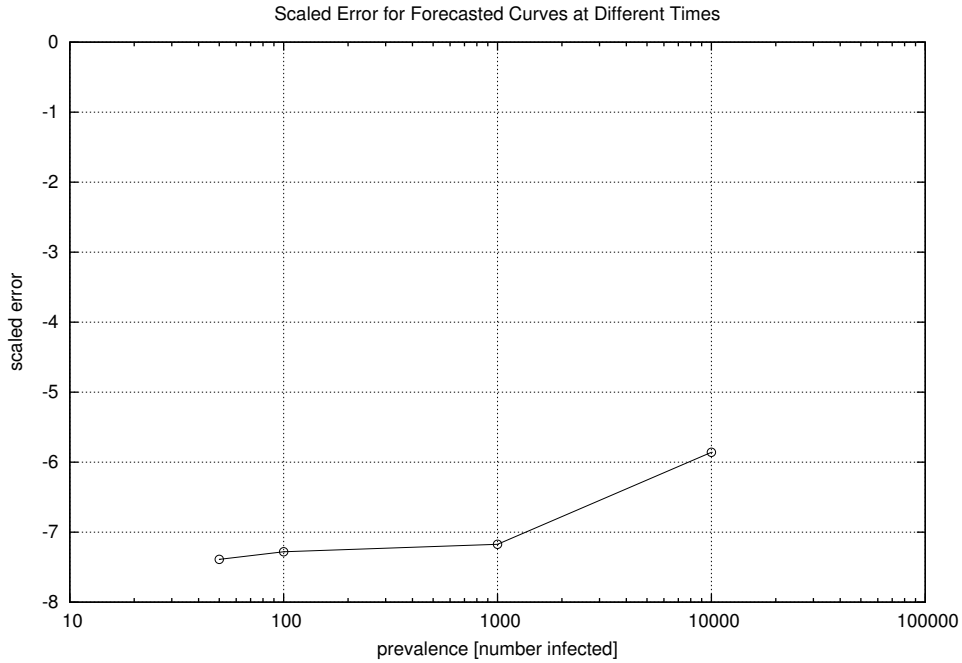

Figure 5.16: Scaled Error for Forecasted Curves at Different Times

plan, and even authorize Rx to initiate a response. Rx deployed automated defenses when it identified the worm at the defined trigger point resulting in a propagation delay for the worm and increased machine survival rate for implementing organizations as the below discussion shows.

M_2: Characterize Malware

Rx characterized the worm and made inferences on the deployment rate within the Internet. Figure 5.17 gives the the estimated probing rate, α, as a function of the number of hosts that become infected over time. The probing rate was used to judge the virulence of the worm, the saturation of the worm in the network, and the necessary response time for containment [2].

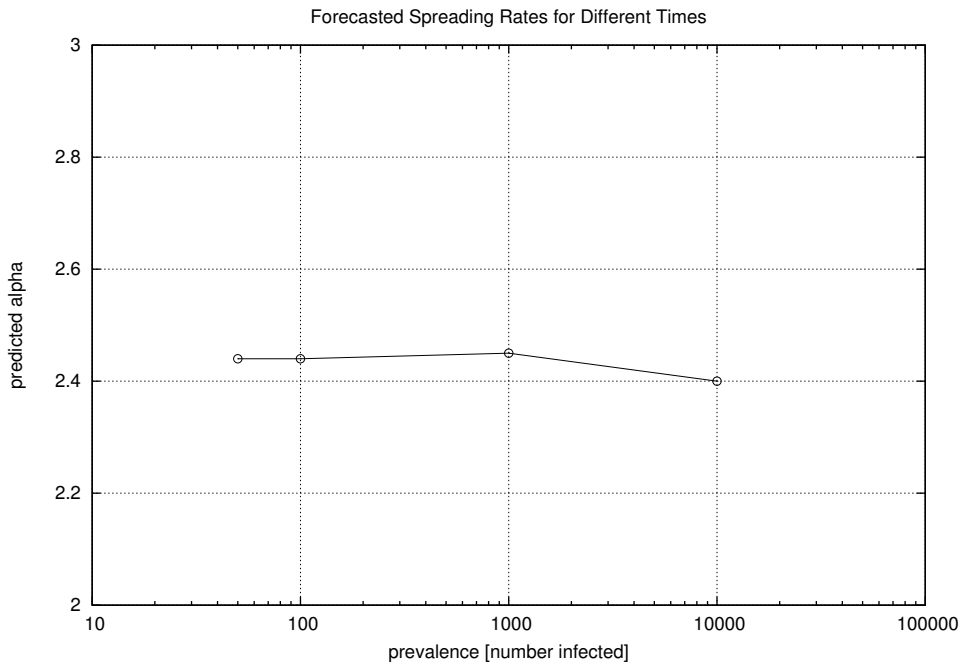

Figure 5.17: Forecasted Spreading Rates for Different Times

Rx accurately estimated the probing rate at roughly 2.4 probes per hour with a response of 2.45 probes per hour for the selected trigger point of 1,000 observed infected hosts. This closely matched the actual value of 5 probes per second resulting in an effective probing rate of 2.41 probes per hour given that $ScanRate_{effective} = ScanRate * Probability(infection) = \frac{5scans}{second} * \frac{3600seconds}{hour} * \frac{575{,}000}{2^{32}} = \frac{2.41scans}{hour}$. The characterization process is continued with Figure 5.18 which gives the probability of deployment as estimated by Rx.

Figure 5.18 indicates that Rx computed the deployment rate of malware sensors sensitive to the simulated Code Red worm at about 25% of the population. This differs from the actual value of the deployment rate of 50% due to differences between the deterministic model and the simulator that were counteracted by balancing the deployment rate and the

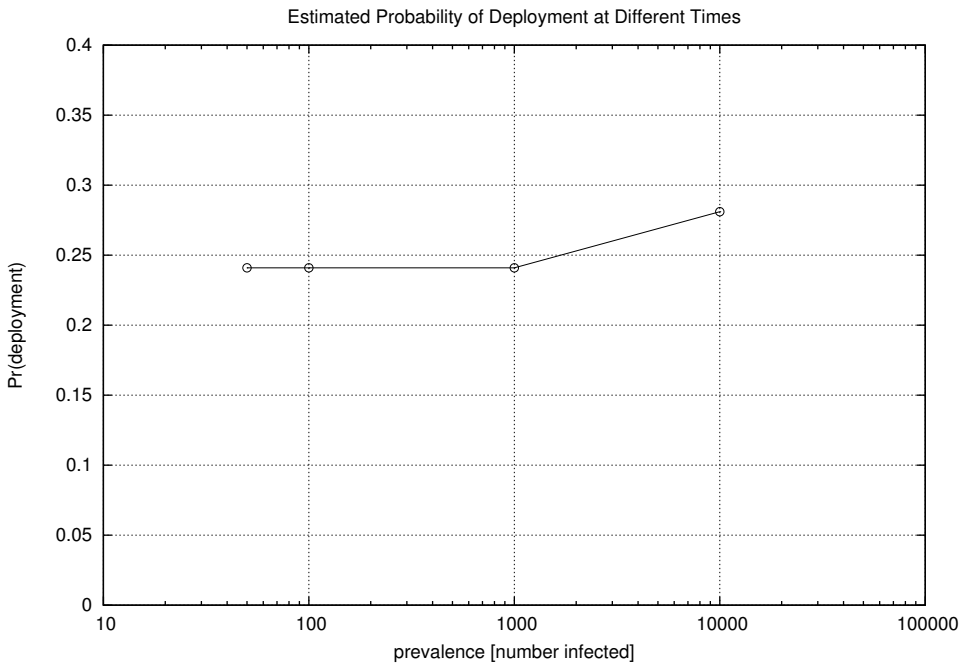

Figure 5.18: Estimated Probability of Deployment at Different Times

probing rate. The next plot in Figure 5.19 gives the forecasted spread for Code Red at the trigger point of 1,000 observed infected hosts.

Figure 5.19 shows that Rx accurately forecasted the spread of the Code Red worm through the first five hours of propagation duration. At this point, the simulated spread of the worm dramatically fell due to the deployment of human-mediated countermeasures and the bandwidth consumption caused by the worm itself. Such information could have indicated the success of the deployed countermeasures by comparing the forecasted and actual graphs of worm spread.

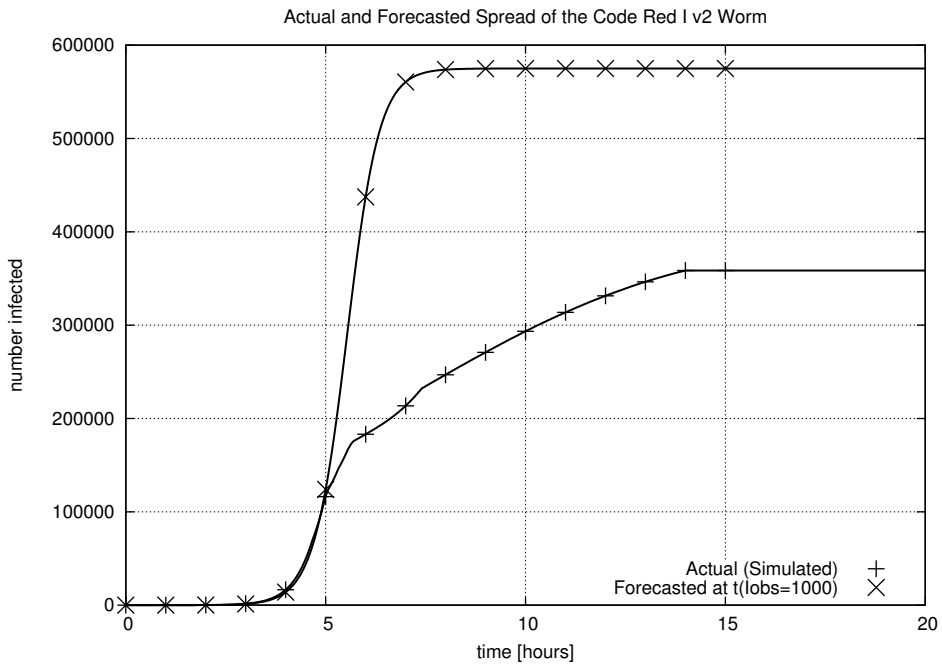

Figure 5.19: Actual and Forecasted Spread of the Code Red I v2 Worm

M_3: Effect Temporal Lag on Malware Spread

The automated response of Rx resulted in a temporal propagation delay of the Code Red worm. The system initiated a response 3.78 hours into worm propagation; this corresponded to the selected trigger point of 1,000 infected and a confidence measure below -1. Rx altered network-device rules to quarantine infected and vulnerable machines in *advance* of the spreading worm.

Figure 5.20 shows the response of Rx under this scenario. The curve indicated with '+' symbols gives the actual, simulated spread of Code Red including the effects of countermeasures and bandwidth consumption. The curve at the bottom with 'squares' provides the observed detection data for Rx given 50% deployment of anomaly detectors and an anomaly

reporting delay of 15 minutes. The top curve with 'x' symbols gives the forecasted spread of the worm while the trace with circles provides spreading data for the worm after a response by Rx.

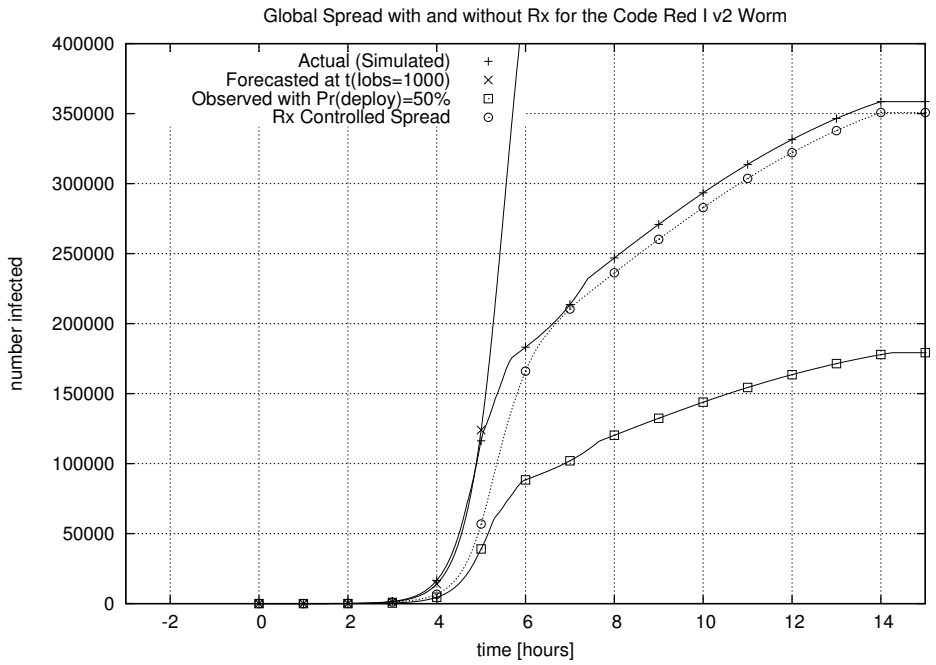

Figure 5.20: Global Spread with and without Rx for the Code Red I v2 Worm

Rx resulted in a delay in the spread of Code Red, per Figure 5.20, and lowered the number of infected machines versus no response by the system. Figure 5.21 provides a closer view of the end of the infection cycle for the Code Red worm with and without a response by Rx to better show the temporal delay in worm spread.

Figure 5.21 shows that Rx caused Code Red to experience a constant spreading delay in propagation. Table 5.2 quantifies the temporal lag resulting from the countermeasures deployed by Rx. The delay is measured as the difference between the number infected

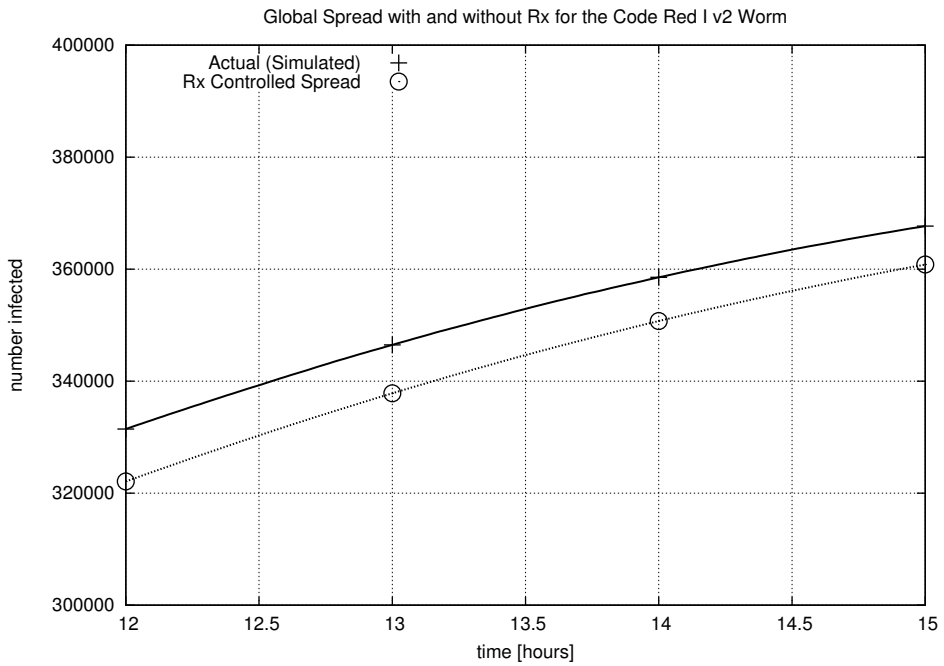

Figure 5.21: Zoom of the Global Spread with and without Rx for the Code Red I v2 Worm

without a response by Rx and those infected with a demographic response.

Table 5.2 shows that Rx caused the simulated Code Red worm to experience a temporal lag of 20 and 30 minutes with prevalence points between 5,000 and 100,000 infected machines. Researchers determined that the time to compromise the first 10,000 hosts governs the total infection time for a worm in the Internet [7]. Rx would have slowed the time for

Table 5.2: Delay of Propagation of Code Red Worm

Number Compromised	**Delay (mins)**
5,000	22.8
10,000	25.2
50,000	26.4
100,000	28.8

Code Red to achieve this number by over 25 minutes. In this simulation, this delay could provide the additional time needed to broadcast security alerts and data, manually deploy countermeasures, and trigger defensive responses such as automatic patch generation [31].

M_4: Enhance Average Survival Rate of Individuals

The Rx framework directly resulted in the infection of 7,807 fewer machines by Code Red versus a scenario without the system. Rx identified and quarantined 1,552 infected hosts and isolated 228,758 devices in *advance* of the spreading worm. This demonstrated the Internet-scale effect of Rx given 50% deployment and a 15 minute reporting delay. The system would have prevented the compromise of many more machines; however, Rx was not fully deployed and thus could not have directly affected the network access of each computer. While Rx blocked many worm infections on the implementing network, Code Red still exhibited epidemic growth by infecting machines unprotected by Rx.

The implementing organizations benefited from Rx as the system quarantined infected machines, and, moreover, prevented the infection of vulnerable machines on the local networks. Those not implementing Rx were afforded no such response until security manufacturers manually deployed defenses. The Internet community as a whole, even those without Rx, benefited from the global delay in worm propagation introduced by the system. This would have allowed for more time to manually deploy defenses and for other automated systems to respond.

5.5 Implementation Testing

The following results validate the approach used by Rx on a physical test-network as discussed in Section 3.5. A Java-based worm exercised Rx by emulating the spread of a network worm through an isolated network. The analysis below specifically establishes the ability of Rx to achieve the validation metrics defined in Section 3.7 on real systems.

A Java-based worm emulated the spread of a network worm through an isolated test-network. The worm used a random-scanning spreading strategy with a probing rate of one probe every 30 seconds. The possible targets of infectious probes by the worm were limited to the population. The worm spread through a test network of 10 machines connected by a private switch. The Java-based worm software installed on the 10 nodes served as a mechanism to abstract the exploit used by malcode and allow greater control and flexibility over the behavior of the worm. The software reported infections to an additional node that managed the experiment and executed Rx.

Figure 5.22 shows the unconstrained spread of the worm through the test-network in terms of the number infected over time. The infection cycle started with one infected individual at time 0. The worm infected half the population in about 1.5 minutes and compromised the entire population in just under four minutes.

The graph of the propagation for the Java-based worm clearly differs from earlier plots showing the spread of network worms. Most notably, the Java-based worm appeared to be spreading in a stairstep fashion. The Java-based worm transmitted probe attempts at a set interval of 30 seconds causing the cumulative number of infected individuals to change

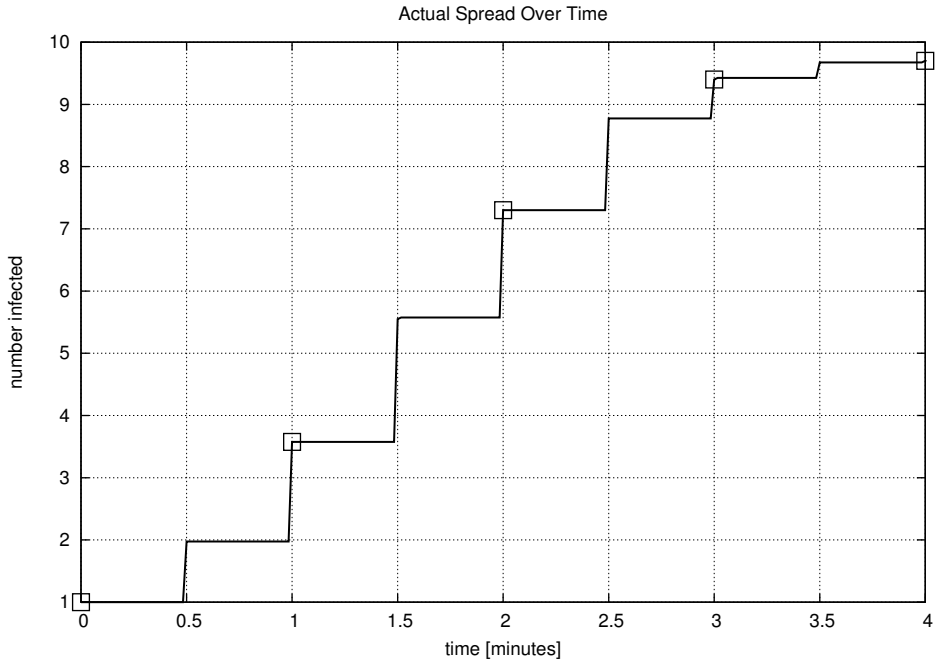

Figure 5.22: Actual Spread Over Time

around that interval and to remain constant otherwise. The probabilistic-based simulator, however, computed worm propagation by giving a worm a probabilistic chance to spread at each infinitesimally small time step. This spreading dynamic better captured the behavior of a worm in large populations where heterogeneous networks, systems, and software resulted in asynchronous scanning attempts and subsequent infections.

M_{1A}: Identify Malware Through Incidence Data

Figure 5.23 demonstrated the ability of Rx to identify a network worm with confidence measures below -4. This satisfied metric M_{1A}. The confidence measures appear to abruptly improve for the infection of three through six machines. This is based on the ability of Rx to provide a better-fit epidemic curve for the stairstep propagation pattern of the Java-based

worm at these points. Figure 5.23 demonstrated the ability of Rx to provide early detection of the worm with as few as 2 infected machines.

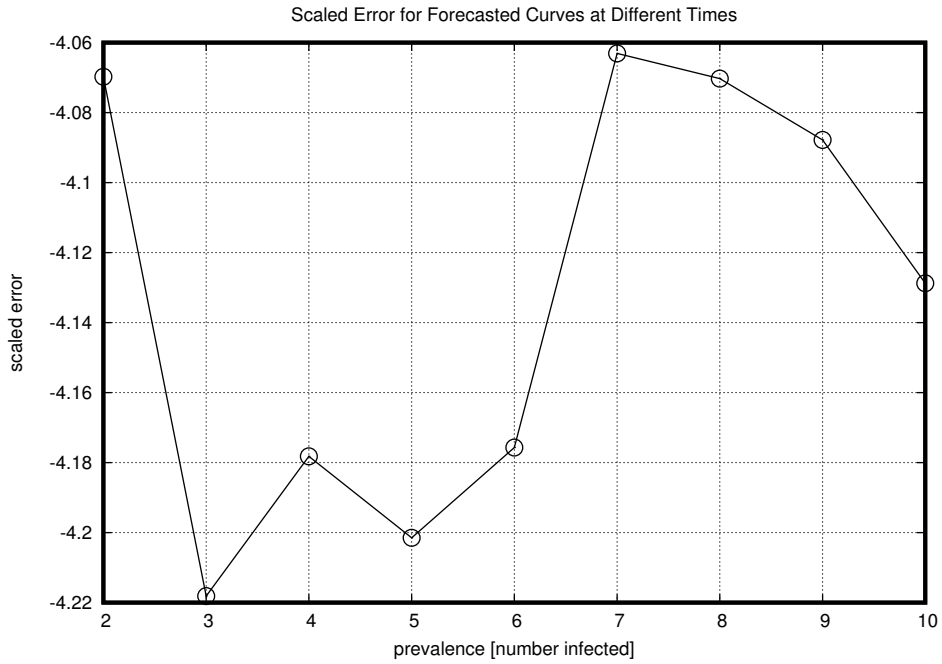

Figure 5.23: Scaled Error for Forecasted Curves at Different Times

M_2: Characterize Malware

Next, Rx characterized the Java-based worm. Figure 5.24 gives the the estimated probing rate, α, as a function of the number of hosts infected by the worm over time. The probing rate was used to judge the virulence of the worm, the saturation of the worm in the network, and the necessary response time for containment [2].

Rx produced probing rate estimates from as high as 1.69 to as low as 1.62. Thus, Rx characterized the probing rate of the worm to be one probe every 35.5 to 37 seconds. This closely approximated the actual, ideal probing rate of one probe every 30 seconds for an α

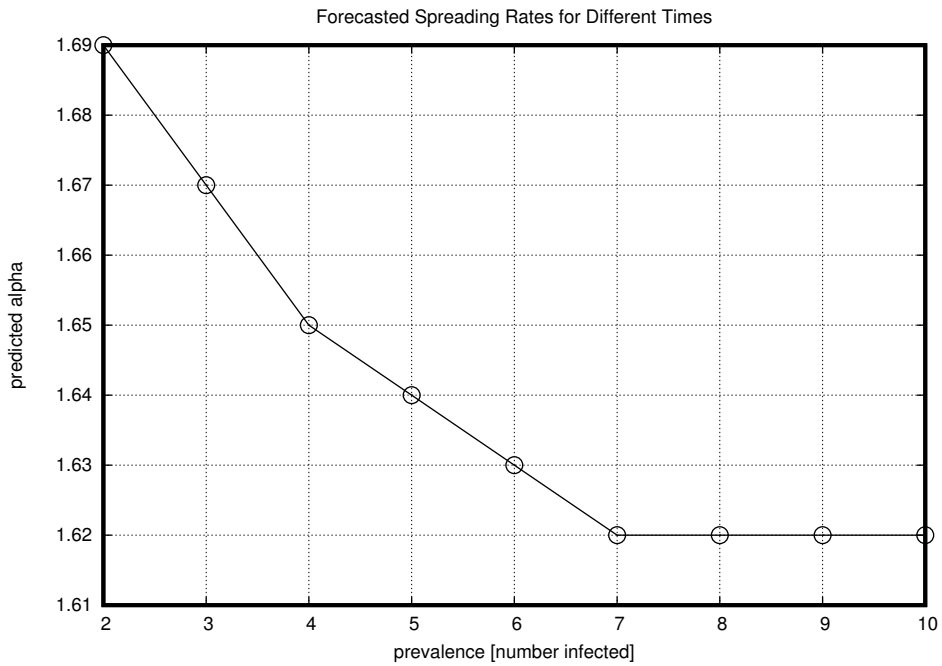

Figure 5.24: Forecasted Spreading Rates for Different Times

of 2 given that $\alpha_{ideal,actual} = \frac{1probe}{30seconds} = \frac{2probes}{minute} = 2$.

Rx produced a forecast for the propagation of the worm based on the estimated probing rate at the trigger point of four infected machines. Figure 5.25 shows the actual spread and predicted spread for the Java-based worm. The trigger point occurred about one minute after the release of the worm.

The forecasted curve in Figure 5.25 provides reasonable insight into the temporal spread of the Java-based worm through the test-network. The predicted curve matches the trend in the trace of the actual propagation of the worm. Differences in the shapes of the curve are apparent. This resulted from differences in the discrete-interval spreading behavior of the Java-based worm and the propagation computed by the epidemiological model based

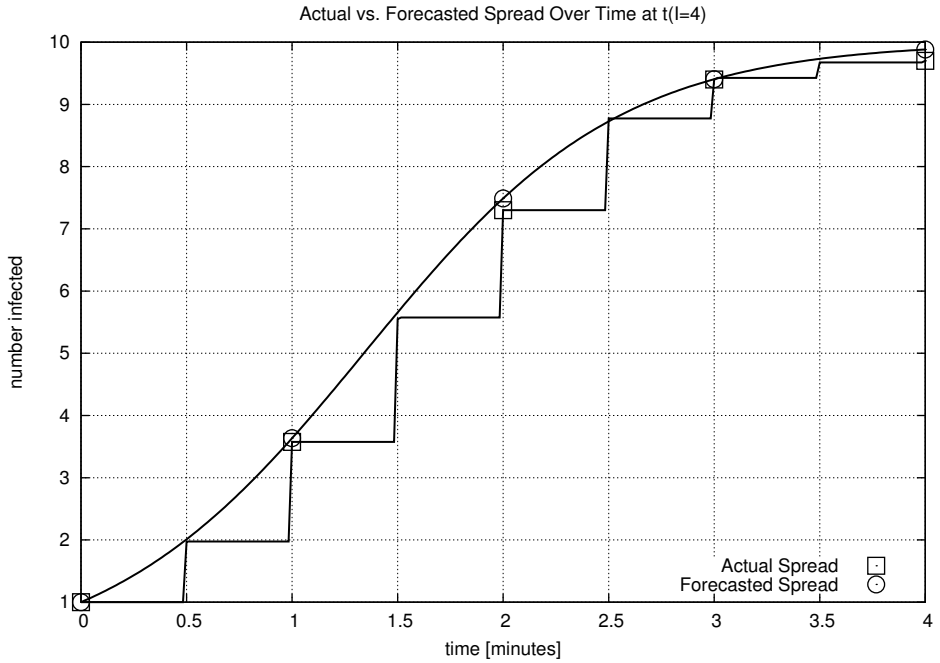

Figure 5.25: Actual vs. Forecasted Spread Over Time at t(I=4)

on continuous-time, deterministic equations. Such a prediction on the temporal spread and saturation of the worm could have helped security professionals understand the virulence of the worm and direct further response and recovery resources.

Rx performed well given a significantly smaller population than the intended deployment scenario discussed in Section 3.3. Stochastic models are typically used to describe epidemic events involving small populations while deterministic models are reserved for similar events for much larger populations [78]. The deterministic model implemented by Rx and used in this trial performed very well, trading improved accuracy with the stochastic model for computation speed offered by the deterministic model.

5.6 Conclusions

This chapter established the efficacy of the approach and the Rx system. The discussion above presented the testing results and analysis from the evaluation trials of the system. This chapter ultimately demonstrated that Rx satisfied the validation metrics set forth in Section 3.7 as exercised by verified tools.

Validation objectives, presented in Section 5.1, focused the scope of the research and validation testing to slow, random-scanning network stealth worms that spread on campus or corporate-sized networks. Both incidence and symptom-based data evaluated Rx. The former data set consisted of absolute detection alerts while the latter included anomaly data composed of worm infection reports and false-alarms.

Section 5.3 validated a single Rx node through simulation. The results showed that Rx effectively identified and characterized the epidemic propagation behavior of network worms from aggregated detection reports over broad ranges of network sizes and probing rates possibly used by the malware. Moreover, Rx demonstrated its sensitivity to worm spread early in the infection cycle with as few as two machines infected and significant accuracy in characterization at four machines infected.

Subsequent investigations showed Rx to profoundly enhance the survival rates of computers under attack by network stealth worms when deployed on campus or corporate-sized networks of 20,000 machines. The system identified and characterized a spreading worm early in the infection cycle. The automated containment response issued by Rx prevented the compromise of one to two orders of magnitude of machines. The system proved effective

given either incidence detection or anomaly report data from malcode sensors and under deployment scenarios of partial deployment and reduced probability of detection. Rx caused the worm to experience a temporal propagation delay of 33 minutes for the infection of the first 1,000 machines.

The framework defined by Rx for large-scale deployment, explored in Section 5.4, was evaluated against the simulated, Internet-scale spread of the Code Red I v2 worm. The results showed that the system significantly impacted the propagation rate of the worm with only 50% deployment across the Internet. Rx responded to the spread of the worm on the implementing networks by quarantining infected machines and isolating vulnerable machines in *advance* of the worm. This caused the worm to experience a temporal lag in propagation of over 25 minutes to infect the first 10,000 machines, thus extending the time available for data and report transfer, human responses, and other automated defenses.

Finally, Section 5.5 validated Rx by exercising an implementation of the system on a test-network. The test environment used 10 machines with a Java-based worm to emulate the spread of a network worm. The Java worm abstracted the vulnerability exploited by a worm and allowed for greater control over the malcode during the test. The results demonstrated the ability of Rx to both accurately identify and characterize a worm on a physical network.

The analysis in this chapter showed Rx to have a profound impact in the defense of computer networks against surreptitious, self-propagating code. Two factors strongly influenced the effectiveness of Rx and provided suggestions for optimal deployment of the system. These included the accuracy of demographic data and conditions leading to detection

and response delays.

Demographic data served a two-fold purpose. It provided population statistics to the bio-mathematical model and ultimately guided the quarantine response by isolating infected and vulnerable machines during a worm attack through the manipulation of routing and firewall rules. Errors in demographic data lessened the effectiveness of the defensive response by failing to remove the infected and vulnerable individuals from the population during a worm attack. This significantly delayed the average spreading rate of the worm, although it still allowed the worm to continue to propagate within the network. The distribution of demographic scanners, especially those implemented as cooperative end-host software, throughout the network could have greatly improved the accuracy of demographic data for use by Rx.

Probability of detection or anomaly generation by malware sensors, partial deployment of malware sensors, partial deployment of Rx nodes outside of an implementing organization, and data reporting delays affected the performance of Rx. These factors all caused Rx to experience identification and response delays by time-shifting the observed propagation of the worm. The combined probabilities of deployment and malware detection or anomaly generation proved the most important. These strongly governed the identification and response time for malware by Rx and thus directly impacted the number of systems prevented from infection.

Rx, however, was not a suitable network defense solution under certain circumstances. This resulted from basic limitations of the approach or from the target scenario that guided

the design of the system. Undesired deployment situations for Rx included malware scanning strategies other than a random-scanning mechanism and fast-spreading worms.

Rx thwarted worms that implemented a random-scanning spreading strategy. Malcode could have also spread through other mechanisms such as open network shares, file-sharing and other networked applications, or social groups established by the user via e-mail and instant messenger contacts. The SIR model implemented by Rx was not sufficient for describing these unique spreading strategies. However, the epidemiological model in Rx could have been extended to show sensitivity to other contact structures.

Rx was designed to counter surreptitious, slow-spreading worms thus the system was ineffective against fast-spreading worms. The limiting factors for system response time were the reporting delays from host or network sensor to Rx as well as the processing time for the epidemiological model to analyze the aggregated data. The latter could have been improved with a more efficient parameter search algorithm or with parallel processing. Researchers estimated that a TCP-based worm could have compromised 95% of the one million vulnerable machines across the Internet in 1.3 seconds; a UDP version of this worm would have required only 510 milliseconds [81]. The short infection cycle duration did not allow sufficient time for sensor reporting, data processing, and response by Rx.

The next chapter summarizes the research conducted for this effort and highlights the significant contributions made in the course of this endeavor. The chapter concludes this document with suggestions on future research direction and closing thoughts.

Chapter 6

Conclusions

This research effort explored and developed techniques to defend computer networks against network stealth worms through the real-time application of concepts from biological epidemiology. Rx, a system created as a product of this research, stood as the embodiment of the biological approach to computer security. The system specifically used bio-mathematical modeling and demographic analysis to identify, characterize, and control the spread of surreptitious, self-propagating code. Rx profoundly increased the survival rate of computers under attack by a network stealth worm. This chapter summarizes the work and significant advances accomplished in the course of this research and discusses future direction for the continued development of this knowledge and approach.

6.1 Research Summary

The purpose of this research was to develop a cyber security system to mitigate the threat of network stealth worms. The system, Rx, defined a reporting and response framework to identify, characterize, forecast, and control stealth worms at the network-level based on host-

oriented incidence and anomaly reports. Rx adapted concepts from biological epidemiology to include bio-mathematical modeling and demographic analysis in defense against digital malware. The system depended on network or host-based sensors to be sensitive to the effects of a network stealth worm. The system did not require new hardware, changes to existing protocols, or participation outside the implementing organization. Through simulation and implementation, Rx demonstrated its ability to dramatically increase the average survival rate of computers under attack by a network stealth worm.

This dissertation documents the approach, results, and analysis of the unique approach to the defense against network stealth worms chosen by this research. Such knowledge will benefit the continued development of defense mechanisms against surreptitious, self-propagating code. Chapter 1 introduced and motivated the problem addressed by this research. It further established the goals and contributions of this effort.

Chapter 2 provided background information and a review of literature on the subject of network worms and subsequent mitigation techniques. It continued by presenting a detailed discussion of biological epidemiology and its use in digital networks. Most importantly, the chapter showed the extensive body of knowledge surrounding the use of epidemiology with network worms focused on historic and hypothetical propagation studies as well as speeding large-scale network worm spreading simulations.

Chapter 3 presented the design of the system developed through this research, Rx, to address the threat of network stealth worms through biological epidemiology. The chapter began by mapping biological terminology and techniques into the digital problem space and

specifically identified those concepts adapted by Rx. The design of Rx was defined in three stages: the function of individual modules, the integration of those modules into a system, and the framework for configuring the cooperation of multiple Rx nodes. Following this came a description of the simulated and implemented versions of the security system. The chapter closed by defining five metrics to validate and measure the effectiveness of Rx.

Chapter 4 verified the bio-mathematical model incorporated into Rx as well as the simulator and implemented a Java-based worm used to generate propagation data to exercise the system. Data from the spread of the biological disease Hong Kong Flu and the digital contagion Code Red I v2 worm verified the model and simulation in both the biological and digital paradigms. A stochastic, epidemiological model verified the implemented Java-based worm.

Chapter 5 provided and analyzed the results from system testing, thus evaluating the efficacy of Rx. Simulation demonstrated the early identification and characterization of network stealth worms by Rx as well as the effectiveness of the system over broad ranges of network sizes and malware probing rates. Subsequent simulations showed the ability of Rx to profoundly enhance the survival rate by several orders of magnitude of computers under attack by a network stealth worm given deployment on a campus-sized network. The chapter continued to provide results detailing the global effect of temporal lag in worm propagation due to a response by Rx given partial deployment. The chapter ended with implementation testing that showed the system accurately identified and characterized worm behavior realized by Java-based software on an isolated test-network.

6.2 Significant Contributions

The primary contribution of this research was the development of an approach for the real-time adaptation and use of epidemiological concepts on digital networks for the early identification, characterization, and effective control of network stealth worms. Previous usage of epidemiology in computer networks focused on mathematical models to study the propagation of classic and hypothetical viruses and worms or to speed large-scale network simulations. This research significantly extended that work by using epidemiology as an approach to real-time network security in terms of *early identification*, *threat evaluation*, and *containment* of surreptitious, self-propagating code.

Techniques from biological epidemiology were shown to support the real-time detection, characterization, forecasting, and containment of network stealth worms. The bio-mathematical model served as a network-level mechanism to identify worm infection behavior buried in end-host anomaly data. The model acted as a correlation tool to detect epidemic propagation trends in the anomaly data. The early detection of self-propagating code proved crucial for the effective containment of the malcode.

The models further parameterized and forecasted the temporal spread of a slow, random scanning network stealth worm early in the infection cycle. This could have allowed network administrators to assess the virulence and threat of the malcode. Security professionals could have used this information to manage response resources especially when under attack by multiple strains of malware.

Demographic analysis demonstrated its profound impact on the survival rate of com-

puters under attack by a network worm. This technique from epidemiology analyzed the infected hosts to determine and then quarantine the subset of vulnerable machines within the population, a response termed *advanced quarantine*. The response from demographic analysis not only prevented the infection of isolated machines but also caused a spreading worm to exhibit a temporal propagation lag, thereby allowing more time for data and alert transfer, manual countermeasures, and other automated responses.

This research also resulted in the definition of a scalable, fault-tolerant reporting and response framework for epidemic events. This strategy was valid on networks from the relatively small wireless networks through medium-sized corporate or campus networks. The framework showed its ability to enhance the survival rate of hosts under worm attack with Internet-scale participation. This approach did not require new hardware, changes to existing protocols, or participation outside the implementing organization.

Finally, this work showed application to a wider range of challenges. The bio-mathematical model is inherently extensible, allowing augmentations of this to respond to variations of self-propagating code. The overall approach was applicable to other forms of malware by interchanging the epidemic model with one more appropriate. Finally, the strategy allowed anomaly detectors to be sensitive to lower reporting thresholds and a variety of often benign yet potentially useful events.

6.3 Future Research Direction

This research contributed the design and analysis of a cyber security system to defend computer networks against surreptitious, self-propagating code by embracing concepts from biological epidemiology. Significant advancement of this work could be realized by formally implementing and deploying Rx on a campus-sized network to evaluate and enhance the effectiveness of the system in the identification, characterization, and control of network stealth worms. Additional improvements could extend the range of malcode to which Rx was sensitive. This research focused on malcode that exhibited epidemic growth with a random-scanning spreading strategy. Other pressing threats to network security included malcode that spread via point-source infections common to spyware, through logical and social networks created by the user such as p2p connections and e-mail and instant messenger contacts as used by network worms, and manual compromise of machines as often observed with DDoS tools.

6.4 Concluding Thoughts

This research extended the knowledge concerning the use of biological epidemiology in digital networks. It showed that the real-time use of epidemiological techniques–namely, biomathematical modeling and demographic analysis–possessed the capability to identify, characterize and control the spread of surreptitious, self-propagating code on computer networks. Rx embodied this approach in a defined system and framework for the effective reporting

and control of epidemic events.

The incremental deployment of Rx in support of or as part of existing security packages will prove critical to the adoption of the approach and even the complete Rx system. Few ideas, especially those incorporating new approaches, receive immediate and comprehensive adoption. The gradual release of the system into the mainstream of network security will facilitate the use of the system and its concepts. The modules constituting Rx and the resultant functionality of the system were largely independent. Malware identification, parameterization, forecasting, and control were the four functions that could have been divided and added piece-wise to existing computer security tools. The system as a whole must easily interface with existing commercial-off-the-shelf security products.

Finally, epidemiological models received significant use in the study of historic and hypothetical propagation of viruses and worms and also found application in speeding large-scale worm spread simulations. Unfortunately, this inadvertently stereotyped epidemiological models into the *study* of worm propagation through *simplistic* analysis techniques. Epidemiology, however, is a field rich with history, knowledge, techniques, and successes for understanding and controlling the spread of biological disease in living organisms. This science and even biology in general stand to contribute further to digital systems and their security.

Bibliography

[1] Symantec, "Symantec Internet Security Threat Report Identifies More Attacks Now Targeting e-Commerce, Web Applications." Downloaded 28 Mar. 2005, http://enterprisesecurity.symantec.com/content.cfm?articleid=4689&EID=0, Sept. 2004.

[2] T. Timmreck, *An Introduction to Epidemiology*. Sadbury, MA: Jones and Bartlett, 1998.

[3] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the Slammer Worm," *IEEE Security and Privacy*, pp. 33–39, 2003.

[4] D. Moore and P. Wilson, "The Spread of the Code Red Worm (CRv2)," 2001. Downloaded 14 Apr. 2005, http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml.

[5] N. Joukov and T. Chiueh, "Internet Worms as Internet-Wide Threat." Downloaded 5 May 2003, http://citeseer.ist.psu.edu/joukov03internet.html, Sept. 2003.

[6] M. Ward, "Money motive drove virus suspects," *BBC News website*, Sept. 2005. Downloaded 7 Sept. 2005, http://news.bbc.co.uk/1/hi/technology/4205220.stm.

[7] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in Your Spare Time," *Proceedings of the 11th USENIX Security Symposium*, 2002.

[8] S. Singh, C. Estan, G. Varghese, and S. Savage, "Automated Worm Fingerprinting," *Proceedings of USENIX OSDI'04*, pp. 45–60, 2004.

[9] T. Chen and J.-M. Robert, "Worm Epidemics in High-Speed Networks," *Computer*, vol. 37, pp. 48–53, June 2004.

[10] "Octave." http://www.octave.org, 2005.

[11] "VMware." http://www.vmware.com, 2006.

[12] J. Shoch and J. Hupp, "The 'Worm' Programs- Early Experience with a Distributed Computation," *Communications of the ACM*, vol. 25, no. 3, pp. 172–180, 1982.

[13] J. Brunner, *The Shockwave Rider*. Canada: Del Ray, 1975.

[14] "Morris Worm," *Wikipedia*, 2005. Downloaded 21 Mar. 2005, http://en.wikipedia.org/wiki/Morris_worm.

[15] M. W. Eichin and J. A. Rochlis, "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988," *IEEE Symposium on Research in Security and Privacy*, pp. 326–343, May 1989.

[16] GovExec, "A worm that turned: A new approach to hacker hunting." Downloaded 20 Mar. 2005, http://www.govexec.com/dailyfed/0103/012903worm.htm, Jan. 2003.

[17] J. Nazario, J. Anderson, R. Wash, and C. Connelly, "The Future of Internet Worms," *Technical Report, Crimelabs Research*, 2001. Downloaded 8 Nov. 2004, http://www.crimelabs.net/docs/worms/worm.pdf.

[18] Dildog, "Tao of Windows Buffer Overflow," May 1998. Downloaded 2 Apr. 2005, http://www.cultdeadcow.com/cDc_files/cDc-351.

[19] B. Krebs, "Hacking Made Easy," *washingtonpost.com*, Mar. 2006. Downloaded 17 Mar. 2006, http://www.washingtonpost.com/wp-dyn/content/article/2006/03/16/AR2006031600916.html.

[20] M. Zalewski, "I don't think I really love you or Writing Internet Worms for Fun and Profit." Downloaded 11 Nov. 2004, http://lcamtuf.coredump.cx/worm.txt, 2000.

[21] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, *Internet Denial of Service: Attack and Defense Mechanisms*. Upper Saddle River, NJ: Prentice Hall, 2005.

[22] T. H. Project and R. Alliance, "Know your Enemy: Tracking Botnets," Mar. 2005. Downloaded 15 Mar. 2005, http://www.honeynet.org/papers/bots.

[23] J. Nazario, *Defense and Detection Strategies against Internet Worms*. Norwood, MA: Artech House, Inc., 2004.

[24] J. Bigus and J. Bigus, *Constructing Intelligent Agents Using Java*. New York, NY: John Wiley and Sons, Inc, 2 ed., 2001.

[25] M. Resnick, *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press, 1994.

[26] S. H. Serge Fenet, "A distributed Intrusion Detection and Response System based on a mobile autonomous agents using social insects communication paradigm," *Proceedings of the 1st International Workshop on Security of Mobile Multiagent Systems (SEMAS)*, 2001.

[27] S. Staniford-Chenu and T. Heberlein, "Holding Intruders Accountable on the Internet," *IEEE Symposium on Security and Privacy, Proceedings*, pp. 39–49, May 1995.

[28] S. Forrest, A. Somayaji, and D. Ackley, "Building Diverse Computer Systems," *Operating Systems, The Sixth Workshop on Hot Topics in*, pp. 67–72, May 1997.

[29] T. Wassenaar and M. Blaser, "Contagion on the Internet," *Emerging Infectious Diseases*, vol. 8, pp. 335–336, Mar. 2002.

[30] M. Williamson, "Biologically Inspired Approaches to Computer Security," *Information Infrastructure Laboratory, HP Laboratories Bristol*, Jun. 2002. Downloaded 8 Nov. 2004, http://www.hpl.hp.com/techreports/2002/HPL-2002-131.html.

[31] S. Sidiroglou and A. Keromytis, "Countering Network Worms Through Automatic Patch Generation," 2003. Technical Report CU CS-029-03.

[32] "The Honeynet Project." http://www.honeynet.org, 2005.

[33] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Network Telescopes: Observing Small or Distant Security Events," *Technical Report, Invited Presentation at the 11th USENIX Security Symposium*, Aug. 2002.

[34] "Snort Intrusion Detection System." http://snort.org, 2005.

[35] "Bro Intrusion Detection System." http://bro-ids.org, 2005.

[36] "DShield." http://www.dshield.org, 2005.

[37] "Wormrader." http://www.wormradar.org, 2005.

[38] "Tripwire." http://www.tripwire.com, 2005.

[39] V. Berk and G. Bakos, "Designing a Framework for Active Worm Detection on Global Networks," *Information Assurance, First IEEE International Workshop on, Proceedings*, pp. 13–23, Mar. 2003.

[40] D. Ellis, J. Aiken, K. Attwood, and S. Tenaglia, "A Behavioral Approach to Worm Detection," *Conference on Computer and Communications Security, Proceedings of the 2004 ACM Workshop on Rapid Malcode*, pp. 43–53, 2004.

[41] S. Forrest, A. Perelson, L. Allen, and R. Cherukuri, "Self-Nonself Discrimination in a Computer," *IEEE Computer Society Symposium on Research in Security and Privacy, Proceedings*, pp. 202–212, May 1994.

[42] J. Newsome and D. Song, "Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploit on Commodity Software," *Proceedings of the 12th Annual Network and Distributed System Security Symposium*, 2005.

[43] M. Costa, J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L. Zhang, and P. Barham, "Vigilante: End-to-End Containment of Internet Worms," *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles*, pp. 133 – 147, 2005.

[44] J. Kephart and W. Arnold, "Automatic Extraction of Computer Virus Signatures," *Proceedings of the 4th Virus Bulletin International Conference*, pp. 178–184, 1994.

[45] H.-A. Kim and B. Karp, "Autograph: Toward Automated Distributed Worm Signature Detection," *Proceedings of the USENIX Security Symposium*, Aug. 2004.

[46] S. Singh, C. Estan, G. Vergese, and S. Savage, "The EarlyBird System for Real-time Detection of Unknown Worms." Technical Report CS2003-0761, CSE Department, UCSD, May 2003.

[47] "Welcome to My Tarpit: The Tactical and Strategic Use of LeBrea." Downloaded 13 Nov. 2004, http://www.labreatechnologies.com.

[48] H. Toyoizumi and A. Kara, "Predators: Good Will Mobile Codes Combat against Computer Viruses," *Workshop on, Session: Intrusion Detection and Response, Proceedings*, pp. 11–17, 2002.

[49] B. McWilliams, "New worms seek out destroy code red." Downloaded Apr. 2002, http://www.commoncriteria.org/news/newsarchive/Sept01/sept02.htm, Sept. 2001.

[50] F. Castaneda, E. C. Sezer, and J. Xuy, "WORM vs. WORM: Preliminary Study of an Active Counter-Attack Mechanism," *Conference on Computer and Communications Security, Proceedings of the 2004 ACM Workshop on Rapid Malcode*, Oct. 2004.

[51] R. Pande, "Using Plant Epidemiological Methods to Track Computer Network Worms." Thesis, Computer Science, Virginia Polytechnic and State University, May 2004.

[52] N. Bailey, *The Mathematical Theory of Infectious Diseases*. New York, NY: Hafner Press, 1975.

[53] V. Capasso, *Mathematical Structures of Epidemic Systems*. Berlin, Heidelberg: Springer-Verlag, 1993.

[54] C. Duncan and S. Scott, *Biology of Plagues: Evidence from Historical Populations*. Edinburgh, United Kingdom: Cambridge University Press, 2001.

[55] "Dictionary.com." http://www.dictionary.com, 2005.

[56] T. Timmreck, *An Introduction to Epidemiology*. Sadbury, MA: Jones and Bartlett, 3 ed., 2002.

[57] F. Cohen, "Computer Viruses Theory and Experiments." Downloaded 30 Mar. 2005, http://vx.netlux.org/lib/afc01.html, 1984.

[58] J. Kephart and S. White, "Directed-Graph Epidemiological Models of Computer Viruses," *IEEE Computer Symposium on Research in Security and Privacy, Proceedings*, pp. 343–359, May 1991.

[59] J. Kephart and S. White, "Measuring and Modeling Computer Virus Prevalence," *Research in Security and Privacy, 1993, Proceedings, 1993 IEEE Computer Society Symposium on*, pp. 2–15, May 1993.

[60] J. Kephart, S. White, and D. Chess, "Computers and Epidemiology," *IEEE Spectrum*, 1993.

[61] M. Williamson and J. Leveille, "An Epidemiological Model of Virus Spread and Cleanup," *Information Infrastructure Laboratory, HOP Laboratories Bristol*, Feb. 2003. Downloaded 8 Nov. 2004, http://www.hpl.hp.com/techreports/2003/HPL-2003-39.html.

[62] C. Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis," *Proceedings of the 9th ACM Symposium on Computer and Communications Security*, pp. 138–147, Oct. 2002.

[63] Z. Chen, L. Gao, and K. Kwiat, "Modeling the Spread of Active Worms," *INFOCOM 2003, Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE*, vol. 3, pp. 1890–1900, Mar./Apr. 2003.

[64] C. Zou, W. Gong, and D. Towsley, "Worm Propagation Modeling and Analysis under Dynamic Quarantine Defense," *Quarantine Defense, ACM CCS Workshop on Rapid Malcode (WORM'03)*, 2003.

[65] M. Liljenstam, D. Nicol, V. Berk, and R. Gray, "Simulating Realistic Network Worm Traffic for Worm Warning System Design and Testing," *Proceedings of the 2003 ACM Workshop on Rapid Malcode*, pp. 24–33, 2003.

[66] "ns2." http://www.isi.edu/nsnam/ns, 2005.

[67] "Center for Disease Control and Prevention," 2005. http://www.cdc.org.

[68] "Nessus Open Source Vulnerability Scanner Project." http://www.nessus.org, 2005.

[69] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *scheduled for presentation at the IEEE Symposium on Security and Privacy*, May 2005.

[70] M. Roberts, "Bird flu 'has pandemic potential'," *BBC News website*, Feb. 2005. Downloaded 15 Sept. 2005, http://news.bbc.co.uk/1/hi/health/4280817.stm.

[71] G. Davis, "Mathematical Models for the Urban-Suburban Spread of Disease." Honors Thesis, Duke University, 1988.

[72] D. Smith and L. Moore, "The SIR Model for Spread of Disease," *Technical Report, Connected Curriculum Project, Duke University*, 2004. Downloaded 18 Jul. 2005, http://www.math.duke.edu/education/postcalc/sir.

[73] K. Carlyle, "Understanding biases in epidemic models important when making public health predictions," *Medical News Today*, Jul. 2005. Downloaded 9 Sept. 2005, http://www.medicalnewstoday.com/medicalnews.php?newsid=28061.

[74] J. Ira Longini, E. Halloran, A. Nizam, and Y. Yang, "Containing Pandemic Influenza with Antiviral Agents," *American Journal of Epidemiology*, pp. 159:623–633, 2004.

[75] D. Smith and L. Moore, *Calculus: Modeling and Application*. D. C. Heath and Co., 1996.

[76] "Cooperative Association for Internet Data Analysis (CAIDA)." http://www.caida.org/home/, 2006.

[77] R. Gallop, "Modeling General Epidemics: SIR MODEL," *Proceedings of the 12th Annual NorthEast SAS Users Group Conference (NESUG)*, 1999.

[78] D. Daley and J. Gani, *Epidemic Modeling: an Introduction*. Cambridge, United Kingdom: Press Syndicate of the University of Cambridge, 1999.

[79] H. Trottier and P. Philippe, "Deterministic Modeling Of Infectious Diseases: Theory And Methods," *The Internet Journal of Infectious Diseases*, vol. 1, no. 2, 2001.

[80] "Virginia Tech DShield." http://dshield.cirt.vt.edu, 2005.

[81] S. Staniford, D. Moore, V. Paxson, and N. Weaver, "The Top Speed of Flash Worms," *Conference on Computer and Communications Security, Proceedings of the 2004 ACM Workshop on Rapid Malcode*, Oct. 2004.